Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Diseño de un teclado capacitivo reprogramable

Autor: Pablo Manuel Guzmán Manzanares
Tutor: Manuel Ángel Perales Esteve

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Diseño de un teclado capacitivo reprogramable

Autor:
Pablo Manuel Guzmán Manzanares

Tutor:
Manuel Ángel Perales Esteve
Profesor titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Carrera: Diseño de un teclado capacitivo reprogramable

Autor: Pablo Manuel Guzmán Manzanares

Tutor: Manuel Ángel Perales Esteve

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El secretario del Tribunal

A mi familia

A mis amigos

Y en especial a Lidia.

Agradecimientos

Tras estar acabando este proyecto, me paro a pensar en todo lo que he vivido desde que comenzó esta etapa de la universidad, todos los amigos que he hecho, todas las relaciones que he desarrollado y cómo ha cambiado todo desde que el Pablo pequeño tomó la decisión de venir a estudiar GIERM en la US porque el nombre sonaba *guay*. Han sido 4 años durísimos, cada uno más duro que el anterior por diversos motivos. Y este último ha arrasado con absolutamente todo lo que me mantenía con pasión por lo que realizaba en la universidad, así que me agarré a lo único que me quedaba: trabajo duro constante.

Este modo de hacer las cosas mío ha producido muchos roces con mi familia y amigos, y ahora se termina, por fin, veo la luz al final del túnel. He avanzado a ciegas durante tanto tiempo, andando por disciplina más que por gusto que estar ya en ese punto de ver el final me hace valorar a todas esas personas que han aguantado junto a mí esa parte del camino.

Gracias a mi familia por no colapsar sobre mí porque mis planes chocaban con sus vacaciones, y espero que entender por fin que los quiero como a nada en este mundo, solo que a veces no saben ver cuando necesito esa palmadita extra. No pasa nada, sé que la próxima vez seremos mejores. Y pese a todo lo que pueda pasar o salir, siempre seréis una de mis piedras angulares porque os quiero muchísimos a todos, con vuestros defectos incluidos.

Gracias a Lidia por dar todo lo posible por echarme cables que no tenía cuando estaba al borde de la rendición. También gracias por darme mi espacio cuando estaba distante por las preocupaciones que me producía todo este proceso. Has sido una parte crucial de que yo haya llegado aquí de una pieza. Te quiero muchísimo.

Y, por último, gracias al gran grupo de gente que puedo llamar mi segunda familia que he desarrollado aquí en Sevilla. Cuando se os ha dicho que se necesitaba estar un ratillo hablando para distraerme, habéis estado. Si necesitaba ayuda para programar algo o revisar un proceso, ahí habéis estado. Si necesitaba ayuda para gestionar información, ahí habéis estado. Sois una parte fundamental de lo que soy ahora.

Pablo Manuel Guzmán Manzanares

Futuro graduado de GIERM

Sevilla, 2021

Resumen

El proyecto consiste en la creación de un teclado capacitivo reprogramable y personalizable, usando un launchPad EK-TM4C1294XL al que se le conecta una pantalla ME813A-WH50C. En esta pantalla se generarán una serie de botones mediante el uso de modos de edición en diferentes páginas para crear perfiles encapsulados. Se conectará a un ordenador de manera que este detecte el periférico como un teclado HID, siendo los botones creados en la pantalla, sus teclas. Para la personalización se crea un servidor web Local contenido dentro del launchPad desde el que se tendrá la opción de personalizar gráficamente el dispositivo, así como la funcionalidad de sus botones.

Abstract

The project consists in the making of a reprogrammable and customizable capacitive keyboard, using a EK-TM4C1294XL launchPad connected to a ME813A-WH50C screen. By the use of different edition modes, a series of buttons will appear in the screen distributed across a variety of pages aiming to create encapsulated profiles. This gadget will be connected to a computer for it to detect the peripheral device as a HID keyboard, turning those screen buttons into its keys. For the customization, a Local Web Server will be created within the own launchPad, enabling a graphic customizing of the device and the functioning of its buttons.

Introducción

Este proyecto nace de la inquietud del alumno por los teclados mecánicos y su inquietud por probar de primera mano un desarrollo de un proyecto desde cero, ya que piensa que en el futuro serán habilidades que le serán útiles de cara al desarrollo de proyectos personales y profesionales.

El teclado capacitivo reprogramable que se quiere conseguir pretende ser una alternativa funcional real a teclados parecidos que ya existen en el mercado, dando un gran abanico de opciones con sus opciones de personalización al usuario para que sea un periférico que se pueda usar en cualquier campo.

Se considera que este trabajo fin de carrera ha cumplido los objetivos propuestos en principio, pese al limitado abanico de opciones debido a que cabía la posibilidad de que el hardware seleccionado para el problema no funcionase o no estuviese disponible, debido a las partes tan específicas de las que consta el proyecto. Pese a esto, es una alegría poder escribir que todo ha ido bien y el sistema es completamente funcional.

Se ha trabajado planteando los diferentes bloques principales del proyecto, tras eso se cogía el bloque determinado y empezaba a desarrollarse hasta terminar con todo lo que se podía hacer con el sistema en ese punto temporal. Tras esto se pasaba al siguiente y se hacía lo mismo, para después comprobar si esta nueva integración implicaba un desarrollo mayor del resto de los bloques. Así, hasta dar el trabajo por finalizado. Esta es la forma que se ha encontrado mejor al estar el trabajo desarrollado por una sola persona y ser la mayor parte del código funcional hecha desde 0.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Introducción **xv**

Índice **xvii**

Índice de Tablas **xxi**

Índice de Figuras **xxiii**

Notación **xxv**

1 Estudio del problema, actualidad y soluciones existentes **1**

1.1 Planteamiento del problema *1*

1.2 Actualidad de Teclados Táctiles Reprogramables *2*

1.2.1 MacBook TouchBar 3

1.2.2 ScreenPad ASUS ZenBook Pro 14 4

1.2.3 ASUS ZenBook Pro Duo 5

1.2.4 Stream Deck de ElGato 6

1.2.5 Numpad reconfigurables, teclados de macros y switches electro-capacitivos 7

2 Resolución planteada del proyecto, Componentes usados y Estructura del proyecto **11**

2.1 Resolución del problema planteado *11*

2.1.1 Pasos previos, planteamiento de diseños erróneos y descartes 11

2.1.2 Idea final 12

2.2 Hardware del Proyecto *12*

2.2.1 ME813A-WH50C 12

2.2.2 EK-TM4C1294XL 14

2.2.3 Conexión de los elementos 14

2.3 Software del Proyecto *16*

2.3.1 Pantalla 16

2.3.2 Teclado HID 16

2.3.3 Servidor Local WEB 17

2.3.4 Mejoras de calidad de vida para el usuario 17

3 Pantalla: EVE Block **19**

3.1 Archivos desde los que se parte, capa base *19*

3.1.1 Capa de pantalla 19

3.1.2 Capa de micro 19

3.1.3 Ficheros extra útiles 19

3.1.4 EVE_TM4C1294XL.c 20

3.2 Bases teóricas de diseño: Estructura de la sintaxis de la creación de pantallas *22*

3.3 Estructura custom principal: BOTON *25*

3.3.1 Campo físico 26

3.3.2 Campo BotAcción 26

3.3.3 Estrucutra de BOTON conjunta 26

3.4 Creando funciones básicas para el funcionamiento *27*
3.4.1 Primeras modificaciones en EVE_API.c 27
3.5 Funciones para botones *28*
3.5.1 Funcion base EVE_CMD_BUTTON 29
3.5.2 Botón con animación realista, EVE_BOTON_P 29
3.5.3 Evolucion a EVE_BOTON_P_2 30
3.5.4 Almacenamiento en tabla: CREAR_BOTON 30
3.5.5 Modo de edición libre: EVE_EDIT_MODE_L 31
3.5.6 Modo de edición automática: EVE_EDIT_MODE_MATRIX 34
3.5.7 Botones visibles e invisibles, EVE_MOSTRAR_BOTONES_GLOBAL 37
3.5.8 Borremos botones, EVE_ERASE() 38
3.5.9 Encapsulizando procesos, INIT_ESTRUCT_PRINC 38
3.5.10 El sistema nervioso de los botones, EVE_HANDLER_ACCION 39
3.6 Consideraciones finales *42*
3.7 Avances en el main.c *43*
3.8 Variables globales únicas *43*

4 HID: Teclado USB **44**
4.1 Human Interface Device *44*
4.1.1 Integración de HID con el launchPad 44
4.1.2 Descriptores de características 44
4.1.3 DeviceDescriptor 46
4.1.4 Manejador del Teclado 46
4.2 USB en el EK-TM4C1294XL *48*
4.2.1 Atajos de configuración, PinoutSet 48
4.2.2 Modos de USB, USBStackModeSet 48
4.2.3 Funcion final, Inicia_Teclado 49
4.3 Teclado virtual, interacción del teclado mediante la pantalla, EVE_API.c II *50*
4.3.1 Códigos HID de teclas físicas, KeypUsageCodes 50
4.3.2 Comandos de transmisión, EVE_ACTUADOR 51
4.3.3 Comando de envio, KeyStateChange 51

5 Personalización: Servidor Local Web **53**
5.1 Inicialización del modulo *53*
5.1.1 Función Principal, Init_Ethernet_Server 53
5.1.2 Extras de la función ejemplo 55
5.2 Creación y comunicación del servidor *55*
5.2.1 Bases de creación web 55
5.2.2 JavaScript en el Proyecto 56
5.3 Comunicación de parte de la placa *60*
5.3.1 Detectando mensajes, io_fs.c 61
5.3.2 Acciones asigandas a los mensajes 62
5.4 Implicaciones de personalización en la pantalla, EVE_API.c III *66*
5.4.1 Variables de personalización 66
5.4.2 Actualización de botón, WEB_SUBMITTER 66
5.4.3 Actualización de colores, modificaciones en bloques básicos 67
5.4.4 Usando los Códigos Custom, EVE_ACTUADOR II 68
5.4.5 Mostrando IP a conectar 69

6 Memoria FLASH y revisión final **70**
6.1 La memoria FLASH *70*
6.2 Usando la FLASH en el sistema *70*
6.2.1 Variables a almacenar: estructura principal, colores y calibración 70
6.2.2 Estrategia de guardado 71
6.2.3 Estrategia de recuperación 73
6.2.4 Estrategia de actualización 74

7 Guía de Usuario 75
7.1 Primer Encendido 75
7.2 Pantalla Principal 75
7.3 Modos de Edición 76
7.3.1 Edición Automática 76
7.3.2 Edición Libre 78
7.4 Botones de funcionalidad 79
7.4.1 Botón de calibración 79
7.4.2 Botón de IP 79
7.4.3 Botón de borrado 80
7.5 Personalización en la Web 80
7.5.1 Apertura de la Web 80
7.5.2 Pantalla Principal de la Web 81
7.5.3 Editor de colores 82
7.5.4 Editor de botones 83
7.6 Diferencias principales al ejecutar con una memoria FLASH vacía 84

Bibliografía 86

ANEXO A 88
Raiz principal 88
Main.c 88
Driverlib2.h 90
Evethernet.h 91
Eveheader.h 91
Usbkeyboardheader.h 92
RecursosEve 92
Eve_api.c 92
Eve_calibrate.c 126
EVE_MCU_TM4C1294XL.c 127
Eve_Hal.c 133
Eve_fonts.c 139
Eve_calibrateplus.h 144
Eve_config.h 145
EVE.h 146
FT8xx.h 152
HAL.h 164
MCU.h 168
usbkeyboard 171
usb_keyb_structs.c 171
usb_keyb_structs.h 178
ethernet 179
io_fs.c 179
io.c 188
io.h 196
fs 197
index.htm 197
info.htm 198
instrucciones.htm 199
principal.htm 200
editorPag.htm 202
javascript.js 203
javascript_load.js 210
style.ccs 210

ÍNDICE DE TABLAS

Table 1. Tabla de conexionado de los elementos 14

ÍNDICE DE FIGURAS

Figure 1.Integración del Touchpad en MacBook 3
Figure 2. Integración del ScreenPad en el ASUS Zenbook 4
Figure 3. Evolución del ScreenPad de ASUS 5
Figure 4. Nueva integración del MousePad en ASUS 5
Figure 5. StreamDeck de ELGATO 6
Figure 6. Numpad externo estándar 7
Figure 7. Ejemplo de la diversificación del NumPad1 8
Figure 8. Ejemplo de la diversificación del NumPad2 8
Figure 9. Ejemplo de la diversificación del Numpad 3 9
Figure 10. Ejemplo de la diversificación del Numpad 4 9
Figure 11. Mecanismo de funcionamiento del switch electro-capacitivo 10
Figure 12.Parte posterior de la ME813A-WH50C 13
Figure 13. Parte superior de la ME813A-WH50C 13
Figure 14. EK-TM4C1294XL 14
Figure 15.Conexionado final de los elementos 15
Figure 16.Esquema de sintaxis de comandos de pantalla 23
Figure 17.Múltiples pantallas a mostrar, esquematización 24
Figure 18. Esquema de contenedor de estructura BOTON 27
Figure 19. Registro de coordenadas de la pantalla táctil 27
Figure 20. Esquematización de pantalla 28
Figure 21.Pantalla principal 75
Figure 22.Pantalla de Edición Automática 76
Figure 23. Pantalla de Edición Automática con uso 77
Figure 24. Matriz generada 77
Figure 25. Uso de la edición libre 1 78
Figure 26. Uso de la edición libre 2 78
Figure 27. botón generado por la edición libre 79
Figure 28. pantalla de calibración 79
Figure 29. Zona donde se muestra la dirección IP 80
Figure 30. Dirección IP 80
Figure 31. Navegador estándar con dirección IP 80
Figure 32. Pagina inicial de la WEB 81
Figure 33.Página inicial de la Web segmentada 81
Figure 34.Pagina de edición de pantalla 82

Figure 35.Edicion de pantalla tras uso 83
Figure 36.Pagina de edición de botones 83
Figure 37.Edición de botones tras uso 84

Notación

TI	Texas Instruments
micro	Microcontrolador
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
I2C.	Inter-Integrated Circuit
FLASH	Evolución de Electrically Erasable Programmable Read-Only Memory
HID	Human Interface Device
USB	Universal Serial Bus

1 ESTUDIO DEL PROBLEMA, ACTUALIDAD Y SOLUCIONES EXISTENTES

Los teclados, junto con internet, los ordenadores, los ratones, las pantallas y demás piezas tecnológicas, nos llevan acompañando en nuestra vida ya más de 70 años desde que las computadoras personales se empezasen a desarrollar, allá por los años 40. Todos los sistemas han pasado por diferentes etapas, dando lugar a dispositivos clásicos que viven en nuestros recuerdos como podrían ser el ratón de bola táctil, o las pantallas anchas o "televisores culones" que arreglábamos en casa a golpe, nunca mejor dicho, de porrazo.

Ahondando en el tema central de este proyecto, los teclados, estos han pasado por diferentes etapas y cambios que han afectado tanto a su tipo de conexión, pasando por el formato y cantidad de las teclas que contenían, y terminando en la tecnología de detección de la pulsación. Hoy en día, podemos ver como este dispositivo tiene un gran abanico de opciones en precio, forma y tecnología pasando de poder conseguir un teclado funcional por menos de 5€ en un bazar, a piezas únicas en el mundo cuyo precio se codea con joyas tradicionales y que son hechas de forma totalmente personalizada con la tecnología más puntera actualmente disponible.

Actualmente, la forma más común de dar información a las máquinas con las que convivimos son las pantallas táctiles variante espiritual con la misma finalidad que estos dispositivos con teclas. Véase los dispositivos móviles que usamos a diario, en los que utilizamos un teclado virtual para escribir a nuestros seres queridos por la aplicación de mensajería correspondiente, o los nuevos puestos táctiles que se están masificando en tiendas para hacer las compras u obtener información, donde pulsamos diferentes teclas asignadas a prendas para poder centrarnos en ellas y sus características.

La intencionalidad real del proyecto y el planteamiento de los puntos clave a cumplir, nacerán de crear una adaptación de un dispositivo mixto para su integración en el mundo de la producción audiovisual. En este campo son muy típicos los teclados matriciales a los que asignan, depende del campo del que hablemos, diferentes pistas de audio, acciones de escena, cámaras, etc. Esto ha dado pie a que grandes empresas del campo saquen diferentes versiones de teclados físicos que usar en estudios y retransmisiones para facilitar la vida a los realizadores y equipo técnico. Estos teclados físicos han inspirado el proyecto, que ha tenido como objetivo sacar un teclado que pudiese tener técnicamente mayor número de teclas y funcionalidades que un teclado estándar comercial por, presumiblemente, un menor precio.

Tras esta breve introducción, se pasará a la introducción del problema y sus puntos clave, continuando con una explicación de las soluciones que se han ido adaptando en el mundo real para cumplir sendos objetivos.

1.1 Planteamiento del problema

Al analizar y simplificar las capas de funcionamiento de lo que se quiere conseguir, se puede ver como el proyecto se puede dividir en una serie de objetivos a cumplir de forma clara. Estos son:

1. Proporcionar al usuario un entorno intuitivo con el que trabaja.
2. El usuario interactuará con el dispositivo a través de sus manos.
3. Permitir al usuario modificar en el dispositivo para adaptar los inputs a sus gustos y necesidades, tanto en forma como en número.
4. Proporcionar opciones para la configuración rápida del dispositivo.
5. Dar al usuario la capacidad de la segmentación del espacio de trabajo de cara a poder disponer de diversas configuraciones en el dispositivo que se adapten a diferentes situaciones.
6. Permitir personalizar la apariencia del dispositivo.
7. Permitir personalizar el dispositivo en cuanto a los comandos que se deseen usar.

8. Diseñar una forma no intrusiva para que el usuario tenga acceso a opciones de personalización.
9. Uso del dispositivo en cualquier PC, Windows o iOS.
10. Que sea plug-and-play, sin necesidad de preconfiguración.
11. Al ser externo, que tenga un tamaño lo más pequeño posible, siendo el óptimo el de un a matriz numpad 4*5.

Tras establecer estos objetivos, se establecen una serie de objetivos secundarios, opcionales para realización y dependientes del tiempo que se disponga tras acabar los principales. En el caso de que estos no puedan cumplirse, estos puntos serán tomados como posibles mejoras en el futuro:

1. Mantener la configuración entre encendidos
2. Calibración única, con posibilidad de repetición no obligatoria
3. Uso de audio o video para la personalización
4. Uso de audio o vibración para el pulsado de teclas
5. Inclusión de interactividad con objetos diferentes a botones, como ruedas de fast-track
6. Inclusión de imágenes en la pantalla a través de la aplicación de customización
7. Conexión inalámbrica

Establecido todo esto, se pasará a describir los diferentes productos que se han ido desarrollando en el mercado en cuanto a la solución de esta serie de objetivos, de cara a sentar unas bases reales y actuales para el proyecto, así como dar un contexto valioso para su valoración.

1.2 Actualidad de Teclados Táctiles Reprogramables

En la actualidad, la oferta y demanda de los teclados a nivel general se ha disparado. La creación del concepto de teclado mecánico como prácticamente joyas únicas ha hecho que muchas grandes empresas giren la cabeza hacia el campo de los periféricos de PC, debido, también en parte, a la falta de competidores fuertes que había hasta hace escasos años en un campo que era exclusivamente llevado por unos cuantos particulares con conocimientos avanzados de electrónica.

Esta inclusión de las empresas ha producido a la vez una diversificación y una estandarización del campo. Esto se puede explicar con la comparación al campo de los teléfonos móviles: todas las marcas ofrecen más o menos lo mismo, y cuando una de las empresas hace un cambio significativo de diseño, el resto de empresas se hacen eco a su manera para incluir esos avances dentro de sus modelos, tras esto se vuelve a un estado de estandarización hasta el siguiente avance importante.

En este campo es parecido, existen modelos muy parecidos entre teclados, ratones y dispositivos llegando a tener modelos con igual diseño y pegatina de marca diferente. Cuando una empresa se desmarca de la estandarización, al cabo de unos meses el resto de marcas cuentan con una alternativa al diseño novedoso. Así es como se ha ido avanzando en los modelos y uso de tecnología dentro de los periféricos de PC.

Dicho esto, también existen excepciones a esta norma como puede ser Apple. Esta empresa se ha caracterizado siempre por la opacidad de sus sistemas, llegando a desarrollar herramientas y piezas no estandarizadas para controlar que no pudiesen ser reproducidas por la competencia o particulares, y así controlar la producción mayoritaria. La principal arma para vender de la empresa es la exclusividad y el marketing.

Este tipo de empresas tiende a desmarcarse de los modelos estándares de periféricos hechos por el grueso de la industria, presentando dispositivos que rompen con la norma establecida. Esto hace que, a la hora de sacar nuevos dispositivos, se presenten ideas innovadoras a nivel de hardware o software vinculado a su línea de dispositivos, y esto puede presentar ideas interesantes para observar, en proyectos como este, el trato que se le ha dado a la integración de sistemas que cumplan los requisitos citados en el punto anterior.

Antes de pasar a citar sistemas específicos, se tiene que tener en cuenta que la combinación de características del proyecto hace complicado el estudio de productos con las mismas características al no existir tantos modelos finales con los módulos que se usaran, es decir, pantalla táctil como teclado únicamente. Se abordarán los proyectos indicando por qué se ha elegido como modelo a estudiar, que partes son similares y, presumiblemente, la razón detrás de las diferencias.

Con todo esto dicho, se puede empezar con el análisis de algunos productos.

1.2.1 MacBook TouchBar

Figure 1.Integración del Touchpad en MacBook

- El TouchBar de Apple fue introducido en el MacBookPro 2016 y ha estado presente en esta línea de portátiles hasta la fecha.
- Consiste en una pantalla táctil multi-touch de retina, situada done las teclas Fi funcionales deberían estar colocadas en el teclado del portátil, sustituyéndolas y por lo tanto siendo, más que una pantalla táctil en sí, una tira o barra táctil.
- Esta barra, reacciona a las aplicaciones del escritorio, modificando y pudiendo crear perfiles dentro del dispositivo en función de la aplicación que se esté usando.
- Al ser tecnología desarrollada por la propia empresa, presenta una integración completa en las aplicaciones desarrolladas también por esta, permitiendo un amplio rango de funcionalidades específicas y características especiales, a parte de la posibilidad de la creación y comandos personalizados.
- Permite el movimiento en todo momento entre el perfil actual y el perfil de teclas funcionales y tecla ESC, para evitar la limitación a la que puede dar lugar la eliminación de estas teclas.
- Este dispositivo puede ser, a nivel funcional, uno de los más parecidos dentro del mercado a lo que se quiere realizar. Tiene las principales características de reprogramabilidad e integración, solo teniendo la inconveniencia de que está integrado por completo y no es un periférico en sí.
- Se especula que el TouchPad desaparecerá en las siguientes versiones del MacBook debido a los costes de su integración en la gestión de espacio, que obligó a usar un sistema de carga menos avanzado que sus anteriores versiones, además de limitar mucho la capacidad de expansión de los puertos que contiene este dispositivo.

1.2.2 ScreenPad ASUS ZenBook Pro 14

Figure 2. Integración del ScreenPad en el ASUS Zenbook

- Asus empezó a introducir esta segunda pantalla en su modelo zenBook a partir de 2018, y a seguido desarrollando diversas ideas relacionadas con pantallas secundarias táctiles dentro de la línea de productos.
- Esta pantalla es una LCD de 5,5 pulgadas, IPS Full HD con MultiTouch
- La pantalla está colocada donde tradicionalmente iría el touchPad. Esta pantalla cambiará entre el uso de touchPad y pantalla, no siendo un sustituto totla de la función tradicional.
- La pantalla no es reprogramable, pero las aplicaciones integradas la usan para facilitar comandos integrados como las opciones de formato de texto en Word, o el control de la música en Spotify.
- Mientras estas aplicaciones no son usadas, este segmento ahce las funciones de segunda pantalla donde podemos pasar ventanas.
- Com función extra, se puede usar para incializar aplicaciones de forma rápida al contener un menú de aplicaciones.
- En formato, es el dispositivo que ams se acerca a la idea final que será una pantalla rectangular táctil. Pese a esto, las funcionalidades son diametralmente diferente, no permitiendo este dispositivo la inclusión de macros o capa de personalización, adema´s de estar integrado en el sistema.
- Asus ha hehco un all-in con la integración de pantallas auxiliares en sus dispositivos portatiles y es una de la empresas que más modelos tienen con este tipod e característica. Han seguido desarrollando esta tecnología hasta llegar a los modelos actuales, ejemplo de ello, el siguiente punto.

1.2.3 ASUS ZenBook Pro Duo

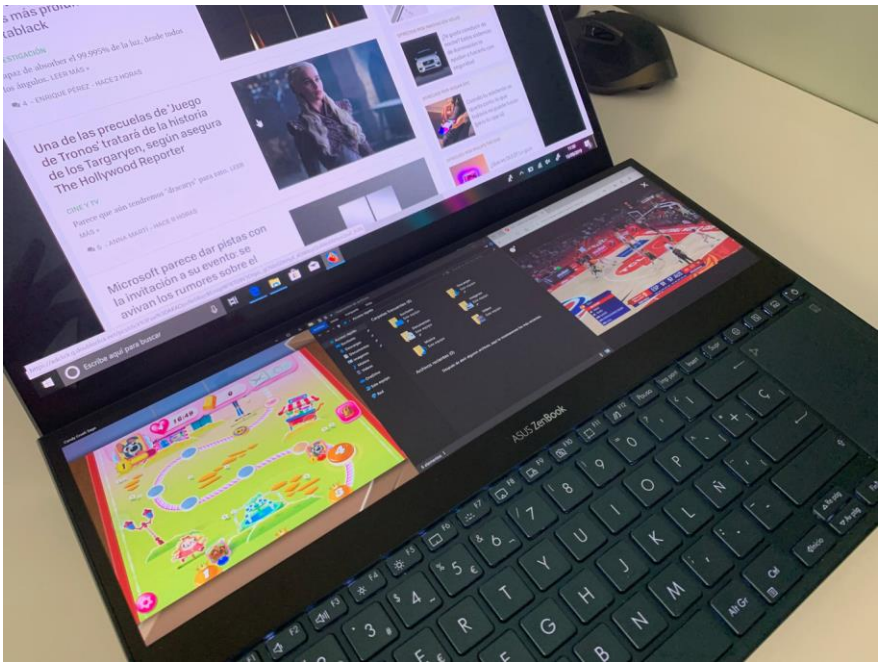

Figure 3. Evolución del ScreenPad de ASUS

Figure 4. Nueva integración del MousePad en ASUS

- La versión definitiva de ASUS de la integración de pantallas capacitivas auxiliares en portátiles.
- Este modelo de portátil fue sacado al mercado en 2019 y la pantalla táctil secundaria LCD es de 14 pulgadas, IPS 4K, y también introdujo una pantalla capacitiva en el touchpad para añadirle funciones extra.
- La segunda pantalla se sitúa en la parte de arriba del teclado, desplazando este a una zona inferior, mientras que el touchpad se encaja, por falta de espacio, donde normalmente estaría el numpad.
- La pantalla LCD es el desarrollo directo del modelo anterior, evolucionado en una pantalla mucho más grande para poder dar un soporte mayor al uso de segunda pantalla.

- No pierde las capacidades de integración con aplicaciones para comandos y acciones que tenía la versión anterior, pero al volverse más grande se presupone que su uso principal será en ser una extensión de la pantalla, función alejada del planteamiento del proyecto.
- Una inclusión muy interesante es la capacidad que se le da al usuario de convertir el touchpad en un numpad táctil, pudiendo asignar macros a esos números y activar y desactivar la funcionalidad del numpad a voluntad.
- Este producto es el resultado de la evolución de una idea muy parecida a sobre la que se fundamente el proyecto, llevada por las características de integración y expansión hacia otro objetivo. Aun siendo así, la inclusión el numpad dentro del ratón por falta de espacio se concibe como una solución que podría incluirse dentro de algunos puntos clave del trabajo, aun siendo alejada del producto final que se desarrollará.

1.2.4 Stream Deck de ElGato

Figure 5. StreamDeck de ELGATO

- ElGato, empresa perteneciente a Corsair, se ha posicionado como una de las principales compañías en cuanto al desarrollo de dispositivos y periféricos para medios audiovisuales. Se creó en 1992 y desde entonces han estado introduciendo tecnologías punteras al medio.
- Este teclado matricial se comercializa de forma masiva desde 2018, y se ha ido actualizando con diferentes versiones que ha modificado el número de teclas totales o la tecnología de sus botones.

- El cometido de este teclado es el de un teclado de Macros, con el añadido de usar las pantallas LCD incluidas en los botones individuales para personalizar estos. Además, cuenta con integraciones para diversas aplicaciones famosas en el mundo audiovisual actual gracias a sus acuerdos comerciales y el uso extendido que se le está dando en el mundo de las retransmisiones por internet.
- Se conecta mediante un USB, y tiene una aplicación dedicada para el diseño de las teclas que se descarga desde su página web.
- Es el sistema en el que se ha inspirado principalmente l proyecto, cumpliendo por lo tanto muchos objetivos principales.
- Las mejoras propuestas para realizar en el sistema en comparación con este sistema tienen principalmente una relación con su formato físico, que limita su formato de forma de forma tajante, además de dar funciones limitadas relacionado con su número de botones limitado.
- Fuera de ser un inconveniente para la empresa, esta vende mejoras de este sistema a los que le van añadiendo botones, siendo exponencialmente más caro cada vez que se le añade una fila.

1.2.5 Numpad reconfigurables, teclados de macros y switches electro-capacitivos

Sin tener puntos específicos debido a que no existe como modelo de una compañía, todo este conjunto de tecnología existente podría ser la combinación alternativa de factores más cercana al producto final que se tendrá.

Los teclados numpad de macros reconfigurables llevan en el mercado desde que el teclado perdió el formato completo en portátiles y teclados de ordenador, de forma que, pese a no disponer en el teclado principal de esta zona, se pudiese contar con esta funcionalidad cuando se necesitase.

Figure 6. Numpad externo estándar

A raíz de estos numpad, nacieron los teclados de macros, que usaban estas teclas que ya no se usaban de forma tan asidua para la asignación de combinaciones específicas de teclas. Además, el sencillo formato de las teclas y PCB, y la extensión de su popularidad hizo que mucha gente empezase a aprender como configurar estos teclados para poder conectarlos a su PC. Esta propagación de la información y democratización de la creación hizo que los formatos se ampliasen, junto con la personalización.

He aquí algunos ejemplos de este movimiento:

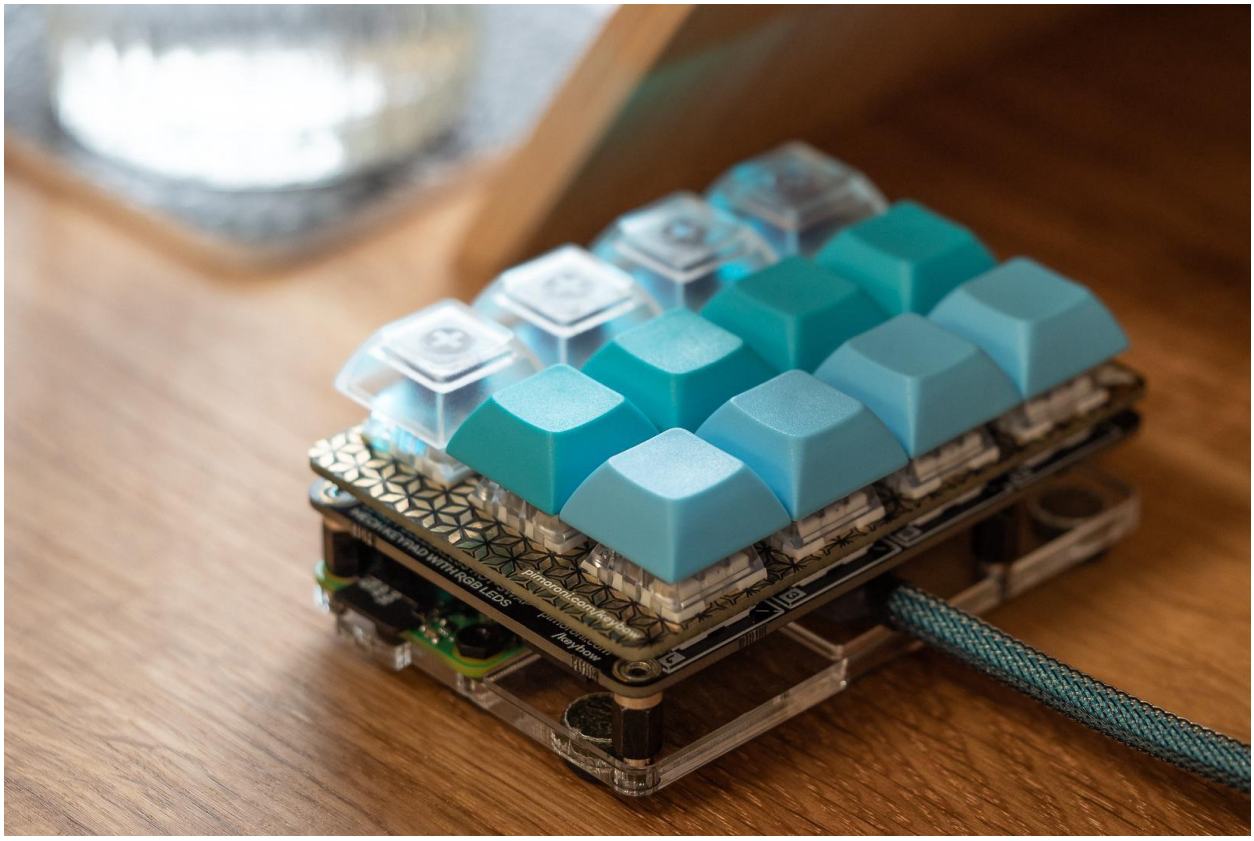

Figure 7. Ejemplo de la diversificación del NumPad1

Figure 8. Ejemplo de la diversificación del NumPad2

Figure 9. Ejemplo de la diversificación del Numpad 3

Figure 10. Ejemplo de la diversificación del Numpad 4

Como se observa, la apariencia de las teclas ha cambiado físicamente, creando teclas intercambiables para encajar en las diferentes configuraciones.

Esta parte es paralela a los objetivos del proyecto planteado, salvo por un detalle: se pretende dar al usuario la posibilidad de la personalización de las teclas de forma cómo da y rápida. Estas opciones implicaban la necesidad de tener las teclas previamente creadas, y para hacer el cambio de configuración se necesitaba que el usuario conociese de primera mano el código que este necesitaba para programar las macros correspondientes.

Por la parte del uso de tecnología capacitiva, tenemos los switches capacitivos. El nombre de switches, se le da a la tecnología que contienen las diferentes teclas de un teclado para detectar la pulsación. A lo largo de los años se han diversificado enormemente, siendo las más famosas y utilizadas actualmente los teclados de malla, switches de botón, y los mecánicos, switches que tienen un mecanismo con muelle y disparador para detectar el pulso.

Entre toda esta tecnología, una corriente de diseño se fijó en el fenómeno capacitivo de las PCB, permitiendo quedarse en un lugar intermedio entre los mecánicos y de malla. Estos funcionan con un muelle que se presiona ligeramente y hace que una placa capacitiva se active al acercarse. Esta tecnología es parecida a la que se planteará en un primer momento en el proyecto, salvo que, para promover la interactividad del usuario con la

pantalla, se quitará la parte mecánica y usaremos nuestros dedos como activador, lo que es una pantalla táctil.

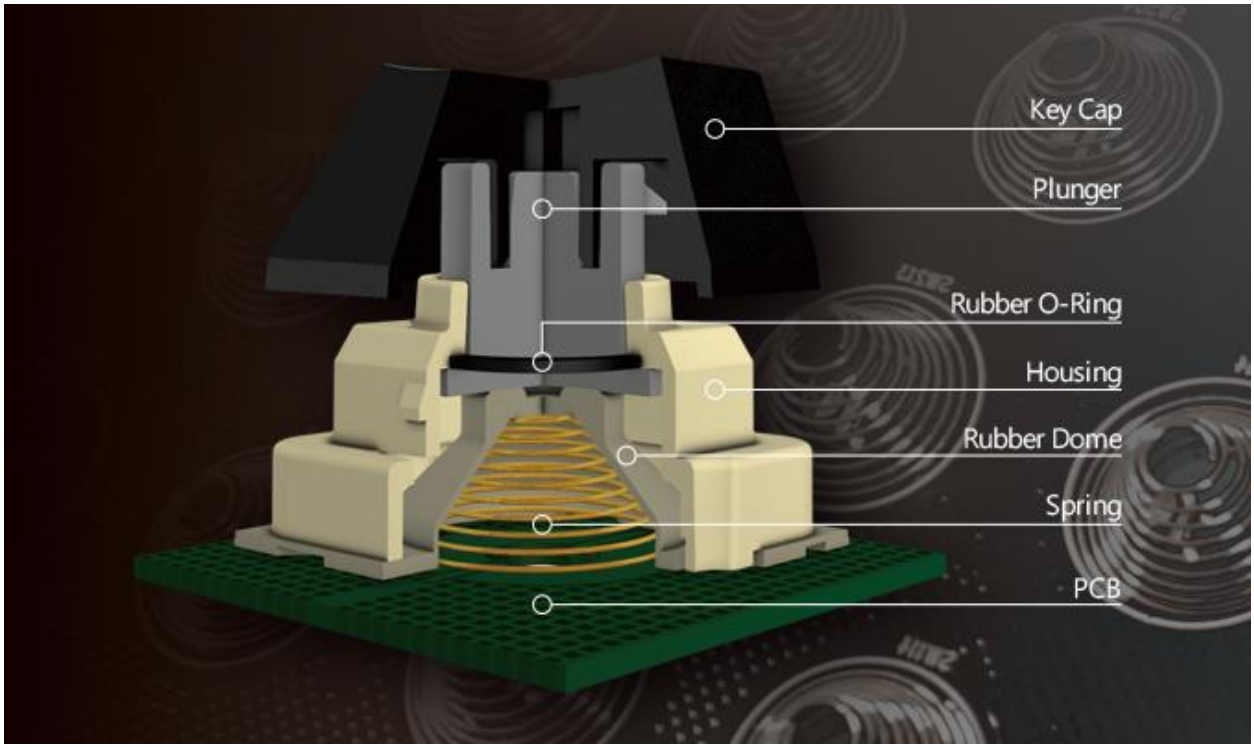

Figure 11. Mecanismo de funcionamiento del switch electro-capacitivo

Tras mencionar todas estas tecnologías desarrolladas actualmente, pasemos a la resolución del problema planteado.

2 RESOLUCIÓN PLANTEADA DEL PROYECTO, COMPONENTES USADOS Y ESTRUCTURA DEL PROYECTO

Trás la descripción de las soluciones existentes descritas en el anterior apartado, se especificará a partir de aquí que estrategia se ha decidido seguir para cumplir todos las exigencias de funcionamiento planteadas.

Estas soluciones se han visto limitadas a los dispositivos que el tutor y alumno tenían, formando estos un amplio abanico de opciones, pero dejando de lado opciones que podrían estar mejor preparadas para enfrentarse al proyecto por motivos económicos.

Pese a esto, se cree firmemente que los elementos planteados para la resolución son totalmente válidos y cubren, punto por punto, todas las exigencias posibles que el sistema pudiese plantear, pudiendo llegar a competir con proyectos de mayor inversión y disponibles en el mercado.

2.1 Resolución del problema planteado

Para dejar constancia de las decisiones que llevaron hasta la decisión final, se comentarán algunas opciones de planteamiento que se adoptaron en un inicio, y como evolucionó el planteamiento del proyecto hasta llegar al actual.

2.1.1 Pasos previos, planteamiento de diseños erróneos y descartes

Este proyecto nació con la idea principal, después descartada, del diseño físico de nuestro propio teclado físico capacitivo. Este se imprimiría en una pcb y se conectaría a un microcontrolador de nuestra elección para realizar la lectura de los parches capacitivos.

Para desarrollar algo más la idea, se diseñó una serie de patrones para los parches capacitivos de manera que se pudiese leer el porcentaje del contacto con este por el dedo o dispositivo que interactuase con el dispositivo, de cara a poder mapear el contacto en los máximos puntos de la pantalla. Esto como se puede ver, sería plantear el diseño de una pantalla táctil al uso.

Esta idea plantea una serie de problemas a solucionar:

- La impresión de una PCB en estos tiempos puede resultar cara para hacer exclusivamente un prototipo
- La lectura, de la manera que se quería hacer, estaba solo al alcance de algunos launchPad específicos cuyos pines pudiesen leer entradas capacitivas
- Una vez terminado el proyecto, se tendría que solucionar la falta de una pantalla que diese feedback al usuario de cara a que pudiese ver que se estaba pulsando o si el dispositivo reaccionaba.

Pese a estos problemas, se buscaron componentes de cara a la realización de estos objetivos. Se plantearon diferentes placas de TexasInstruments que contenían estos pines capacitivos, además de la compra de algún sistema intermediario que pudiese convertir esas entradas capacitivas en entradas directas de teclado.

Al final, todo esto se descartó debido a que los microcontroladores propuestos de TI tenían los pines capacitivos, pero no existía ni existe soporte funcional ni documental por parte de la empresa para entender como sus dispositivos se inician y gestionan para esta característica. Además, otros micros fueron descartados por faltarles puertos necesarios para el desarrollo de todas las funciones requeridas, como podía ser el de soporte USB para crear el teclado HID.

De cara al teclado HID, se planteó el uso de Bluetooth desde un principio. Pero por recomendación del tutor se

descartó la idea debido a la reciente para de soporte del BT 2.0 por parte de iOS y Windows, para pasar a ser BT 5.0, menos extendió a la hora de su programación que el 2.0, y por lo tanto mucho más obtuso a la hora de realizar las comunicaciones. En el caso de usarlo, el proyecto se encontraría en un punto donde esta característica pudiese funcionar dependiendo de los adaptadores, protocolos y versión del sistema que tuviese un ordenador, no teniendo una fiabilidad consistente.

2.1.2 Idea final

Tras repasar tutor y alumno las posibilidades disponibles, descartando la creación de una pantalla táctil propia y el uso del bluetooth, se decide darle una vuelta al planteamiento del proyecto. El proyecto se centrará en la capacidad del usuario para personalizar un teclado mediante diferentes métodos:

- Para paliar la imposibilidad de usar el bluetooth, se decide buscar una placa con posibilidad de conexión USB con el objetivo de poder configurarlo como dispositivo USB HID.
- Para la personalización, se piensa en desarrollar una aplicación que permita al usuario navegar por unas opciones y hacer y deshacer a su antojo. Pero esto no es posible, debido a que el puerto USB por donde presumiblemente se comunicarían estaría ya conectado. Esto plantea una decisión: se crea un driver personalizado para el uso simultaneo de un dispositivo HID y un puerto de transmisión, o por el contrario se usa otro puerto. De aquí nace el uso de un servidor web que se comunique con la placa.
- Con la pantalla, se llega a la conclusión de que lo mejor es trabajar sobre una pantalla táctil capacitiva que de la opción de mostrar por pantalla las acciones que se van realizando de cara a solucionar el problema de feedback existente. Además, aceleraría enormemente el diseño y creación si se contase con una desde el principio, no teniendo que depender de sistemas de envío en verano.

Por lo tanto, se tienen los requerimientos de hardware y software iniciales: un launchPad con capacidad de conexión USB HID, conexión a internet y un puerto extra del tipo que sea para conectar una pantalla que corresponda a este.

Esta serie de características definen en mayor o menor medida al EK-TM4C1294XL, que estaba ya comprado con antelación al haberlo usado previamente en la carrera. El uso de esta placa aceleraría el proceso de adaptación del proyecto al conocer ya los procesos y quehaceres propias de su programación.

Del lado de la pantalla, el tutor proporciona una pantalla ME813A-WH50C que, aunque en principio no existiese soporte para este tipo de microcontrolador, sí que lo tiene para una familia diferente de su empresa: los MSP430. Esto significará que habrá que portear el programa base que proporciona la empresa, y estudiando en profundidad el funcionamiento diseñado para que sea compatible con múltiples micros de diferentes compañías, diseñar las funciones específicas para asegurar su conexión y funcionamiento.

Planteado todo esto tenemos los ingredientes iniciales del proyecto determinados.

2.2 Hardware del Proyecto

El proyecto consta de dos componentes: El Connected LaunchPad EK-TM4C1294XL de Texas Instruments y la ME813A-WH50C (FT813 Development Module with 5.0 Inch WVGA TFT LCD) de FTDI Chip. El segundo dependerá directamente del primero, que se encargará de que todas las funcionalidades de nuestro teclado capacitivo reprogramable interactúen entre sí y le indicará al segundo que hacer en cada determinado momento.

2.2.1 ME813A-WH50C

- La pantalla a utilizar tiene una resolución de 800x480px, variable mediante programación del software, pero ésta será la resolución escogida del proyecto, debido a que aprovecha el máximo espacio posible de la pantalla.
- La pantalla cuenta con la posibilidad de controlar los leds instalados en la parte posterior, así como incluir audio, imágenes e incluso video de forma relativamente cómoda gracias a que cuenta con un

altavoz mono y que, incluso, tiene el soporte para la conexión de un altavoz externo. Pese a que existen estas posibilidades, este trabajo NO usará estas funciones por decisión de diseño.

- Entre otras funcionalidades extra de la pantalla tenemos con que podremos leer hasta 5 toques de pantalla simultáneos, existiendo un registro propio para almacenar las coordenadas de cada uno de estos contactos (REG_CTOUCH_TAGi_XY, siendo i un numero entre el 0 y el 4). En este proyecto solo usamos el primer registro, teniendo la capacidad real de 1 toque simultaneo.
- Por último, la alimentación de la pantalla es de 5V, puerto de alimentación que puede proporcionarle el launchPad sin problemas.

Figure 12.Parte posterior de la ME813A-WH50C

Figure 13. Parte superior de la ME813A-WH50C

2.2.2 EK-TM4C1294XL

Este launchPad, que Texas Instruments denomina como low-cost, tiene las características y potencia perfecta para nuestro proyecto, siendo algunas de estas:

- Procesador basado en ARM-Cortex-M4F, siendo el TM4C1294NCPDTI
- Puerto ETHERNET y puerto micro USB para conexiones a PC de tipo 2.0
- 2 botones y 4 leds del usuario
- 2 conexiones de 40 pines pensadas para el uso de boosterpacks, expansiones de la placa con usos como conexión wifi o recolección de datos físicos.
- Dispositivo para debug incorporado con controlador y puerto USB dedicado.
- Capacidad de hibernación con botón asociado.
- Capacidad de alimentación directamente por el puerto de conexión USB-PC
- Capacidad de alimentación de 5V en varios puntos.
- Múltiples puertos UART, SPI e I2C
- Pines analógicos, digitales y mixtos

Figure 14. EK-TM4C1294XL

2.2.3 Conexión de los elementos

Conectaremos la pantalla por puerto SPI al launchPad del microcontrolador, correspondiendo los pines de la siguiente forma:

Table 1. Tabla de conexionado de los elementos

PIN EK-TM4C1294XL	PIN ME813A-WH50C
PD0	MISO
PD1	MOSI
PD3	CLK
PN2	PD
PN3	CS

Por lo tanto, se usará el puerto 2 de SPI (SSI2) de la EK-TM4C1294XL para la configuración.

Tras esto, el resto de conexiones serán las de los cables USB y ETHERNET.

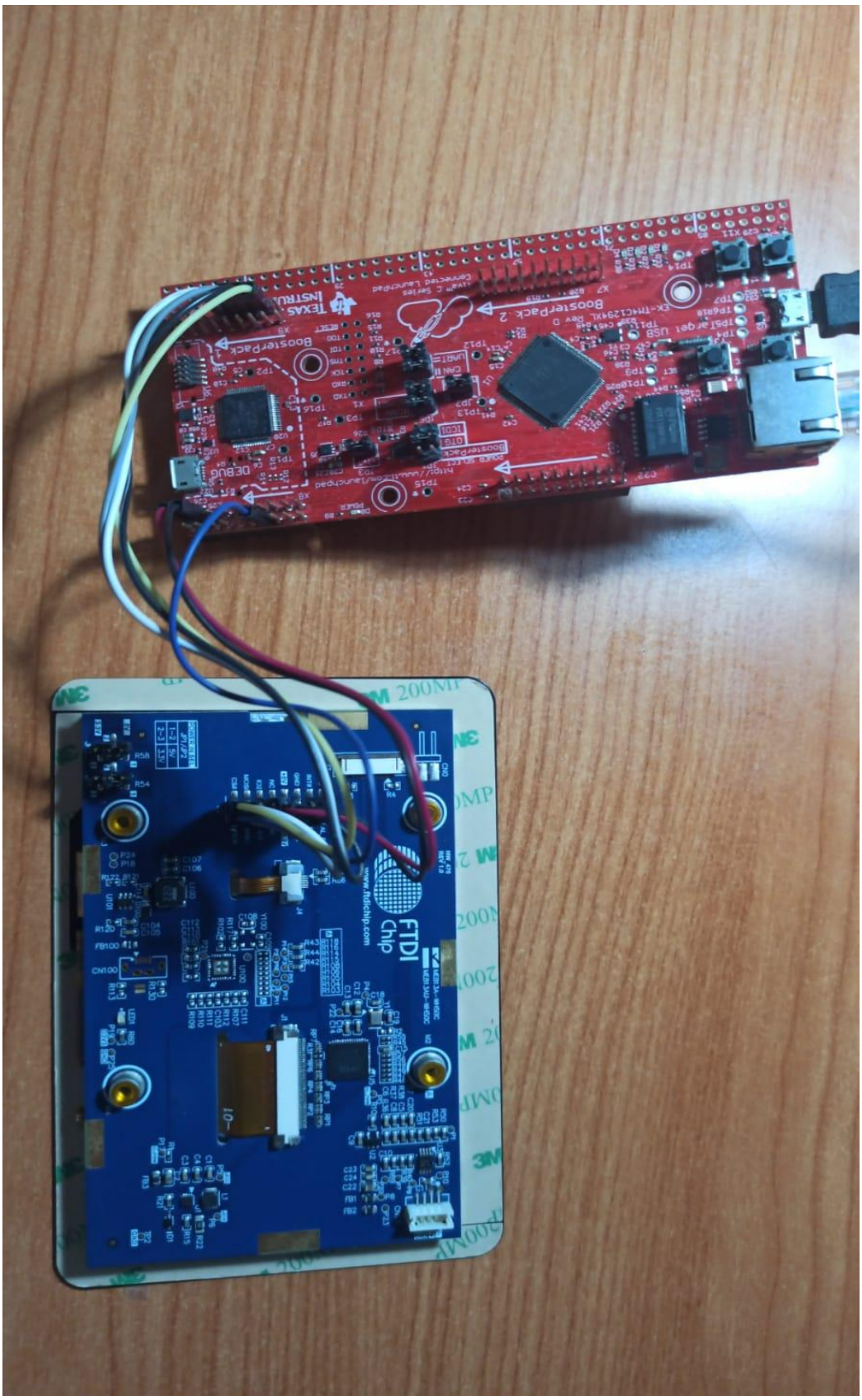

Figure 15.Conexionado final de los elementos

2.3 Software del Proyecto

En cuanto a la estructura interna del funcionamiento del proyecto, se plantea una estructura por bloques, siendo cada uno de los bloques una de las funcionalidades principales que se busca en el proyecto:

2.3.1 Pantalla

- El proyecto recae a nivel visual, casi por completo, en este componente. Es el componente principal al ser la principal fuente de feedback máquina-usuario y podremos determinar el buen funcionamiento del proyecto si este responde acorde a lo esperado en el diseño. Todos los bloques de software tienen implicaciones con este componente ya que interactuará a nivel de personalización-configuración con el servidor y a nivel funcional con el teclado HID, por lo que contará con canales de comunicación con todas las partes del proyecto
- La pantalla utiliza el controlador serie FT813 para manejar la funcionalidad de gráficos y audio, por lo que usaremos la guía de programación para el usuario FT81x, proporcionada por FTD Chip, para desarrollar y entender las funciones principales del dispositivo, así como desarrollar otras nuevas mediante el uso de los registros internos de este y descripciones proporcionadas por los fabricantes.
- Se parte de la base de un ejemplo proporcionado directamente por FTDI Chip, compatible con la placa MSP-EXP430G2ET, por lo que habrá que portear el trabajo hacia nuestro LaunchPad deseado.
- Junto con el ejemplo proporcionado, se da, por parte de la empresa que lo desarrolla, una serie de indicaciones que facilitarán la exportación del proyecto, así como diferentes indicaciones si queremos excavar en capas inferiores a nivel de registros.

2.3.2 Teclado HID

- Es el bloque del proyecto que interactúa directamente con el ordenador y envía los comandos programados al ordenador para que este los interprete y realice las acciones. Los comandos serán seleccionados mediante la pantalla por lo que este bloque tiene que ser capaz de comunicarse con el bloque de la pantalla para que conozca el mensaje a enviar.
- La implementación es sencilla en principio, debido a que TexasInstruments proporciona diversos ejemplos de implementación junto con la librería TIVAWARE, compatible con nuestro launchPad EK-TM4C1294XL.
- Estos ejemplos son acompañados de librerías que facilitan mucho la comprensión de este paso. Para personas no experimentadas en implementación de Dispositivos de Interfaz para Humanos o Human Interface Device (HID), este proceso exclusivamente, sin tener en cuenta los requerimientos necesarios para disponer del dispositivo en su modo de conexión por puerto USB, puede llegar a resultar muy tedioso ya que la programación clásica de estos dispositivos recae en el envío de estructuras de números planos que el PC correspondiente reinterpretará según los protocolos como un comando, una tecla determinada, un estado de tecla… Por ejemplo, el comando {0x01, 0x06} en el momento correcto de emisión, significa que, de la página de dispositivos estándar, queremos configurar el dispositivo entrante como teclado.
- Gracias a estas librerías y la guía de usuario que Texas Instruments proporciona sobre los dispositivos USB, podremos controlar sin muchos problemas: la temporización de los mensajes, el tipo de mensaje, el tipo de dispositivo a programar, los comandos de dispositivo, etc.
- El dispositivo se llamará “EVE Teclado ONE” y será configurable en iOS y en Windows.
- Es un teclado con configuración española, aunque esto en realidad no tiene más importancia, excepto por el uso de la ñ, a la que no doy espacio a usar por los problemas de compatibilidad de muchos programas al usar comandos con esta tecla dependiente de la región. En resumidas cuentas, es un teclado estándar configurado en español, aunque no haya diferencias reales con un teclado inglés, por ejemplo.
- Como justificación del uso de la “simulación” de un teclado podrían ser diversas:

1. A la hora de probar diferentes modos de Inputs como mandos, ratones con teclado configurables, ratones clásicos, teclados extendidos… He visto como los únicos dispositivos que detectaban como entrada de comandos eran los que estaban configurados como teclados.
2. Pese a que se podría hacer un dispositivo custom, digamos un mando-teclado, según dispositivos custom comerciales como el Mars Gaming MM4, ratón-teclado, el software inicializa 2 dispositivos a la vez y no 1 conjunto, por lo que determino que no es rentable realizar el esfuerzo en tiempo y recursos de intentar reunir varios dispositivos en uno y así simplifico el problema.
3. Además, el problema principal a nivel de funcionalidad que nos enfrentamos es: que los programas puedan reconocer los comandos que enviemos. Esto nos hace depender exclusivamente de lo que los programas estén codificados para entender, dejando muy poco lugar a la resolución creativa del problema, como hacer un mando de botones infinitos y cada botón ser una tecla de macro, ya que el programa no detectaría que este botón fuera pulsado.
4. La configuración de un teclado con lo que nos da TexasInstruments es prácticamente un problema resuelto. Nos da la configuración principal iniciada, y el trabajo es, prácticamente, de limpieza de funcionalidades que no vamos a necesitar.

2.3.3 Servidor Local WEB

- Al uso, será la aplicación de personalización del teclado. Se usa un servidor web aprovechando las capacidades del boosterpack de conectarse a internet mediante un cable ethernet y generar un servidor web interno basado en HTML, CCS y JavaScript, que será necesario conocer para el desarrollo de este bloque funcional. Como será la manera que tendremos de personalizar el teclado al gusto del usuario, los cambios tendrán que verse reflejados en el bloque de la pantalla de forma clara, por lo que es necesario que este bloque se comunique de forma directa con la pantalla.
- Podríamos decir que es el bloque que más se aleja del trabajo típico desarrollado en la carrera, ya que, académicamente, nunca nos hemos enfrentado al diseño web, así como tampoco se nos ha ni mencionado la funcionalidad de HTML, CCS y JavaScript.
- Pese a lo anterior, aprovechar la capacidad del LaunchPad de iniciar un servidor interno y conectarse de forma local al dispositivo nos soluciona un aspecto que suele ser básico en este tipo de proyectos: una aplicación dedicada para hacer mantenimiento, configuraciones y personalizar el dispositivo.
- Además, esta opción me parece mucho mejor que la instalación de molestas aplicaciones que se usan de forma esporádica las primeras veces que usamos el dispositivo. Ahorramos el espacio, la descarga, la desinstalación y damos comodidad al usuario dando la posibilidad de que, en el caso de que no quiera ni que el servidor interactúe con su PC, no sea necesario conectar el cable de configuración.
- La comunicación servidor-placa se hace mediante la publicación y lectura de una serie de mensajes en un registro interno del launchPad encargado del inicio y mantenimiento del servidor. La placa buscará una serie de mensajes específicos que le programemos, mientras el servidor publica mensajes dependiendo de las acciones que realice el usuario en él.
- Toda esta aplicación no sería posible sin el desarrollo de uno de los ejemplos y sus librerías proporcionados por Texas Instruments, debido a mi poco conocimiento previo sobre servidores web y el proceso de conexión que tienen. Además, he de denotar que mi conocimiento de HTML, CCS y JavaScript previo a este proyecto eran absolutamente nulos, y una de las grandes partes de este ha sido el aprendizaje de estos tres idiomas tan ligados al desarrollo web.

2.3.4 Mejoras de calidad de vida para el usuario

- Esta ha sido la parte final del proyecto, donde introduje algunas mejoras de calidad de vida, como el mantenimiento de la configuración de botones y pantalla entre encendidos, el mantenimiento de la calibración de la pantalla y la capacidad de recalibrar en cualquier momento para no tener que realizar el proyecto cada vez que se inicie el dispositivo, la capacidad de eliminar botones, y mejoras en el HUD de usuario basadas en mi propia experiencia a lo largo de su desarrollo.

- A nivel general, son mejoras que afectan exclusivamente a la pantalla por lo que serán incluidas en las librerías de esta, aunque se trate de forma apartada en este documento. Y, también en general, son mejoras relacionadas con otra de las grandes funcionalidades de nuestro EK-TM4C1294XL: la memoria FLASH.
- Depende del modelo, los bloques de memoria flash de cada launchPad son de diferentes tamaños. Y en este caso, se nos proporciona diversos de bloques de unos 16KB que usaremos para almacenar de forma separada los datos de configuración de botones, pantalla y calibración.
- Los guardamos de forma separada ya que el funcionamiento de la FLASH consiste en que un 1 lógico se puede convertir en un 0, pero un 0 lógico no puede convertirse en un 1. Por lo que para hacer cambios de forma consistente en FLASH es necesario borrar el contenido, que solo se hacer si borramos un bloque de memoria al completo. En resumen, lo separamos para que cuando necesitemos cambiar la configuración de los botones no tengamos que volver a escribir toda la configuración de la pantalla, y viceversa.
- Se explicará más adelante, pero solo se podrán recuperar datos en forma de cuartetos de bytes, por lo que será necesario tener muy claro cuantos bytes tiene cada variable y hacer cuentas para asegurarnos de que la información que mandamos y la que recuperamos es la misma.

3 PANTALLA: EVE BLOCK

La configuración y puesta a punto de la ME813A-WH50C pasa por diferentes fases a lo largo del proyecto. Desde su arranque inicial con el ejemplo dado para otro launchPad, pasando por la definición de la estructura de información que dominaría a nivel funcional el proyecto basado en las peticiones de la pantalla, terminando por la comunicación con los diferentes bloques.

A continuación, se irá desglosando el trabajo desarrollado en los diferentes niveles, acompañando del código correspondiente.

3.1 Archivos desde los que se parte, capa base

El ejemplo proporcionado por el fabricante está configurado para la inicialización del periférico en la MSP430 correspondiente, se usará la estructura de este ejemplo para diseñar la nuestra. (Bridgetek, 2019)

3.1.1 Capa de pantalla

Esta capa del proyecto corresponde a la gestión única y exclusiva de la pantalla, desde su capa más alta a la más baja:

- **EVE_API.c**: capa de código más cercana a la aplicación principal, y que por lo tanto contiene la mayor parte de funciones de alto nivel que se podrán usar dentro de la capa de aplicación. Será uno de los archivos que más modificaremos en el futuro de cara a desarrollar nuestras propias funciones de alto nivel para introducir funcionalidades nuevas en nuestra aplicación.
- **EVE_HAL.c**: capa de código intermedia. Si se estudiase, se vería como prototipos de funciones de alto nivel que usan registros y funciones intermedias a partes iguales. Se encarga de que la temporización de los comandos se la correcta ya que contiene los protocolos de uso completos, como el de activación o lectura de registros específicos. En principio, no se necesitará modificar nada de este nivel para obtener funcionalidades nuevas, y tocar cualquier función que contiene este archivo podría resultar en el malfuncionamiento total del trabajo.
- **FT8xx.h**: Es una hoja de datos con información del procesador que tiene la pantalla instalada para la gestión automática externa de la pantalla. Es algo especifico que dan los fabricantes para que se pueda trabajar de forma cómoda con el procesador integrado del dispositivo.
- **EVE.h**: fichero cabecera de **EVE_API.c**. Lo necesitaremos modificar para incluir nuestras funciones nuevas, entre otras cosas.
- **EVE_config.h**: Fichero con la información de la configuración específica que tendrá la pantalla. Todas las variables como la versión de FT8xx, el alto y ancho de la pantalla, la primera línea activa, etc.

3.1.2 Capa de micro

Archivos específicos para el funcionamiento del periférico con un microcontrolador específico, se necesitará diseñar y modificar el procedimiento para nuestro proyecto. Los archivos serán:

- **EVE_MCU_MSP430.c (proyecto ejemplo) a EVE_MCU_TM4C1294XL.c (proyecto final)**: Este archivo presenta los protocolos de inicio del micro específico, tratamiento del puerto SPI, inicialización del puerto determinado con la conexión de pines correcta, etc. Se tratará más adelante los pasos necesarios para la configuración del nuestro.
- **MCU.h**: Fichero cabecera del anterior.

3.1.3 Ficheros extra útiles

A parte de los archivos anteriores, el ejemplo proporcionado trae diferentes archivos auxiliares que se

guardarán para uso de diferentes funcionalidades, aparte de mantener el archivo main.c para comprobar las rutinas de inicio seguidas:

- **Eve_calibrate.c**: Demostración del proceso de calibración mediante el proceso de tocar tres puntos en la pantalla. Este proceso se reutilizará de forma completa para la calibración del producto final, salvo por un cambio que se indicará más adelante relacionado con cuando ha de incoarse el módulo de calibración.
- **Eve_example.c**: Contiene el código principal a ejecutar del ejemplo.
- **Eve_fonts.c**: Demostración del manejo de fuentes diferentes de las preprogramadas por la pantalla.
- **Eve_helper.c**: Pese a ser un archivo que podría considerarse auxiliar, de él sale la idea principal del manejo de la pulsación de botones que más tarde generaremos en el teclado final. Investigando en la documentación, la ME813A-WH50C contiene una serie de registros que asignan y leen una serie de etiquetas o TAGS, que podemos asignar a cada uno de los elementos que vamos dibujando en la pantalla. En el caso de que estemos tocando un elemento con un tag asignado, los registros de REG_CTOUCH_TAG, REG_CTOUCH_TAG_XY y derivados de la posibilidad del multipunto táctil, mostrarán el TAG asociado al elemento, por lo que si leeos esos registros podremos determinar que elemento piensa la pantalla que estamos tocando. Este archivo se usa para detectar el toque de los puntos de calibración en el archivo **Eve_calibrate.c**.
- **Eve_images.c**: Demostración del manejo de imágenes para poder incluir estas y mostrarlas por pantalla.
- **Eve_expample.h:** funciona como archivo cabecera de todo lo anterior y es lo que se incluye en el **main.c** para manejar los diferentes módulos del ejemplo.
- **main.c**: Enseña los procesos de inicio de la pantalla y posteriormente del uso de esta para mostrar diferentes funcionalidades en pantalla.

Habiendo estudiado la documentación, la estructura y los archivos necesarios, se puede proceder con el diseño de la capa específica para nuestro launchPad EK-TM4C1294XL.

3.1.4 EVE_TM4C1294XL.c

Para el archivo, se tendrán que conocer las diferentes características únicas del launchPad, así como sus procesos específicos para el tratamiento de puertos. Estos procesos se pueden resumir en:

1. Inicialización:

   Tanto la inicialización del reloj del launchPad como la del puerto SPI con sus pines asociados para el nuevo launchPad, acorde a sus especificaciones técnicas de puertos y pines.

2. Tratamiento del puerto SPI:

   El TM4 no trata al puerto como una interrupción programable, sino que establece una serie de registros de pines específicos para que actúen con las diferentes funcionalidades de un puerto de este tipo.

3. Lectura/Escritura de los búferes SPI correspondientes:

   Haremos una la lectura/ escritura del búfer o búferes asociado al puerto SPI específico que estamos usando, de los 4 posibles, mediante una función de escritura/lectura directa que desarrollaremos en la librería, extrapolando después a diferentes tamaños de palabras.

Viendo estas diferencias, podemos comprobar como la documentación dada puede confirmar nuestro planteamiento, indicando los diferentes espacios que tendremos que modificar para el correcto funcionamiento del port (Bridsgetek, 2019), recogidas en el punto 6 del documento: MCU Layer. Estos campos son:

- Manejabilidad del CS (High/Low): Asociado al cambio de pines.

- Manejabilidad del PD (High/Low): Asociado al cambio de pines.
- Lectura/Escritura de 8 bits: Teniendo en cuenta que el resto de lecturas/escrituras se realizan basadas en el de 8 bits, solo tendremos que preocuparnos de que esta lectura se realice correctamente y el tamaño de los datos sea el especificado.
- Todo lo relacionado al MCU_Init: relacionado con la diferencia de reloj, procesos de inicio del GPIO y módulos de SPI.

Por lo tanto, los cambios hechos son los siguientes:

3.1.4.1 Incializacion Reloj

```
  void initClock()
  {
      RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
SYSCTL_CFG_VCO_480), 120000000);
  }
```

Acorde a las especificaciones dadas de inicio del reloj, no se sale de lo que normalmente se realiza para iniciar este módulo.

3.1.4.2 Incializacion SPI

```
  void initSPI()
  {
      SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
      //
      //  pin PN3 es /CS
      //
      GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_3);
      CS_PORT=GPIO_PORTN_BASE;
      CS_PIN=GPIO_PIN_3;

      //
      // pin PN2 es /PD
      //
      GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_2);
      PD_PORT=GPIO_PORTN_BASE;
      PD_PIN=GPIO_PIN_2;

      //
      //  PD0 es SSI2 SSI2XDAT1
      //
      GPIOPinConfigure(GPIO_PD0_SSI2XDAT1);
      GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_0);

      //
      // PD1 es SSI2 SSI2XDAT0
      //
      GPIOPinConfigure(GPIO_PD1_SSI2XDAT0);
      GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_1);

      //
      // PD3 es SSI2 SSI2CLK
      //
      GPIOPinConfigure(GPIO_PD3_SSI2CLK);
      GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_3);
      SSI_BASE=SSI2_BASE;

      SSIConfigSetExpClk(SSI_BASE, RELOJ, SSI_FRF_MOTO_MODE_0,
                         SSI_MODE_MASTER, 1000000, 8);
      //
```

```
    // Enable the SSI module.
    //
    SSIEnable(SSI_BASE);

    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);

}
```

Inicializamos el puerto 2 de SPI, pese a que se podría haber iniciado cualquier de los otros puertos con la conexión de pines propia del puerto iniciado. Tras esto configuramos los pines especificados en el punto 2 del documento, y hacemos el procedimiento requerido para la inicialización del módulo SPI. Además, permití el uso de un led de la placa de cara a debuggear y saber en qué estado de la inicialización se encontraba para encontrar diferentes fallos en las primeras estancias del proyecto.

3.1.4.3 Lectura/Escritura del buffer

```
uint8_t MCU_SPIReadWrite8(uint8_t DataToWrite)
{
    uint32_t DataRead;
    SSIDataPut(SSI_BASE,DataToWrite);
    while(SSIBusy(SSI_BASE));

    SSIDataGet(SSI_BASE, &DataRead);

    return (uint8_t)DataRead;
}
```

A nivel funcional, el dato pasado es puesto en el buffer y el sistema espera la respuesta cuando el dato es totalmente transmitido. Por último, devuelve el dato, que será de 8 bits mediante un casteo hacia ese tipo de variable. A nivel funcional esto es clave para mantener la estructura desarrollada de la librería y hacer los mínimos cambios posibles.

Con estos cambios, el proyecto debería ser ejecutable por nuestro launchPad objetivo.

3.2 Bases teóricas de diseño: Estructura de la sintaxis de la creación de pantallas

Uno de los problemas principales del proyecto es dar con la estrategia para la creación de forma sistemática de bloques que podamos mostrar por pantalla sin crear malfuncionamientos a causa de pisar comandos en diferentes bloques creados. Sin afrontar este problema, no se podría mostrar el trabajo por pantalla, y el trabajo carecería de sentido.

Pues Bridgetek proporciona una forma muy eficiente de poder declarar diversos bloques funcionales en una pantalla y mostrarla sin problemas: La creación de una estructura de bloques para la declaración de las funcionalidades con comandos específicos para el almacenamiento y demostración de la pantalla.

La estructura a seguir es la siguiente:

1. La cabecera: Específicamente es la línea de comando `EVE_LIB_BeginCoProList();`

   Esta línea inicializa la declaración del bloque y es totalmente obligatoria. Si esto no se pusiera, la declaración del bloque seria ignorada y podrían producirse errores de declaración al usar comandos de cierre de bloque sin utilizar uno de apertura.

2. El cuerpo del bloque:

   Aquí va, literalmente, cualquier comando que no sea de cierre o inicio. Existen determinados comandos

que nos interesará utilizar juntos en algunos procesos típicos, además de comandos específicos sueltos para realizar tareas comunes. Estos serán:

- Limpiar pantalla e inicializar nueva pantalla a un color RGB dado:

```
EVE_CMD_DLSTART();
EVE_CLEAR_COLOR_RGB(R,G,B);
EVE_CLEAR(1,1,1);
```

- Terminar la pantalla y mostrarla:

```
EVE_DISPLAY();
EVE_CMD_SWAP();
```

- Cambiar los elementos (botones, sliders, ruedas…) de la pantalla a un color específico RGB dado:

```
EVE_CMD_FGCOLOR(RGB);
```

  Con una pequeña aclaración, RGB será una variable de 24b contenida en 32b, donde se reparten a partes iguales de 8b los campos para cada color.

- Cambiar el texto y números de la pantalla (incluido los de dentro de elementos, como los botones) a un color RGB dado:

```
EVE_COLOR_RGB(R,G,B);
```

3. Cierre del bloque: Combinación de dos líneas que sirven como cola de bloque para cerrar la instrucción.

```
EVE_LIB_EndCoProList();
EVE_LIB_AwaitCoProEmpty();
```

   Esto cierra el bloque y produce una espera para poder transferirlo de forma correcta. Cuando termina la transmisión, sale de la última indicación.

Por proporcionar algo más gráfico, se puede pensar como un sándwich:

Figure 16.Esquema de sintaxis de comandos de pantalla

Al estructurar las ordenes de la pantalla de esta forma tenemos la libertad de programar cada funcionalidad individual en un “sándwich” y añadirlo poco a poco a la lista completa que se mostrará por pantalla. Dicho esto, podemos apilar infinitos sándwiches, pero solo se mostrarán aquellos que se almacenen antes de una orden de mostrar la pantalla, pese a que después de dar esta orden sea posible seguir almacenando bloques de comandos. Es más, cualquier bloque de comandos que se almacene tras una orden de mostrar la pantalla contará para la siguiente lista a mostrar, cosa que podría resultar un problema o una ventaja, depende de nuestra maestría organizando los comandos.

Por esquematizarlo:

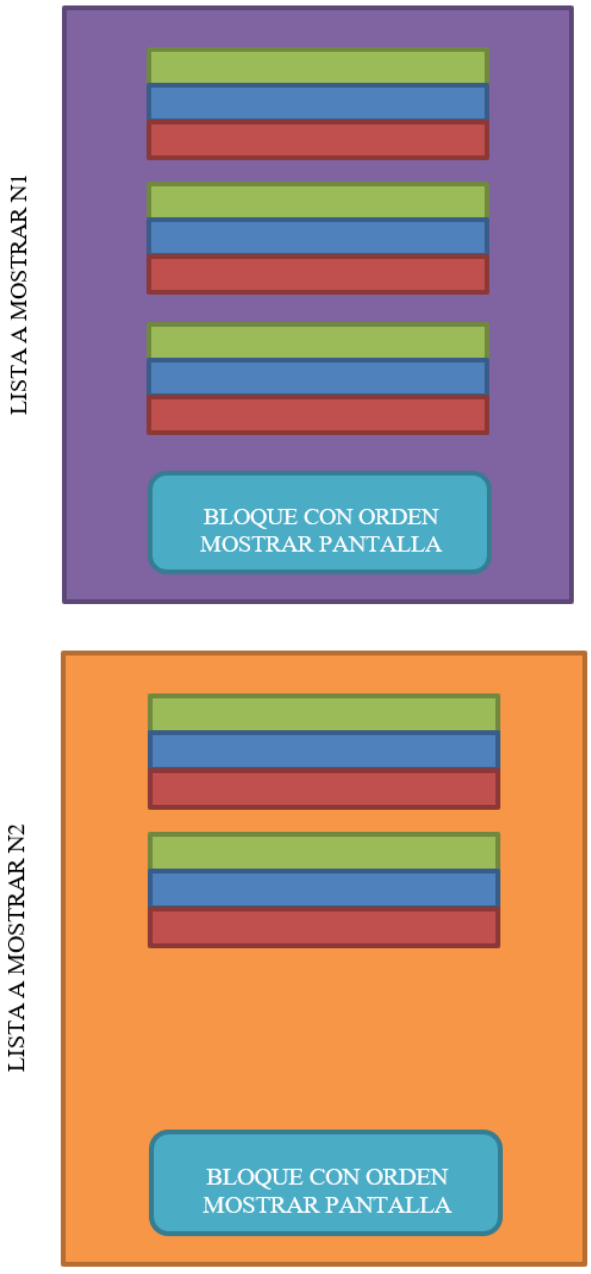

Figure 17.Múltiples pantallas a mostrar, esquematización

Pese a la posibilidad de hacer 2,3,4 o bloques de mostrar pantalla, tiene poco sentido o ninguno a nivel funcional salvo para lograr efectos muy específicos. La explicación se puede aplicar a 2 casos y es tan simple como que, en el caso de no estar dentro de un bucle, la última pantalla sobrescribirá a las primeras y a efectos prácticos solo veríamos la última, y en el caso de que si lo esté, veremos una superposición constante de ambas pantallas, no permitiendo una visión clara.

Con la teoría aquí explicada, se pueden realizar funciones específicas que, pese a no ser estrictamente necesarias, son beneficiosas para nuestra comodidad a la hora de programar las diversas funcionalidades a las que nos enfrentaremos. Por citar los bloques:

```
void EVE_INICIALIZAR_NUEVA_PANTALLA(){
EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_CMD_DLSTART(); // When executed, EVE will begin a new DL
    EVE_CLEAR_COLOR_RGB(0, 0, 0); // Select color to clear screen to
    EVE_CLEAR(1,1,1); // Clear

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}
void EVE_MOSTRAR_FIN_PANTALLA(){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_DISPLAY(); // Tells EVE that this is the end
    EVE_CMD_SWAP(); // Swaps new list into foreground buffer

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}
void EVE_LIMPIAR_PANTALLA_A(uint8_t R,uint8_t G,uint8_t B){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_CLEAR_COLOR_RGB(R, G, B); // Select color to clear screen to
    EVE_CLEAR(1,1,1); // Clear

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}
void EVE_COLOR_ELEMENTO(){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD
//    uint32_t Auxiliar=;
    EVE_COLOR_RGB(256,256,256);
    EVE_CMD_FGCOLOR(0x00ff00ff);

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}
```

Tras esto, tenemos los cimientos establecidos para poder empezar a crear las funcionalidades de forma consistente.

3.3 Estructura custom principal: BOTON

Esta estructura con nombre tan poco usado dentro del proyecto será la estructura principal, el esqueleto del proyecto, la estrella que guiará al sistema. A partir de aquí, todos los sistemas estarán pensados para que toda la información relevante sea volcada en un array de esta estructura.

Pasemos a explicar los diferentes campos escogidos y su justificación.

3.3.1 Campo físico

Tan sencillo como los parámetros físicos de posición(X-Y) y tamaño(W-H) asociados al botón. La justificación es bastante trivial, sin estos valores no existiría el botón.

Las variables existirán dentro de este campo, y se referenciará a ellas mediante la declaración Boton.físico.[aquí la variable]

En código:

```
struct fisico{
    uint16_t x;
    uint16_t y;
    uint16_t w;
    uint16_t h;
};
```

3.3.2 Campo BotAcción

Aqui nos estamos adelantando a acontecimientos ya que, pese a que aún no nos hemos metido dentro de las acciones que realizarán nuestros botones, podemos prever que necesitaremos un lugar donde guardar que acciones queremos asignar a determinados botones. Este espacio se proporcionará con la variable TIPO, de la que se dará una tabla completa de comandos y referencias cuando se cree la función que maneje este valor, o función handler.

Además, vamos a seguir reservando espacio que, por la naturaleza del proyecto, prevemos que utilizaremos en el futuro. Este es el caso de los campos COMANDO y la estructura de 4bits MOD. Estos campos serán usados cuando empecemos a desarrollar las funcionalidades de teclado USB junto con la modificación mediante el servidor web, y se explicará según se vaya necesitando para no adelantar funcionalidades.

Su declaración en código:

```
struct Accion{
    uint8_t tipo;
//    Por ej:
//    1-EditMatrixmode
//    2-EditLibre
//    3-IncPantalla
//    4-DecPantalla
//    5-ComandoTeclado
//    6-Sync?...
    uint16_t comando; //Si lo hubiera
    bool mod[4];
};
```

3.3.3 Estrucutra de BOTON conjunta

Tras declarar los campos anteriores, pasamos a definir en esta estructura de bloques, el contenedor final. Vamos a añadir a los campos anteriores variables interesantes que nos vendrán bien a la hora de facilitarnos la vida a nivel de programación y creación de características como son TAGASOC, para detectar la pulsación, y PANTALLAASOC, para la creación de diferentes pantallas virtuales con una sola estructura, así como una variable de identificación extra en NAME, para poder guardar nombre de botones para etiquetarlos encima del elemento.

En código, la estructura final queda:

```
struct Boton{
    struct fisico fis;
    char name[6];
    uint8_t tagAsoc;
```

```
    uint8_t pantallaAsoc; //16 es territorio de nadie, 0 es presente en todas as
pantallas
    struct Accion BotAccion;
};
```

Con todo esto hecho, hemos definido prácticamente todo el soporte funcional de nuestro trabajo. Para cualquier gestión correspondiente a los botones, solo tendremos que referenciar el proceso al campo del botón especifico dentro del array, y accederá a la información que considere oportuna.

Por tener un pequeño esquemático:

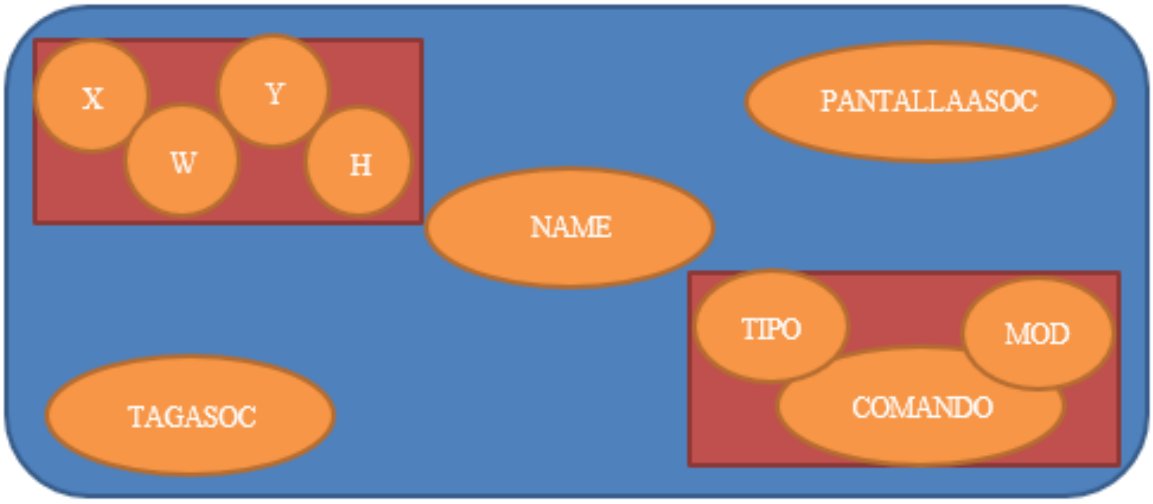

Figure 18. Esquema de contenedor de estructura BOTON

3.4 Creando funciones básicas para el funcionamiento

Tras poder correr el ejemplo básico en nuestra TM4C1294XL pasamos al siguiente nivel de integración: necesitamos algunas funciones de funcionamiento muy básico para fundamentar nuestras aplicaciones más avanzadas. Con esto me refiero a la capacidad de leer la posición del toque en pantalla y leer elementos que pulsemos.

3.4.1 Primeras modificaciones en EVE_API.c

Comenzaremos por la función para saber que coordenadas tienen nuestro contacto con la pantalla. Como podemos comprobar en documentación, el registro que tiene la información que buscamos es el EVE_REG_TOUCH_SCREEN_XY, siendo este de 32 bits y correspondiendo 16b a cada coordenada 2D.

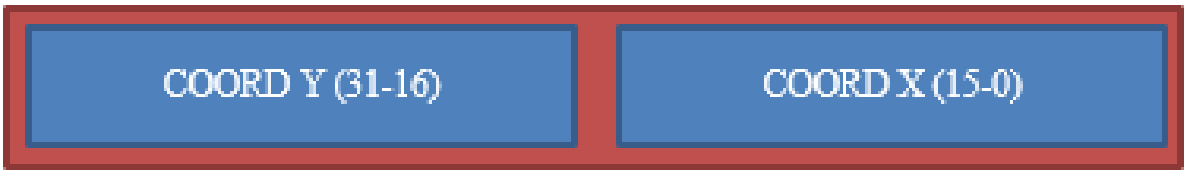

Figure 19. Registro de coordenadas de la pantalla táctil

Por lo tanto, hacemos lo siguiente, y con esto introduzco la función de funcionamiento básico **EVE_LEER_PANTALLA**:

```
void EVE_LEER_PANTALLA(){
    BufferXY = HAL_MemRead32(EVE_REG_TOUCH_SCREEN_XY);
    POSX=(BufferXY&0xffff0000)>>16;
```

```
    POSY=BufferXY&0x0000FFFF;
}
```

Aquí, almacenaremos la lectura del registro en una variable intermedia de 32b que usamos para asignar los valores por separado a POSX y POSY, de 16b, mediante un proceso de desplazamiento y enmascaramiento, respectivamente.

Por el bien del lector, haré un pequeño recordatorio de que los ejes en las pantallas funcionan de forma diferente al clásico sistema de ejes 2D que se suele enseñar en matemáticas o física. Estos ejes son iguales en perpendicularidad y dirección del eje X, pero el eje Y invierte su dirección:

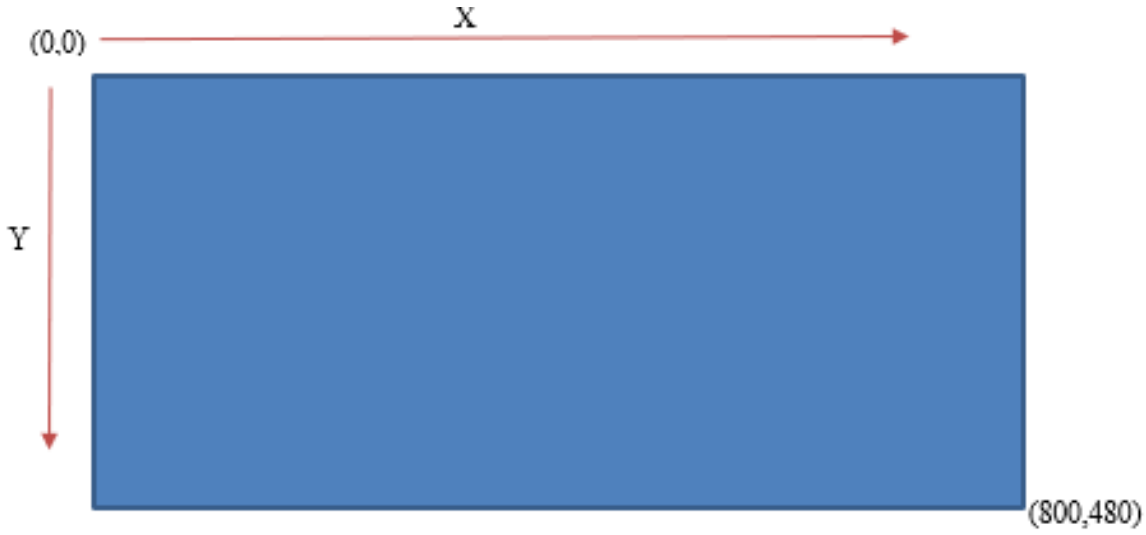

Figure 20. Esquematización de pantalla

Tras esto, adoptaremos el modo de funcionamiento de la función helper del ejemplo original, aislándola para que sirva en un propósito más general, quedándonos la función **EVE_LEER_TAG**:

```
uint8_t EVE_LEER_TAG(){
    uint8_t tag=0;
    uint8_t Read_tag;
    Read_tag = HAL_MemRead8(EVE_REG_TOUCH_TAG);
    if (!(HAL_MemRead16(EVE_REG_TOUCH_RAW_XY) & 0x8000))
        tag = Read_tag;
    return tag;
}
```

Esta devolverá el tag asociado al elemento tras leer el registro determinado y comprobar que está dentro del espacio interactuarle.

Este concepto de tag, no es más que un numero indicador que se le asigna a cada elemento creado en pantalla y en el futuro, cuando gestionemos diversos elementos que se están dibujando, junto con otros que están preprogramados, pero no visibles, tendremos que tener mucho cuidado con problemas de compatibilidad y tag duplicados.

3.5 Funciones para botones

El uso y la creación de botones es la funcionalidad en la que se fundamenta el proyecto, por lo que se busca un uso cómodo de éstos, permitiendo al usuario modificar y personalizar rápidamente los aspectos que considere

necesario.

Para empezar, tenemos que entender cómo funciona a nivel básico la función por defecto para este fin que trae las librerías proporcionadas, calco de lo que indica la guía de usuario para FT81x.

3.5.1 Funcion base EVE_CMD_BUTTON

Esto se amplía por el manual en la página 174, punto 5.28. (FTDI, 2015)

Para empezar, las variables que tendremos que pasarle a la función:

1. X-Y: coordenada x-y de la esquina superior izquierda
2. W-H: referentes a width(ancho) y height(alto), las dimensiones del botón.
3. FONT: es la fuente con la que queremos que ese botón en específico sea, en el caso de que exista un nombre asociado, etiquetado con su indicador.
4. OPCIONES: Usaremos esta función para cosas como posicionar el texto dentro del botón, y lo más importante, hacer que parezca que sobresale cuando está en reposo y aplastado cuando lo tocamos, es decir, un efecto de pulsación física.
5. NOMBRE: nombre, literalmente, del botón. Se escribirá encima todo lo que se detecte que va delante del típico terminador '\0'. Esto será importante más adelante a la hora modificar los nombres para diferenciar botones e indicar su comando a realizar.

Pasándole esta información mediante la función, conseguiremos dibujar un botón en pantalla. Si asignásemos un TAG y una acción a la detección específica de este, podríamos hacer que fuese un botón interactuarle.

3.5.2 Botón con animación realista, EVE_BOTON_P

El **BOTON_P**(ulsable) es una función puente que creé en su momento para jugar con las opciones y conseguir el efecto de pulsación realista. Esta evolucionará a la hora de manejar la estructura que se creará más adelante para simplificar sus variables de input y proporcionar un funcionamiento más versátil.

A nivel de variables que pasamos a la función, decidí reducir el número de cara a hacer test rápidos de funcionamiento, quedándome con X, Y, W y H. Además, añado la posibilidad de pasar un TAG para que la función lo asocie directamente al elemento que produzca.

3.5.2.1 Asociación a TAGS

¿Como se asocial un TAG? La realidad, es que es muy sencillo. Existe una función proporcionada por la propia librería, llamada EVE_TAG(t), que nos hará el trabajo. Funciona de forma muy similar a la asignación de colores a los elementos de la pantalla, es decir, cuando ponemos el comando con t siendo un número, todas las entidades creadas tras este comando serán asignadas con ese TAG=t. La única forma de que esto no sea así es que, cada vez que queramos crear un nuevo elemento, designemos su TAG.

3.5.2.2 Animación física

Como he comentado antes, este efecto depende del campo de OPCIONES de la función base EVE_CMD_BUTTON. Para crear el efecto crearemos una función que dependiendo de si el TAG actual es el asignado al botón creará un botón de 2 posibles:

- El primer botón será el reposo, este sale por defecto con efecto 3D por lo que no serán necesarios cambios en las opciones básicas.
- El segundo será el botón pulsado, pasando a ser un botón "2D" al estar aplastado a nivel pantalla. Esto se consigue con la opción EVE_OPT_FLAT.

La función final queda:

```
void EVE_BOTON_P(int16_t x,int16_t y,int16_t L,int16_t L2, uint8_t t){
    extern uint8_t TAG;
```

```
    EVE_TAG(t);
    if(TAG==t)
        EVE_CMD_BUTTON(x, y, L, L2, 28, EVE_OPT_FLAT, "\0");
    else
        EVE_CMD_BUTTON(x, y, L, L2, 28, EVE_OPT_CENTER, "\0");
}
```

Tras esto, y antes de diseñar más funciones, pasé a diseñar la estructura que sería la espina dorsal del proyecto. De esta manera, cualquier función específica de botones a partir de tener esto terminado, podría trabajar con la información que terminaría dominando el proyecto y evitarme hacer adaptaciones y readaptaciones.

3.5.3 Evolucion a EVE_BOTON_P_2

Tras la creación de la estructura, podemos evolucionar la función anterior de EVE_BOTON_P a **EVE_BOTON_P_2**, a la que única y exclusivamente tendremos que pasarle el índice dentro de la tabla de botones que inicialicemos en el **main.c**, y esta función buscará la información que necesite automáticamente.

En código tendremos lo siguiente:

```
  void EVE_BOTON_P_2(uint8_t n){
      extern uint8_t TAG;
      EVE_TAG(Botones[n].tagAsoc);
      if(TAG==Botones[n].tagAsoc)
          EVE_CMD_BUTTON(Botones[n].fis.x,     Botones[n].fis.y,     Botones[n].fis.w,
Botones[n].fis.h, 28, EVE_OPT_FLAT, Botones[n].name);
      else
          EVE_CMD_BUTTON(Botones[n].fis.x,     Botones[n].fis.y,     Botones[n].fis.w,
Botones[n].fis.h, 28, EVE_OPT_CENTER, Botones[n].name);
  }
```

3.5.4 Almacenamiento en tabla: CREAR_BOTON

Para que los campos inicializados cambien de valor a los que nos interesen, necesitaremos una función que asigne los campos específicos con datos que le proporcionemos. Este es el cometido de la función CREAR_BOTON.

En esta función, además de asignar los valores que le pasemos a la función, haremos un asignador automático de índice dentro de la tabla y de comandos para representar teclas placeholder para el teclado USB.

Antes de explicar el mecanismo, introduciré primero una idea de diseño que afecta a la variable PATALLAASOC. Como se podía leer en el comentario anterior, se asigna una pantalla omnipresente en el valor 0, es decir, cualquier botón con pantalla asociada igual a 0, aparecerá en todas las pantallas diferentes que se muestren, además de los botones que estén asignados a la pantalla especifica. A parte de la omnipresencia que significa el 0, establecemos un numero de pantalla como inicialización en el 16, sin ninguna razón detrás, pero con este número conoceré que botones han sido únicamente inicializados y no usados.

Con esto, el funcionamiento del TAG automático es simple, se revisa la tabla y se cuenta cuantos botones si han sido definidos tras la inicialización y, tras esto, se le asigna el siguiente número que toque. Por ejemplo, inicializamos la tabla, creamos 3 botones en la pantalla 0, 4 en la 1, 5 en la 2, y 0 en la 3 y 4; tras esto el siguiente TAG asociado al botón será la suma de todos los botones anteriores +1, 3+4+5+1=13.

El funcionamiento del placeholder está pensado para que sea un número que se estará asociado a una tabla donde tendremos almacenados diferentes combinaciones de teclas que pasaremos al bloque de teclado USB para que pueda realizar acciones de teclado. Esta tabla se compondrá de diferentes combinaciones de CTRL+SHIFT+NUMPAD, y se usará de funciones básicas. La asignación automática está pensada de forma parecida al del TAG, pero, en vez de contar todos los botones existentes, contará los asignados a su pantalla sin contar los generales.

Para agilizar el proceso, se crea el campo origen que hará más rápida el acceso a las partes que nos interesen de la tabla, y con esta variable llega la decisión de diseño de tener en la tabla que contiene los botones un total de:

- 12 botones de sistema u omnipresentes.
- 26 botones de creación automática. El proceso se explicará más adelante.
- 26 botones de creación manual. Símil a los automáticos.

Tras todo esto, nos queda la siguiente función:

```
  void CREAR_BOTON(uint8_t origen,uint16_t x,uint16_t y,uint16_t w,uint16_t h, char named[5],uint8_t pantallaAsocd, uint8_t tipod, uint16_t comandod){
      int i;
      uint8_t index;
      uint16_t comandoR=1;
      if(origen==0){
          index=0;
          comandoR=comandod;
      }
      else if(origen==1){
          index=OFFSETBOTMATRIX;
          for(i=OFFSETBOTMATRIX;i<64;i++){
              if(Botones[i].pantallaAsoc==PAG && Botones[i].BotAccion.comando!=0){
                  comandoR++;
              }
          }
      }
      else if(origen==2){
          index=OFFSETBOTLIBRE;
          for(i=OFFSETBOTMATRIX;i<64;i++){
              if(Botones[i].pantallaAsoc==PAG && Botones[i].BotAccion.comando!=0){
                  comandoR++;
              }
          }
      }
      while(Botones[index].pantallaAsoc!=16){
          index+=1;
      }
      Botones[index].fis.x=x;
      Botones[index].fis.y=y;
      Botones[index].fis.w=w;
      Botones[index].fis.h=h;
      strcpy(Botones[index].name,named);
      Botones[index].tagAsoc=index+1;
      Botones[index].pantallaAsoc=pantallaAsocd;
      Botones[index].BotAccion.tipo=tipod;
      Botones[index].BotAccion.comando=comandoR;
      cambioF1=true;
  }
```

3.5.5 Modo de edición libre: EVE_EDIT_MODE_L

La primera función con la que le daremos poder al usuario para crear los botones que estime oportuno. La idea a realizar es la típica de creación de figuras de toque y arrastre: el primer toque establece la esquina izquierda superior, y al arrastrar redimensionamos el botón.

De cara a entrar en este modo, crearemos un botón con esta función asignada para salir y entrar en la funcionalidad, así permitiremos al usuario crear cuando plazca y no tener que estar atento a no crear botones accidentalmente al tocar la pantalla por error.

La función empezará con un modo anti rebote, es decir, que el sistema espere a que se suelte el botón para empezar el modo y que no ande cambiando constantemente entre modo de edición y modo normal cuando mantenemos el botón pulsado. Esto se realiza de la siguiente forma:

```
while(TAG!=0){
            EVE_LEER_PANTALLA();
```

```
            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }
```

Es decir, mientras que el TAG sigue siendo el que produjo la entrada en la función soltado y no se ha soltado (TAG=0), sigue generando pantallas y detectado donde está el toque y que TAG está asignado.

Tras esto, la lectura del buffer que almacena la posición del contacto se vuelve muy importante ya que la primera lectura de este corresponderá a la X e Y de nuestro botón a crear, y la diferencia entre las siguientes lecturas y este punto origen serán nuestro W y H. Para esto, creamos las variables auxiliares pertinentes y creamos una máquina de estados que recorre los diferentes estados en la creación de estos botones, haciendo que sea visible constantemente el botón que estamos creando para permitir buen feedback del dispositivo hacia el usuario:

```
while(TAG!=Botones[1].tagAsoc)
    {
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        ////////////////////////////////////////////
        switch (caso) {
        case 0:
            if(BufferXY!=0x80008000)
                caso=1;
            break;
        case 1:
            Xi=POSX;
            Yi=POSY;
            Xf=POSX;
            Yf=POSY;
            caso=2;
            break;
        case 2:
            EVE_LIB_BeginCoProList();

            EVE_CMD_BUTTON(Xi,Yi,Xf-Xi,Yf-Yi,28,EVE_OPT_CENTER,"\0");

            EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
            EVE_LIB_AwaitCoProEmpty();

            if(BufferXY==0x80008000)
                caso=3;
            else{
                Xf=POSX;
                Yf=POSY;
            }
            break;
        case 3:

            CREAR_BOTON(2,Xi,Yi,Xf-Xi,Yf-Yi,"LIB",PAG,10,0);
            caso=0;
            break;
        }

        EVE_LEER_PANTALLA();

        EVE_MOSTRAR_BOTONES_GLOBAL();

        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
```

```
    }
```

Una pequeña nota: nos mantendremos en esta función hasta que detecte que se está pulsando el TAG asociado a la función de edición libre, de ahí la condición del while asociada a esta parte.

Cuando se pulsa este botón mencionado, hacemos otra rutina anti-rebote y terminamos:

```
while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
```

Quedando la función completa:

```
void EVE_EDIT_MODE_L(){
    int Xi,Yi,Xf,Yf;
    int caso=0;
    while(TAG!=0){
            EVE_LEER_PANTALLA();

            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }

    while(TAG!=Botones[1].tagAsoc)
    {
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        ////////////////////////////////////////
        switch (caso) {
        case 0:
            if(BufferXY!=0x80008000)
                caso=1;
            break;
        case 1:
            Xi=POSX;
            Yi=POSY;
            Xf=POSX;
            Yf=POSY;
            caso=2;
            break;
        case 2:
            EVE_LIB_BeginCoProList();

            EVE_CMD_BUTTON(Xi,Yi,Xf-Xi,Yf-Yi,28,EVE_OPT_CENTER,"\0");

            EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
            EVE_LIB_AwaitCoProEmpty();

            if(BufferXY==0x80008000)
                caso=3;
            else{
                Xf=POSX;
                Yf=POSY;
            }
            break;
```

```
            case 3:

                CREAR_BOTON(2,Xi,Yi,Xf-Xi,Yf-Yi,"LIB",PAG,10,0);
                caso=0;
                break;
            }

            EVE_LEER_PANTALLA();

            EVE_MOSTRAR_BOTONES_GLOBAL();

            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();

        }
        while(TAG!=0){
                    EVE_LEER_PANTALLA();

                    EVE_INICIALIZAR_NUEVA_PANTALLA();
                    EVE_COLOR_ELEMENTO();
                    EVE_MOSTRAR_BOTONES_GLOBAL();
                    EVE_MOSTRAR_FIN_PANTALLA();

                    TAG=EVE_LEER_TAG();
                }
    }
```

3.5.6 Modo de edición automática: EVE_EDIT_MODE_MATRIX

Esta función proporcionará una herramienta para crear matrices de botones automáticamente con el único requisito de pasarle el número de botones que deseamos que esa matriz tenga. Esto separa a un proceso que tomará las decisiones pertinentes como, el tamaño y la posición de los botones, incluyendo el número de filas y columnas que encaja mejor dentro de las condiciones dadas. Para explicar todo, empezaré con la función interna que toma las decisiones de diseño y, tras esto, describiré el proceso que sigue el botón, muy parecido al del punto anterior.

3.5.6.1 Función Interna de Decisión: EVE_MATRIX_BOTON

Para entender y diseñar este proceso interno, hay que pararse a entender, primero, como haríamos nosotros este proceso para extrapolarlo al programa:

- Para empezar, toda la pantalla no está disponible. Tenemos que contar con que habrá botones de funcionalidad en los bordes de la pantalla (decisión de diseño de pantalla para facilitar las cosas, aparte de ser donde normalmente están por norma general), y que además, por temas visuales, tendremos que dejar un pequeño espacio entre botones para facilitar su entendimiento. Este concepto da lugar a los conceptos de los offset: LILOFF (Little offset), BIGOFF (Big offset) y SUPOFF (Superior Offset). Correspondiendo: el pequeño, al offset dejado de margen en las zonas laterales; el grande, al dejado en la zona baja de la pantalla (donde acumulamos más botones) y el superior, al de la zona superior (donde solo estará en un futuro el número de pantalla actual).
- Después de esto, elegiríamos una estructura de botones que normalmente nos dejara tener los botones más grandes que pudiésemos, descartando así configuraciones que nos dejaran botones por debajo de una medida. Por esto se establece en LMIN (lado mínimo), que corresponde al lado del cuadrado más pequeño que aceptaremos como botón dibujable.
- Por último, de las variables a declarar, tenemos que no pondríamos los botones uno al lado del otro inmediatamente, puesto que daría la sensación de agobio visual. Esto lo arreglamos dejando una pequeña separación igual para todos que hace que esa sensación desaparezca y se vea ordenado, a la vez que limpio. De aquí nace SEPMIN (separación mínima).

Tras declarar estos parámetros iniciales para el diseño y la decisión, nos queda establecer que estrategia vamos a seguir para decidir la estructura de la matriz. Ésta es a la que he llegado yo:

1. Empezamos con un contador de columnas igual a 1, indicando que tenemos un total de 1 columna.
2. Dividimos el espacio disponible de la pantalla tras aplicar los parámetros iniciales entre el número de columnas actual, para saber el lado máximo posible que cabría en pantalla sin afectar a la columna siguiente
3. Como sabemos el número de columnas, el número de filas será el número de botones entre el de columnas.
4. El lado del botón ajustado a fila será el resultado de calcular el espacio disponible en pantalla en el eje x, tras quitarle las separaciones pertinentes, entre en número de filas.
5. Ahora comprobamos que los valores sean aceptables: ¿El lado ajustado es mayor que el lado máximo calculado en el punto 2? Si es así lo limitamos igualándolo al lado máximo, y en el caso de que no sea así ¿Es el lado ajustado menor que el lado mínimo aceptable que hemos establecido? Si es que sí, incrementamos el número de columnas en 1 sobre el que trabajamos y volvemos al punto 2 de esta lista.

En código queda algo así:

```
Bcol=1;
    Xu=800-2*LilOff;
    Yu=480-SupOff-BigOff;

    while(L<Lmin){
        Lmax=Yu/(Bcol+1)-20;
        Bfil=n/Bcol;
        L=(Xu-Bfil*SepMin)/Bfil;
        if(L>Lmax)
            L=Lmax;
        else if(L<Lmin)
            Bcol+=1;
    }
```

Tras esto tendremos los parámetros de los botones, así como la posibilidad de sacar sus posiciones de forma directa mediante operaciones matemáticas básicas, por lo que creamos los botones pertinentes con la función creada en puntos anteriores:

```
for (i=1; i<=Bcol; i++){
       for (j=1; j<=Bfil; j++){
           x=j*(Xu/(Bfil+1))+LilOff;
           y=i*(Yu/(Bcol+1))+LilOff;
           CREAR_BOTON(1,x-L/2,y-L/2,L,L,"MAT",PAG,10,0);
       }
```

La función general será:

```
void EVE_MATRIX_BOTON(int n){
    const int LilOff= 50;
    const int SupOff= 20;
    const int BigOff=65;
    const int Lmin=60;
    const int SepMin=50;
    int Xu,Yu,Bfil,Bcol,Lmax,i,j,x,y;
    int L=0;

    Bcol=1;
    Xu=800-2*LilOff;
    Yu=480-SupOff-BigOff;
```

```
    while(L<Lmin){
        Lmax=Yu/(Bcol+1)-20;
        Bfil=n/Bcol;
        L=(Xu-Bfil*SepMin)/Bfil;
        if(L>Lmax)
            L=Lmax;
        else if(L<Lmin)
            Bcol+=1;
    }
   for (i=1; i<=Bcol; i++){
       for (j=1; j<=Bfil; j++){
           x=j*(Xu/(Bfil+1))+LilOff;
           y=i*(Yu/(Bcol+1))+LilOff;
           CREAR_BOTON(1,x-L/2,y-L/2,L,L,"MAT",PAG,10,0);
       }
   }
}
```

3.5.6.2 Funcion del proceso general

Tras implementar la función específica, vamos a crear la interfaz con la que interactuará el usuario.

A nivel general, se parece en estructura a la de edición libre, es decir, dos funciones anti rebote a la salida y a la entrada con un cuerpo intermedio que gestiona el proceso. Obviaré referirme otra vez al proceso contra los rebotes y pasaré directamente al cuerpo del proceso.

Esta parte central generará una pantalla temporal intermedia con un botón en el centro, un contador debajo de este, y el botón que se ha pulsado para entrar en el modo. Cada vez que se pulse el botón central se incrementará en 2 un contador, indicando éste el número de botones deseados por el usuario. Cuando el número sea el deseado, se pulsará el botón con el que se accederá a este modo para volver a la pantalla que se estaba editando, donde se habrán generado los botone que se indicó mediante la función explicada en el subapartado anterior.

El contador se incrementa de 2 en 2 debido a la naturaleza de las operaciones de la función de decisión, que ignorarán los números impares y mostrarán únicamente el par inferior.

La función completa, con anti rebotes, quedará así:

```
void EVE_EDIT_MODE_MATRIX(){
    int n=0;
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    }

    while(TAG!=Botones[0].tagAsoc){
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();

        EVE_LIB_BeginCoProList();

        EVE_BOTON_P(350,190,100,100,Botones[0].tagAsoc+1);
        EVE_BOTON_P_2(0); //Boton de EditMatrix
        EVE_CMD_NUMBER(400,300,28,EVE_OPT_CENTER,n);

        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

        EVE_MOSTRAR_FIN_PANTALLA();
```

```
            TAG=EVE_LEER_TAG();

            if(TAG==Botones[0].tagAsoc+1){
                n+=2;
                while(TAG==Botones[0].tagAsoc+1){
                    EVE_LEER_PANTALLA();
                    TAG=EVE_LEER_TAG();

                    EVE_INICIALIZAR_NUEVA_PANTALLA();
                    EVE_COLOR_ELEMENTO();
                    EVE_LIB_BeginCoProList();

                    EVE_BOTON_P(350,190,100,100,Botones[0].tagAsoc+1);
                    EVE_BOTON_P_2(0); //Boton de EditMatrix
                    EVE_CMD_NUMBER(400,300,28,EVE_OPT_CENTER,n);

                    EVE_LIB_EndCoProList();
                    EVE_LIB_AwaitCoProEmpty();
                    EVE_MOSTRAR_FIN_PANTALLA();
                }
            }
        }
        while(TAG!=0){
            EVE_LEER_PANTALLA();
            TAG=EVE_LEER_TAG();
        }
        EVE_MATRIX_BOTON(n);
    }
```

Por si el lector no se ha dado cuenta, en esta función existe un posible problema de TAG cruzados. Este se soluciona asignando un TAG dependiente del TAG del único botón que se mantiene entre los modos de edición automática y normal. Por lo que, pese a que sea un TAG que se asigna a otro elemento dentro del sistema, ese elemento no se está mostrando dentro de la pantalla en ese momento, evitando el problema.

3.5.7 Botones visibles e invisibles, EVE_MOSTRAR_BOTONES_GLOBAL

Tras editar los botones, no ha habido ningún tipo de declaración explicita de los comandos para mostrar botones en pantalla, es decir, **CMD_BUTTON** o **EVE_BOTON_P** en cualquiera d sus versiones. Pero entonces, ¿cómo estamos mostrando los botones en la pantalla? Para esto creamos una función que se invoca en cada refresco de pantalla y que selecciona que botones de la tabla tiene que mostrar en cada momento.

La funcionalidad es muy simple, usamos la estructura de la sintaxis de los bloques de comandos para la creación de las pantallas y encapsulamos un bucle for que recorre la tabla buscando los botones omnipresentes y los botones contenidos en la pantalla actual, el índice se lo pasamos a la función anteriormente creada **EVE_BOTON_P_2** y esta se encarga de la gestión.

El código queda sencillo tal que así:

```
void EVE_MOSTRAR_BOTONES_GLOBAL(){
    EVE_LIB_BeginCoProList();
    for(i=0;i<64;i++){
        if(Botones[i].pantallaAsoc==0 || Botones[i].pantallaAsoc==PAG){
            EVE_BOTON_P_2(i);
        }
    }
    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();

}
```

3.5.8 Borremos botones, EVE_ERASE()

La base funcional de esta característica es muy sencilla: devolvemos el botón seleccionado a los valores de inicialización, lo que hace que esté virtualmente borrado. Para esto repasamos la tabla buscando el TAG asociado al botón seleccionado, copiamos el indicie que encontremos como correcto, y cambiamos esos valores.

La función general:

```
void EVE_ERASE(uint8_t boton){
    uint8_t index=OFFSETBOTMATRIX;
    while(Botones[index].tagAsoc!=boton && index<65){
        index+=1;
    }
    if(index<65){
        Botones[index].fis.x=0;
        Botones[index].fis.y=0;
        Botones[index].fis.w=0;
        Botones[index].fis.h=0;
        Botones[index].name[0]='\0';
        Botones[index].tagAsoc=0;
        Botones[index].pantallaAsoc=16;
        Botones[index].BotAccion.tipo=0;
        Botones[index].BotAccion.comando=0;
        Botones[index].BotAccion.mod[0]=0;
        Botones[index].BotAccion.mod[1]=0;
        Botones[index].BotAccion.mod[2]=0;
        Botones[index].BotAccion.mod[3]=0;
        cambioF1=true;
    }
}
```

Ese `cambioF1=true`, es algo relacionado con la memoria FLASH y será explicado más adelante.

3.5.9 Encapsulizando procesos, INIT_ESTRUCT_PRINC

Quería mencionar esto antes de pasar al handler principal que maneja todas las acciones relacionadas con botones. Es un proceso que inicializa la tabla de estructura botón y asigna los valores iniciales antes mencionados.

```
void INIT_ESTRUCT_PRINC(){
    uint8_t i;
    for(i=0;i<64;i++){

            Botones[i].fis.x=0;
            Botones[i].fis.y=0;
            Botones[i].fis.w=0;
            Botones[i].fis.h=0;
            strcpy(Botones[i].name,"\0\0\0\0\0\0");
            Botones[i].tagAsoc=0;
            Botones[i].pantallaAsoc=16;
            Botones[i].BotAccion.tipo=0;
            Botones[i].BotAccion.comando=0;
            Botones[i].BotAccion.mod[0]=0;
            Botones[i].BotAccion.mod[1]=0;
            Botones[i].BotAccion.mod[2]=0;
            Botones[i].BotAccion.mod[3]=0;
        }

}
```

Lo único mencionable es la composición de `Botones[i].name`, que está hecha de esa manera para que en futuras modificaciones el exceso de caracteres no sea un problema.

3.5.10 El sistema nervioso de los botones, EVE_HANDLER_ACCION

Tras la estructura que sustenta el proyecto, necesitaremos un sistema que tenga registrados que hacer dependiendo de que circunstancias. Estas circunstancias dependerán del campo `tipo` dentro de `BotAccion`, de la estructura principal.

Con la creación de esta función se nos presenta una decisión: que comandos se asignarán a que cosas. Esta es la lista final:

1. Modo de edición automática
2. Modo de edición libre
3. Incrementar la página
4. Decrementar la página
5. Borrar botones
6. Mostrar IP del servidor web, explicado más adelante en el documento
7. Función de calibración, explicado más adelante en el documento
10. Función de pulsación de un botón normal

A nivel de asignación, en los comandos 1 y 2 no hay que hacer mucho, solo inicializarlos, ya que la función tiene programada los filtros de rebote y la salida del modo.

Con respecto a los puntos 3 y 4, el manejo de páginas, se aplica un filtro de rebotes de entrada, y cuando se sale de este se incremente o decrementa dependiendo del comando. Por tener un ejemplo antes de la función general:

```
case 3:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            if(PAG>=4)
                PAG=1;
            else
                PAG+=1;
            break;
```

En los casos 5 a 7, incluimos los filtros pertinentes de rebotes rodeando a la ejecución de una función programada. Pondré como ejemplo el único punto explicado hasta ahora:

```
case 5:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            };
            while(TAG!=5){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();
```

```
                TAG=EVE_LEER_TAG();
                if(TAG>=OFFSETBOTMATRIX)
                    EVE_ERASE(TAG);
            }
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            };
            break;
```

Y, para terminar, el comando 10 estará asignada a una función que se creará cuando el teclado USB este configurado, al estar relacionada las teclas con comandos de éste.

```
case 10:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            EVE_ACTUADOR(TAG_P);
            break;
        }
```

El conjunto sería el siguiente:

```
void EVE_HANDLER_ACCION(){
    uint8_t accion=0;
    uint8_t TAG_P=TAG;
    uint8_t i;

    if(SubmitState){
        SubmitState=false;
        WEB_SUBMITER();
    }

    if(TAG!=0){
        accion=Botones[TAG-1].BotAccion.tipo;
        switch (accion) {
        default:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            break;

        case 1:
            EVE_EDIT_MODE_MATRIX();
```

```
    break;
case 2:
    EVE_EDIT_MODE_L();
    break;
case 3:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    }
    if(PAG>=4)
        PAG=1;
    else
        PAG+=1;
    break;

case 4:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    }
    if(PAG<=1)
        PAG=4;
    else
        PAG-=1;
    break;

case 5:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    };
    while(TAG!=5){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
        if(TAG>=OFFSETBOTMATRIX)
            EVE_ERASE(TAG);
    }
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
```

```
            };
            break;

        case 6:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            muestraIP=!muestraIP;
            break;

        case 7:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            for(i=0;i<6;i++){
                calib.transform[i]=0;
            }
            eve_calibrate();
            cambioF2=true;

            break;

        case 10:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            EVE_ACTUADOR(TAG_P);
            break;
        }

    }
}
```

3.6 Consideraciones finales

Pese a que este apartado de la pantalla se acaba, la biblioteca donde hemos realizado la mayoría de cambios se seguirá actualizando con funcionalidades cruzadas del teclado USB, el servidor web y la gestión de memoria FLASH. El lector no ha de preocuparse por nada ya que estas vueltas hacia atrás serán claramente indicadas.

Para redondear lo máximo posible, indicaré a continuación la pantalla que se está creando en el main.c, junto con variables que se han usado hasta ahora que sean globales y una breve explicación.

3.7 Avances en el main.c

A nivel de main.c, este apartado se lleva la mayor parte de código obviando los comandos referidos a las inicializaciones pertinentes de los siguientes módulos.

Fuera del bucle haremos todas las iniciaciones como **INIT_ESTRUCT_PRINC()**, **EVE_INIT(),** y **eve_calibrate(),** además de crear los botones de sistema asociados a las pantallas, los modos de edición, la papelera de botones, la IP y la recalibración. Estos botones serán así y no sufrirá cambios a lo largo del proyecto:

```
CREAR_BOTON(0,0,420,90,60,"Mtrx",0,1,0);
    CREAR_BOTON(0,710,420,90,60,"Draw",0,2,0);
    CREAR_BOTON(0,750,100,50,280,"->",0,3,0);
    CREAR_BOTON(0,0,100,50,280,"<-",0,4,0);
    CREAR_BOTON(0,154*3+90-35,420,70,60,"BIN",0,5,0);
    CREAR_BOTON(0,154*2+90-35,420,70,60,"IP",0,6,0);
    CREAR_BOTON(0,154*1+90-35,420,70,60,"CAL",0,7,0);
```

Con los nombres, ilustramos que función cumplen.

Dentro del bucle while, simplemente creamos la estructura de la pantalla a mostrar, pidiendo la actualización de los registros de toque y tag, y enviándole la información pertinente a las funciones que lo necesiten.

```
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();

        EVE_LIB_BeginCoProList();

//        EVE_CMD_NUMBER(100,50,28,EVE_OPT_CENTER,POSX);
//        EVE_CMD_NUMBER(700,50,28,EVE_OPT_CENTER,POSY);
        EVE_CMD_NUMBER(400,20,28,EVE_OPT_CENTER,PAG);

        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
        EVE_HANDLER_ACCION();
```

Este bucle tendrá algunas adiciones con el paso de la aplicación, pero estaría completo al 75% al recaer prácticamente todo el peso de programación en librerías al haber encapsulado lo máximo posible.

3.8 Variables globales únicas

```
uint16_t POSX, POSY;
uint32_t BufferXY,ui32SysClock;
uint8_t TAG=0;
uint8_t PAG=1;
struct Boton Botones[64];
#define OFFSETBOTMATRIX 12
#define OFFSETBOTLIBRE 38
```

Son variables que necesitaremos usar en EVE_API.c declarándolas como externas, creando así una vía de comunicación global entre main.c y API que podríamos usar para otras funcionalidades futuras.

En general son auto explicativas, y se sobreentienden si se ha leído la información expuesta arriba.

4 HID: TECLADO USB

Esta parte del proyecto es técnicamente invisible para el usuario, igual que cuando conectamos cualquier dispositivo a nuestros ordenadores y, como por arte de magia, reacciona a lo que pulsamos y movemos. Este punto tratará de esa magia: ¿Cómo funciona realmente un teclado?

4.1 Human Interface Device

Una descripción de dispositivos que, por norma general, nos rodea cada vez mas día a día. Cualquier dispositivo que usemos como personas humanas para introducir datos en una máquina suele corresponder al marco de los HID, desde los mandos más complejos para navegación aeroespacial a el ratón más simple que podamos encontrar. Si es verdad que, como es obvio, el grado de complejidad no es la misma pero no deja de ser una forma diferente de usar una serie de piezas disponibles para todos.

El HID es una normativa estándar creada por la organización (Universal Serial Bus, 2001), reconocido nombre, para proveer soporte de forma consistente a este tipo de dispositivos. Esta normativa cuenta con descripciones detalladas de las estructuras que deberemos usar para que el dispositivo que programemos funcione de forma predecible en el dispositivo al que lo conectemos. USB, como organización, actualiza esta normativa cuando se hacen necesarias adiciones a la normativa en cuanto a posibilidades funcionales (la última actualización trajo a los queridísimos y globales emojis).

Tras esta pequeña aclaración sobre el HID, revisaremos en este punto diferentes aspectos que necesitaremos usar de este origen: tablas, procesos, comandos…

4.1.1 Integración de HID con el launchPad

El EK-TM4C1294XL nos pone una serie de facilidades al proporcionar la librería de drivers correspondiente con una serie de ejemplos, librerías y manuales desarrollados por el fabricante (Texas Instruments, 2020) para facilitarnos las cosas a la hora de la programación. Esto es enormemente importante, ya que aligera mucho el proceso de puesta en marcha de la función y comprobación de funcionamiento con el resto de componentes ya que, si no hacemos las cosas de forma correcta, podríamos crear errores de temporización en la transmisión de datos de cualquiera de los módulos individuales.

Para empezar, en la librería original determinada para el uso del teclado se encontraban diferentes declaraciones clave de funcionamiento. Esta librería es posteriormente readaptada para el proyecto, así que iré indicando a continuación las diferentes estructuras y procesos que sea adoptaron en el proyecto exigidas por los estándares del protocolo.

4.1.2 Descriptores de características

Nombre del teclado, idiomas que maneja, fabricante… Es la ficha técnica a la que podríamos acceder desde dentro del Sistema Operativo correspondiente si tenemos alguna duda. Tras las declaraciones individuales, convergen todas dentro de un descriptor general más grande que las contiene para poder usarlo en el futuro.

El código se nos facilita debido a la previa declaración de códigos numéricos específicos en los que se basa, casi por completo, este tipo de programación. Códigos como “el idioma español” que originalmente sería el número hexadecimal C0A pasa a ser algo más legible como “USB_LANG_ES_MODERN”. Para más indicaciones recomiendo revisar las bibliotecas relacionadas con los dispositivos USB y HID proporcionadas por TexasInstruments, así como el manual USB completo que los acompaña. Sea como sea el código final corresponde al siguiente:

```
const uint8_t g_pui8LangDescriptor[] =
{
    4,
    USB_DTYPE_STRING,
    USBShort(USB_LANG_ES_MODERN)
};

const uint8_t g_pui8ManufacturerString[] =
{
    (17 + 1) * 2,
    USB_DTYPE_STRING,
    'G', 0, 'u', 0, 'z', 0, 'M', 0, 'a', 0, 'n', 0, 'z', 0, ' ', 0, 'I', 0,
    'n', 0, 'd', 0, 'u', 0, 's', 0, 't', 0, 'r', 0, 'y', 0, ' ', 0,
};

const uint8_t g_pui8ProductString[] =
{
    (16 + 1) * 2,
    USB_DTYPE_STRING,
    'E', 0, 'V', 0, 'E', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0, 'o', 0,
    'a', 0, 'r', 0, 'd', 0, ' ', 0, 'O', 0, 'N', 0, 'E', 0
};

const uint8_t g_pui8SeriailNumberString[] =
{
    (8 + 1) * 2,
    USB_DTYPE_STRING,
    '1', 0, '2', 0, '3', 0, '4', 0, '5', 0, '6', 0, '7', 0, '8', 0
};

const uint8_t g_pui8HIDInterfaceString[] =
{
    (22 + 1) * 2,
    USB_DTYPE_STRING,
    'H', 0, 'I', 0, 'D', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0,
    'o', 0, 'a', 0, 'r', 0, 'd', 0, ' ', 0, 'I', 0, 'n', 0, 't', 0,
    'e', 0, 'r', 0, 'f', 0, 'a', 0, 'c', 0, 'e', 0
};

const uint8_t g_pui8ConfigString[] =
{
    (26 + 1) * 2,
    USB_DTYPE_STRING,
    'H', 0, 'I', 0, 'D', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0,
    'o', 0, 'a', 0, 'r', 0, 'd', 0, ' ', 0, 'C', 0, 'o', 0, 'n', 0,
    'f', 0, 'i', 0, 'g', 0, 'u', 0, 'r', 0, 'a', 0, 't', 0, 'i', 0,
    'o', 0, 'n', 0
};

const uint8_t * const g_ppui8StringDescriptors[] =
{
    g_pui8LangDescriptor,
    g_pui8ManufacturerString,
    g_pui8ProductString,
    g_pui8SeriailNumberString,
    g_pui8HIDInterfaceString,
    g_pui8ConfigString
};
```

4.1.3 DeviceDescriptor

Tras esta lista descriptora o ficha técnica del dispositivo, pasamos a declarar el propio dispositivo mediante un descriptor de dispositivo. Este descriptor de dispositivo tiene una serie de campos que definirán a nivel funcional el producto final, como el consumo o el ID de producto para detectar el teclado. Estos campos son:

- El ID del vendedor: TexasInstruments te recomienda usar el suyo propio que viene programado en la librería de IDs para USBs.
- El ID del producto: esto define de qué manera entiende el PC el dispositivo, en nuestro caso pondremos el de teclado: 0x1
- Corriente máxima consumida por el dispositivo: normalmente dejamos la estándar para este tipo de productos, 500mA, salvo en casos específicos que necesitemos una cantidad diferente por tener un requerimiento mayor debido a LEDS, pantallas integradas…
- Flags de características de corriente: Según los bits activados, el sistema tendrá unas características u otras. Dejaremos los que recomienda TI.
- Puntero a manejador de eventos: será una estructura que explicaremos más adelante.
- Puntero a la estructura misma: esta estructura se inicializará con un nombre. Esto lo usaremos para crear un puntero a la información de esta estructura y enviarlo para que sirva como ID de ida y vuelta para el sistema, para que sepa de que dispositivo son las peticiones y hacia donde tiene que enviar las respuestas.
- Descriptor de característica: el del anterior punto.
- Numero de campos en el descriptor de características: la única forma de que varie es añadiéndole multiplicidad de idiomas.

4.1.4 Manejador del Teclado

Es un manejador del estado en relación con el puerto USB, y no un manejador de las acciones del teclado como podrían ser las pulsaciones de teclas, es decir, un manejador de la conexión del dispositivo. Al ser así, este se tendrá que encargar de monitorizar los diferentes estados por los que el dispositivo podría pasar en una conexión, transmisión, desconexión, etc.

Para la monitorización del estado, se declara primero una variable que marcará el estado actual del teclado. Se compondrá de un enum, que contendrá los 3 estados posibles del teclado: No configurado, preparado y a la espera, y enviando datos. A nivel código quedará:

```
volatile enum
{

    STATE_UNCONFIGURED,

    STATE_IDLE,

    STATE_SENDING
}
g_eKeyboardState = STATE_UNCONFIGURED;
```

Como es obvio, inicializaremos la variable como no configurada. Tras el proceso apropiado, esto cambará e irá alternándose entre los otros dos estados posibles.

Tras la declaración de la variable indicadora, pasamos a declarar la función manejador. Esta función manejador se explica en detalle en el manual de uso de USB proporcionado por TI, en la que se justifican los inputs de la función y los estados totales. La función es la siguiente:

```
uint32_t KeyboardHandler(void *pvCBData, uint32_t ui32Event, uint32_t ui32MsgData,
                 void *pvMsgData)
{
    switch (ui32Event)
    {

        case USB_EVENT_CONNECTED:
        {
            g_bConnected = true;
            g_bSuspended = false;
            break;
        }

        case USB_EVENT_DISCONNECTED:
        {
            g_bConnected = false;
            break;
        }

        case USB_EVENT_TX_COMPLETE:
        {

            g_eKeyboardState = STATE_IDLE;
            break;
        }

        case USB_EVENT_SUSPEND:
        {
            g_bSuspended = true;
            break;
        }

        case USB_EVENT_RESUME:
        {
            g_bSuspended = false;
            break;
        }

        case USBD_HID_KEYB_EVENT_SET_LEDS:
        {

            MAP_GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,
                             (ui32MsgData & HID_KEYB_CAPS_LOCK) ? GPIO_PIN_0 :
                             0);

            break;
        }

        default:
        {
            break;
        }
    }

    return(0);
}
```

Es un handler básico en el que se limita a cambiar una serie de variables troncales del proceso del teclado, como indicar si se encuentra apagado, suspendido o conectado y los estados generales.

Como pequeño extra de funcionamiento, se proporciona la típica función en los teclados de encender un led cuando se tiene el CAP_LOCK activado, comprobando si el mensaje es la susodicha tecla y cambiando el estado del led.

La combinación de estas estructuras será la base de cualquier posible futuro proyecto en el que se pretenda usar un HID, siendo los valores anteriores modificables para adaptarlos a un ratón, un mando, un dispositivo mixto

o cualquier cosa que el lector pueda pensar. Para eso solo será necesario que se incluya en los manuales HID de USB, aunque no estén programados directamente por TI ya que, me parece pertinente recordar lo siguiente, no es necesario escribir los comandos adaptados proporcionados por la librería para que el proyecto funcione. Mientras la persona encargada de la declaración y conversión sea conocedora de los comandos numéricos, el proyecto tendrá las capacidades que el programador le proporcione.

4.2 USB en el EK-TM4C1294XL

De cara al trabajo que se hace en el launchPad, es necesario inicializar previamente el modo USB y, tras esto, configurar las variables y llamadas correspondientes para indicar que el dispositivo será de tipo HID.

4.2.1 Atajos de configuración, PinoutSet

Las librerías proporcionadas tienen atajos de funcionamiento que son muy convenientes a la hora de inicializar sus puertos. Es el caso de la función PinoutSet, que permie, de forma rápida y cómoda, inicializar los pines, registros y variables internas correspondientes al uso del USB y del ETHERNET, extra que será útil más adelante en la explicación. La función depende de dos variables booleanas que se pasan como input a la función, una para cada puerto correspondiente. Depende de las que sean activadas, un IF impedirá o no que la función entre a configurar los pines correspondientes.

La configuración del USB corresponde a los pines PB0-1 y PL6-7 como analógicos de USB, y PD6 como digital de USB y puerto USB. Además de la inicialización del PD6, se necesita hacer una serie de ajustes en el puerto D con la intención de que las configuraciones puedan realizarse y no las bloquee la protección del puerto. Como extra se puede asignar un PIN I/O como detector de fallos en corriente, que la función establece en el PQ4.

En el caso de que se mande un FALSE la bool correspondiente al USB, se configura el PD6 como PIN I/O estándar y la función se asegura de que cualquier módulo relacionado con el USB es apagado o reconfigurado.

A nivel ETHERNET, se explicará en el punto correspondiente para mantener la estructura del documento.

Pese a que la función permite controlar exclusivamente la decisión de inicialización de USB y ETHERNET, ésta inicializa algunos módulos adicionales como los LEDS o la UART0, pensando en su utilización en interacciones con los dos puertos controlados, pero siendo totalmente independientes del valor de las variables de entradas y, significando esto, su inicialización cada vez que esta función es llamada.

Por cerrar, las variables pasadas dentro del proyecto serán ambas verdaderos lógicos debido al uso de ambos puertos relacionados. Para no reestablecer pines y puertos en una segunda ejecución del código, parte de este proceso será del ETHERNET, explicado en puntos posteriores.

4.2.2 Modos de USB, USBStackModeSet

Otra función importante a la hora de la configuración en el launchPad que facilita la librería es USBStackModeSet. Ésta permite indicar el modo de USB que tiene el proyecto, siendo este el modo de dispositivo. Los modos disponibles son:

- eUSBModeDevice: modo de dispositivo. Correspondiente a la indicación de que nuestro proyecto será comprendido como un dispositivo conectado por USB en el sistema. Este modo tiene la implicación de que el bus usado para la comunicación será monitorizado de forma continua en pos de leer y recibir mensajes. Por último, para que esta declaración sea correcta, tiene que existir las declaraciones de ID del dispositivo ya que, si esto no es así cuando se termine la configuración, el dispositivo no se configurará.
- eUSBModeHost: configuración como receptor de dispositivos. Si el proyecto consistiera en la conexión de un teclado a nuestro launchPad y escribir con este en un serial, este sería el modo seleccionado. Tiene

la misma implicación del bus que el anterior y, por el contrario, no tiene que existir ningún ID de producto para la configuración, debido a que este recibirá los ID y no los dará.

- eUSBModeOTG: dispositivo on-the-go. Son dispositivos capaces de actuar como host y dispositivo, pudiendo abrir una sesión master/slave a voluntad. Está pensado para dispositivos como móviles, dándoles flexibilidad a la hora de decidir su actuación a la hora de ser conectados por USB.
- eUSBModeNone: place holder para indicar que el dispositivo USB aún no está configurado.
- eUSBModeForceHost y eUSBModeForceDevice: similares a los dos primeros, pero con la diferencia de que la gestión de buses e IDs pasa a estar fuera del manejo del controlador USB programado, dándole el control a la otra parte del sistema, dispositivo o receptor respectivamente.

4.2.3 Funcion final, Inicia_Teclado

Se encapsulará en esta función todo lo anteriormente introducido junto con declaraciones e inicializaciones de variables útiles para el uso y la conversión del dispositivo USB HID:

1. La declaración de la información comienza con la creación de la variable ui32PLLRate, que es necesaria para almacenar datos relevantes en la temporización de la búsqueda de información necesaria para el protocolo USB.
2. Tras esto, se declara la función previamente explicada de PinoutSet, con sus dos variables de entrada a true.
3. Se inicializan las variables previamente declaradas de conexión y suspensión del dispositivo. Esto es lógico debido a que cuando el dispositivo se enciende, el dispositivo ni estará automáticamente conectado y mucho menos suspendido, sino no configurado.
4. Se declara la función USBStackModeSet con su modo dispositivo indicado en las variables que se le ofrecen.
5. Tras lo anterior, se introducen una serie de funciones que establecen el valor de la variable inicializada en el punto 1. Posteriormente, se le da la información del reloj y la variable inicializadas al gestor de características de la conexión USB.
6. Se indica que el dispositivo declarado es un HID de forma teclado mediante el uso de la función USBHIDKeyboardInit, a la que se le pasa un puntero hacia la declaración de la estructura básica desarrollada con las exigencias del HID.
7. Por último, se configura la temporización de las comprobaciones de sistema, y se habilitan estas junto con el uso de interrupciones

Tras este proceso, el launchPad estará a punto para su uso como teclado. La declaración del código queda de la siguiente forma:

```
extern uint32_t ui32SysClock;
void Inicia_Teclado(){

    uint32_t ui32PLLRate;

    PinoutSet(true, true);

    g_bConnected = false;
    g_bSuspended = false;

    USBStackModeSet(0, eUSBModeDevice, 0);

    SysCtlVCOGet(SYSCTL_XTAL_25MHZ, &ui32PLLRate);
    USBDCDFeatureSet(0, USBLIB_FEATURE_CPUCLK, &ui32SysClock);
    USBDCDFeatureSet(0, USBLIB_FEATURE_USBPLL, &ui32PLLRate);

    USBDHIDKeyboardInit(0, &g_sKeyboardDevice);

    MAP_SysTickPeriodSet(ui32SysClock / SYSTICKS_PER_SECOND);
```

```
    MAP_SysTickIntEnable();
    MAP_SysTickEnable();

}
```

4.3 Teclado virtual, interacción del teclado mediante la pantalla, EVE_API.c II

Tras la realización de toda la configuración, el sistema está preparado para enviar al ordenador que se conecte los comandos determinados para indicar la pulsación e las teclas pertinentes. Para esto será necesario el entendimiento de la estructura que se seguirá para el envío de información, así como de los códigos para las teclas individuales y sus modificadores.

Como se indica en el título, los cambios siguientes se realizarán en las librerías de la pantalla, debido a que será a través del módulo que se realizará la interacción usuario-teclado. Esto no es necesario, y se puede agrupar las funciones y variables mediante referencia externas entre las librerías dedicadas, pero al compartir la mitad del peso entre ambas, se tomó la decisión de diseño de realizarlo así.

4.3.1 Códigos HID de teclas físicas, KeypUsageCodes

Como todo lo referenciado a HID dentro del proyecto, las librerías proporcionadas facilitan la programación de forma sustancial. Con éstas, se muestran las teclas de forma limpia, transcritas de su código numérico al nombre de variable correspondiente que indica su la letra asignada, dando así la posibilidad de programar sin tablas de transcripción delante.

Pese a tener esa posibilidad, el proyecto se queda en un estado medio en el que obtenemos los códigos de las librerías, ya que nos ahorra buscarlo en el manual, pero se pasan como códigos numéricos. Esto es debido a la necesidad de exportar las librerías correspondientes a la transcripción de los códigos y los problemas que traía esta inclusión.

Para empezar, el proyecto necesita una serie de combinaciones que funcionen de placheholders para las teclas que se vayan creando con los modos de edición automática y libre. Estos botones, resultado de la ejecución de la función pertinente, se crean con la función explicada anteriormente de CREAR_BOTON, y se les asigna un número correspondiente a cuantos botones hay dentro de la pantalla por delante de él, es decir, si hay 6 botones en pantalla cada uno tendrá un numero único del 1 al 6. El número asignado puede ser usado como índice en una tabla de asignaciones predefinidas únicas, teniendo como resultado una asignación única a cada tecla individual.

Para la tabla necesaria se escoge usar una combinación del LEFT_CTRL, código de referencia 0x01, LEFT_ALT, código de referencia 0x04, y los números y signos del NUMPAD, con códigos que van del 0x54 al 0x67. Estas opciones se recombinarán para crear 4 secciones: CTRL + números NUMPAD, CTRL+ ALT + números NUMPAD, CTRL + símbolos NUMPAD y CTRL+ ALT + números NUMPAD. Reflejados en código de la siguiente forma:

```
static const int8_t EveKeypUsageCodes[][2]={
{0x01,0x59},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_1},
{0x01,0x5A},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_2},
{0x01,0x5B},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_3},
{0x01,0x5C},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_4},
{0x01,0x5D},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_5},
{0x01,0x5E},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_6},
{0x01,0x5F},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_7},
{0x01,0x60},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_8},
{0x01,0x61},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_9},
{0x01,0x62},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_0},

{0x01|0x04,0x59},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_1},
{0x01|0x04,0x5A},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_2},
{0x01|0x04,0x5B},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_3},
{0x01|0x04,0x5C},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_4},
{0x01|0x04,0x5D},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_5},
```

```
{0x01|0x04,0x5E},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_6},
{0x01|0x04,0x5F},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_7},
{0x01|0x04,0x60},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_8},
{0x01|0x04,0x61},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_9},
{0x01|0x04,0x62},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_0},

  {0x01,0x54},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_SLASH},
  {0x01,0x55},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_STAR},
  {0x01,0x56},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_MINUS},
  {0x01,0x57},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_PLUS},
  {0x01,0x63},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_PERIOD},
  {0x01,0x67},//{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_EQUAL},

{0x01|0x04,0x54},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_SLASH},
{0x01|0x04,0x55},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_STAR},
{0x01|0x04,0x56},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_MINUS},
{0x01|0x04,0x57},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_PLUS},
{0x01|0x04,0x63},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_PERIOD}
,
{0x01|0x04,0x67},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,HID_KEYB_USAGE_KEYPAD_EQUAL},
  };
```

Como podemos observar, se inicializa un array con 2 columnas: la primera para las teclas modificadoras y la segunda con las teclas principales. Este modo de trabajar se verá recompensado después a la hora de trabajar con el comando que envía las pulsaciones al ordenador mediante el módulo USB.

Como última cosa a comentar, la declaración de este array la contiene la cabecera EVE.h y no el EVE_API.c por motivos de limpieza más que funcionales, ya que esta estructura no será usada por ningún otro módulo externo.

4.3.2 Comandos de transmisión, EVE_ACTUADOR

Por poner en contexto al lector, varios puntos atrás, en el manejador de acciones de la pantalla, le asignamos un comando y una acción al hecho de pulsar una tecla, junto con su función. Esta función se dejó para explicarla en este momento.

EVE_ACTUADOR será la responsable de ejecutar los comandos de comunicación para indicar el estado de las teclas que el usuario vaya seleccionando en la pantalla táctil. Para esto, nos apoyaremos en la tabla creada anteriormente y en una variable de entrada que se corresponderá al TAG del botón, de cara a leer la información almacenada en la estructura principal del proyecto dentro de los campos correspondientes en BotAccion.

Para dar forma a la función final, se tienen en cuenta las dos situaciones posibles del botón: con placeholder y con tecla asignada. Esto se diferencia mediante el uso del último bit de la variable Mod, que tiene como única función indicarnos si aún no se ha modificado la estructura, 0, o si por el contrario el usuario ya ha definido su valor, 1. El resto de la explicación para el caso en el que el usuario ha podido determinar la tecla mediante el servidor web será desarrollado en el apartado de éste.

De cara a la situación de los placeholder, usaremos el campo comando como un indicador que meteremos dentro de la tabla para extraer información de esta y pasar los valores hacia la función de envío. Esta función tendrá un campo para modificadores y un campo para valores principales, por lo que sus valores serán los correspondientes a:

- Modificador de tecla: `EveKeypUsageCodes[Botones[tag-1].BotAccion.comando-1][0]`
- Tecla principal: `EveKeypUsageCodes[Botones[tag-1].BotAccion.comando-1][1]`

4.3.3 Comando de envio, KeyStateChange

Las indicaciones e instrucciones para los diferentes comandos de librerías explicados en el manual proporcionado, explican como uno de los comandos incluidos, USBDHIDKeyboardKeyStateChange, encaja

perfectamente dentro de las peticiones del proyecto. Este comando puede indicarle al ordenador que teclas "individuales" están pulsadas y en reposo, junto con sus modificadores. Todas las teclas empiezan en reposo por defecto, y toda tecla que sea modificada se mantendrá modificada hasta que se retoque con el mismo comando.

Para su funcionamiento, es necesario proporcionarle:

- El puntero al dispositivo HID para recuperar su funcionamiento e información.
- Las Teclas modificadoras: CTRL, ALT, SHIFT…
- La tecla principal
- Si estamos pulsándola, indicado con un 1, o si está en reposo, con un 0

Tras esto, se tiene todo listo para escribir las líneas de código que permitirían la interacción con el ordenador. En el caso del proyecto:

```
  void EVE_ACTUADOR(uint8_t tag){
  USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][0],EveKeypUsageCodes[Botones[tag-1].BotAccion.comando-
1][1],1);
          SysCtlDelay(20*40000);

USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,0,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][1],0);
          SysCtlDelay(20*40000);
  }
```

Tras todo esto, el proyecto debería estar en un punto en el que la pantalla es capaz de interactuar con el ordenador mediante las teclas que generemos con los modos de edición. Las funcionalidades faltantes se incluirán con el servidor WEB para, posteriormente, introducir mejoras de calidad para el usuario con el uso de la FLASH.

5 PERSONALIZACIÓN: SERVIDOR LOCAL WEB

Este punto recogerá todos los pasos necesarios para crear el servidor web local que se comunicará con nuestro dispositivo para modificar las diferentes características incluidas.

Para comenzar, no existe, o no he encontrado, ningún documento oficial que desarrolle el proceso, así como las librerías dedicadas, para el correcto funcionamiento de esta característica. Todo lo que proporciona Texas Instruments son los 4 ejemplos con funcionalidades diversas enfocados en la conexión ETHERNET, sin más notas aparte que los comentarios hechos en sendos archivos.

Partiendo de este punto inicial, añadiremos la dificultad de que para completar este punto se necesitan unas herramientas añadidas que no hemos tenido la necesidad de usar en los módulos desarrollados anteriormente como son los idiomas de programación HTML5, CCS y JavaScript. Por lo que respecta al nivel requerido, sin ningún tipo de información proporcionada como era al inicio de este bloque en el desarrollo del proyecto, se necesitaría un nivel medio-alto de JavaScript de cara a entender y optimizar esta parte del proyecto. Espero que este documento le sirva al lector para aclarar dudas y problemas que este proyecto tuvo a causa de la nula información existente sobre esta vía de comunicación.

Como último añadido de dificultad, entender cómo funciona la conexión a internet no es un problema trivial. Así como el entendimiento de los diferentes protocolos, métodos de comunicación y procedimientos que conviven entre sí a la vez en este campo. El ejemplo que usó este módulo del proyecto como base tenía diversos canales de comunicación. Se escogió el que se ha incluido debido a que es una metodología que usa el archivo de JavaScript y el archivo de C dedicado al servidor como bases de comunicación, mientras que la ruta alternativa pedía entender la forma en las que los puertos de internet almacenaban información, en conjunto con otro idioma extra.

Con respecto a HTML5 y CCS, los idiomas son más simples, cortando así su dificultad y estando pensados para usarlos con una finalidad específica: estructurar y decorar, respectivamente. Prácticamente no se mencionará nada de estos archivos salvo que algún archivo de JavaScript lo requiera.

5.1 Inicialización del modulo

Antes de comenzar, se vuelve a mencionar que lo que se explicará a continuación no tiene ningún tipo de documentación de donde sacar la información o que no se contó con ella en el proceso de creación de esta pieza del proyecto. Se darán las explicaciones acordes a lo que se ha podido investigar haciendo una regresión en las funciones hacia las capas inferiores.

5.1.1 Función Principal, Init_Ethernet_Server

La función que encapsula casi todo el proceso de arranque de las funcionalidades ETHERNET y el servidor local web. Ese casi es el comando anteriormente explicado PinoutSet, que tenía implicaciones dentro de este campo y que explicaremos antes de entrar en la función principal.

5.1.1.1 PinoutSet de ETHERNET

Técnicamente, lo menos importante de todo el proceso ya que, en contraste con la declaración para USB, esta función solo tiene el objetivo de iniciar LEDs de información para el puerto ETHERNET, configurando el PF0 y PF4 como tal. Cuando esta variable es falsa, se configuran los dichos LEDs como estándares y se apagan por defecto para no crear órdenes excedentes no útiles.

Se declarará la función de inicio principal tras esta para no crear problemas de compatibilidad, aunque en principio no debería haber ningún problema si no se hace así.

5.1.1.2 Función principal

Como información previa que habría que conocer tenemos el uso de la FLASH, debido a que el proceso que

ejemplo sigue para la inicialización del módulo señala a las zonas de memoria correspondientes a USER1 y USER2 de la FLASH somo los contenedores de la dirección MAC del launchPad. La gestión de este tipo de memoria se explicará más adelante y lo único que tenemos que ver ahora mismo es que se recuperan los datos de la zona mencionada y se pasan a una variable que contiene la información necesaria de la MAC de la EK-TM4C1294XL que usaremos para inicializar y comunicarnos con el servidor.

Tras esto, se inicializa el gestor lwIP que se encarga de la gestión de IP y el protocolo TCP. A ésta le pasaremos el reloj, la MAC recuperada y la variable correspondiente a la adquisición de IPs mediante DHCP.

Tras esto preparamos el entorno para la inicialización del servidor web mediante la ejecución de las funciones con raíz LOCATOR, en las que además le pasamos la MAC y el nombre del servidor.

Tras esto, inicializamos el servidor local. La estructura de archivos, así como los pasos para crear este servidor local serán especificados más adelante debido a su implicación con la comunicación placa-servidor.

La última cosa es indicar el orden de prioridades de interrupción, dejando por encima del resto a las relacionadas con el puerto ETHERNET. Esto ocurre, según se ha investigado, porque la temporización usando Internet es crítica, por lo que si se interrumpiese la transmisión o recepción de estos datos en mitad de una conexión se podrían producir problemas graves como la pérdida total de conexión o falla total del sistema. En el transcurso de este proyecto se han producido estos casos, por lo que se recomienda seguir este paso.

Tras esta explicación se tendría lo siguiente:

```
void Init_Ethernet_Server(){
    uint32_t ui32User0, ui32User1;
    uint8_t pui8MACArray[8];
    SysCtlMOSCConfigSet(SYSCTL_MOSC_HIGHFREQ);
    MAP_SysTickPeriodSet(ui32SysClock / SYSTICKHZ);
    MAP_SysTickEnable();
    MAP_SysTickIntEnable();
    MAP_FlashUserGet(&ui32User0, &ui32User1);

    pui8MACArray[0] = ((ui32User0 >>  0) & 0xff);
    pui8MACArray[1] = ((ui32User0 >>  8) & 0xff);
    pui8MACArray[2] = ((ui32User0 >> 16) & 0xff);
    pui8MACArray[3] = ((ui32User1 >>  0) & 0xff);
    pui8MACArray[4] = ((ui32User1 >>  8) & 0xff);
    pui8MACArray[5] = ((ui32User1 >> 16) & 0xff);

    lwIPInit(ui32SysClock, pui8MACArray, 0, 0, 0, IPADDR_USE_DHCP);

    LocatorInit();
    LocatorMACAddrSet(pui8MACArray);
    LocatorAppTitleSet("TFG Web Program");

    httpd_init();

    MAP_IntPrioritySet(INT_EMAC0, ETHERNET_INT_PRIORITY);
    MAP_IntPrioritySet(FAULT_SYSTICK, SYSTICK_INT_PRIORITY);

}
```

5.1.1.3 Comprobador en bucle, CHECK_FLAGS

Rápido y sencillo, es un comprobador activo que busca flags relacionadas con el puerto ETHERNET. Va en el bucle principal, en contra parte a la función de inicialización, que va antes de este.

```
void CHECK_FLAGS(){
if(g_ulFlags){
            HWREGBITW(&g_ulFlags, FLAG_TICK) = 0;
}
}
```

5.1.2 Extras de la función ejemplo

En este apartado se han incluido una serie de funciones que pese a no referenciarse dentro de las consideradas vitales para el funcionamiento, son funciones con raíces profundas dentro de los archivos de ejemplo y que hace difícil su exclusión sin forzar un error en otro módulo si esencial. Esto hace que se decida dejar tras limpiar las funcionalidades no necesarias dentro del proyecto incluidas en el ejemplo.

5.1.2.1 lwIPHostTimerHandler

La primera de estas era la función manejadora lwIPHostTimerHandler, que pese a parecer una función necesaria en funcionamiento por su nombre, originalmente no era más que una manera de comunicar al usuario mediante el uso del puerto serie del estado actual de configuración de la IP del servidor.

5.1.2.2 AnimTimerIntHandler

Originalmente era la función que se encargaba de la velocidad de animación de apagado y encendido de un LED mediante el control por el servidor. Podría eliminarse si no fuera porque el uso de esta hubiese forzado el cambio en el fichero base Starup_ccs.c dentro de su tabal de datos fundamental y que, en el caso de borrar el dato asociado, produzca una falla grave en el proyecto dejando todos los módulos fuera de juego.

Hablando a nivel de autor, esto me hizo preguntarme si realmente había algo que no estuviese viendo y esa función si se utilizase para sincronizar algo, pero no he encontrado ni una sola referencia en código de esta, por la que planteo la hipótesis de que esta se puede eliminar sin problemas si se quita justo lo necesario y arreglando cualquier referencia externa.

5.2 Creación y comunicación del servidor

Este apartado engloba la parte de HTML5, CCS y JavaScript del proyecto y como se crea, de parte del servidor, el canal de interacción y comunicación con el launchPad.

5.2.1 Bases de creación web

Se incluye este punto de cara a sentar unas bases comunes sobre las que se redactará lo hecho en el proyecto.

Hay diversas maneras de hacer una página web, la seleccionada para este proyecto consiste en: estructurar la idea con HTML5, decorar y organizar la estructura con CCS, y crear las funcionalidades de la página con JavaScript. Sólo mediante la interacción de estos tres idiomas en sus diferentes archivos será posible que la página web creada dentro del servidor local sea y actúe como se planease en un principio.

Como añadido, la carpeta contenedora se tendrá que ver sometida a un archivo que nos proporciona TI: el makefsfile. El proceso se indica en el ejemplo y se hace para cargar los cambios que se produzcan en el código de la placa referido al servidor, es decir, el launchPad no lee directamente estos tres idiomas, sino una transcripción.

5.2.1.1 HTML5

A nivel de archivos, cada archivo .htm dentro del contenedor del servidor, es decir, la carpeta contenedora del servidor, corresponderá a una página que podremos cargar individualmente en nuestro navegador si accedemos a su ruta. Estos archivos se dividen en:

- Básicos con nombres específicos: como podrían ser “index.htm” para el archivo principal desde el cual nos iremos moviendo al resto, “404.htm” para indicar una búsqueda de archivo fallida o “perror.htm” para casos de errores en el servidor.
- Archivos propios: como los archivos editores, “editorBot.htm” y “editorPag.htm”, para la modificación de valores objetivo, o “instrucciones.htm” para indicar el funcionamiento de la página.

Todos estos archivos podrán ser comunicados internamente mediante links de HTML5 o externamente mediante

acciones onClick de JavaScript, incluso conteniendo dentro de un archivo a otras páginas, como se hará finalmente en el resultado final usando a index de contenedor para incluir al resto.

Otra parte importante de HTML5 es que es una programación por bloques, creando diferentes entidades una tras otra que después trataremos con el resto de idiomas para indicar su aspecto y habilidades. Lo que sí hace HTML5 para facilitar las vías de comunicación entre idiomas es asignar una serie de características para la fácil detección de elementos, como pueden ser: ID únicos, clases, cajas preestablecidas como , <a> y <p>, etc.

Tras estructurar la información mediante las posibilidades que nos brinda el idioma, pasamos a CCS para decorar y organizarlos dentro de la pantalla.

5.2.1.2 CCS

Es un idioma que sirve para propósitos puramente estéticos mediante la configuración de los bloques creados con HTML5, conteniendo todos los estilos en un solo archivo, normalmente. No tiene ningún uso a nivel funcional, pero teniendo en cuenta que una gran parte de la aceptación de una página web se hace por motivos visuales, es una pieza igual de importante que cualquier otra.

No hay mucho más que mencionar aquí salvo que preparamos la página index para resultar el contenedor donde se mostrará en su parte central e inferior las páginas pertinentes que se vayan requiriendo.

5.2.1.3 JavaScript

JavaScript corresponde a la parte funcional de esta triada de idiomas que se encarga de la creación de la página. Cualquier tipo de función como podría ser la creación de un contador, un reloj, un submitter o un dispositivo interactivo, requiere del uso de este idioma.

Se suelen crear los que se precisen, exportando después las funciones dentro del HTML5 mediante un comando de inclusión propio. Por lo tanto, se asemejan enormemente a las librerías de C donde se contienen funciones encapsuladas para usar en el archivo principal.

5.2.2 JavaScript en el Proyecto

Entender el funcionamiento del modus operandi de la comunicación en estos archivos ha sido la parte más complicada de todo el proyecto sin lugar a dudas. Enfrentar esto con un conocimiento nulo de este idioma ha hecho que otras opciones que seguramente fueran mejores soluciones para la funcionalidad requerida se hayan tenido que descartar al no ser capaz de lograr una comunicación consistente web-placa. Aun así, se considera que la solución aportada es satisfactoria debido a que se ha proporcionado al usuario todas las herramientas planteadas desde un inicio, sin dejar ninguna funcionalidad atrás.

En nuestro caso los archivos de este idioma serán el responsable de la inclusión de páginas en index, de la comunicación con el launchPad y de plasmar los datos que este le indique al usuario mediante la página. Todas estas funcionalidades se contendrán dentro de un único archivo con nombre igual al idioma.

El otro archivo de este idioma, “javascript_load.js”, se encargará de cargar las páginas correspondientes al inicio del servidor, para proporcionar la vista compacta desde un primer momento. No será necesario volver a mencionar este archivo ya que es un espejo de la primera función incluida en el otro. Cualquier cambio en esa declaración, tendrá que ser correspondida aquí.

5.2.2.1 OnClick y OnLoad, cargando la información y moviéndonos por los archivos

Estas funciones corresponden a la carga de archivos en dos situaciones:

- OnLoad cargará el archivo definido en la declaración cuando se produce la primera llamada a la página y se carga la web en el navegador. Se usa para inicialización de funciones, y en este caso se indican las funciones onClick para la navegación del usuario por los diferentes archivos. La función cargará en una ventana dentro de index estos archivos.

- OnClick realizará la función pertinente cuando se seleccione en un elemento con un ID determinado, indicado en la declaración de la función. Estos ID se referencian a los puestos en HTML5 a los bloques. Como uso extra de estas funciones, se encargan de ejecutar la función de obtención de datos para recuperar estados de variables que se usan dentro de la página cuando nos movemos hacia esta.

Tras establecer estas funciones, se tiene un servidor por el que el usuario puede moverse a voluntad, que está visualmente refinado y que reacciona ante las interacciones.

5.2.2.2 Planificando las conexiones

Antes de introducirnos en la función de comunicación y en la estrategia de interacción que esta sigue, sería positivo para el proyecto recapitular y ver de cuantas variables se quiere que el usuario disponga para interactuar dentro del servidor.

Como objetivos se ha planteado en primer lugar la modificación de las teclas en su comando designado, significando esto:

- Capacidad para decidir la página de la tecla objetivo.
- Capacidad para decidir el número de la tecla objetivo.
- Capacidad para decidir el modificador que se aplicará a la tecla entre shift, alt, control o una combinación de estos.
- Capacidad para elegir la tecla principal que se aplicará entre las del teclado principal.
- Capacidad de decisión de cuando se efectúa el cambio, en caso de que se desee cambiar al final.

El primer objetivo tiene vinculadas un total de 5 variables posibles a modificar.

Como segundo objetivo de personalización se tiene la modificación del color del fondo y las teclas para una pantalla determinada, esto implicaría:

- Capacidad para decidir la página objetivo.
- Capacidad de elegir entre si modificar elementos o fondo.
- Capacidad de elegir colores.

Aunque en este caso no es tan directo el cálculo de variables debido a la naturaleza del problema, se concluye que, si la variable de página es igual a la anterior, se tendrá 1 variable de elección más 3 variables de selección de color, una asignada a cada parámetro de los colores RGB por los cuales se rige la pantalla.

Pero esto se incrementa al adoptar la solución propuesta. Si los estados los cambia un botón, será necesario un botón por cada estado posible de la variable, lo que significa que:

- Numero de pantalla: dividido en siguiente pantalla y anterior pantalla.
- Numero de botón: dividido en siguiente botón y anterior botón.
- Modificador de tecla: 3 teclas para los 3 modificadores, además de una tecla para limpiarlos.
- Tecla: una tecla por cada tecla básica de teclado que queramos incluir, siendo esto alrededor de 65 teclas
- Actualizador: se queda en un botón.
- Selector fondo/elemento: podemos dejarlo en un botón que invierta el estado.
- Rojo, Verde y Azul: la forma más pensada se corresponde al de escribir un numero en un teclado numérico, que corresponderá con el valor R, G o B del color de la pantalla, por lo que serían 10 teclas más.

Además, tendremos una variable extra no asociada a botones para mostrar el valor actual en el launchPad y dar así feedback al usuario.

Esto dejaría el problema en más o menos 110 posibilidades, haciendo que sea bastante prioritario la optimización de la gestión, creación y recepción de los mensajes por parte de ambos extremos del canal.

5.2.2.3 Estructura de comunicación en ambos sentidos por parte del servidor

Cuando se estudia el ejemplo original se reconoce una estructura bien establecida que, si nos paramos a leer, contiene variables que parecen de la placa. Al desarrollarla, veremos cómo esta estructura es lo único que necesitaremos a este lado del canal para empezar a enviar y recibir información.

La estructura corresponde con una función declarada en HTML5 como onClick al botón correspondiente. Cuando el botón es pulsado, la función se ejecutará y hará lo que le indiquemos. Y aquí está el kit de la cuestión, que tenemos que hacer para que la conexión sea exitosa.

La función comienza con la creación de otra función interna llamada ToggleComplete que, tras unas comprobaciones que pueden parecer un poco arcaicas, menciona a la variable contenedor no asociada al botón y que se encarga de dar feedback sobre el estado al usuario. Se usará de ejemplo la función que se encarga de seleccionar si lo personalizamos los elementos o el fondo:

```
function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("WindButState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }
```

Tras esto la función pasa a comprobar si existe una petición de HTTP o algún objeto activo para asignarlo a la variable troncal del proceso, lo que se supone que se hace de cara a mandar la información con un identificador o a un sitio determinado.

```
if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
```

Despúes de esto, se indica una línea de texto que se publicará al otro lado de la conexión. Esto es realmente importante ya que se está declarando que la comunicación entre el dispositivo y el servidor ocurre de manera que se publica un mensaje que el otro sistema busca activamente, y al encontrar este mensaje el sistema remoto decide que hacer, modificando una variable asignada al campo de feedback del usuario, que el servidor recupera para hacer la asignación nueva.

Finalmente, y tras la publicación, se ejecuta la función previamente establecida. Esto hace que se especule sobre que esta función se encarga de recuperar el valor nuevo de la variable determinada.

```
if(req)
  {
    req.open("GET", "/WindBut" + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
```

Tras la declaración de lo anterior quedará una función de la forma:

```
function WindBut()
{
  var req = false;
```

```
function ToggleComplete()
{
  if(req.readyState == 4)
  {
    if(req.status == 200)
    {
      document.getElementById("WindButState").innerHTML = "<div>" +
      req.responseText + "</div>";
    }
  }
}

if(window.XMLHttpRequest)
{
  req = new XMLHttpRequest();
}
else if(window.ActiveXObject)
{
  req = new ActiveXObject("Microsoft.XMLHTTP");
}
if(req)
{
  req.open("GET", "/WindBut" + Math.random(), true);
  req.onreadystatechange = ToggleComplete;
  req.send(null);
}
```

Ésta será la base para todas las variables, aunque será necesaria una modificación general debido a que el problema principal de esta parte del proyecto es evitar crear una función de este tipo para cada una de las 110 posibilidades.

Esto se consigue simplemente añadiendo una variable de entrada a la función y sabiendo algo de gestión de cadenas de caracteres en JavaScript. La explicación es la siguiente: si podemos asignar una variable de entrada que sea a su vez una cadena de caracteres que podamos asignar al mensaje que se publica cuando se interactúe con ese botón se tendrá la capacidad de reducir los 110 a la cantidad total de las variables de feedback para el usuario debido a la dependencia de recuperación que tienen las funciones con sus datos a recuperar. En resumen, tras introducir esta mejora de características se reducirá las funciones requeridas a:

- Control del número de página.
- Control del número de botón.
- Control de los modificadores de tecla seleccionados.
- Control de la tecla seleccionada.
- Control de configuración fondo/elemento.
- Control de Rojo.
- Control de Verde.
- Control de Azul.
- Control de actualizar botón.
- Control de numero de página alternativo (no necesario, por pruebas realizadas)
- Función de obtención de variables sin interactuar para inicializar o mantener variables entre pantallas, necesaria para funcionamiento en algunos casos.

Como se puede observar, la introducción de esta característica implica la reducción del 90% del problema. La función tipo corresponderá a lo siguiente:

```
function edTec(a)
{
  var req = false;
```

```
  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("TecState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Tec' + a + '?id';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
}
```

Mientras que la declaración estándar del comando onClick será, poniendo de ejemplo un teclado numérico hecho para los colores:

```
<p>Seleccione la tonalidad de Rojo:</p><a id="RedState">-</a>
<div id=DivTecNum>
        <input id="Tec4" value="7" onclick="edRed('7');" type="button">
        <input id="Tec4" value="8" onclick="edRed('8');" type="button">
        <input id="Tec4" value="9" onclick="edRed('9');" type="button">
        <div></div>
        <input id="Tec4" value="4" onclick="edRed('4');" type="button">
        <input id="Tec4" value="5" onclick="edRed('5');" type="button">
        <input id="Tec4" value="6" onclick="edRed('6');" type="button">
        <div></div>
        <input id="Tec4" value="1" onclick="edRed('1');" type="button">
        <input id="Tec4" value="2" onclick="edRed('2');" type="button">
        <input id="Tec4" value="3" onclick="edRed('3');" type="button">
        <div></div>
        <input id="Tec4" value="0" onclick="edRed('0');" type="button">
      </div>
```

5.3 Comunicación de parte de la placa

De parte del EK-TM4C1294XL, la comunicación se hace mediante dos bloques: uno que hace las de detector de los mensajes publicados por el servidor y otro que ejecuta el bloque con la rutina de acción determinada para ese mensaje.

La primera de estas funciones la cumple el archivo io_fs.c, que contiene las funciones de apertura, clausura y lectura del canal de comunicación.

La segunda función esta descrita por parte del archivo io.c, donde también están contenidas las funciones de arranque del módulo ETHERNET. Este archivo contiene las funciones que se ejecutarán cuando se detecten los mensajes que envía el servidor y tiene una estructura de librería típica, en contraste con su contraparte fs.

5.3.1 Detectando mensajes, io_fs.c

El problema al que nos enfrentamos en este archivo es el reflejo de su contraparte en el servidor. Necesitamos optimizar la detección de mensajes de manera que no tengamos que escribir una secuencia de detección por cada una de las 110 posibilidades.

5.3.1.1 Estructura básica de rastreo

Para empezar, observamos el funcionamiento de la estructura de detección:

1. Una variable se compara con un mensaje dados un número total de caracteres para la comparación, si es igual en ese número se introduce en el contenido del IF.
2. Se ejecuta la funciona signada al mensaje, a la que se le pasa una variable referente a un búfer y un numero referente al tamaño total del que dispone para pasar datos. Se modificará este último valor para dar el espacio justo y necesario para que la variable a cambiar llegue sin problemas.
3. Tras esto, una variable es desglosada en diferentes valores y se les asigna a sus campos una serie de valores. Esto es lo equivalente a una ficha de datos del mensaje.
4. Esta variable que recoge diferentes datos se devuelve, presumiblemente hacia el servidor.

Tras comprender esto, se puede plantear la solución para afrontar el problema principal planteado.

5.3.1.2 Solución del problema de rastreo de mensajes

Aplicando las bases de la solución se llega a la conclusión de que utilizar el manejo de cadenas de caracteres para automatizar algunos de estos procesos reduce el problema en un porcentaje elevado. Siguiendo esta línea de pensamiento, se crean diferentes variables auxiliares en forma de arrays de caracteres:

- Teclado: array inicializado con los valores posibles que tendría el teclado en el servidor. O sea, números del ‘0’ al ‘9’, y letras de la ‘A’ a la ‘Z’. Esto servirá de variable auxiliar para completar automáticamente las cadenas que se usarán para la comprobación final, pudiendo construir cualquier combinación con prefijo + este array.
- auxTecString: array que inicializaremos con la palabra /Tec en todos sus campos, a la que posteriormente le añadiremos las variables de teclado, dando así los mensajes de control referentes al teclado.
- auxRedString, auxGrnString y auxBluString: cadenas inicializadas con su correspondiente célula de tres letras a las que se le añadirá los números de teclado del 0 al 9, para obtener todos los mensajes de control del teclado numérico que maneja los colores del RGB.

Tras esto, se hacen bucles while para identificar el mensaje de dentro de las variables auxiliares y determinar lo antes posible si se contiene o no. Como ejemplo:

```
ip=0;
      while(ustrncmp(pcName, auxTecString[ip], 5) != 0 && ip<36){
          ip++;
      }

      if(ustrncmp(pcName, auxTecString[ip], 5) == 0 && ip<36){
          static char pcBuf[2];

          io_change_Tec(ip,pcBuf, 2);

          psFile->data = pcBuf;
          psFile->len = strlen(pcBuf);
          psFile->index = psFile->len;
          psFile->pextension = NULL;

          return(psFile);
      }
```

Esto, junto con la introducción de una variable en las funciones asignadas que será explicada a continuación, agilizan la escritura de un código especialmente repetitivo.

Como punto final a esta explicación, los cambios se realizan específicamente en la función de apertura, dejando el resto por defecto. La función total de apertura se incluirá en un anexo debido a que su longitud rompería la estructura del documento.

5.3.2 Acciones asigandas a los mensajes

La estrategia seguida en la programación será: creación de una función para manejar la variable a la que podremos introducirle una serie de parámetros de cara a generalizar el máximo posible para englobar el mayor número de casos en el menor número de funciones, y junto con esta una función de inicialización de la variable de cara a la primera ejecución del servidor.

A partir de aquí se repasarán las funciones de manejo creadas y la estrategia seguida en ellas.

5.3.2.1 Numero de Página

Se le añade la introducción de un bool, cuyo estado indica si la página se incrementa o se reduce.

Para la devolución de la variable, que tiene que ser una string, se le suma el número determinado al carácter '0', que correspondería con el número en forma de carácter al tener limitada la variable en un rango muy específico.

La variable a controlar será `page`.

```
void io_change_numPag(bool PlusMin,char * pcBuf, int iBufLen)
{
    if(PlusMin)
        page--;
    else if(!PlusMin)
        page++;

    if(page<1)
        page=4;
    else if(page>4)
        page=1;

    char pageString[2] = {'0' + page};
    usnprintf(pcBuf, iBufLen, pageString);
}
```

5.3.2.2 Numero de Tecla

En cuanto a la variable de entrada es prácticamente igual a la función anterior, un bool que controla si suma o resta.

Por parte de la devolución, al ser un número que puede superar la decena, se crean dos situaciones:

- Cuando estamos por debajo de esta, 0 a 9, se sigue la estrategia del punto anterior.
- Si se está por encima, se asigna un '1' al primer carácter, se le resta 10 a la variable de control y se asigna el segundo carácter como en el punto 1.

La variable a controlar será `VirTec`.

```
void io_change_numVirTec(bool PlusMin,char * pcBuf, int iBufLen)
{
    char VirTecString[3]={'0'};
    if(PlusMin)
        VirTec--;
    else if(!PlusMin)
        VirTec++;
```

```
    if(VirTec<1)
        VirTec=12;
    else if(VirTec>12)
        VirTec=1;

    if(VirTec>9){
        VirTecString[0]='1';
        VirTecString[1]='0' + VirTec-10;
    }
    else{
        VirTecString[1]='0'+ VirTec;
        VirTecString[0]='0';

    }
    usnprintf(pcBuf, iBufLen, VirTecString);
}
```

5.3.2.3 Modificadores de Tecla

La variable de entrada es un número del 1 al 4, que indica:

1. Pulsado CTRL
2. Pulsado ALT
3. Pulsado SHIFT
4. Pulsado Limpiar

Se introducirá la variable en un switch, que asignará los bits a 1 y 0 de un array de 3 bits creado, representación de CTRL|ALT|SHIFT y su estado, pulsación o reposo. El estado de limpiar establecerá todos estos valores a 0, reposo.

Por parte de la devolución, se hace que dependa de la variable que hemos modificado con el estado de las teclas y, en una cadena previamente inicializada con espacios, copiamos el acrónimo específico en un sitio específico de la cadena para dar feedback al usuario.

La variable a controlar será ModTec.

```
void io_change_ModTec(uint8_t Tecla,char * pcBuf, int iBufLen)
{
    char ModTecString[15]={"_____________"}; //ctrl+shift+alt = 15
    switch (Tecla){
    case 1:
        ModTec[0]=1;
        break;
    case 2:
        ModTec[1]=1;
        break;
    case 3:
        ModTec[2]=1;
        break;
    case 4:
        ModTec[0]=0;
        ModTec[1]=0;
        ModTec[2]=0;
        break;
    default:
        ModTec[0]=0;
        ModTec[1]=0;
        ModTec[2]=0;
        break;
    }

    if(ModTec[0])
        strcpy(&ModTecString[0],"CTRL");

    if(ModTec[1])
```

```
        strcpy(&ModTecString[5],"ALT");

    if(ModTec[2])
        strcpy(&ModTecString[9],"SHIFT");

    ModTecString[4]='|';
    ModTecString[8]='|';

    usnprintf(pcBuf, iBufLen, ModTecString);
}
```

5.3.2.4 Teclas principales

La variable de entrada es el ID de rastreo usado para la detección en el rastreo de mensajes, teniendo ese número se puede reutilizar para mirar el índice de la tabla teclado correspondiente y copiar la tecla a la parte de la devolución directamente.

La variable a controlar será Tec, y corresponderá con el ID y no con la tecla en sí.

```
void io_change_Tec(uint8_t ip,char * pcBuf, int iBufLen)
{
    Tec=ip;
    char TecString[2]={*teclado[Tec]};

    usnprintf(pcBuf, iBufLen, TecString);
}
```

5.3.2.5 Cambio fondo/elemento

No se usa variable de entrada en esta función. Esto es debido a que con una simple acción de negación del anterior estado siendo la variable de control un bool se puede conseguir el efecto deseado.

La devolución se hará asignando una cadena directamente debido a la simplicidad del problema de cambio de variable.

La variable a declarar por lo tanto será un bool llamado WindBut.

```
void io_change_windbut(char * pcBuf, int iBufLen){
    char WindButString[5];
    WindBut=!WindBut;
    if(WindBut)
        strcpy(WindButString,"Boton");
    else
        strcpy(WindButString,"Fondo");

    usnprintf(pcBuf, iBufLen, WindButString);
}
```

5.3.2.6 Gestión de colores

El problema a solucionar para los tres canales de colores es muy similar: se necesita leer el número que sea pulsado y ponerlo en las unidades del número actual, desplazando el resto de dígitos. Además, dependiendo del valor de la variable anterior y del valor de la página actual, se modificará un dato de un índice especifico asociado al fondo o a los elementos.

La variable a pasar volverá a ser el id usado en el bucle while para detectar si el mensaje correspondía con alguno de la lista del color especificado. Esta variable será reconvertida en un valor numérico del 0 al 9, se en que rango está el número actual y dependiendo de si se encuentra por debajo de 100 o no se seguirá una línea u otra:

- Por debajo de 100: Se desplazan los valores hacia la izquierda, con una multiplicación en el caso de la variable y una asignación en el caso de la cadena de caracteres de la variable a devolver. El nuevo valor

se asigna en el sitio de las unidades.

- Por encima de 100 o igual a 0: se receta el contador con todo a 0 y se asigna el valor nuevo a las unidades en las variables de control y de devolución.

Las variables a controlar serán dos arrays de tamaño igual al número de páginas totales, asignados a los elementos y al fondo. Se llamarán RedVal, RedBot, GrnVal, GrnBot, BluVal y BluBot.

Como ejemplo, se pone la gestión del rojo:

```
void io_change_RedVal(uint8_t ip,char * pcBuf, int iBufLen)
{
    static char RedString[4]={"000"};
    uint16_t AuxNum;
    if(ip>8)
        AuxNum=0;
    else
        AuxNum=ip+1;
    if(WindBut){
        if(RedBot[page-1]<100 && RedBot[page-1]>0){
                RedString[0]=RedString[1];
                RedString[1]=RedString[2];
                RedString[2]='0' + AuxNum;
                RedBot[page-1]=RedBot[page-1]*10;
                RedBot[page-1]=RedBot[page-1]+AuxNum;
            }
            else{
                RedString[0]='0';
                RedString[1]='0';
                RedString[2]='0' + AuxNum;
                RedBot[page-1]=AuxNum;
            }

            if(RedBot[page-1]>255)
                RedBot[page-1]=255;
            }
    else{
    if(RedVal[page-1]<100 && RedVal[page-1]>0){
        RedString[0]=RedString[1];
        RedString[1]=RedString[2];
        RedString[2]='0' + AuxNum;
        RedVal[page-1]=RedVal[page-1]*10;
        RedVal[page-1]=RedVal[page-1]+AuxNum;
    }
    else{
        RedString[0]='0';
        RedString[1]='0';
        RedString[2]='0' + AuxNum;
        RedVal[page-1]=AuxNum;
    }

    if(RedVal[page-1]>255)
        RedVal[page-1]=255;
    }

    usnprintf(pcBuf, iBufLen, RedString);
}
```

5.3.2.7 Función de actualización de botón

La más simple, indica al sistema que hay que actualizar los botones mediante la asignación de su variable de control tipo bool a verdadero.

La variable se llama SubmitState.

5.4 Implicaciones de personalización en la pantalla, EVE_API.c III

De cara a reflejar los cambios que el usuario realiza en la web local dentro de la pantalla se necesitaran desarrollar una serie de sistemas extras para manejar la serie de variables nuevas declaradas en este punto.

5.4.1 Variables de personalización

Se van a mencionar todas las variables personalización nuevas para recogerlas en un mismo apartado y tener la posibilidad de referenciarse a ellas. Son las siguientes:

```
extern int page;
extern int VirTec;
extern bool ModTec[3];
extern uint8_t Tec;

extern uint16_t RedVal[4];
extern uint16_t GrnVal[4];
extern uint16_t BluVal[4];

extern uint16_t RedBot[4];
extern uint16_t GrnBot[4];
extern uint16_t BluBot[4];

extern bool SubmitState;
```

Como se puede observar, son variables externas cuyo origen es la librería io.c.

5.4.2 Actualización de botón, WEB_SUBMITTER

Esta función se encargará de reflejar los cambios hechos por el usuario en las teclas virtuales de la pantalla táctil actualizando los campos de la estructura principal del proyecto, Botones.

Si el problema se resumiese a que cuando se detecte una activación de la variable de actualización, los campos determinados por el servidor se actualizasen, recayendo la búsqueda del índice en una simple cuenta relacionada con la página y el número de tecla a modificar, el problema se solucionaría de forma trivial mediante asignaciones directas. Pero de cara a ofrecer una diferenciación en las teclas para evitar conclusiones y tener una información clara de lo que se ha programado en una tecla específica, vamos a asignar el nombre de la tecla dependiendo de los modificadores asignados y la tecla principal.

Esto lo desarrollaremos de la siguiente manera:

- Creamos una variable auxiliar que recorrerá la posición de los diferentes caracteres de la cadena.
- En el caso de que se hayan elegido teclas a modificar, se pone su inicial en la posición actual que indique la variable auxiliar y se le añade en el siguiente campo, aux+1, un signo +, para indicar que por ejemplo la tecla fuese C+8. Tras esto incrementamos en 1 la variable auxiliar.
- Se repite este proceso comprobando todas las modificaciones. Si hay más de 1, el símbolo + será machacado por la inicial correspondiente y añadido de nuevo a continuación.
- Por último, se comprueba si en la posición actual que indica la variable auxiliar hay o no un signo +:
    - Si lo hay, se entiende que se han incluido modificadores y se incrementa en 1 más la variable auxiliar para apuntar a un campo no ocupado.
    - Si no lo hay no se hace nada.
- Se coloca la tecla en el carácter que indique el dato auxiliar.

Tras esto, se pasan los datos de la web a sus símiles en la estructura principal, y se establece el bit menos significativo de la estructura BotAccion.mod a 1 para indicar que esta entrada de la tabla ha sido modificada por el usuario.

Por fin se tendrá la capacidad de personalizar los botones mediante la web.

```
void WEB_SUBMITER(){
    uint8_t comandoR=0;
    uint8_t i=OFFSETBOTMATRIX;
    uint8_t TeclaAPulsar;
    uint8_t aux=0;
    char NameTeclaAPulsar[6]="\0\0\0\0\0\0";
    while(comandoR!=VirTec && i<64){
        i++;
        if(Botones[i-1].pantallaAsoc==page){
            comandoR++;
        }


    }
    if(ModTec[0]){
        NameTeclaAPulsar[aux]= 'C';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;
    }
    if(ModTec[1]){
        NameTeclaAPulsar[aux]= 'A';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;

    }
    if(ModTec[2]){
        NameTeclaAPulsar[aux]= 'S';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;
    }
    if(NameTeclaAPulsar[aux]=='+'){
        aux++;
    }

    if(Tec<8){
        TeclaAPulsar=30 + Tec;
        NameTeclaAPulsar[aux]= '1' + Tec;
    }
    else if(Tec==9){
        TeclaAPulsar=0x27;
        NameTeclaAPulsar[aux]= '0';
    }
    else{
        TeclaAPulsar=0x04+Tec-10;
        NameTeclaAPulsar[aux]= 'A'+ Tec-10;
    }
    strcpy(Botones[i-1].name,NameTeclaAPulsar);
    Botones[i-1].BotAccion.comando=TeclaAPulsar;
    Botones[i-1].BotAccion.mod[0]=ModTec[0];
    Botones[i-1].BotAccion.mod[1]=ModTec[1];
    Botones[i-1].BotAccion.mod[2]=ModTec[2];
    Botones[i-1].BotAccion.mod[3]=1;
    cambioF1=true;
}
```

5.4.3 Actualización de colores, modificaciones en bloques básicos

La actualización de los colores de los elementos será en tiempo real, debido a la no inclusión intencionada de un botón de actualización asignado al color. Con esto el usuario podrá ver el cambio en directo de los colores que vaya introduciendo mediante la web.

Con esto se necesitará una manera de detectar cambios en el color por parte del sistema, siendo esta la introducción de variables de recuerdo. Estas se compararán con el valor actual y se actualizarán en caso de cambio, detectando así un cambio de color.

La asignación de colores se hará directamente con los valores asignados a la página actual y no con valores

preestablecidos, como antes.

Cambiarán la inicialización de pantalla:

```
  void EVE_INICIALIZAR_NUEVA_PANTALLA(){ //Pantalla con fondo negro
      if(RedVal[PAG-1]!=RedValPast[PAG-1]     ||     GrnVal[PAG-1]!=GrnValPast[PAG-1]||
BluVal[PAG-1]!=BluValPast[PAG-1]||RedBot[PAG-1]!=RedBotPast[PAG-1]    ||    GrnBot[PAG-
1]!=GrnBotPast[PAG-1]|| BluBot[PAG-1]!=BluBotPast[PAG-1]){
          cambioF2=true;
          RedValPast[PAG-1]=RedVal[PAG-1];
          GrnValPast[PAG-1]=GrnVal[PAG-1];
          BluValPast[PAG-1]=BluVal[PAG-1];
          RedBotPast[PAG-1]=RedBot[PAG-1];
          GrnBotPast[PAG-1]=GrnBot[PAG-1];
          BluBotPast[PAG-1]=BluBot[PAG-1];
      }
      EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

      EVE_CMD_DLSTART(); // When executed, EVE will begin a new DL
      EVE_CLEAR_COLOR_RGB(RedVal[PAG-1], GrnVal[PAG-1], BluVal[PAG-1]); // Select color
to clear screen to
      EVE_CLEAR(1,1,1); // Clear

      EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
      EVE_LIB_AwaitCoProEmpty();
  }
```

Y la función de color de elemento:

```
  void EVE_COLOR_ELEMENTO(){
      EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD
  //    uint32_t Auxiliar=;
      EVE_COLOR_RGB((512-RedBot[PAG-1]-RedVal[PAG-1])/2,            (512-GrnBot[PAG-1]-
GrnVal[PAG-1])/2, (512-BluBot[PAG-1]-BluVal[PAG-1])/2);
      EVE_CMD_FGCOLOR(RedBot[PAG-1]*0x10000+GrnBot[PAG-1]*0x100+BluBot[PAG-1]);

      EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
      EVE_LIB_AwaitCoProEmpty();
  }
```

En esta última, se decidirá el color del texto como un término a medio camino del color de fondo y el color de elemento.

5.4.4 Usando los Códigos Custom, EVE_ACTUADOR II

Si se tiene en cuenta que el comando de tecla se almacena dentro del campo correspondiente de comando y puede aplicarse directamente sobre la función de teclado HID correspondiente, solo queda resolver el uso de los modificadores.

Este problema se solucionará con el uso de la acción OR aplicada sobre una variable auxiliar del tamaño del campo correspondiente a los modificadores de la función HID. Si comprobamos los modificadores 1 a 1 y los superponemos, obtendremos el modificador conjunto de tecla.

Se desarrolla esa idea a código, y se obtiene la función final:

```
  void EVE_ACTUADOR(uint8_t tag){
      if(Botones[tag-1].BotAccion.mod[3]){
          uint8_t AuxModTec=0;
          if(Botones[tag-1].BotAccion.mod[0])
              AuxModTec=AuxModTec|0x01;
          if(Botones[tag-1].BotAccion.mod[1])
              AuxModTec=AuxModTec|0x04;
```

```
            if(Botones[tag-1].BotAccion.mod[2])
                AuxModTec=AuxModTec|0x02;

            USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,AuxModTec,Botones[tag-
1].BotAccion.comando,1);
            SysCtlDelay(40*40000);
            USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,0,Botones[tag-
1].BotAccion.comando,0);
            SysCtlDelay(40*40000);

        }
        else{

USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][0],EveKeypUsageCodes[Botones[tag-1].BotAccion.comando-
1][1],1);
            SysCtlDelay(20*40000);

USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,0,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][1],0);
            SysCtlDelay(20*40000);
        }
    }
```

5.4.5 Mostrando IP a conectar

Como simple recordatorio: anteriormente en el manejador de botones se programó una funcionalidad asociada a un botón que se encargaría de mostrar y ocultar la IP necesaria para la conexión al servidor local.

Esta decisión de diseño se toma debido a que solo es necesario ver esta dirección 1 vez, y tras esto es un gasto de espacio en pantalla que puede evitarse.

El botón asignado a la función controlará un bool que indicará el estado de mostrar, 1, o no mostrar, 0. Con este se hará una comprobación en el main.c y se ejecutará, o no, esta función.

La función vuelca el contenido de la variable que contiene la IP configurada en la librería io.c, pasada a la librería EVE_API.c como extern, en una cadena de texto que mostramos con el comando correspondiente.

```
    void MOSTRAR_IP(){

        EVE_LIB_BeginCoProList();

        usprintf(pcBuf, "%d.%d.%d.%d", g_ui32IPAddress & 0xff, (g_ui32IPAddress >> 8) &
0xff,
                 (g_ui32IPAddress >> 16) & 0xff, (g_ui32IPAddress >> 24) & 0xff);
        EVE_CMD_TEXT(400,380,28,EVE_OPT_CENTER,pcBuf);

        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

    }
```

Tras esto, se termina la integración total de los tres bloques principales del proyecto y se cuenta con un teclado capacitivo reprogramable mediante un servidor web que interactúa con cualquier ordenador Windows e iOS. Es decir, se han cumplido los objetivos puestos en un primer momento.

6 MEMORIA FLASH Y REVISIÓN FINAL

Tras la comprobación del funcionamiento de los anteriores bloques implementados, se tiene una serie de situaciones que pueden afectar negativamente a la experiencia del usuario, haciendo muy tedioso el uso de forma continuada del dispositivo. La principal de ellas es: todas las configuraciones, colores y comandos propios, entre otros, que se introduzcan serán borrados del launchPad al desconectar este de una fuente de alimentación.

En este momento es donde se plantea la solución típica para estos casos, el uso de la memoria FLASH.

6.1 La memoria FLASH

La memoria FLASH es un bloque contenido dentro de nuestro launchPad que tiene como función principal el almacenamiento de datos. Lo especial de esta memoria es que es capaz de establecer datos físicamente dentro de unas células mediante el uso de impulsos eléctricos, y una vez establecidos no necesita de ningún tipo de corriente para mantener su estado, por lo que, si se elimina la fuente de energía, al reintroducirla podremos acceder a esos datos guardados en esta.

Este tipo de memoria se usa principalmente para el guardado de comandos, rutinas y datos de sistema básicos para el funcionamiento correcto de los diferentes launchPad del mercado. Además de esto, se suelen dejar reservadas para el uso del usuario una serie de posiciones, dándole la posibilidad al programador de jugar con estas características que se mantienen entre encendidos.

Uno de los problemas que presentan este tipo de memoria y el que se solucionará en el proyecto es que, a la hora de cambiar los valores de la memoria FLASH, dependiendo de su tecnología, no podremos convertir ceros en unos. Esto provocará que, si queremos hacer cambios de forma consistente y no producir datos alterados a causa de la conjunción de diferentes bits de datos, tendremos que borrar el campo antes de modificarlo para mantener la integridad. Pero esto provoca el problema principal: para la EK-TM4C129XL, cuando borramos una posición de memoria de la FLASH, esta provocará el borrado del bloque completo que contiene a esta posición, es decir, unas 16kb en posiciones.

Este problema está presente en todos los launchPad que tengan esta memoria accesible, debido a que todas las memorias FLASH tienen este problema. La única variabilidad entre los problemas es el tamaño del bloque, que depende de la tipología de la FLASH y cambia entre sistemas.

6.2 Usando la FLASH en el sistema

Se van a necesitar dos estrategias para interactuar con la FLASH, una para almacenar información en la FLASH y otra para recuperar información de esta.

Para esto, será muy importante que tengamos claro cuáles serán los tamaños de las variables que almacenaremos y recuperaremos debido a que estos procesos estarán principalmente limitados por la cantidad de información que se mueve de una vez.

6.2.1 Variables a almacenar: estructura principal, colores y calibración

A la hora de plantear la información más útil que se puede mantener entre apagados tenemos: la estructura funcional principal de botones, los valores de personalización de los colores de elementos y fondo, y los valores de calibración. Estos últimos se incluyen para que no sea necesario recalibrar la pantalla cada vez que se enciende el sistema, dando la posibilidad al usuario de recalibrar cuando desee mediante el botón asignado.

Ahora se hará una lista de las variables que nos interesan y sus tamaños.

6.2.1.1 Estructura principal, Botones

- Coordenada X, 16b
- Coordenada Y, 16b
- Tamaño W, 16b
- Tamaño H, 16b
- Nombre asociado, 6 caracteres * 8b cada uno
- Tag Asociado, 8b
- Pantalla Asociada, 8b
- Tipo de Acción, 8b
- Comando de Acción, 16b
- Modificador de tec, 4b

6.2.1.2 Colores

Todas las variables relevantes de colores se duplicarán debido a la dualidad elemento/fondo. Además, teniendo en cuenta que los tres colores son iguales tendremos:

- Color, 16b por evitar sobrepasos de 256 * 3 colores * 2 situaciones = 6 colores de 16b

Pese a que la variable original es de 16b para evitar situaciones de sorpaso de los 256 que el usuario puede provocar, cuando esta información llega a la capa de la pantalla estará limitada a 256, lo que son 8b. Esto hace que se pueda interpretar esto como si tuviese mos 6 colores de 8b.

6.2.1.3 Calibración

Según la declaración de la variable de calibración, tendremos que asignar un total de 6 registros de 32b cada uno.

6.2.2 Estrategia de guardado

Para el guardado en FLASH se usará la función FLASHPROGRAM. A esta función tendremos que pasarle un puntero al inicio del dato que queremos guardar, la posición de la FLASH donde la guardaremos y el tamaño total del dato en octetos pares de bytes.

Las posiciones elegidas serán 2, de cara a partir el problema de sobreescritura para hacer el cambio menos frecuente en cada bloque. Se almacenará en la posición 0x50000 todo lo relacionado a botones, y en la 0x60000 todo lo relacionado a colores y calibración. Se elige este offset de posiciones ya que coincide con el tamaño de bloque de la FLASH, 16kbytes, ya que 0x10000 representa en decimal el numero 16k.

Tras esto, el protocolo a seguir será el de enviar los datos a la memoria FLASH en paquetes de 32 bits, agrupando estos para ocupar el máximo tamaño dentro de los paquetes. Los primeros paquetes tendrán la forma:

1. Coordenadas X e Y, 16b + 16b = 32b
2. Tamaños W y H, 16b + 16b = 32b
3. 4 caracteres de name, 4 * 8b = 32b
4. 2 caracteres de name, tag y pantalla, 2 * 8b + 8b + 8b = 32b
5. Tipo de Acción, Comando de Acción y Modificador, 8b + 16b + 4b = 28b

Esto hace que haya 4b no usados en el último paquete, pero no existe problema con eso. Se asignará un paquete intermedio vacío para ver gráficamente en la memoria con un lector de memoria FLASH la posición de las

variables de forma más clara.

Con respecto a los segundos paquetes tendremos en colores:

1. Rfondo, Gfondo, Bfondo y Rboton, 4 * 8b = 32b
2. Gboton y Bboton, 2 * 8b = 16b

Estos se almacenarán primero, usando los 16b no usados como separador visual. Tras estos se colocarán los registros de calibración:

1. Registro 1, 32b
2. Registro 2, 32b
3. Registro 3, 32b
4. Registro 4, 32b
5. Registro 5, 32b
6. Registro 6, 32b

Por lo tanto, la función estándar lo único que tendrá que hacer es almacenar en una palabra de 32b las variables a guardar en memoria mediante una operación OR, tras esto esa palabra será guardada y el índice que maneja las posiciones donde se va operando, incrementado en 4 unidades. Como ejemplo, la primera función:

```
void FLASH_STOCKER_I(uint8_t i){
    uint32_t posicionAux=posicion+i*24;
    uint32_t posAux;

    if(Botones[i].fis.w!=0 && Botones[i].fis.h!=0){
    posAux=0;
    posAux|=Botones[i].fis.x;
    posAux|=Botones[i].fis.y<<16;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].fis.w;
    posAux|=Botones[i].fis.h<<16;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].name[0]<<0;
    posAux|=Botones[i].name[1]<<8;
    posAux|=Botones[i].name[2]<<16;
    posAux|=Botones[i].name[3]<<24;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].name[4]<<0;
    posAux|=Botones[i].name[5]<<8;
    posAux|=Botones[i].tagAsoc<<16;
    posAux|=Botones[i].pantallaAsoc<<24;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].BotAccion.tipo<<0;
    posAux|=Botones[i].BotAccion.comando<<8;
    posAux|=Botones[i].BotAccion.mod[0]<<24;
    posAux|=Botones[i].BotAccion.mod[1]<<25;
    posAux|=Botones[i].BotAccion.mod[2]<<26;
```

```
      posAux|=Botones[i].BotAccion.mod[3]<<27;
      FlashProgram(&posAux,posicionAux,4);
      }
  }
```

Tras esto, tendremos nuestras variables hasta ahora volátiles guardadas en memoria para su recuperación cuando estimemos oportuno.

6.2.3 Estrategia de recuperación

Una vez tenemos almacenada los datos interesantes para el proyecto en la memoria, el proyecto pasa a necesitar el desarrollo de un bloque que pueda encargarse de recuperar esa memoria y asignarla a las variables correspondientes.

Esto se conseguirá mediante el uso de la función HWREG, que permite leer una posición de memoria y devuelve un dato de tamaño 32b. Este tipo de devolución del dato, es en parte causante de la decisión y enviar paquetes de 32b en la estrategia de guardado, de cara a hacer el cálculo necesario con antelación y controlar los números asignados para la recuperación de los datos enviados. Habiendo hecho ese trabajo, lo único que tenemos que hacer es asignar las variables en el mismo orden que se mandaron, correspondiendo los bits de derecha a izquierda con el orden de envió.

Lo más complicado, planteado lo anterior, es el enmascaramiento y desplazamiento de los bits de cara a recuperar la forma original que tenían a partir de una palabra de 32b conjunta. Poniendo de nuevo el ejemplo del bloque botones:

```
  void FLASH_FISHER_I(uint8_t i){
      uint32_t posicionAux=posicion+i*24;
      uint32_t posAux=0;

      if(HWREG(posicionAux)!=0xffffffff    &&    HWREG(posicionAux+4)!=0xffffffff    &&
HWREG(posicionAux+8)!=0xffffffff && HWREG(posicionAux+12)!=0xffffffff){
      posAux=HWREG(posicionAux);
      Botones[i].fis.x=(uint16_t)(posAux & 0xffff);
      Botones[i].fis.y=(uint16_t)(posAux >>16);
      posicionAux+=4;

      posAux=0;
      posAux=HWREG(posicionAux);
      Botones[i].fis.w=(uint16_t)(posAux & 0xffff);
      Botones[i].fis.h=(uint16_t)(posAux >>16);
      posicionAux+=4;

      posAux=0;
      posAux=HWREG(posicionAux);
      Botones[i].name[0]=(char)(posAux & 0xffffff);
      Botones[i].name[1]=(char)((posAux & 0xffff)>>8);
      Botones[i].name[2]=(char)(posAux>>16 & 0xff);
      Botones[i].name[3]=(char)(posAux >> 24);
      posicionAux+=4;

      posAux=0;
      posAux=HWREG(posicionAux);
      Botones[i].name[4]=(char)(posAux & 0xffffff);
      Botones[i].name[5]=(char)((posAux & 0xffff)>>8);
      Botones[i].tagAsoc=(uint8_t)(posAux>>16 & 0xff);
      Botones[i].pantallaAsoc=(uint8_t)(posAux >> 24);
      posicionAux+=4;

      posAux=0;
      posAux=HWREG(posicionAux);
      Botones[i].BotAccion.tipo=(uint8_t)(posAux & 0x000000ff);
      Botones[i].BotAccion.comando=(uint16_t)((posAux & 0x00ffff00)>>8);
      Botones[i].BotAccion.mod[0]=(bool)(posAux>>24 & 0x01);
      Botones[i].BotAccion.mod[1]=(bool)(posAux>>25 & 0b0000001);
      Botones[i].BotAccion.mod[2]=(bool)(posAux>>26 & 0b000001);
      Botones[i].BotAccion.mod[3]=(bool)(posAux>>27 & 0b00001);
```

```
        }
    }
```

El proceso de recuperación se dará únicamente al principio del programa, copiando estos valores a sus variables si volátiles para trabajar de forma rápida con ellas y no tener que estar accediendo a memoria para la recuperación constantemente. Y debido a que los cambios se ejecutarán en las variables volátiles, cargándose en memoria cada vez que se efectúen, el número necesario de recuperaciones será de 1.

6.2.4 Estrategia de actualización

He aquí el problema principal del uso de FLASH. A falta de una ocurrencia mejor, el proyecto tratará este problema de manera que cada vez que se detecte un cambio que tenga que ser guardado en la memoria, recupere el resto de campos almacenados previamente.

Esto se hará mediante unas variables marcadoras, que se activarán cuando los procesos de cambio de botones, calibración o la detección del cambio de color, sean activadas. El estado de estas se comprobará en cada bucle, y en el caso de que estén activas se ejecutará la función FLASH_RENEW, que se encargará del proceso de borrado mediante FLASHERASE, y la recuperación de todos los campos actualizados junto con los no modificados.

A nivel funcional, el bloque ejecutará las funciones individuales hechas para almacenar los datos en memoria pasando todos los posibles índices que aceptan de cara a no dejar ningún dato olvidado.

```
void FLASH_RENEW(){
    uint8_t i;
    if(cambioF1){
    FlashErase(posicion);
    for(i=0;i<64;i++){
        FLASH_STOCKER_I(i);
        cambioF1=false;

    }
    }
    if(cambioF2){
        FlashErase(posicionColor);
        for(i=1;i<5;i++){
            FLASH_STOCKER_C(i);
            CALIB_STOCKER();
            cambioF2=false;
        }
    }
}
```

Al diferenciar entre los dos apartados de memoria, los cambios en cada bloque son la mitad de frecuentes. Esto es totalmente opcional, pero el proyecto quería ser testeado en bloques diferentes.

7 GUÍA DE USUARIO

En este apartado se desarrollará la guía de uso para usuario, en la que se indicarán las instrucciones para el correcto funcionamiento del dispositivo.

7.1 Primer Encendido

- Por norma general y en el caso del dispositivo final de este TFG, el dispositivo contará con una versión precargada de prueba. Si el lector está desarrollando su propio sistema esto no será así, para este caso se desarrollará un último punto en el que se listaran las diferencias entre la versión de prueba y la que sería resultante de una carga de código directa.
- Se recomienda encarecidamente que se produzca un calibrado al iniciar, pulsando en el botón CAL de la pantalla principal.
- Con la versión precargada el usuario puede experimentar en un entorno ya iniciado las funcionalidades del dispositivo

7.2 Pantalla Principal

La pantalla principal tendrá la siguiente estructura:

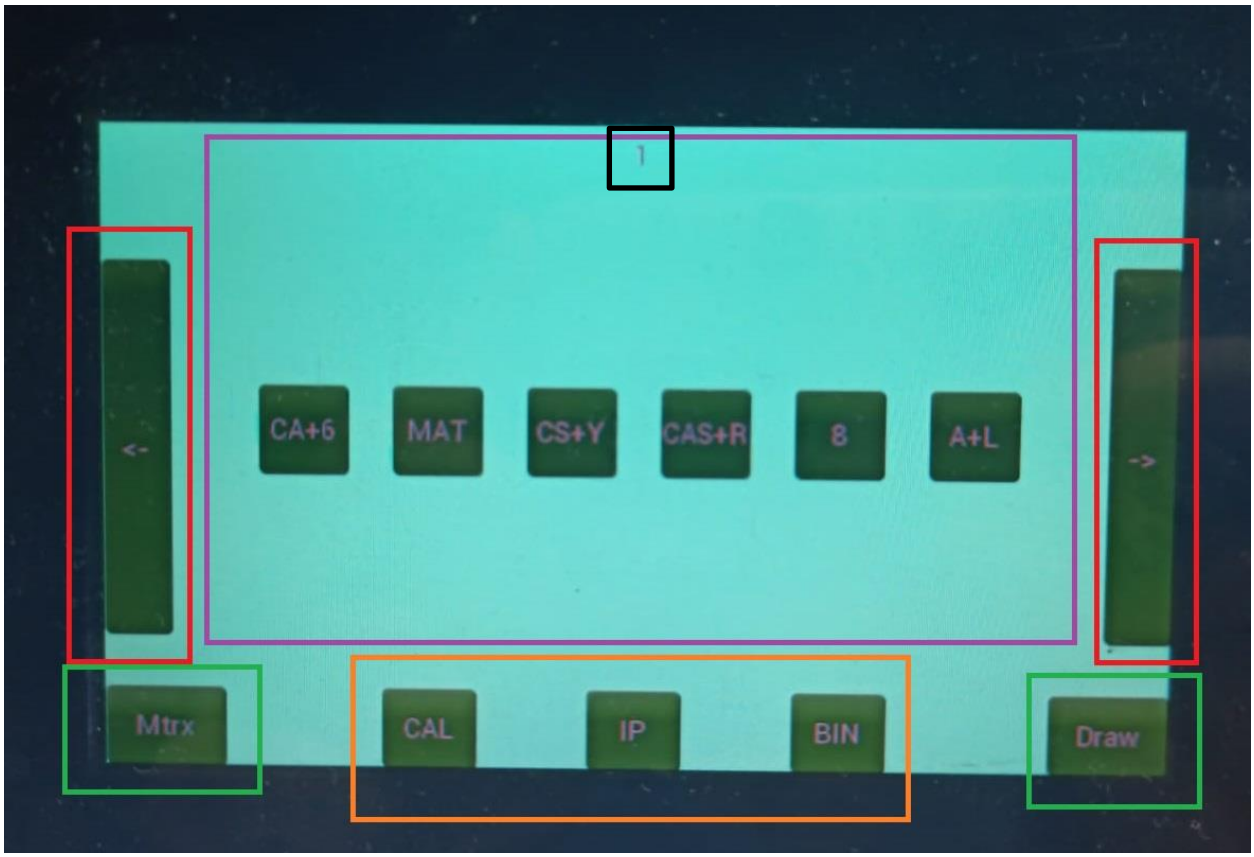

Figure 21.Pantalla principal

- ROJO: botones de desplazamiento por la pantalla. Derecha incrementa, izquierda decrementa.
- VERDE: modos de edición.

  - MTRX es el modo automático.
  - DRAW el modo manual.
- NARANJA: botones de función.
  - CAL ejecuta la secuencia de calibrado.
  - IP muestra la IP justo encima de esta zona.
  - BIN ejecuta el protocolo de borrado de botones.
- NEGRO: número de pantalla actual.
- MORADO: Cuerpo principal útil de la pantalla. Es la zona que usará el modo automático para crear botones, con el modo libre podríamos saltarnos estos límites, pero eso queda bajo la responsabilidad del usuario.

7.3 Modos de Edición

7.3.1 Edición Automática

1. Al pulsar sobre el botón MTRX entraremos en la pantalla asociada a este modo.

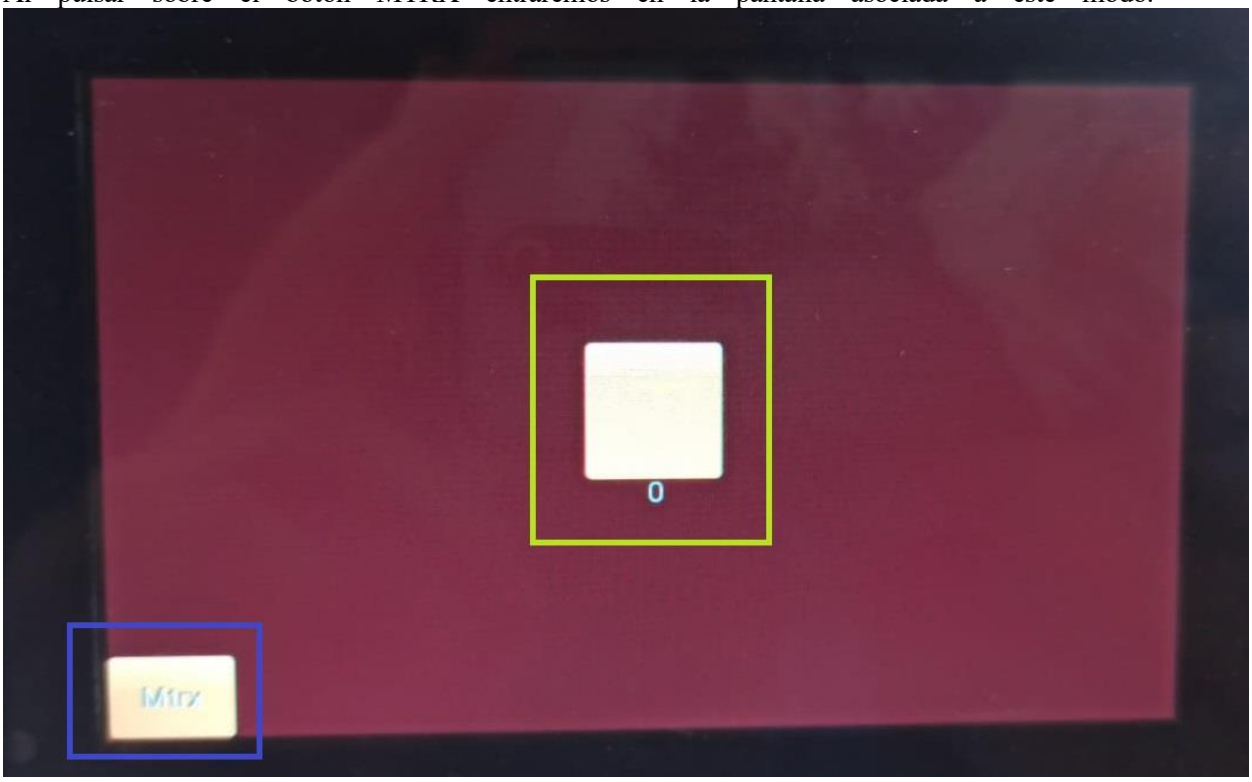

Figure 22.Pantalla de Edición Automática

2. El botón central servirá para incrementar el número de botones deseados.

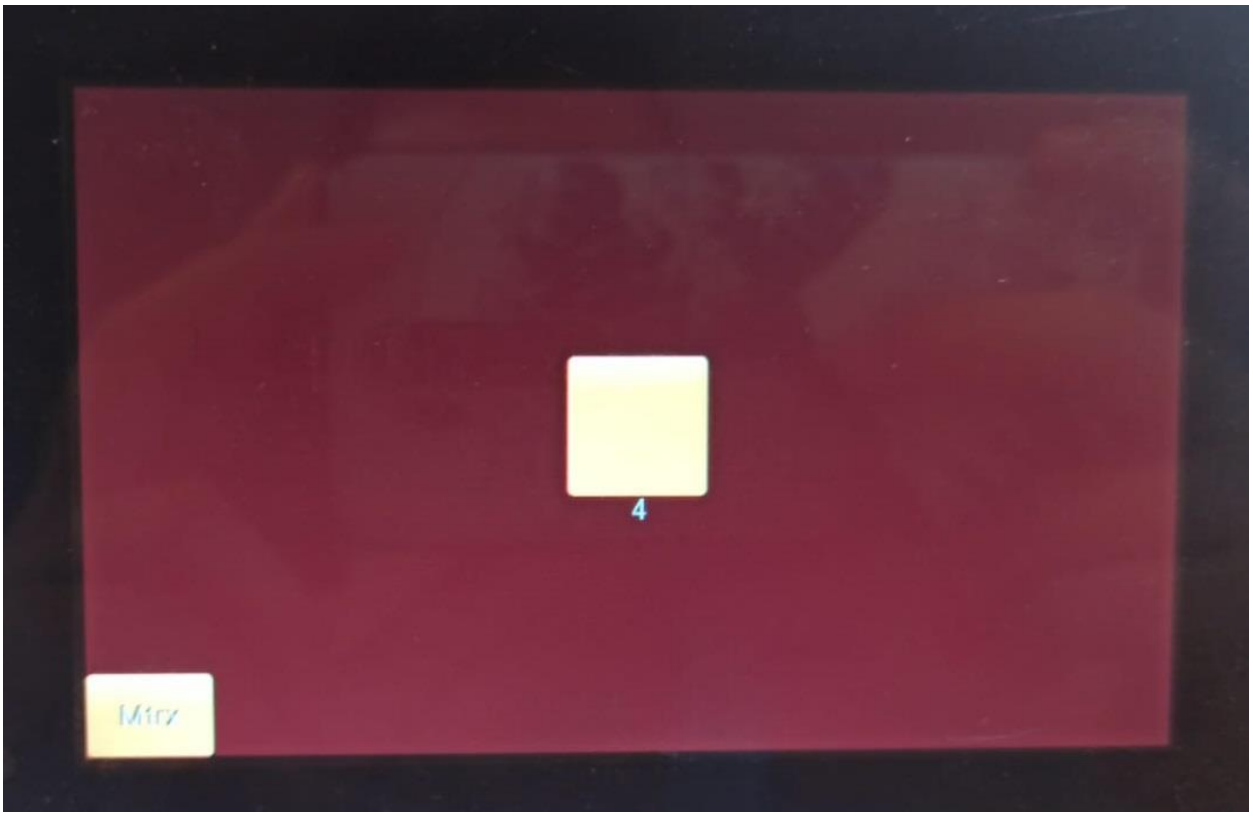

Figure 23. Pantalla de Edición Automática con uso

3. Cuando se termine, se pulsará de nuevo el botón MTRX para salir del modo y generar la actriz deseada.

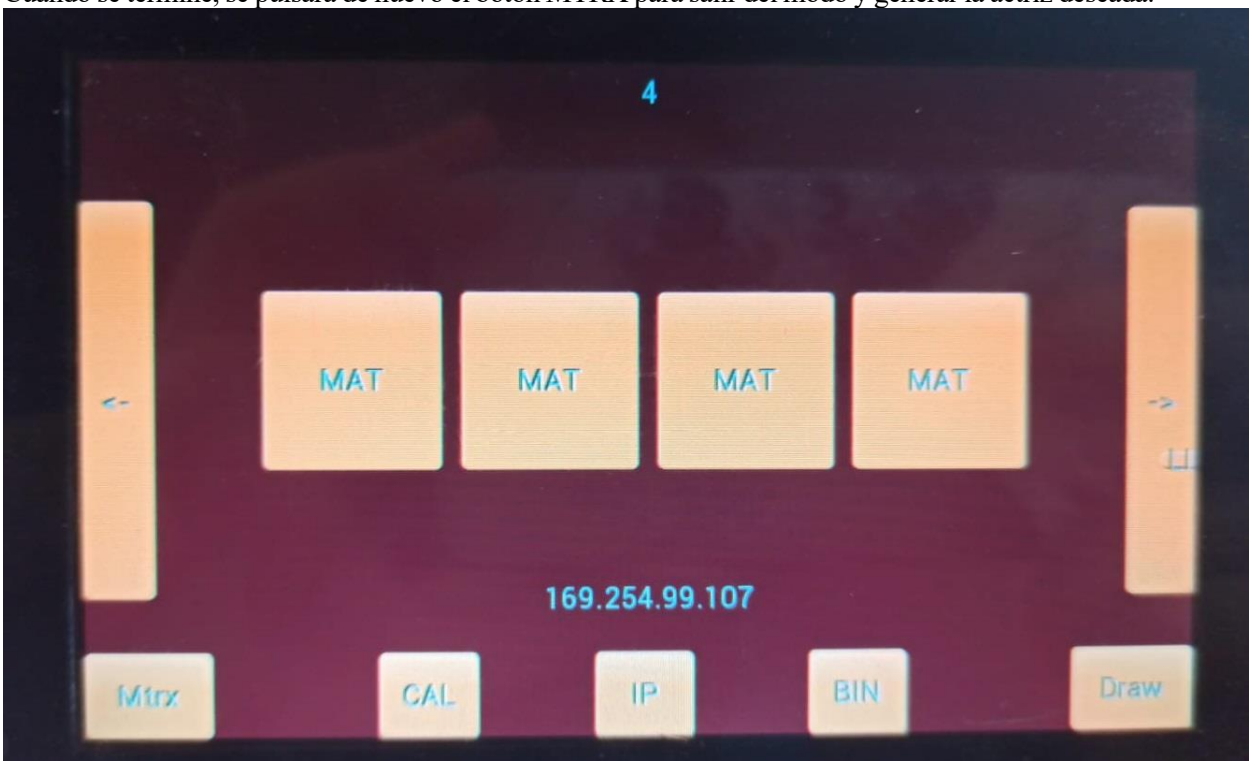

Figure 24. Matriz generada

7.3.2 Edición Libre

1. Para entrar en el modo de edición libre, se pulsará el botón correspondiente DRAW. La pantalla es igual a la principal, pero habrá perdido el número de pantalla y al tocar los botones estos no reaccionarán.
2. Para crear un botón, pulse y mantenga en algún punto de la pantalla

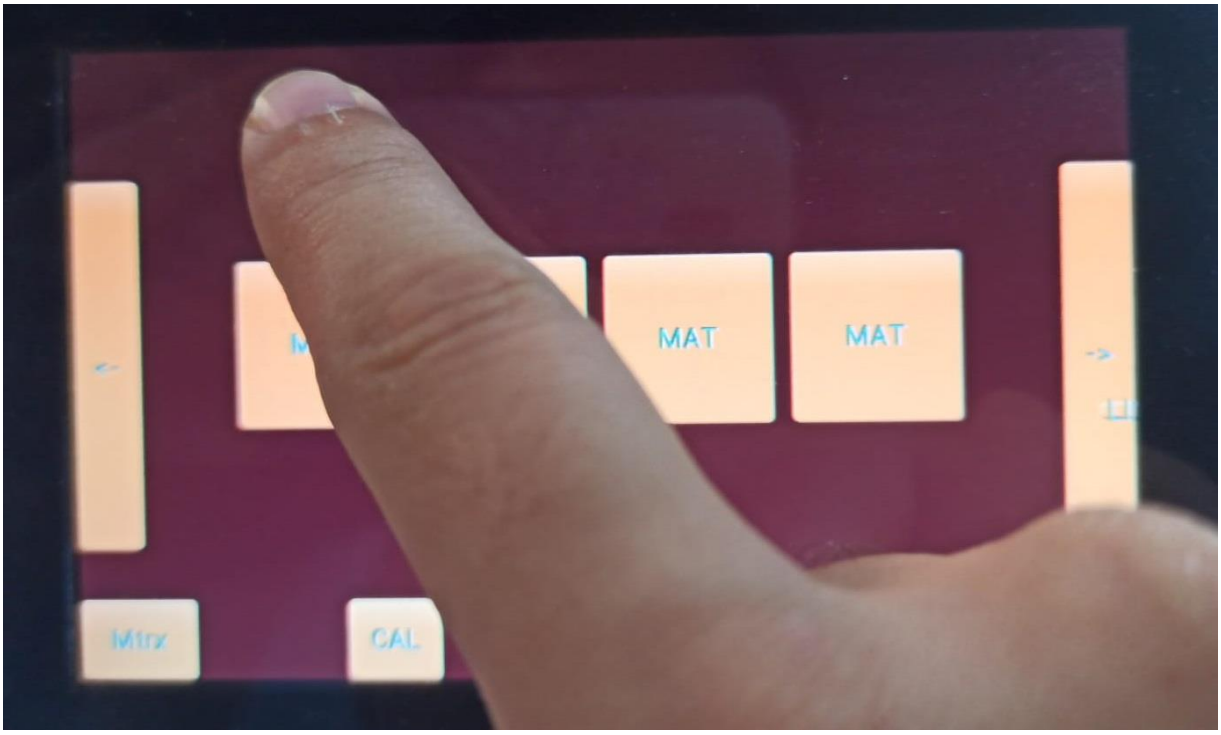

Figure 25. Uso de la edición libre 1

3. Tras esto, deslice por la pantalla hasta tener el tamaño de botón deseado.

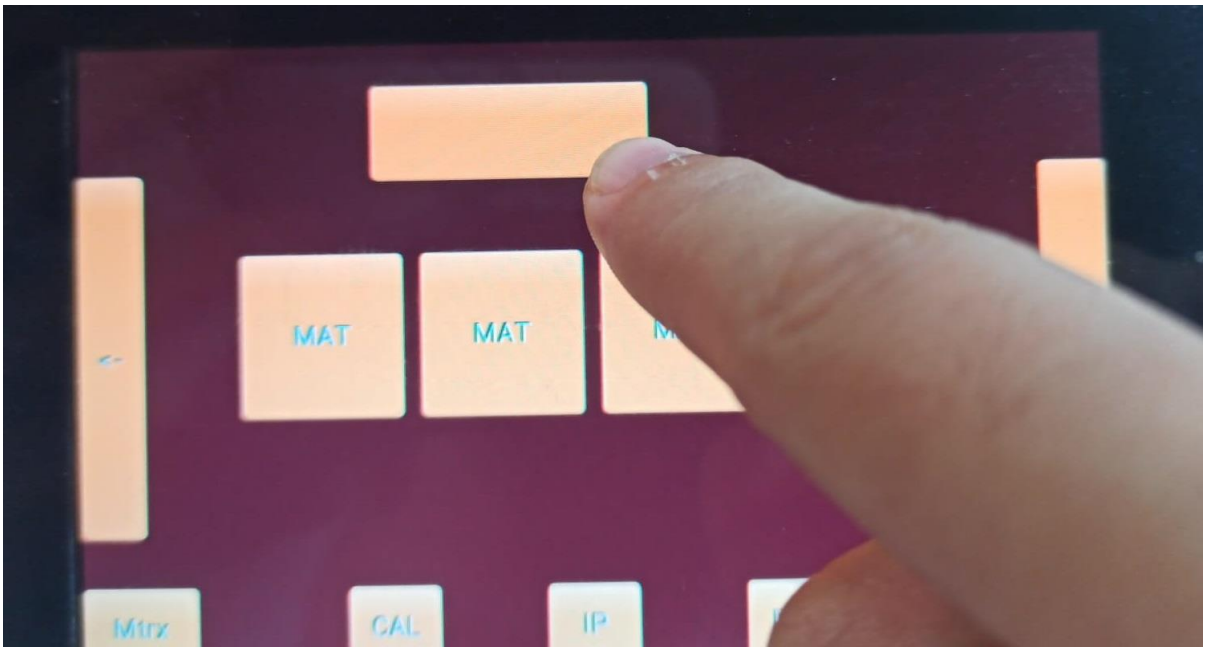

Figure 26. Uso de la edición libre 2

4. Suelte para crear el botón. Se confirmará la creación cuando en este aparezca la etiqueta LIB.

Figure 27. botón generado por la edición libre

5. Siga creando botones a antojo y, cuando termine, pulse el botón de LIB para volver al estado de funcionamiento normal con los nuevos botones creados.

7.4 Botones de funcionalidad

7.4.1 Botón de calibración

Al pulsar en este botón, aparecerá una pantalla interactiva que le pedirá que pulse en los puntos indicados.

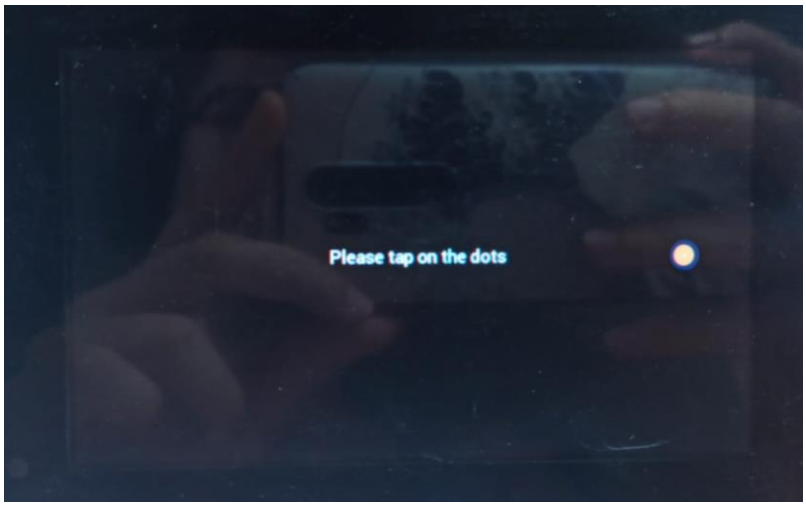

Figure 28. pantalla de calibración

Tras esto saldrá automáticamente del modo, habiendo calibrado la pantalla satisfactoriamente.

7.4.2 Botón de IP

Pulse para cambiar el estado entre mostrar la IP u ocultarla

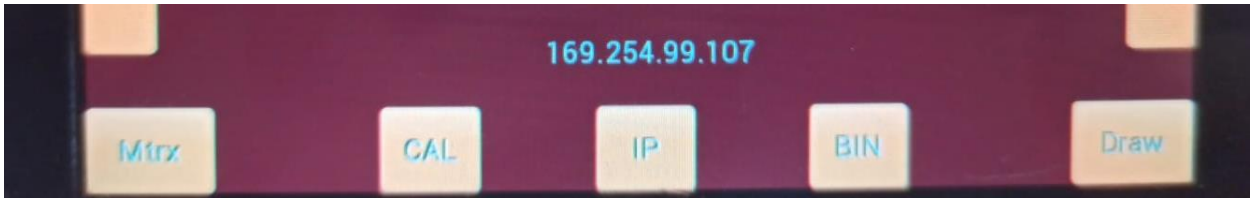

Figure 29. Zona donde se muestra la dirección IP

7.4.3 Botón de borrado

1. Asignado al botón BIN, de papelera. Al pulsar éste, se entrará en un estado parecido al de la edición libre, desapareciendo el texto en pantalla fuera de las etiquetas de botones.
2. En este estado cualquier botón que no sea de sistema será borrado al pulsar sobre él.
3. Para volver, pulsar el botón BIN de nuevo.

7.5 Personalización en la Web

7.5.1 Apertura de la Web

De cara al uso de la web para la personalización, el usuario tendrá que abrir la web siguiendo los siguientes pasos:

1. Leemos la dirección IP proporcionada por la pantalla.

Figure 30. Dirección IP

2. Abrimos nuestro navegador de confianza y copiamos tal cual la dirección IP en la barra de navegación.

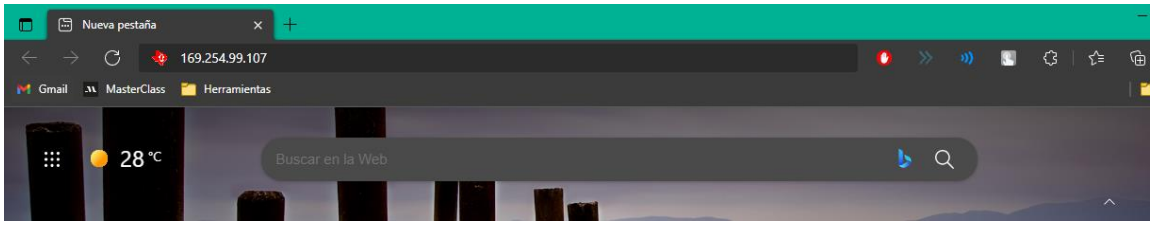

Figure 31. Navegador estándar con dirección IP

3. Pulsamos Intro para acceder a la dirección y nos encontramos con la pantalla principal

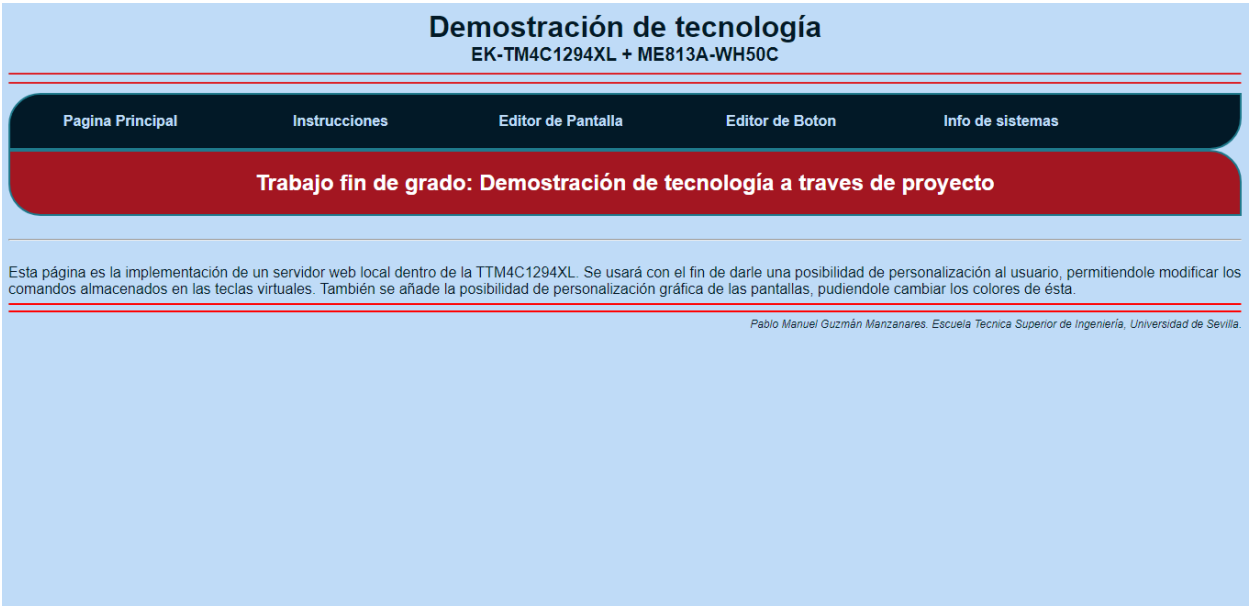

Figure 32. Pagina inicial de la WEB

7.5.2 Pantalla Principal de la Web

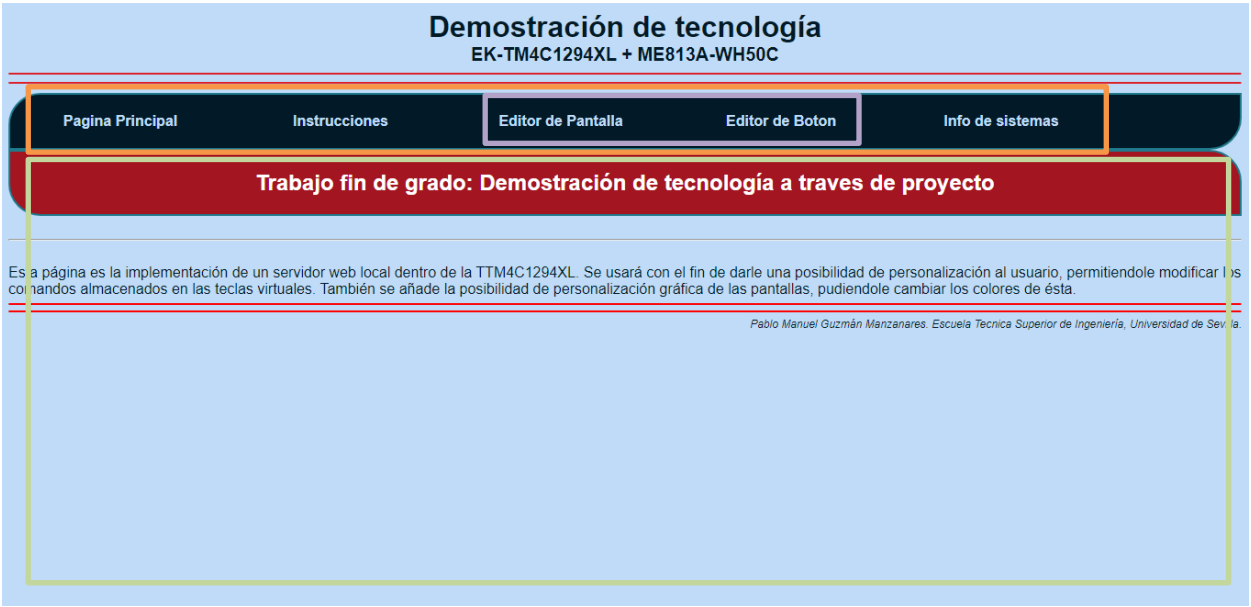

Figure 33.Página inicial de la Web segmentada

- NARANJA: Barra principal de navegación. Contiene:
  - Página Principal, la que se ve actualmente.
  - Instrucciones, página espejo de este apartado con las indicaciones para el uso de la página.
  - Editores, se explicarán a continuación.
  - Página de referencia con links a los fabricantes y recursos
- VERDE: cuerpo principal de la página y donde se irán mostrando el contenido relacionado con las pestañas seleccionadas.
- MORADO: editores de características de la pantalla.

7.5.3 Editor de colores

Se accede mediante la pestaña EDITOR DE PANTALLA.

Figure 34.Pagina de edición de pantalla

Contiene los siguientes campos:

- Selector de Página a modificar: se incrementa el número con los botones del campo
- Selector fondo/elemento
- Teclado para ROJO, se introduce el numero deseado en el orden centenas, decenas y unidades
- Teclado para VERDE, igual a rojo
- Teclado para AZUL, igual a rojo

Un ejemplo de una pantalla que queda tras un uso normal sería:

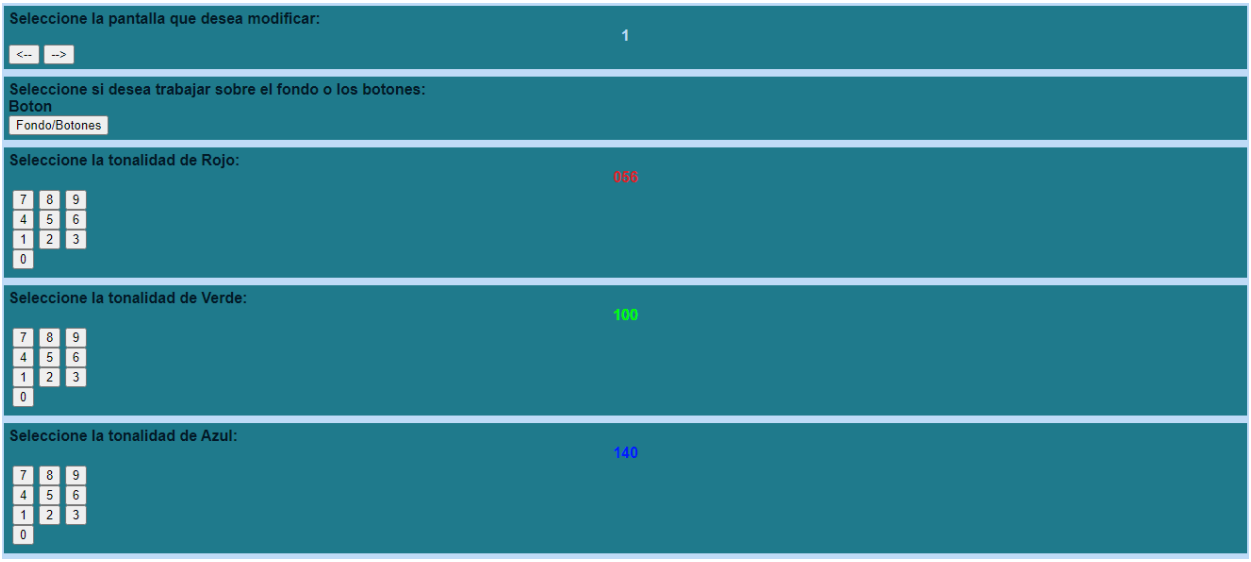

Figure 35.Edicion de pantalla tras uso

7.5.4 Editor de botones

Se accede mediante la pestaña EDITOR DE BOTON.

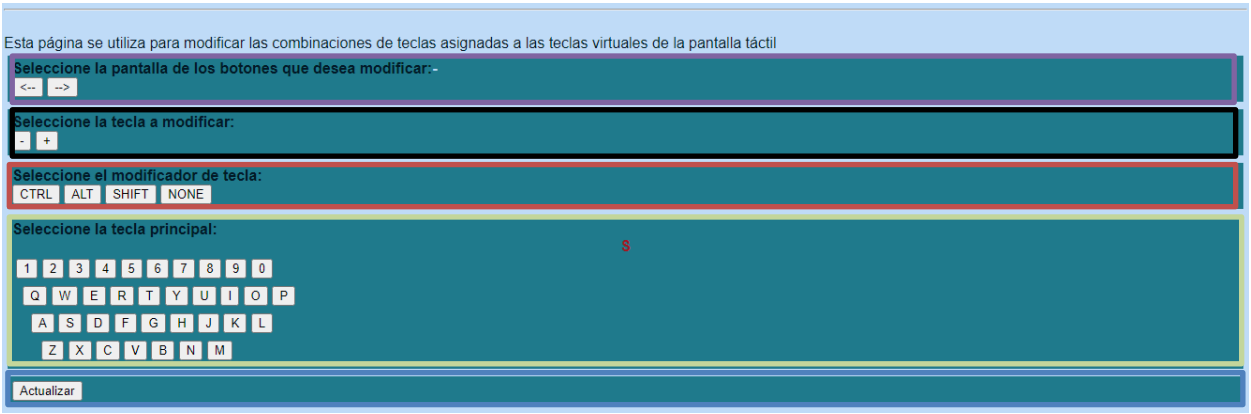

Figure 36.Pagina de edición de botones

Contiene los siguientes campos:

- Selector de Página a modificar: se incrementa el número con los botones del campo.
- Selector de tecla: se incrementa y decrementa con los botones asignados al campo.
- Selector de modificador para tecla, se seleccionan los modificadores con las teclas de nombres correspondientes
- Selector de Tecla, se selecciona la tecla principal deseada para el comando custom pinchando sobre la tecla del teclado virtual deseada
- Actualización, pulsar este botón actualiza los valores del botón seleccionado en el dispositivo

Un ejemplo de una pantalla que queda tras un uso normal sería:

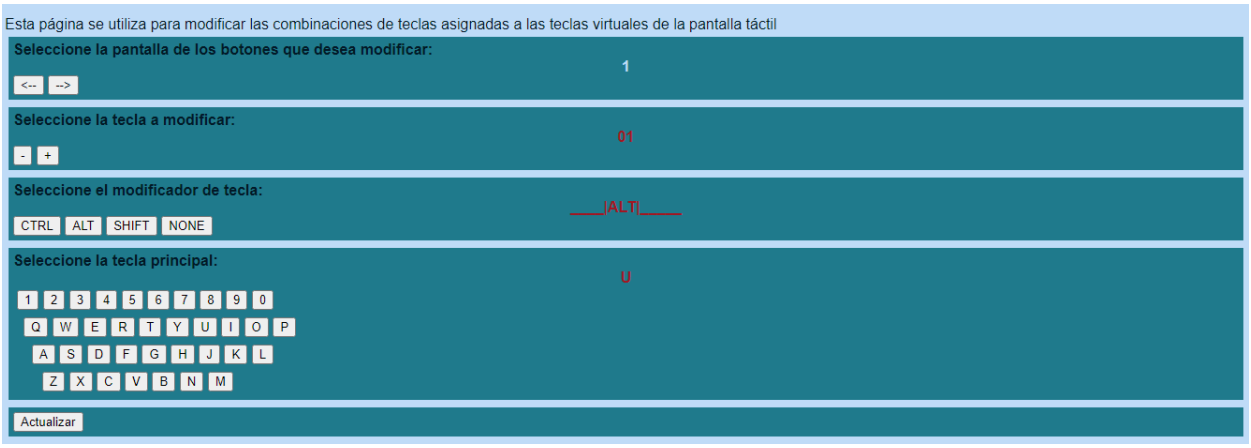

Figure 37.Edición de botones tras uso

7.6 Diferencias principales al ejecutar con una memoria FLASH vacía

- Se produce una petición de calibrado nada más comenzar al no existir una configuración previa.
- Al producirse la recuperación de la información al principio del programa por parte de la FLASH, se recuperará el color 0xFFFFFF, o sea, blanco. Esto se aplicará tanto en fondo como en elemento, por lo que se hace muy confuso trabajar en la pantalla.
- No existe ningún botón en ninguna pantalla, a parte de los establecidos de sistema

En vista de esto, se aconseja:

- Cambiar los colores de fondo y botones con la web, antes de hacer cualquier cosa, en pos de trabajar en un entorno claro y menos confuso.
- Crear una serie de botones de los dos tipos de edición en todas las pantallas para hacer pruebas de funcionamiento.

BIBLIOGRAFÍA

APPLE SUPPORT, 2021. Usar la Touch Bar en Mac. En: *Manual del usuario de macOS* [en línea]. Disponible en: Usar la Touch Bar en la Mac - Soporte técnico de Apple [consulta: 10 septiembre 2021].

BRIDGETEK, 2019. *FT8xx Portable MCU Library: Application Note* [en linea]. Singapore: Bridgetek Pte Ltd (BRT Chip) [consulta 10 septiembre 2021]. Disponible en: BRT_AN_025_FT8xx_Portable_MCU_Library_DRAFT.pdf

FUTURE TECHNOLOGY DEVICES INTERNATIONAL (FTDI), 2015. *FT81x Series Programmers Guide* [en línea]. United Kingdom: Future Technology Devices International Limited [consulta 3 agosto 2021]. doi: FT_001173. Disponible en: FT81x Series Programmers Guide (ftdichip.com)

PASTOR, Javier, 2018. El ASUS Zenbook Pro 14 quiere revolucionar el segmento con un ScreenPad que es touchpad y pantalla auxiliar. *Xataka* [en línea]. Disponible en: Nuevo ASUS Zenbook Pro 14: características, precio y ficha técnica (xataka.com) [consulta: 10 septiembre 2021].

PENALVA, Javier, 2019. ASUS Zenbook Pro Duo, análisis: el portátil de doble pantalla 4K intenta ser el portátil definitivo. *Xataka* [en línea]. Disponible en: ASUS Zenbook Pro Duo, análisis: review con características, precio y especificaciones (xataka.com) [consulta: 10 septiembre 2021].

POMEYROL, J., 2021. MacBook Pro: ¿adiós a la Touch Bar? *Muy Computer* [en línea]. Disponible en: MacBook Pro: ¿adiós a la Touch Bar? (muycomputer.com) [consulta: 10 septiembre 2021].

TEXAS INSTRUMENTS, 2016. *Tiva C Series TM4C1294 Connected LaunchPad Evaluation Kit: User's guide* [en línea]. Austin, TX: Texas Instruments Incorportated [consulta 3 agosto 2021]. doi: SPMU365C. Disponible en: Tiva C Series TM4C1294 Connected LaunchPad Evaluation Kit (Rev. C)

TEXAS INSTRUMENTS, 2020. *TivaWare USB Library: User's guide* [en línea]. Austin, TX: Texas Instruments Incorporated [consulta: 16 agosto 2021]. doi: SW-TM4C-USBL-UG-2.2.0.295. Disponible en: TivaWare™ USB Library for C Series User's Guide (Rev. E)

UNIVERSAL SERIAL BUS, 2001. *Device Class Definition for Human Interface Devices (HID)*[en línea]. World: USB Enabling Connections™[consulta 20 agosto 2021]Disponible en: Microsoft Word - HID1_11.doc (usb.org)

ANEXO A

Aquí se incluirá todo el código del proyecto dividido en sus diferentes bloques:

Raiz principal

Main.c

```
  #include <stdbool.h>
  #include <stdint.h>
  #include <string.h>

  #include <eveheader.h>

  #include <usbkeyboardheader.h>

  #include "driverlib2.h"

  #include "evethernet.h"

  #define PARAM_ERROR_RESPONSE    "/perror.htm"

  #define JAVASCRIPT_HEADER                                                   \
          "<script type='text/javascript' language='JavaScript'><!--\n"
  #define JAVASCRIPT_FOOTER                                                   \
          "//--></script>\n"

  #define OFFSETBOTMATRIZ 12
  #define OFFSETBOTLIBRE 38

  #ifdef DEBUG
  void
  __error__(char *pcFilename, uint32_t ui32Line)
  {
  }
  #endif

  char auxTecString[36][5];
  char auxRedString[10][5];
  char auxGrnString[10][5];
  char auxBluString[10][5];
  char
teclado[36][1]={"1","2","3","4","5","6","7","8","9","0","A","B","C","D","E","F","G","H
","I",

"J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z"};

  uint8_t i;
  uint16_t POSX, POSY;
  uint32_t BufferXY,g_ui32IPAddress,ui32SysClock;
  uint8_t TAG=0;
  uint8_t PAG=1;

  bool cambioF1=false;
  bool cambioF2=false;
  bool muestraIP=false;

  struct Boton Botones[64];   //12-reservados
                              //26-botones de matriz
                              //26-botones a mano alzada
```

```
int main(void){
    ui32SysClock = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
            SYSCTL_OSC_MAIN |
            SYSCTL_USE_PLL |
            SYSCTL_CFG_VCO_240), 120000000);

    INIT_ESTRUCT_PRINC();

    Init_Cad_Comms();
    Init_Ethernet_Server();

    Inicia_Teclado();

    CALIB_FISHER();
    EVE_Init();
    eve_calibrate();
    cambioF2=true;



    CREAR_BOTON(0,0,420,90,60,"Mtrx",0,1,0);
    CREAR_BOTON(0,710,420,90,60,"Draw",0,2,0);
    CREAR_BOTON(0,750,100,50,280,"->",0,3,0);
    CREAR_BOTON(0,0,100,50,280,"<-",0,4,0);
    CREAR_BOTON(0,154*3+90-35,420,70,60,"BIN",0,5,0);
    CREAR_BOTON(0,154*2+90-35,420,70,60,"IP",0,6,0);
    CREAR_BOTON(0,154*1+90-35,420,70,60,"CAL",0,7,0);

    for(i=0;i<64;i++){
        FLASH_FISHER_I(i);
    }
    for(i=1;i<5;i++){
        FLASH_FISHER_C(i);
    }

////////////////////////////////////////////////////////////////////////////////
//////////////
    while(1)
    {

        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        if(muestraIP)
            MOSTRAR_IP();

        EVE_LIB_BeginCoProList();

//        EVE_CMD_NUMBER(100,50,28,EVE_OPT_CENTER,POSX);
//        EVE_CMD_NUMBER(700,50,28,EVE_OPT_CENTER,POSY);
        EVE_CMD_NUMBER(400,20,28,EVE_OPT_CENTER,PAG);

        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
        EVE_HANDLER_ACCION();

        if(cambioF1 || cambioF2){
            FLASH_RENEW();
            cambioF1=false;
            cambioF2=false;
        }

        CHECK_FLAGS();
    }
}
```

Driverlib2.h

```
//INCLUIR TODAS LAS LIBRERÍAS de DRIVERLIB
#ifndef DRIVERLIB_H_
#define DRIVERLIB_H_

#include "driverlib/adc.h"
#include "driverlib/aes.h"
#include "driverlib/can.h"
#include "driverlib/comp.h"
#include "driverlib/cpu.h"
#include "driverlib/crc.h"
#include "driverlib/debug.h"
#include "driverlib/des.h"
#include "driverlib/eeprom.h"
#include "driverlib/emac.h"
#include "driverlib/epi.h"
#include "driverlib/flash.h"
#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
#include "driverlib/hibernate.h"
#include "driverlib/i2c.h"
#include "driverlib/interrupt.h"
#include "driverlib/lcd.h"
#include "driverlib/mpu.h"
#include "driverlib/onewire.h"
#include "driverlib/pin_map.h"
#include "driverlib/pwm.h"
#include "driverlib/qei.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/rtos_bindings.h"
#include "driverlib/shamd5.h"
#include "driverlib/ssi.h"
#include "driverlib/sw_crc.h"
#include "driverlib/sysctl.h"
#include "driverlib/sysexc.h"
#include "driverlib/systick.h"
#include "driverlib/timer.h"
#include "driverlib/uart.h"
#include "driverlib/udma.h"
#include "driverlib/usb.h"
#include "driverlib/watchdog.h"

#include "inc/hw_adc.h"
#include "inc/hw_aes.h"
#include "inc/hw_can.h"
#include "inc/hw_ccm.h"
#include "inc/hw_comp.h"
#include "inc/hw_des.h"
#include "inc/hw_eeprom.h"
#include "inc/hw_emac.h"
#include "inc/hw_epi.h"
#include "inc/hw_fan.h"
#include "inc/hw_flash.h"
#include "inc/hw_gpio.h"
#include "inc/hw_hibernate.h"
#include "inc/hw_i2c.h"
#include "inc/hw_ints.h"
#include "inc/hw_lcd.h"
#include "inc/hw_memmap.h"
#include "inc/hw_nvic.h"
#include "inc/hw_onewire.h"
#include "inc/hw_pwm.h"
#include "inc/hw_qei.h"
```

```
#include "inc/hw_shamd5.h"
#include "inc/hw_ssi.h"
#include "inc/hw_sysctl.h"
#include "inc/hw_sysexc.h"
#include "inc/hw_timer.h"
#include "inc/hw_types.h"
#include "inc/hw_uart.h"
#include "inc/hw_udma.h"
#include "inc/hw_usb.h"
#include "inc/hw_watchdog.h"

//#include "drvlibh/tm4c1294ncpdt.h"

#endif /* DRIVERLIB_H_ */
```

Evethernet.h

```
/*
 * evethernet.h
 *
 *  Created on: 27 ago. 2021
 *      Author: pablo
 */

#ifndef EVETHERNET_H_
#define EVETHERNET_H_

#include <utils/locator.h>
#include <utils/lwiplib.h>
#include <utils/ustdlib.h>
#include "httpserver_raw/httpd.h"
#include "httpserver_raw/fs.h"
#include "httpserver_raw/fsdata.h"
#include "ethernet/io.h"


#endif /* EVETHERNET_H_ */
```

Eveheader.h

```
/*
 * evekeyboardheader.h
 *
 *  Created on: 17 ago. 2021
 *      Author: pablo
 */

#ifndef EVEHEADER_H_
#define EVEHEADER_H_

#include <RecursosEve/eve_calibrateplus.h>
#include <RecursosEve/EVE.h>
#include <RecursosEve/EVE_config.h>
#include <RecursosEve/FT8xx.h>
#include <RecursosEve/HAL.h>
#include <RecursosEve/MCU.h>


#endif /* EVEHEADER_H_ */
```

Usbkeyboardheader.h

```
/*
 * Usbkeyboardheader2.h
 *
 *  Created on: 17 ago. 2021
 *      Author: pablo
 */

#ifndef USBKEYBOARDHEADER_H_
#define USBKEYBOARDHEADER_H_

#include "usblib/usblib.h"
#include "usblib/usbhid.h"
#include "usblib/usb-ids.h"
#include "usblib/device/usbdevice.h"
#include "usblib/device/usbdhid.h"
#include "usblib/device/usbdhidkeyb.h"
#include "drivers/buttons.h"
#include "drivers/pinout.h"
#include "usbkeyboard/usb_keyb_structs.h"
#include <utils/uartstdio.h>

#endif /* USBKEYBOARDHEADER_H_ */
```

RecursosEve

Eve_api.c

```
#include <string.h>
#include <stdint.h> // for Uint8/16/32 and Int8/16/32 data types
#include <stdbool.h>

#include "driverlib2.h"

#include<ethernet/io.h>

#include <eveheader.h>

#include <usbkeyboardheader.h>

void EVE_Init(void)
{
    uint8_t regGpio;
      int i;

    HAL_EVE_Init();
```

```
// ------------------------ Display settings -----------------------------

// LCD display parameters
// Active width of LCD display
HAL_MemWrite16(EVE_REG_HSIZE,   EVE_DISP_WIDTH);
// Total number of clocks per line
HAL_MemWrite16(EVE_REG_HCYCLE,  EVE_DISP_HCYCLE);
// Start of active line
HAL_MemWrite16(EVE_REG_HOFFSET, EVE_DISP_HOFFSET);
// Start of horizontal sync pulse
HAL_MemWrite16(EVE_REG_HSYNC0,  EVE_DISP_HSYNC0);
// End of horizontal sync pulse
HAL_MemWrite16(EVE_REG_HSYNC1,  EVE_DISP_HSYNC1);
// Active height of LCD display
HAL_MemWrite16(EVE_REG_VSIZE,   EVE_DISP_HEIGHT);
// Total number of lines per screen
HAL_MemWrite16(EVE_REG_VCYCLE,  EVE_DISP_VCYCLE);
// Start of active screen
HAL_MemWrite16(EVE_REG_VOFFSET, EVE_DISP_VOFFSET);
// Start of vertical sync pulse
HAL_MemWrite16(EVE_REG_VSYNC0,  EVE_DISP_VSYNC0);
// End of vertical sync pulse
HAL_MemWrite16(EVE_REG_VSYNC1,  EVE_DISP_VSYNC1);
// Define RGB output pins
HAL_MemWrite8(EVE_REG_SWIZZLE,  EVE_DISP_SWIZZLE);
// Define active edge of PCLK
HAL_MemWrite8(EVE_REG_PCLK_POL, EVE_DISP_PCLKPOL);


  // Write initial display list
  HAL_MemWrite32((EVE_RAM_DL + 0), EVE_ENC_CLEAR_COLOR_RGB(0,0,0));
HAL_MemWrite32((EVE_RAM_DL + 4), EVE_ENC_CLEAR(1,1,1));
HAL_MemWrite32((EVE_RAM_DL + 8), EVE_ENC_DISPLAY());
HAL_MemWrite8(EVE_REG_DLSWAP, EVE_DLSWAP_FRAME);


// Read the  GPIO register for a read/modify/write operation
regGpio = HAL_MemRead8(EVE_REG_GPIO);
// set bit 7 of  GPIO register (DISP) - others are inputs
regGpio = regGpio | 0x80;
// Enable the DISP signal to the LCD panel
HAL_MemWrite8(EVE_REG_GPIO, regGpio);

// Now start clocking data to the LCD panel
HAL_MemWrite8(EVE_REG_PCLK, EVE_DISP_PCLK);
HAL_MemWrite8(EVE_REG_PWM_DUTY, 127);

// ---------------------- Touch and Audio settings -------------------------

// Eliminate any false touches
HAL_MemWrite16(EVE_REG_TOUCH_RZTHRESH, 1200);

// turn recorded audio volume down
HAL_MemWrite8(EVE_REG_VOL_PB, EVE_VOL_ZERO);
// turn synthesizer volume down
HAL_MemWrite8(EVE_REG_VOL_SOUND, EVE_VOL_ZERO);
// set synthesizer to mute
HAL_MemWrite16(EVE_REG_SOUND, 0x6000);


// --------------------- Clear screen ready to start ----------------------
```

```
    EVE_LIB_BeginCoProList();
    EVE_CMD_DLSTART();
    EVE_CLEAR_COLOR_RGB(0, 0, 0);
    EVE_CLEAR(1,1,1);
    EVE_DISPLAY();
    EVE_CMD_SWAP();
    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();

      // ---------------------- Reset all bitmap properties ------------------------
      EVE_LIB_BeginCoProList();
      EVE_CMD_DLSTART();
    EVE_CLEAR_COLOR_RGB(0, 0, 0);
      EVE_CLEAR(1,1,1);
      for (i = 0; i < 16; i++)
      {
            EVE_BITMAP_HANDLE(i);
            EVE_CMD_SETBITMAP(0,0,0,0);
      }
      EVE_DISPLAY();
      EVE_CMD_SWAP();
    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();

}

// Begins co-pro list for display creation
void EVE_LIB_BeginCoProList(void)
{
    // Wait for command FIFO to be empty and record current position in FIFO
    EVE_LIB_AwaitCoProEmpty();

    // Begins SPI transaction
    HAL_ChipSelect(1);
    // Send address for writing as the next free location in the co-pro buffer
    HAL_SetWriteAddress(EVE_RAM_CMD + HAL_GetCmdPointer());
}

// Ends co-pro list for display creation
void EVE_LIB_EndCoProList(void)
{
    // End SPI transaction
    HAL_ChipSelect(0);
    // Update the ring buffer pointer to start decode
    HAL_WriteCmdPointer();
}

// Waits for the read and write pointers to become equal
void EVE_LIB_AwaitCoProEmpty(void)
{
    // Await completion of processing
    HAL_WaitCmdFifoEmpty();
}

// Writes a block of data to the RAM_G
void EVE_LIB_WriteDataToRAMG(const uint8_t *ImgData, uint32_t DataSize, uint32_t
DestAddress)
{
```

```
    // Begins SPI transaction
    HAL_ChipSelect(1);
    // Send address to which first value will be written
    HAL_SetWriteAddress(DestAddress);

    // Pad data length to multiple of 4.
    DataSize = (DataSize + 3) & (~3);

    // Send data as 32 bits.
    while (DataSize)
    {
        HAL_Write32(*(uint32_t *)ImgData);
        ImgData += 4;
        DataSize -= 4;
    }

    // End SPI transaction
    HAL_ChipSelect(0);
}

// Reads a block of data from the RAM_G
void EVE_LIB_ReadDataFromRAMG(uint8_t *ImgData, uint32_t DataSize, uint32_t
SrcAddress)
{
    // Begins SPI transaction
    HAL_ChipSelect(1);
    // Send address to which first value will be written
    HAL_SetReadAddress(SrcAddress);

    // Pad data length to multiple of 4.
    DataSize = (DataSize + 3) & (~3);

    // Send data as 32 bits.
    while (DataSize)
    {
        *(uint32_t *)ImgData = HAL_Read32();
        ImgData += 4;
        DataSize -= 4;
    }

    // End SPI transaction
    HAL_ChipSelect(0);
}

// Write a block of data to the coprocessor
void EVE_LIB_WriteDataToCMD(const uint8_t *ImgData, uint32_t DataSize)
{
    uint32_t CurrentIndex = 0;
    uint32_t ChunkSize = 0;
    const uint32_t MaxChunkSize = 128;
    uint8_t IsLastChunk = 0;
    uint16_t Freespace = 0;

    // This code works by sending the data in a series of one or more bursts.
    // If the data is more than MaxChunkSize bytes, it is sent as a series of
    // one or more bursts and then the remainder. MaxChunkSize is a size which
    // is smaller than the command buffer on the EVE and small enough to gain
    // maximum buffering effect from the MCU SPI hardware.

    // Pad data length to multiple of 4.
    DataSize = (DataSize + 3) & (~3);
```

```
    // While not all data is sent
    while (CurrentIndex < DataSize)
    {
        // If more than ChunkSize bytes to send
        if ((DataSize - CurrentIndex) > MaxChunkSize)
        {
            // ... then add ChunkSize to the current target index to make new target
            ChunkSize = MaxChunkSize;
            // ... and this is not the last chunk
            IsLastChunk = 0;
        }
        // or if all remaining bytes can fit in one chunk
        else
        {
            // ... then add the amount of data to the current target
            ChunkSize = DataSize - CurrentIndex;
            // .. and this is the last chunk
            IsLastChunk = 1;
        }

        // Wait until there is space
        Freespace = 0;
        while (Freespace < MaxChunkSize)
        {
            Freespace = HAL_CheckCmdFreeSpace();
        }

        // Begin an SPI burst write
        HAL_ChipSelect(1);

        // to the next location in the FIFO
        HAL_SetWriteAddress(EVE_RAM_CMD + HAL_GetCmdPointer());

        HAL_Write(ImgData, ChunkSize);
        ImgData += ChunkSize;
        CurrentIndex += ChunkSize;

        // End the SPI burst
        HAL_ChipSelect(0);

        // Calculate where end of data lies
        HAL_IncCmdPointer(ChunkSize);
        HAL_WriteCmdPointer();

        // If this is the last chunk of the data,
        if (IsLastChunk)
        {
            break;
        }
    }
}

// Writes a string over SPI
uint16_t EVE_LIB_SendString(const char* string)
{
    uint16_t length;
    uint16_t CommandSize;
```

```
    // Include the terminating null character in the string length.
    // Pad string length to a multiple of 4.
    length = ((strlen(string) + 1) + 3) & (~3);
    // Store command length to return.
    CommandSize = length;

    // Send string as 32 bit data.
    while (length)
    {
        HAL_Write32(*(uint32_t *)string);
        string += 4;
        length -= 4;
    }

    return CommandSize;
}

void EVE_LIB_GetProps(uint32_t *addr, uint32_t *width, uint32_t *height)
{
    uint32_t WritePointer;

    WritePointer = HAL_GetCmdPointer();
    EVE_LIB_BeginCoProList();
    // To read the result from CMD_GETPROPS we need to be clever and find out
    // where the CoProcessor is writing the command. We can then retrieve the
    // results from the place where they were written.
    // Send the command to the CoProcessor.
    EVE_CMD_GETPROPS(0, 0, 0);
    // Wait for it to finish.
    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();
    // Obtain the results from the EVE_RAM_CMD in the CoProcessor.
    *addr = HAL_MemRead32(EVE_RAM_CMD + ((WritePointer + (1 * sizeof(uint32_t))) &
(EVE_RAM_CMD_SIZE - 1)));
    *width = HAL_MemRead32(EVE_RAM_CMD + ((WritePointer + (2 * sizeof(uint32_t))) &
(EVE_RAM_CMD_SIZE - 1)));
    *height = HAL_MemRead32(EVE_RAM_CMD + ((WritePointer + (3 * sizeof(uint32_t))) &
(EVE_RAM_CMD_SIZE - 1)));
}

//##############################################################################
// Display List commands for co-processor
//##############################################################################

void EVE_CMD(uint32_t c)
{
    HAL_Write32(c);
    HAL_IncCmdPointer(4);
}

void EVE_CLEAR_COLOR_RGB(uint8_t R, uint8_t G, uint8_t B)
{
    HAL_Write32(EVE_ENC_CLEAR_COLOR_RGB(R, G, B));
    HAL_IncCmdPointer(4);
}

void EVE_CLEAR_COLOR(uint32_t c)
{
    HAL_Write32(EVE_ENC_CLEAR_COLOR(c));
    HAL_IncCmdPointer(4);
```

```
}

void EVE_CLEAR(uint8_t C, uint8_t S, uint8_t T)
{
    HAL_Write32(EVE_ENC_CLEAR((C & 0x01),(S & 0x01),(T & 0x01)));
    HAL_IncCmdPointer(4);
}

void EVE_COLOR_RGB(uint8_t R, uint8_t G, uint8_t B)
{
    HAL_Write32(EVE_ENC_COLOR_RGB(R, G, B));
    HAL_IncCmdPointer(4);
}

void EVE_COLOR(uint32_t c)
{
    HAL_Write32(EVE_ENC_COLOR(c));
    HAL_IncCmdPointer(4);
}

void EVE_VERTEX2F(int16_t x, int16_t y)
{
    HAL_Write32(EVE_ENC_VERTEX2F(x, y));
    HAL_IncCmdPointer(4);
}

void EVE_VERTEX2II(uint16_t x, uint16_t y, uint8_t handle, uint8_t cell)
{
    HAL_Write32(EVE_ENC_VERTEX2II(x, y, handle, cell));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_HANDLE(uint8_t handle)
{
    HAL_Write32(EVE_ENC_BITMAP_HANDLE(handle));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_SOURCE(uint32_t addr)
{
    HAL_Write32(EVE_ENC_BITMAP_SOURCE(addr));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_LAYOUT(uint8_t format, uint16_t linestride, uint16_t height )
{
    HAL_Write32(EVE_ENC_BITMAP_LAYOUT(format, linestride, height));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_SIZE(uint8_t filter, uint8_t wrapx, uint8_t wrapy, uint16_t width,
uint16_t height)
{
    HAL_Write32(EVE_ENC_BITMAP_SIZE(filter, wrapx, wrapy, width, height));
    HAL_IncCmdPointer(4);
}

void EVE_CELL(uint8_t cell)
```

```
{
    HAL_Write32(EVE_ENC_CELL(cell));
    HAL_IncCmdPointer(4);
}

void EVE_TAG(uint8_t s)
{
    HAL_Write32(EVE_ENC_TAG(s));
    HAL_IncCmdPointer(4);
}

void EVE_ALPHA_FUNC(uint8_t func, uint8_t ref)
{
    HAL_Write32(EVE_ENC_ALPHA_FUNC(func, ref));
    HAL_IncCmdPointer(4);
}

void EVE_STENCIL_FUNC(uint8_t func, uint8_t ref, uint8_t mask)
{
    HAL_Write32(EVE_ENC_STENCIL_FUNC(func, ref, mask));
    HAL_IncCmdPointer(4);
}

void EVE_BLEND_FUNC(uint8_t src, uint8_t dst)
{
    HAL_Write32(EVE_ENC_BLEND_FUNC(src, dst));
    HAL_IncCmdPointer(4);
}

void EVE_STENCIL_OP(uint8_t sfail, uint8_t spass)
{
    HAL_Write32(EVE_ENC_STENCIL_OP(sfail, spass));
    HAL_IncCmdPointer(4);
}

void EVE_POINT_SIZE(uint16_t size)
{
    HAL_Write32(EVE_ENC_POINT_SIZE(size));
    HAL_IncCmdPointer(4);
}

void EVE_LINE_WIDTH(uint16_t width)
{
    HAL_Write32(EVE_ENC_LINE_WIDTH(width));
    HAL_IncCmdPointer(4);
}

void EVE_CLEAR_COLOR_A(uint8_t alpha)
{
    HAL_Write32(EVE_ENC_CLEAR_COLOR_A(alpha));
    HAL_IncCmdPointer(4);
}

void EVE_COLOR_A(uint8_t alpha)
{
    HAL_Write32(EVE_ENC_COLOR_A(alpha));
    HAL_IncCmdPointer(4);
}

void EVE_CLEAR_STENCIL(uint8_t s)
{
```

```
    HAL_Write32(EVE_ENC_CLEAR_STENCIL(s));
    HAL_IncCmdPointer(4);
}

void EVE_CLEAR_TAG(uint8_t s)
{
    HAL_Write32(EVE_ENC_CLEAR_TAG(s));
    HAL_IncCmdPointer(4);
}

void EVE_STENCIL_MASK(uint8_t mask)
{
    HAL_Write32(EVE_ENC_STENCIL_MASK(mask));
    HAL_IncCmdPointer(4);
}

void EVE_TAG_MASK(uint8_t mask)
{
    HAL_Write32(EVE_ENC_TAG_MASK(mask));
    HAL_IncCmdPointer(4);
}

void EVE_SCISSOR_XY(uint16_t x, uint16_t y)
{
    HAL_Write32(EVE_ENC_SCISSOR_XY(x, y));
    HAL_IncCmdPointer(4);
}

void EVE_SCISSOR_SIZE(uint16_t width, uint16_t height)
{
    HAL_Write32(EVE_ENC_SCISSOR_SIZE(width, height));
    HAL_IncCmdPointer(4);
}

void EVE_CALL(uint16_t dest)
{
    HAL_Write32(EVE_ENC_CALL(dest));
    HAL_IncCmdPointer(4);
}

void EVE_JUMP(uint16_t dest)
{
    HAL_Write32(EVE_ENC_JUMP(dest));
    HAL_IncCmdPointer(4);
}

void EVE_BEGIN(uint8_t prim)
{
    HAL_Write32(EVE_ENC_BEGIN(prim));
    HAL_IncCmdPointer(4);
}

void EVE_COLOR_MASK(uint8_t r, uint8_t g, uint8_t b, uint8_t a)
{
    HAL_Write32(EVE_ENC_COLOR_MASK(r, g, b, a));
    HAL_IncCmdPointer(4);
}
```

```
void EVE_END(void)
{
    HAL_Write32(EVE_ENC_END());
    HAL_IncCmdPointer(4);
}

void EVE_SAVE_CONTEXT(void)
{
    HAL_Write32(EVE_ENC_SAVE_CONTEXT());
    HAL_IncCmdPointer(4);
}

void EVE_RESTORE_CONTEXT(void)
{
    HAL_Write32(EVE_ENC_RESTORE_CONTEXT());
    HAL_IncCmdPointer(4);
}

void EVE_RETURN(void)
{
    HAL_Write32(EVE_ENC_RETURN());
    HAL_IncCmdPointer(4);
}

void EVE_MACRO(uint8_t m)
{
    HAL_Write32(EVE_ENC_MACRO(m));
    HAL_IncCmdPointer(4);
}

void EVE_DISPLAY(void)
{
    HAL_Write32(EVE_ENC_DISPLAY());
    HAL_IncCmdPointer(4);
}

//##############################################################################
// Co-Processor Widgets
//##############################################################################

void EVE_CMD_TEXT(int16_t x, int16_t y, int16_t font, uint16_t options, const char*
string)
{
    uint32_t CommandSize;
    uint32_t StringLength;

    HAL_Write32(EVE_ENC_CMD_TEXT);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (font & 0xffff));
    CommandSize = 12;

    StringLength = EVE_LIB_SendString(string);
    CommandSize = CommandSize + StringLength;

    HAL_IncCmdPointer(CommandSize);
}

void EVE_CMD_BUTTON(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font,
uint16_t options, const char* string)
{
    uint32_t CommandSize;
```

```
    uint32_t StringLength;

    HAL_Write32(EVE_ENC_CMD_BUTTON);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (font & 0xffff));
    CommandSize = 16;

    StringLength = EVE_LIB_SendString(string);
    CommandSize = CommandSize + StringLength;

    HAL_IncCmdPointer(CommandSize);
}

void EVE_CMD_KEYS(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t
options, const char* string)
{
    uint32_t CommandSize;
    uint32_t StringLength;

    HAL_Write32(EVE_ENC_CMD_KEYS);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (font & 0xffff));
    CommandSize = 16;

    StringLength = EVE_LIB_SendString(string);
    CommandSize = CommandSize + StringLength;

    HAL_IncCmdPointer(CommandSize);
}

void EVE_CMD_NUMBER(int16_t x, int16_t y, int16_t font, uint16_t options, int32_t n)
{
    HAL_Write32(EVE_ENC_CMD_NUMBER);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (font & 0xffff));
    HAL_Write32(n);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_LOADIDENTITY(void)
{
    HAL_Write32(EVE_ENC_CMD_LOADIDENTITY);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_TOGGLE(int16_t x, int16_t y, int16_t w, int16_t font, uint16_t options,
uint16_t state, const char* string)
{
    uint32_t CommandSize;
    uint32_t StringLength;

    HAL_Write32(EVE_ENC_CMD_TOGGLE);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)font << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)state << 16)|options);
    CommandSize = 16;
```

```
    StringLength = EVE_LIB_SendString(string);
    CommandSize = CommandSize + StringLength;

    HAL_IncCmdPointer(CommandSize);
}

/* Error handling for val is not done, so better to always use range of 65535 in
order that needle is drawn within display region */
void EVE_CMD_GAUGE(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t major,
uint16_t minor, uint16_t val, uint16_t range)
{
    HAL_Write32(EVE_ENC_CMD_GAUGE);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (r & 0xffff));
    HAL_Write32(((uint32_t)minor << 16) | (major & 0xffff));
    HAL_Write32(((uint32_t)range << 16) | (val & 0xffff));
    HAL_IncCmdPointer(20);
}

void EVE_CMD_REGREAD(uint32_t ptr, uint32_t result)
{
    HAL_Write32(EVE_ENC_CMD_REGREAD);
    HAL_Write32(ptr);
    HAL_Write32(result);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_GETPROPS(uint32_t ptr, uint32_t w, uint32_t h)
{
    HAL_Write32(EVE_ENC_CMD_GETPROPS);
    HAL_Write32(ptr);
    HAL_Write32(w);
    HAL_Write32(h);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_MEMCPY(uint32_t dest, uint32_t src, uint32_t num)
{
    HAL_Write32(EVE_ENC_CMD_MEMCPY);
    HAL_Write32(dest);
    HAL_Write32(src);
    HAL_Write32(num);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_SPINNER(int16_t x, int16_t y, uint16_t style, uint16_t scale)
{
    HAL_Write32(EVE_ENC_CMD_SPINNER);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)scale << 16) | (style & 0xffff));
    HAL_IncCmdPointer(12);
}

void EVE_CMD_BGCOLOR(uint32_t c)
{
    HAL_Write32(EVE_ENC_CMD_BGCOLOR);
    HAL_Write32(c);
    HAL_IncCmdPointer(8);
}
```

```
void EVE_CMD_SWAP(void)
{
    HAL_Write32(EVE_ENC_CMD_SWAP);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_INFLATE(uint32_t ptr)
{
    HAL_Write32(EVE_ENC_CMD_INFLATE);
    HAL_Write32(ptr);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_TRANSLATE(int32_t tx, int32_t ty)
{
    HAL_Write32(EVE_ENC_CMD_TRANSLATE);
    HAL_Write32(tx);
    HAL_Write32(ty);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_STOP(void)
{
    HAL_Write32(EVE_ENC_CMD_STOP);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_SLIDER(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t range)
{
    HAL_Write32(EVE_ENC_CMD_SLIDER);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)val << 16) | (options & 0xffff));
    HAL_Write32(range);
    HAL_IncCmdPointer(20);
}

void EVE_BITMAP_TRANSFORM_A(long a)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_A(a)); //    ((21UL << 24) |
(((a)&131071UL)<<0))
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_TRANSFORM_B(long b)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_B(b)); //  ((22UL << 24) |
(((b)&131071UL)<<0))
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_TRANSFORM_C(long c)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_C(c)); //  ((23UL << 24) |
(((c)&16777215UL)<<0))
    HAL_IncCmdPointer(4);
}
```

```
void EVE_BITMAP_TRANSFORM_D(long d)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_D(d)); //   ((24UL << 24) |
(((d)&131071UL)<<0))
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_TRANSFORM_E(long e)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_E(e)); //   ((25UL << 24) |
(((e)&131071UL)<<0))
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_TRANSFORM_F(long f)
{
    HAL_Write32(EVE_ENC_BITMAP_TRANSFORM_F(f)); //  ((26UL << 24) |
(((f)&16777215UL)<<0))
    HAL_IncCmdPointer(4);
}

void EVE_CMD_INTERRUPT(uint32_t ms)
{
    HAL_Write32(EVE_ENC_CMD_INTERRUPT);
    HAL_Write32(ms);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_FGCOLOR(uint32_t c)
{
    HAL_Write32(EVE_ENC_CMD_FGCOLOR);
    HAL_Write32(c);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_ROTATE(int32_t a)
{
    HAL_Write32(EVE_ENC_CMD_ROTATE);
    HAL_Write32(a);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_MEMWRITE(uint32_t ptr, uint32_t num)
{
    HAL_Write32(EVE_ENC_CMD_MEMWRITE);
    HAL_Write32(ptr);
    HAL_Write32(num);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_SCROLLBAR(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t size, uint16_t range)
{
    HAL_Write32(EVE_ENC_CMD_SCROLLBAR);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)val << 16) | (options & 0xffff));
    HAL_Write32(((uint32_t)range << 16) | (size & 0xffff));
    HAL_IncCmdPointer(20);
}
```

```
void EVE_CMD_GETMATRIX(int32_t a, int32_t b, int32_t c, int32_t d, int32_t e, int32_t
f)
{
    HAL_Write32(EVE_ENC_CMD_GETMATRIX);
    HAL_Write32(a);
    HAL_Write32(b);
    HAL_Write32(c);
    HAL_Write32(d);
    HAL_Write32(e);
    HAL_Write32(f);
    HAL_IncCmdPointer(28);
}

void EVE_CMD_SKETCH(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t ptr,
uint16_t format)
{
    HAL_Write32(EVE_ENC_CMD_SKETCH);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(ptr);
    HAL_Write32(format);
    HAL_IncCmdPointer(20);
}

void EVE_CMD_MEMSET(uint32_t ptr, uint32_t value, uint32_t num)
{
    HAL_Write32(EVE_ENC_CMD_MEMSET);
    HAL_Write32(ptr);
    HAL_Write32(value);
    HAL_Write32(num);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_GRADCOLOR(uint32_t c)
{
    HAL_Write32(EVE_ENC_CMD_GRADCOLOR);
    HAL_Write32(c);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_BITMAP_TRANSFORM(int32_t x0, int32_t y0, int32_t x1, int32_t y1, int32_t
x2, int32_t y2, int32_t tx0, int32_t ty0, int32_t tx1, int32_t ty1, int32_t tx2,
int32_t ty2, uint16_t result)
{
    HAL_Write32(EVE_ENC_CMD_BITMAP_TRANSFORM);
    HAL_Write32(x0);
    HAL_Write32(y0);
    HAL_Write32(x1);
    HAL_Write32(y1);
    HAL_Write32(x2);
    HAL_Write32(y2);
    HAL_Write32(tx0);
    HAL_Write32(ty0);
    HAL_Write32(tx1);
    HAL_Write32(ty1);
    HAL_Write32(tx2);
    HAL_Write32(ty2);
```

```
    HAL_Write32(result);
    HAL_IncCmdPointer(56);
}

void EVE_CMD_CALIBRATE(uint32_t result)
{
    HAL_Write32(EVE_ENC_CMD_CALIBRATE);
    HAL_Write32(result);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_SETFONT(uint32_t font, uint32_t ptr)
{
    HAL_Write32(EVE_ENC_CMD_SETFONT);
    HAL_Write32(font);
    HAL_Write32(ptr);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_LOGO(void)
{
    HAL_Write32(EVE_ENC_CMD_LOGO);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_APPEND(uint32_t ptr, uint32_t num)
{
    HAL_Write32(EVE_ENC_CMD_APPEND);
    HAL_Write32(ptr);
    HAL_Write32(num);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_MEMZERO(uint32_t ptr, uint32_t num)
{
    HAL_Write32(EVE_ENC_CMD_MEMZERO);
    HAL_Write32(ptr);
    HAL_Write32(num);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_SCALE(int32_t sx, int32_t sy)
{
    HAL_Write32(EVE_ENC_CMD_SCALE);
    HAL_Write32(sx);
    HAL_Write32(sy);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_CLOCK(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t h,
uint16_t m, uint16_t s, uint16_t ms)
{
    HAL_Write32(EVE_ENC_CMD_CLOCK);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (r & 0xffff));
    HAL_Write32(((uint32_t)m << 16) | (h & 0xffff));
    HAL_Write32(((uint32_t)ms << 16) | (s & 0xffff));
    HAL_IncCmdPointer(20);
}
```

```
void EVE_CMD_GRADIENT(int16_t x0, int16_t y0, uint32_t rgb0, int16_t x1, int16_t y1,
uint32_t rgb1)
{
    HAL_Write32(EVE_ENC_CMD_GRADIENT);
    HAL_Write32(((uint32_t)y0 << 16) | (x0 & 0xffff));
    HAL_Write32(rgb0);
    HAL_Write32(((uint32_t)y1 << 16) | (x1 & 0xffff));
    HAL_Write32(rgb1);
    HAL_IncCmdPointer(20);
}

void EVE_CMD_SETMATRIX(void)
{
    HAL_Write32(EVE_ENC_CMD_SETMATRIX);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_TRACK(int16_t x, int16_t y, int16_t w, int16_t h, int16_t tag)
{
    HAL_Write32(EVE_ENC_CMD_TRACK);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(tag);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_GETPTR(uint32_t result)
{
    HAL_Write32(EVE_ENC_CMD_GETPTR);
    HAL_Write32(result);
    HAL_IncCmdPointer(8);
}


void EVE_CMD_PROGRESS(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t range)
{
    HAL_Write32(EVE_ENC_CMD_PROGRESS);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(((uint32_t)val << 16) | (options & 0xffff));
    HAL_Write32(range);
    HAL_IncCmdPointer(20);
}

void EVE_CMD_COLDSTART(void)
{
    HAL_Write32(EVE_ENC_CMD_COLDSTART);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_DIAL(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t val)
{
    HAL_Write32(EVE_ENC_CMD_DIAL);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)options << 16) | (r & 0xffff));
    HAL_Write32(val);
    HAL_IncCmdPointer(16);
```

```
}

void EVE_CMD_LOADIMAGE(uint32_t ptr, uint32_t options)
{
    HAL_Write32(EVE_ENC_CMD_LOADIMAGE);
    HAL_Write32(ptr);
    HAL_Write32(options);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_DLSTART(void)
{
    HAL_Write32(EVE_ENC_CMD_DLSTART);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_SNAPSHOT(uint32_t ptr)
{
    HAL_Write32(EVE_ENC_CMD_SNAPSHOT);
    HAL_Write32(ptr);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_SCREENSAVER(void)
{
    HAL_Write32(EVE_ENC_CMD_SCREENSAVER);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_MEMCRC(uint32_t ptr, uint32_t num, uint32_t result)
{
    HAL_Write32(EVE_ENC_CMD_MEMCRC);
    HAL_Write32(ptr);
    HAL_Write32(num);
    HAL_Write32(result);
    HAL_IncCmdPointer(16);
}

// FT81X-only features
#ifdef FT81X_ENABLE

// ###############       GPU        ###############################################

void EVE_VERTEX_FORMAT(uint8_t frac)
{
    HAL_Write32(EVE_ENC_VERTEX_FORMAT(frac));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_LAYOUT_H(uint8_t linestride, uint8_t height)
{
    HAL_Write32(EVE_ENC_BITMAP_LAYOUT_H(linestride, height));
    HAL_IncCmdPointer(4);
}

void EVE_BITMAP_SIZE_H(uint8_t width, uint8_t height)
{
    HAL_Write32(EVE_ENC_BITMAP_SIZE_H(width, height));
    HAL_IncCmdPointer(4);
}
```

```
void EVE_PALETTE_SOURCE(uint32_t addr)
{
    HAL_Write32(EVE_ENC_PALETTE_SOURCE(addr));
    HAL_IncCmdPointer(4);
}

void EVE_VERTEX_TRANSLATE_X(uint32_t x)
{
    HAL_Write32(EVE_ENC_VERTEX_TRANSLATE_X(x));
    HAL_IncCmdPointer(4);
}

void EVE_VERTEX_TRANSLATE_Y(uint32_t y)
{
    HAL_Write32(EVE_ENC_VERTEX_TRANSLATE_Y(y));
    HAL_IncCmdPointer(4);
}

void EVE_NOP(void)
{
    HAL_Write32(EVE_ENC_NOP());
    HAL_IncCmdPointer(4);
}

// ###############       CO-PRO        ##########################################


void EVE_CMD_SETROTATE(uint32_t r)
{
    HAL_Write32(EVE_ENC_CMD_SETROTATE);
    HAL_Write32(r);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_SETFONT2(uint32_t font, uint32_t ptr, uint32_t firstchar)
{
    HAL_Write32(EVE_ENC_CMD_SETFONT2);
    HAL_Write32(font);
    HAL_Write32(ptr);
    HAL_Write32(firstchar);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_SNAPSHOT2(uint32_t fmt, uint32_t ptr, int16_t x, int16_t y, int16_t w,
int16_t h)
{
    HAL_Write32(EVE_ENC_CMD_SNAPSHOT2);
    HAL_Write32(fmt);
    HAL_Write32(ptr);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_IncCmdPointer(20);
}

void EVE_CMD_MEDIAFIFO(uint32_t ptr, uint32_t size)
{
    HAL_Write32(EVE_ENC_CMD_MEDIAFIFO);
    HAL_Write32(ptr);
```

```
    HAL_Write32(size);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_INT_SWLOADIMAGE(uint32_t ptr, uint32_t options)
{
    HAL_Write32(EVE_ENC_CMD_INT_SWLOADIMAGE);
    HAL_Write32(ptr);
    HAL_Write32(options);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_SYNC(void)
{
    HAL_Write32(EVE_ENC_CMD_SYNC);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_CSKETCH(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t ptr,
uint16_t format, uint16_t freq)
{
    HAL_Write32(EVE_ENC_CMD_CSKETCH);
    HAL_Write32(((uint32_t)y << 16) | (x & 0xffff));
    HAL_Write32(((uint32_t)h << 16) | (w & 0xffff));
    HAL_Write32(ptr);
    HAL_Write32(((uint32_t)freq << 16) | (format & 0xffff));
    HAL_IncCmdPointer(20);
}

void EVE_CMD_ROMFONT(uint32_t font, uint32_t romslot)
{
    HAL_Write32(EVE_ENC_CMD_ROMFONT);
    HAL_Write32(font);
    HAL_Write32(romslot);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_PLAYVIDEO(uint32_t options)
{
    HAL_Write32(EVE_ENC_CMD_PLAYVIDEO);
    HAL_Write32(options);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_VIDEOFRAME(uint32_t dst, uint32_t ptr)
{
    HAL_Write32(EVE_ENC_CMD_VIDEOFRAME);
    HAL_Write32(dst);
    HAL_Write32(ptr);
    HAL_IncCmdPointer(12);
}

void EVE_CMD_VIDEOSTART(void)
{
    HAL_Write32(EVE_ENC_CMD_VIDEOSTART);
    HAL_IncCmdPointer(4);
}

void EVE_CMD_SETBASE(uint32_t base)
{
    HAL_Write32(EVE_ENC_CMD_SETBASE);
```

```
    HAL_Write32(base);
    HAL_IncCmdPointer(8);
}

void EVE_CMD_SETBITMAP(uint32_t source, uint16_t fmt, uint16_t w, uint16_t h)
{
    HAL_Write32(EVE_ENC_CMD_SETBITMAP);
    HAL_Write32(source);
    HAL_Write32(((uint32_t)w << 16) | (fmt & 0xffff));
    HAL_Write32(h);
    HAL_IncCmdPointer(16);
}

void EVE_CMD_SETSCRATCH(uint32_t handle)
{
    HAL_Write32(EVE_ENC_CMD_SETSCRATCH);
    HAL_Write32(handle);
    HAL_IncCmdPointer(8);
}
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
extern unsigned long POSX, POSY, BufferXY;
extern struct Boton Botones[64];
#define OFFSETBOTMATRIX 12
#define OFFSETBOTLIBRE 38
extern struct Boton Botones[64];
extern uint8_t TAG,PAG;

extern int page;
extern int VirTec;
extern bool ModTec[3];
extern uint8_t Tec;

extern uint16_t RedVal[4];
extern uint16_t GrnVal[4];
extern uint16_t BluVal[4];
uint16_t RedValPast[4]={0,0,0,0};
uint16_t GrnValPast[4]={0,0,0,0};
uint16_t BluValPast[4]={0,0,0,0};

extern uint16_t RedBot[4];
extern uint16_t GrnBot[4];
extern uint16_t BluBot[4];
uint16_t RedBotPast[4]={120,120,120,120};
uint16_t GrnBotPast[4]={0,0,0,0};;
uint16_t BluBotPast[4]={120,120,120,120};

extern bool SubmitState;
extern bool cambioF1;
extern bool cambioF2;
extern bool muestraIP;

struct touchscreen_calibration calib={.key=0,.transform={{0}}};
```

```
void EVE_LEER_PANTALLA(){
    BufferXY = HAL_MemRead32(EVE_REG_TOUCH_SCREEN_XY);
    POSX=(BufferXY&0xffff0000)>>16;
    POSY=BufferXY&0x0000FFFF;
}

uint8_t EVE_LEER_TAG(){
    uint8_t tag=0;
    uint8_t Read_tag;
    Read_tag = HAL_MemRead8(EVE_REG_TOUCH_TAG);
    if (!(HAL_MemRead16(EVE_REG_TOUCH_RAW_XY) & 0x8000))
        tag = Read_tag;
    return tag;
}

void EVE_INICIALIZAR_NUEVA_PANTALLA(){ //Pantalla con fondo negro
    if(RedVal[PAG-1]!=RedValPast[PAG-1] || GrnVal[PAG-1]!=GrnValPast[PAG-1]||
BluVal[PAG-1]!=BluValPast[PAG-1]||RedBot[PAG-1]!=RedBotPast[PAG-1] || GrnBot[PAG-
1]!=GrnBotPast[PAG-1]|| BluBot[PAG-1]!=BluBotPast[PAG-1]){
        cambioF2=true;
        RedValPast[PAG-1]=RedVal[PAG-1];
        GrnValPast[PAG-1]=GrnVal[PAG-1];
        BluValPast[PAG-1]=BluVal[PAG-1];
        RedBotPast[PAG-1]=RedBot[PAG-1];
        GrnBotPast[PAG-1]=GrnBot[PAG-1];
        BluBotPast[PAG-1]=BluBot[PAG-1];
    }
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_CMD_DLSTART(); // When executed, EVE will begin a new DL
    EVE_CLEAR_COLOR_RGB(RedVal[PAG-1], GrnVal[PAG-1], BluVal[PAG-1]); // Select color
to clear screen to
    EVE_CLEAR(1,1,1); // Clear

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}


void EVE_MOSTRAR_FIN_PANTALLA(){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_DISPLAY(); // Tells EVE that this is the end
    EVE_CMD_SWAP(); // Swaps new list into foreground buffer

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}

void EVE_LIMPIAR_PANTALLA_A(uint8_t R,uint8_t G,uint8_t B){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD

    EVE_CLEAR_COLOR_RGB(R, G, B); // Select color to clear screen to
    EVE_CLEAR(1,1,1); // Clear

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}

void EVE_COLOR_ELEMENTO(){
    EVE_LIB_BeginCoProList(); // CS low and send address in RAM_CMD
```

```
//    uint32_t Auxiliar=;
    EVE_COLOR_RGB((512-RedBot[PAG-1]-RedVal[PAG-1])/2, (512-GrnBot[PAG-1]-GrnVal[PAG-
1])/2, (512-BluBot[PAG-1]-BluVal[PAG-1])/2);
    EVE_CMD_FGCOLOR(RedBot[PAG-1]*0x10000+GrnBot[PAG-1]*0x100+BluBot[PAG-1]);

    EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
    EVE_LIB_AwaitCoProEmpty();
}

void EVE_BOTON_P(int16_t x,int16_t y,int16_t L,int16_t L2, uint8_t t){
    extern uint8_t TAG;
    EVE_TAG(t);
    if(TAG==t)
        EVE_CMD_BUTTON(x, y, L, L2, 28, EVE_OPT_FLAT, "\0");
    else
        EVE_CMD_BUTTON(x, y, L, L2, 28, EVE_OPT_CENTER, "\0");
}

void EVE_BOTON_P_2(uint8_t n){
    extern uint8_t TAG;
    EVE_TAG(Botones[n].tagAsoc);
    if(TAG==Botones[n].tagAsoc)
        EVE_CMD_BUTTON(Botones[n].fis.x, Botones[n].fis.y, Botones[n].fis.w,
Botones[n].fis.h, 28, EVE_OPT_FLAT, Botones[n].name);
    else
        EVE_CMD_BUTTON(Botones[n].fis.x, Botones[n].fis.y, Botones[n].fis.w,
Botones[n].fis.h, 28, EVE_OPT_CENTER, Botones[n].name);
}

void CREAR_BOTON(uint8_t origen,uint16_t x,uint16_t y,uint16_t w,uint16_t h, char
named[5],uint8_t pantallaAsocd, uint8_t tipod, uint16_t comandod){
    int i;
    uint8_t index;
    uint16_t comandoR=1;
    if(origen==0){
        index=0;
        comandoR=comandod;
    }
    else if(origen==1){
        index=OFFSETBOTMATRIX;
        for(i=OFFSETBOTMATRIX;i<64;i++){
            if(Botones[i].pantallaAsoc==PAG && Botones[i].BotAccion.comando!=0){
                comandoR++;
            }
        }
    }
    else if(origen==2){
        index=OFFSETBOTLIBRE;
        for(i=OFFSETBOTMATRIX;i<64;i++){
            if(Botones[i].pantallaAsoc==PAG && Botones[i].BotAccion.comando!=0){
                comandoR++;
            }
        }
    }

    while(Botones[index].pantallaAsoc!=16){
        index+=1;
    }
```

```
    Botones[index].fis.x=x;
    Botones[index].fis.y=y;
    Botones[index].fis.w=w;
    Botones[index].fis.h=h;
    strcpy(Botones[index].name,named);
    Botones[index].tagAsoc=index+1;
    Botones[index].pantallaAsoc=pantallaAsocd;
    Botones[index].BotAccion.tipo=tipod;
    Botones[index].BotAccion.comando=comandoR;
    cambioF1=true;
}

void EVE_EDIT_MODE_L(){
    int Xi,Yi,Xf,Yf;
    int caso=0;
    while(TAG!=0){
            EVE_LEER_PANTALLA();

            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }

    while(TAG!=Botones[1].tagAsoc)
    {
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        ////////////////////////////////////////////
        switch (caso) {
        case 0:
            if(BufferXY!=0x80008000)
                caso=1;
            break;
        case 1:
            Xi=POSX;
            Yi=POSY;
            Xf=POSX;
            Yf=POSY;
            caso=2;
            break;
        case 2:
            EVE_LIB_BeginCoProList();

            EVE_CMD_BUTTON(Xi,Yi,Xf-Xi,Yf-Yi,28,EVE_OPT_CENTER,"\0");

            EVE_LIB_EndCoProList(); // CS high and update REG_CMD_WRITE
            EVE_LIB_AwaitCoProEmpty();

            if(BufferXY==0x80008000)
                caso=3;
            else{
                Xf=POSX;
                Yf=POSY;
            }
            break;
        case 3:
```

```
            CREAR_BOTON(2,Xi,Yi,Xf-Xi,Yf-Yi,"LIB",PAG,10,0);
            caso=0;
            break;
        }

        EVE_LEER_PANTALLA();

        EVE_MOSTRAR_BOTONES_GLOBAL();

        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();

    }
    while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
}

void EVE_MATRIX_BOTON(int n){
    const int LilOff= 50;
    const int SupOff= 20;
    const int BigOff=65;
    const int Lmin=60;
    const int SepMin=50;
    int Xu,Yu,Bfil,Bcol,Lmax,i,j,x,y;
    int L=0;
    if(n!=0){
        Bcol=1;
        Xu=800-2*LilOff;
        Yu=480-SupOff-BigOff;

        while(L<Lmin){
            Lmax=Yu/(Bcol+1)-20;
            Bfil=n/Bcol;
            L=(Xu-Bfil*SepMin)/Bfil;
            if(L>Lmax)
                L=Lmax;
            else if(L<Lmin)
                Bcol+=1;
        }
        for (i=1; i<=Bcol; i++){
            for (j=1; j<=Bfil; j++){
                x=j*(Xu/(Bfil+1))+LilOff;
                y=i*(Yu/(Bcol+1))+LilOff;
                CREAR_BOTON(1,x-L/2,y-L/2,L,L,"MAT",PAG,10,0);
            }
        }
    }

}
```

```
void EVE_EDIT_MODE_MATRIX(){
    int n=0;
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    }

    while(TAG!=Botones[0].tagAsoc){
        EVE_LEER_PANTALLA();
        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();

        EVE_LIB_BeginCoProList();

        EVE_BOTON_P(350,190,100,100,Botones[0].tagAsoc+1);
        EVE_BOTON_P_2(0); //Boton de EditMatrix
        EVE_CMD_NUMBER(400,300,28,EVE_OPT_CENTER,n);

        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();

        if(TAG==Botones[0].tagAsoc+1){
            n+=2;
            while(TAG==Botones[0].tagAsoc+1){
                EVE_LEER_PANTALLA();
                TAG=EVE_LEER_TAG();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_LIB_BeginCoProList();

                EVE_BOTON_P(350,190,100,100,Botones[0].tagAsoc+1);
                EVE_BOTON_P_2(0); //Boton de EditMatrix
                EVE_CMD_NUMBER(400,300,28,EVE_OPT_CENTER,n);

                EVE_LIB_EndCoProList();
                EVE_LIB_AwaitCoProEmpty();
                EVE_MOSTRAR_FIN_PANTALLA();
            }
        }
    }
    while(TAG!=0){
        EVE_LEER_PANTALLA();
        TAG=EVE_LEER_TAG();
    }
    EVE_MATRIX_BOTON(n);
}

void WEB_SUBMITER(){
    uint8_t comandoR=0;
    uint8_t i=OFFSETBOTMATRIX;
```

```
    uint8_t TeclaAPulsar;
    uint8_t aux=0;
    char NameTeclaAPulsar[6]="\0\0\0\0\0\0";
    while(comandoR!=VirTec && i<64){
        i++;
        if(Botones[i-1].pantallaAsoc==page){
            comandoR++;
        }


    }
    if(ModTec[0]){
        NameTeclaAPulsar[aux]= 'C';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;
    }
    if(ModTec[1]){
        NameTeclaAPulsar[aux]= 'A';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;

    }
    if(ModTec[2]){
        NameTeclaAPulsar[aux]= 'S';
        NameTeclaAPulsar[aux+1]= '+';
        aux++;
    }
    if(NameTeclaAPulsar[aux]=='+'){
        aux++;
    }

    if(Tec<8){
        TeclaAPulsar=30 + Tec;
        NameTeclaAPulsar[aux]= '1' + Tec;
    }
    else if(Tec==9){
        TeclaAPulsar=0x27;
        NameTeclaAPulsar[aux]= '0';
    }
    else{
        TeclaAPulsar=0x04+Tec-10;
        NameTeclaAPulsar[aux]= 'A'+ Tec-10;
    }
    strcpy(Botones[i-1].name,NameTeclaAPulsar);
    Botones[i-1].BotAccion.comando=TeclaAPulsar;
    Botones[i-1].BotAccion.mod[0]=ModTec[0];
    Botones[i-1].BotAccion.mod[1]=ModTec[1];
    Botones[i-1].BotAccion.mod[2]=ModTec[2];
    Botones[i-1].BotAccion.mod[3]=1;
    cambioF1=true;


}

void EVE_HANDLER_ACCION(){
    uint8_t accion=0;
    uint8_t TAG_P=TAG;
    uint8_t i;
```

```
if(SubmitState){
    SubmitState=false;
    WEB_SUBMITER();
}

if(TAG!=0){
    accion=Botones[TAG-1].BotAccion.tipo;
    switch (accion) {
    default:
        while(TAG!=0){
            EVE_LEER_PANTALLA();

            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }
        break;

    case 1:
        EVE_EDIT_MODE_MATRIX();
        break;
    case 2:
        EVE_EDIT_MODE_L();
        break;
    case 3:
        while(TAG!=0){
            EVE_LEER_PANTALLA();

            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }
        if(PAG>=4)
            PAG=1;
        else
            PAG+=1;
        break;

    case 4:
        while(TAG!=0){
            EVE_LEER_PANTALLA();

            EVE_INICIALIZAR_NUEVA_PANTALLA();
            EVE_COLOR_ELEMENTO();
            EVE_MOSTRAR_BOTONES_GLOBAL();
            EVE_MOSTRAR_FIN_PANTALLA();

            TAG=EVE_LEER_TAG();
        }
        if(PAG<=1)
            PAG=4;
        else
            PAG-=1;
        break;
```

```
case 5:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    };
    while(TAG!=5){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
        if(TAG>=OFFSETBOTMATRIX)
            EVE_ERASE(TAG);
    }
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    };
    break;

case 6:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();

        TAG=EVE_LEER_TAG();
    }
    muestraIP=!muestraIP;
    break;

case 7:
    while(TAG!=0){
        EVE_LEER_PANTALLA();

        EVE_INICIALIZAR_NUEVA_PANTALLA();
        EVE_COLOR_ELEMENTO();
        EVE_MOSTRAR_BOTONES_GLOBAL();
        EVE_MOSTRAR_FIN_PANTALLA();
```

```
                TAG=EVE_LEER_TAG();
            }
            for(i=0;i<6;i++){
                calib.transform[i]=0;
            }
            eve_calibrate();
            cambioF2=true;

            break;

        case 10:
            while(TAG!=0){
                EVE_LEER_PANTALLA();

                EVE_INICIALIZAR_NUEVA_PANTALLA();
                EVE_COLOR_ELEMENTO();
                EVE_MOSTRAR_BOTONES_GLOBAL();
                EVE_MOSTRAR_FIN_PANTALLA();

                TAG=EVE_LEER_TAG();
            }
            EVE_ACTUADOR(TAG_P);
            break;
        }

    }
}
void EVE_ERASE(uint8_t boton){
    uint8_t index=OFFSETBOTMATRIX;
    while(Botones[index].tagAsoc!=boton && index<65){
        index+=1;
    }
    if(index<65){
        Botones[index].fis.x=0;
        Botones[index].fis.y=0;
        Botones[index].fis.w=0;
        Botones[index].fis.h=0;
        Botones[index].name[0]='\0';
        Botones[index].tagAsoc=0;
        Botones[index].pantallaAsoc=16;
        Botones[index].BotAccion.tipo=0;
        Botones[index].BotAccion.comando=0;
        Botones[index].BotAccion.mod[0]=0;
        Botones[index].BotAccion.mod[1]=0;
        Botones[index].BotAccion.mod[2]=0;
        Botones[index].BotAccion.mod[3]=0;
        cambioF1=true;
    }
}

int i;
void EVE_MOSTRAR_BOTONES_GLOBAL(){
    EVE_LIB_BeginCoProList();
    for(i=0;i<64;i++){
        if(Botones[i].pantallaAsoc==0 || Botones[i].pantallaAsoc==PAG){
            EVE_BOTON_P_2(i);
        }
    }
    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();
```

```
}

void EVE_ACTUADOR(uint8_t tag){
    if(Botones[tag-1].BotAccion.mod[3]){
        uint8_t AuxModTec=0;
        if(Botones[tag-1].BotAccion.mod[0])
            AuxModTec=AuxModTec|0x01;
        if(Botones[tag-1].BotAccion.mod[1])
            AuxModTec=AuxModTec|0x04;
        if(Botones[tag-1].BotAccion.mod[2])
            AuxModTec=AuxModTec|0x02;

        USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,AuxModTec,Botones[tag-
1].BotAccion.comando,1);
        SysCtlDelay(40*40000);
        USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,0,Botones[tag-
1].BotAccion.comando,0);
        SysCtlDelay(40*40000);

    }
    else{

USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][0],EveKeypUsageCodes[Botones[tag-1].BotAccion.comando-
1][1],1);
        SysCtlDelay(20*40000);

USBDHIDKeyboardKeyStateChange(&g_sKeyboardDevice,0,EveKeypUsageCodes[Botones[tag-
1].BotAccion.comando-1][1],0);
        SysCtlDelay(20*40000);
    }
}
void INIT_ESTRUCT_PRINC(){
    uint8_t i;
    for(i=0;i<64;i++){

            Botones[i].fis.x=0;
            Botones[i].fis.y=0;
            Botones[i].fis.w=0;
            Botones[i].fis.h=0;
            strcpy(Botones[i].name,"\0\0\0\0\0\0");
            Botones[i].tagAsoc=0;
            Botones[i].pantallaAsoc=16;
            Botones[i].BotAccion.tipo=0;
            Botones[i].BotAccion.comando=0;
            Botones[i].BotAccion.mod[0]=0;
            Botones[i].BotAccion.mod[1]=0;
            Botones[i].BotAccion.mod[2]=0;
            Botones[i].BotAccion.mod[3]=0;
        }
}

uint32_t posicion=0x50000;
uint32_t posicionColor=0x60000;

void FLASH_STOCKER_I(uint8_t i){
    uint32_t posicionAux=posicion+i*24;
    uint32_t posAux;
```

```
    if(Botones[i].fis.w!=0 && Botones[i].fis.h!=0){
    posAux=0;
    posAux|=Botones[i].fis.x;
    posAux|=Botones[i].fis.y<<16;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].fis.w;
    posAux|=Botones[i].fis.h<<16;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].name[0]<<0;
    posAux|=Botones[i].name[1]<<8;
    posAux|=Botones[i].name[2]<<16;
    posAux|=Botones[i].name[3]<<24;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].name[4]<<0;
    posAux|=Botones[i].name[5]<<8;
    posAux|=Botones[i].tagAsoc<<16;
    posAux|=Botones[i].pantallaAsoc<<24;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=Botones[i].BotAccion.tipo<<0;
    posAux|=Botones[i].BotAccion.comando<<8;
    posAux|=Botones[i].BotAccion.mod[0]<<24;
    posAux|=Botones[i].BotAccion.mod[1]<<25;
    posAux|=Botones[i].BotAccion.mod[2]<<26;
    posAux|=Botones[i].BotAccion.mod[3]<<27;
    FlashProgram(&posAux,posicionAux,4);
    }
}

void FLASH_FISHER_I(uint8_t i){
    uint32_t posicionAux=posicion+i*24;
    uint32_t posAux=0;

    if(HWREG(posicionAux)!=0xffffffff && HWREG(posicionAux+4)!=0xffffffff &&
HWREG(posicionAux+8)!=0xffffffff && HWREG(posicionAux+12)!=0xffffffff){
    posAux=HWREG(posicionAux);
    Botones[i].fis.x=(uint16_t)(posAux & 0xffff);
    Botones[i].fis.y=(uint16_t)(posAux >>16);
    posicionAux+=4;

    posAux=0;
    posAux=HWREG(posicionAux);
    Botones[i].fis.w=(uint16_t)(posAux & 0xffff);
    Botones[i].fis.h=(uint16_t)(posAux >>16);
    posicionAux+=4;

    posAux=0;
    posAux=HWREG(posicionAux);
    Botones[i].name[0]=(char)(posAux & 0xffffff);
```

```
    Botones[i].name[1]=(char)((posAux & 0xffff)>>8);
    Botones[i].name[2]=(char)(posAux>>16 & 0xff);
    Botones[i].name[3]=(char)(posAux >> 24);
    posicionAux+=4;

    posAux=0;
    posAux=HWREG(posicionAux);
    Botones[i].name[4]=(char)(posAux & 0xffffff);
    Botones[i].name[5]=(char)((posAux & 0xffff)>>8);
    Botones[i].tagAsoc=(uint8_t)(posAux>>16 & 0xff);
    Botones[i].pantallaAsoc=(uint8_t)(posAux >> 24);
    posicionAux+=4;

    posAux=0;
    posAux=HWREG(posicionAux);
    Botones[i].BotAccion.tipo=(uint8_t)(posAux & 0x000000ff);
    Botones[i].BotAccion.comando=(uint16_t)((posAux & 0x00ffff00)>>8);
    Botones[i].BotAccion.mod[0]=(bool)(posAux>>24 & 0x01);
    Botones[i].BotAccion.mod[1]=(bool)(posAux>>25 & 0b0000001);
    Botones[i].BotAccion.mod[2]=(bool)(posAux>>26 & 0b000001);
    Botones[i].BotAccion.mod[3]=(bool)(posAux>>27 & 0b00001);
    }
}


void FLASH_STOCKER_C(uint8_t i){
    i-=1;
    uint32_t posicionAux=posicionColor+i*8;
    uint32_t posAux;

    posAux=0;
    posAux|=RedVal[i];
    posAux|=GrnVal[i]<<8;
    posAux|=BluVal[i]<<16;
    posAux|=RedBot[i]<<24;
    FlashProgram(&posAux,posicionAux,4);
    posicionAux+=4;

    posAux=0;
    posAux|=GrnBot[i];
    posAux|=BluBot[i]<<8;
    FlashProgram(&posAux,posicionAux,4);
}

void FLASH_FISHER_C(uint8_t i){
    i-=1;
    uint32_t posicionAux=posicionColor+i*8;
    uint32_t posAux;

    posAux=HWREG(posicionAux);
    RedVal[i]=(uint16_t)(posAux & 0x000000ff);
    GrnVal[i]=(uint16_t)(posAux>>8 & 0x0000ff);
    BluVal[i]=(uint16_t)(posAux>>16 & 0x00ff);
    RedBot[i]=(uint16_t)(posAux >>24);
    posicionAux+=4;

    posAux=HWREG(posicionAux);
    GrnBot[i]=(uint16_t)(posAux & 0x000000ff);
```

```
    BluBot[i]=(uint16_t)(posAux>>8 & 0x0000ff);


    }


void FLASH_RENEW(){
    uint8_t i;
    if(cambioF1){
    FlashErase(posicion);
    for(i=0;i<64;i++){
        FLASH_STOCKER_I(i);
        cambioF1=false;

    }
    }
    if(cambioF2){
        FlashErase(posicionColor);
        for(i=1;i<5;i++){
            FLASH_STOCKER_C(i);
            CALIB_STOCKER();
            cambioF2=false;
        }
    }
}

void CALIB_STOCKER(){
    uint32_t posicionAux=posicionColor+4*8;
    uint8_t i;
    for(i=0;i<6;i++){
        FlashProgram(&calib.transform[i],posicionAux,4);
        posicionAux+=4;
    }
}
void CALIB_FISHER(){
    uint32_t posicionAux=posicionColor+4*8;
    if(HWREG(posicionAux)!=0XFFFFFFFF){
        for(i=0;i<6;i++){
            calib.transform[i]=HWREG(posicionAux);
            posicionAux+=4;
        }
    }
}

extern uint32_t g_ui32IPAddress;
char pcBuf[16];
#include "ustdlib.h"
void MOSTRAR_IP(){

    EVE_LIB_BeginCoProList();

    usprintf(pcBuf, "%d.%d.%d.%d", g_ui32IPAddress & 0xff, (g_ui32IPAddress >> 8) &
0xff,
             (g_ui32IPAddress >> 16) & 0xff, (g_ui32IPAddress >> 24) & 0xff);
    EVE_CMD_TEXT(400,380,28,EVE_OPT_CENTER,pcBuf);

    EVE_LIB_EndCoProList();
    EVE_LIB_AwaitCoProEmpty();

}
```

```
#endif  /* FT81X_ENABLE */
```

Eve_calibrate.c

```
include <eveheader.h>
#include <stdint.h>


int8_t platform_calib_init(void)
{
    return 1;
}

int8_t platform_calib_write(struct touchscreen_calibration *calib)
{
    return 0;
}

int8_t platform_calib_read(struct touchscreen_calibration *calib)
{
    return -1;
}
extern struct touchscreen_calibration calib;

void eve_calibrate(void)
{
    uint8_t dummy;

    platform_calib_init();

    // If no store of calibration or current screen touch.
    if (calib.transform[0]==0 && calib.transform[1]==0 && calib.transform[2]==0
            && calib.transform[3]==0 && calib.transform[4]==0 &&
calib.transform[5]==0){
        // Wait for end of touch.
        while (eve_read_tag(&dummy));

        EVE_LIB_BeginCoProList();
        EVE_CMD_DLSTART();
        EVE_CLEAR_COLOR_RGB(0, 0, 0);
        EVE_CLEAR(1,1,1);
        EVE_COLOR_RGB(255, 255, 255);
        EVE_CMD_TEXT(EVE_DISP_WIDTH/2, EVE_DISP_HEIGHT/2,
                     28, EVE_OPT_CENTERX | EVE_OPT_CENTERY,"Please tap on the dots");
        EVE_CMD_CALIBRATE(0);
        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();

        calib.transform[0] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_A);
        calib.transform[1] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_B);
        calib.transform[2] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_C);
        calib.transform[3] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_D);
        calib.transform[4] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_E);
        calib.transform[5] = HAL_MemRead32(EVE_REG_TOUCH_TRANSFORM_F);
        platform_calib_write(&calib);
    }
```

```
      else
      {
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_A, calib.transform[0]);
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_B, calib.transform[1]);
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_C, calib.transform[2]);
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_D, calib.transform[3]);
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_E, calib.transform[4]);
        HAL_MemWrite32(EVE_REG_TOUCH_TRANSFORM_F, calib.transform[5]);
    }
}

uint8_t eve_read_tag(uint8_t *key)
{
    uint8_t Read_tag;
    uint8_t key_detect = 0;

    Read_tag = HAL_MemRead8(EVE_REG_TOUCH_TAG);

    if (!(HAL_MemRead16(EVE_REG_TOUCH_RAW_XY) & 0x8000))
    {
        key_detect = 1;
        *key = Read_tag;
    }

    return key_detect;
}
```

EVE_MCU_TM4C1294XL.c

```
#include <eveheader.h>
#include <stdbool.h>
#include <stdint.h>

#include "driverlib2.h"


// ===========================================================================
// Function Declarations
// ===========================================================================
#define dword long
#define byte char
#define CPU_FREQ        1000

int RELOJ;


//
############################################################################################
########################
// User Application - Initialization of MCU / FT800 / Display
//
############################################################################################
########################

uint32_t SSI_BASE,CS_PORT,CS_PIN,PD_PORT,PD_PIN;

//
############################################################################################
########################
// FT800 SPI Functions
```

```
//
//*****************************************************************************
**********************

//
//*****************************************************************************
**********************
//
//*****************************************************************************
**********************
//                                      HAL functions
//
//*****************************************************************************
**********************
//
//*****************************************************************************
**********************

void initClock()
{
    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
SYSCTL_CFG_VCO_480), 120000000);
}

void initSPI()
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI2);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    //
    //  pin PN3 es /CS
    //
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_3);
    CS_PORT=GPIO_PORTN_BASE;
    CS_PIN=GPIO_PIN_3;

    //
    // pin PN2 es /PD
    //
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_2);
    PD_PORT=GPIO_PORTN_BASE;
    PD_PIN=GPIO_PIN_2;

    //
    //  PD0 es SSI2 SSI2XDAT1
    //
    GPIOPinConfigure(GPIO_PD0_SSI2XDAT1);
    GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_0);

    //
    // PD1 es SSI2 SSI2XDAT0
    //
    GPIOPinConfigure(GPIO_PD1_SSI2XDAT0);
    GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_1);

    //
    // PD3 es SSI2 SSI2CLK
    //
```

```
    GPIOPinConfigure(GPIO_PD3_SSI2CLK);
    GPIOPinTypeSSI(GPIO_PORTD_BASE, GPIO_PIN_3);
    SSI_BASE=SSI2_BASE;

    SSIConfigSetExpClk(SSI_BASE, RELOJ, SSI_FRF_MOTO_MODE_0,
                SSI_MODE_MASTER, 1000000, 8);
    //
    // Enable the SSI module.
    //
    SSIEnable(SSI_BASE);

    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);

}

/* configure MCU, SPI and PD pins */
void MCU_Init(void){

    initClock();                            // set Clocks
    initSPI();                              // configure SPI


}

void MCU_Setup(void)
{
/* function to configure QSPI*/
//#ifdef FT81X_ENABLE

//#endif // FT81X_ENABLE
}


// ############################ SPI Send and Receive
#####################################
// --------------------- SPI Read/Write 8 bits ---------------------------------
uint8_t MCU_SPIReadWrite8(uint8_t DataToWrite)
{
    uint32_t DataRead;
    SSIDataPut(SSI_BASE,DataToWrite);
    while(SSIBusy(SSI_BASE));

    SSIDataGet(SSI_BASE, &DataRead);

    return (uint8_t)DataRead;
}

// --------------------- SPI Read/Write 16 bits ---------------------------------
uint16_t MCU_SPIReadWrite16(uint16_t DataToWrite)
{
    uint16_t DataRead = 0;
    DataRead = MCU_SPIReadWrite8((DataToWrite) >> 8) << 8;
    DataRead |= MCU_SPIReadWrite8((DataToWrite) & 0xff);

    return DataRead;
}

// --------------------- SPI Read/Write 24 bits ---------------------------------
uint32_t MCU_SPIReadWrite24(uint32_t DataToWrite)
{
```

```
    uint32_t DataRead = 0;
    uint32_t temp;

    temp = (MCU_SPIReadWrite8((DataToWrite) >> 24));
    DataRead |= (temp<<24);
    temp = (MCU_SPIReadWrite8((DataToWrite) >> 16));
    DataRead |= (temp<<16);
    temp = (MCU_SPIReadWrite8((DataToWrite) >> 8));
    DataRead |= (temp<<8);

    return DataRead;
}

// --------------------- SPI Read/Write 32 bits ---------------------------------
uint32_t MCU_SPIReadWrite32(uint32_t DataToWrite)
{
    uint32_t DataRead = 0;
    uint32_t temp;

    temp = (MCU_SPIReadWrite8((DataToWrite) >> 24));
    DataRead |= (temp << 24);
    temp = (MCU_SPIReadWrite8((DataToWrite) >> 16));
    DataRead |= (temp << 16);
    DataRead |= (MCU_SPIReadWrite8((DataToWrite) >> 8) << 8);
    DataRead |= (MCU_SPIReadWrite8(DataToWrite) & 0xff);

    return DataRead;
}

// --------------------- SPI Read 8 bits ----------------------------------
uint8_t MCU_SPIRead8(void)
{
    uint8_t DataRead = 0;

    DataRead = MCU_SPIReadWrite8(0);

    return DataRead;
}

// --------------------- SPI Write 8 bits ----------------------------------
void MCU_SPIWrite8(uint8_t DataToWrite)
{
    MCU_SPIReadWrite8(DataToWrite);
}

// --------------------- SPI Read 16 bits ----------------------------------
uint16_t MCU_SPIRead16(void)
{
    uint16_t DataRead = 0;

    DataRead = MCU_SPIReadWrite16(0);

    return DataRead;
}

// --------------------- SPI Write 16 bits ----------------------------------
void MCU_SPIWrite16(uint16_t DataToWrite)
{
```

```
    MCU_SPIReadWrite16(DataToWrite);
}

// --------------------- SPI Read 24 bits --------------------------------
uint32_t MCU_SPIRead24(void)
{
    uint32_t DataRead = 0;

    DataRead = MCU_SPIReadWrite24(0);

    return DataRead;
}

// --------------------- SPI Write 24 bits --------------------------------
void MCU_SPIWrite24(uint32_t DataToWrite)
{
    MCU_SPIReadWrite24(DataToWrite);
}

uint32_t MCU_SPIRead32(void)
{
    uint32_t DataRead = 0;

    DataRead = MCU_SPIReadWrite32(0);

    return DataRead;
}

// --------------------- SPI Read/Write 32 bits --------------------------------
void MCU_SPIWrite32(uint32_t DataToWrite)
{
    MCU_SPIReadWrite32(DataToWrite);
}

// --------------------- SPI burst write --------------------------------
void MCU_SPIWrite(const uint8_t *DataToWrite, uint32_t length)
{
    uint16_t DataPointer = 0;
    DataPointer = 0;

    while(DataPointer < length)
    {
        MCU_SPIWrite8(DataToWrite[DataPointer]);  // Send data byte-by-byte from
array
        DataPointer ++;
    }
}

// ############################ GPIO CONTROL ###################################

// --------------------- Chip Select line low --------------------------------
inline void MCU_CSlow(void)
{
    GPIOPinWrite(CS_PORT, CS_PIN, 0);
}

// --------------------- Chip Select line high --------------------------------
inline void MCU_CShigh(void)
{
    GPIOPinWrite(CS_PORT, CS_PIN, CS_PIN);
}
```

```
// ------------------------- PD line low -----------------------------------
inline void MCU_PDlow(void)
{
    GPIOPinWrite(PD_PORT, PD_PIN, 0);
}

// ------------------------- PD line high ----------------------------------
inline void MCU_PDhigh(void)
{
    GPIOPinWrite(PD_PORT, PD_PIN, PD_PIN);
}

// ------------------- msec delay based on MCLK (CPU_FREQ) ---------------------
void delay(int msec){
    //could use a timer here
    while(msec)
    {
        SysCtlDelay(CPU_FREQ);
        msec--;
    }
}
// ---------------------------- 20ms delay -------------------------------------
void MCU_Delay_20ms(void)
{
    SysCtlDelay(20*40000);
}

// ---------------------------- 500ms delay -------------------------------------
void MCU_Delay_500ms(void)
{
    SysCtlDelay(500*40000);
}

// ############################ ENDIAN CONVERSION ###################################

uint32_t bswap32(uint32_t x)
{
    uint32_t s;
    s = ((x) >> 24);
    s |= (((x) & 0x00FF0000) >> 8);
    s |= (((x) & 0x0000FF00) << 8);
    s |= ((x) << 24);

    return s;
}

uint16_t bswap16(uint16_t x)
{
    uint16_t s;
    s = ((x) >> 8);
    s |= ((x) << 8);

    return s;
}

uint16_t MCU_htobe16 (uint16_t h)
{
```

```
    return h;
}

uint32_t MCU_htobe32 (uint32_t h)
{
    return h;
}

uint16_t MCU_htole16 (uint16_t h)
{

        return bswap16(h);
}

uint32_t MCU_htole32 (uint32_t h)
{

        return bswap32(h);
}

uint16_t MCU_be16toh (uint16_t h)
{
    return h;
}
uint32_t MCU_be32toh (uint32_t h)
{
     return h;
}

uint16_t MCU_le16toh (uint16_t h)
{

        return bswap16(h);
}

uint32_t MCU_le32toh (uint32_t h)
{
        return bswap32(h);

}
```

Eve_Hal.c

```
#if !defined(USE_LINUX_SPI_DEV) || defined(USE_MPSSE)

#include <string.h>
#include <stdint.h> // for Uint8/16/32 and Int8/16/32 data types

#include <eveheader.h>

// Used to navigate command ring buffer
static uint16_t writeCmdPointer = 0x0000;


void HAL_EVE_Init(void)
{
```

```
	uint8_t val;
	MCU_Init();

	// Set Chip Select OFF
	HAL_ChipSelect(0);

	// Reset the display
	MCU_Delay_20ms();
	HAL_PowerDown(1);
	MCU_Delay_20ms();
	HAL_PowerDown(0);
	MCU_Delay_20ms();

#ifndef FT81X_ENABLE
	// FT80x_selection - FT80x modules from BRT generally use external crystal
    // You can also send the host command to set the PLL here if you want to change
it from the default of 48MHz (FT80x) or 60MHz (FT81x)
	// Clock selection and clock rate selection will put EVE to sleep and so must
be before the Active command
	// for example:
	HAL_HostCmdWrite(0x44, 0x00); // 0x44 = HostCMD_CLKEXT
	HAL_HostCmdWrite(0x62, 0x00); // 0x64 = HostCMD_CLK48M
#endif

	// Set active
	HAL_HostCmdWrite(0, 0x00);

	MCU_Delay_500ms();

	// Read REG_ID register (0x302000) until reads 0x7C
	while ((val = HAL_MemRead8(EVE_REG_ID)) != 0x7C)
	{
		MCU_Delay_20ms();
	}

	// Ensure CPUreset register reads 0 and so FT8xx is ready
	while (HAL_MemRead8(EVE_REG_CPURESET) != 0x00)
	{
	}

	// This function will not return unless and FT8xx device is present.
	MCU_Setup();
}

// ---------------------- Chip Select line ---------------------------------
void HAL_ChipSelect(int8_t enable)
{
	if (enable)
		MCU_CSlow();
	else
		MCU_CShigh();
}

// ------------------------- Power Down line -------------------------------------
void HAL_PowerDown(int8_t enable)
{
	if (enable)
		MCU_PDlow();
```

```
	else
		MCU_PDhigh();
}

// ------------------ Send FT81x register address for writing ------------------
void HAL_SetWriteAddress(uint32_t address)
{
	// Send three bytes of a register address which has to be subsequently
	// written. Ignore return values as this is an SPI write only.
	// Send high byte of address with 'write' bits set.
	MCU_SPIWrite24(MCU_htobe32((address << 8) | (1UL << 31)));
}

// ------------------ Send FT81x register address for reading ------------------
void HAL_SetReadAddress(uint32_t address)
{
	// Send three bytes of a register address which has to be subsequently read.
	// Ignore return values as this is an SPI write only.
	// Send high byte of address with 'read' bits set.
	MCU_SPIWrite32(MCU_htobe32((address << 8) | (0UL << 31)));
}

// ------------------------ Send a block of data -------------------------
void HAL_Write(const uint8_t *buffer, uint32_t length)
{
	// Send multiple bytes of data after previously sending address. Ignore return
	// values as this is an SPI write only. Data must be the correct endianess
	// for the SPI bus.
	MCU_SPIWrite(buffer, length);
}

// ------------------------ Send a 32-bit data value -------------------------
void HAL_Write32(uint32_t val32)
{
	// Send four bytes of data after previously sending address. Ignore return
	// values as this is an SPI write only.
	MCU_SPIWrite32(MCU_htole32(val32));
}

// ------------------------ Send a 16-bit data value -------------------------
void HAL_Write16(uint16_t val16)
{
	// Send two bytes of data after previously sending address. Ignore return
	// values as this is an SPI write only.
	MCU_SPIWrite16(MCU_htole16(val16));
}

// ------------------------ Send an 8-bit data value -------------------------
void HAL_Write8(uint8_t val8)
{
	// Send one byte of data after previously sending address. Ignore return
	// values as this is an SPI write only.
	MCU_SPIWrite8(val8);
}

// ------------------------ Read a 32-bit data value -------------------------
uint32_t HAL_Read32(void)
{
	// Read 4 bytes from a register has been previously addressed. Send dummy
	// 00 bytes as only the incoming value is important.
	uint32_t val32;
```

```
	// Read low byte of data first.
	val32 = MCU_SPIRead32();

	// Return combined 32-bit value
	return MCU_le32toh(val32);
}

// ------------------------ Read a 16-bit data value --------------------------
uint16_t HAL_Read16(void)
{
	// Read 2 bytes from a register has been previously addressed. Send dummy
	// 00 bytes as only the incoming value is important.
	uint16_t val16;

	// Read low byte of data first.
	val16 = MCU_SPIRead16();

	// Return combined 16-bit value
	return MCU_le16toh(val16);
}

// ------------------------ Read an 8-bit data value --------------------------
uint8_t HAL_Read8(void)
{
	// Read 1 byte from a register has been previously addressed. Send dummy
	// 00 byte as only the incoming value is important.
	uint8_t val8;

	val8 = MCU_SPIRead8();

	// Return 8-bit value read
	return val8;
}

// ################## COMBINED ADDRESSING AND DATA FUNCTIONS ####################

// This section has combined calls which carry out a full write or read cycle
// including chip select, address, and data transfer.
// This would often be used for register writes and reads.

// -------------- Write a 32-bit value to specified address -------------------
void HAL_MemWrite32(uint32_t address, uint32_t val32)
{
	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send address to be written
	HAL_SetWriteAddress(address);
	// Send the data value
	HAL_Write32(val32);
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);
}

// -------------- Write a 16-bit value to specified address -------------------
void HAL_MemWrite16(uint32_t address, uint16_t val16)
{
	// CS low begins the SPI transfer
```

```
	HAL_ChipSelect(1);
	// Send address to be written
	HAL_SetWriteAddress(address);
	// Send the data value
	HAL_Write16(val16);
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);
}

// -------------- Write an 8-bit value to specified address --------------------
void HAL_MemWrite8(uint32_t address, uint8_t val8)
{
	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send address to be written
	HAL_SetWriteAddress(address);
	// Send the data value
	HAL_Write8(val8);
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);
}

// -------------- Read a 32-bit value from specified address --------------------
uint32_t HAL_MemRead32(uint32_t address)
{
	uint32_t val32;

	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send address to be read
	HAL_SetReadAddress(address);
	// Read the data value
	val32 = HAL_Read32();
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);

	// Return 32-bit value read
	return val32;
}
// -------------- Read a 16-bit value from specified address --------------------
uint16_t HAL_MemRead16(uint32_t address)
{
	uint16_t val16;

	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send address to be read
	HAL_SetReadAddress(address);
	// Read the data value
	val16 = HAL_Read16();
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);

	// Return 16-bit value read
	return val16;
}
// -------------- Read an 8-bit value from specified address --------------------
uint8_t HAL_MemRead8(uint32_t address)
{
	uint8_t val8;
```

```
	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send address to be read
	HAL_SetReadAddress(address);
	// Read the data value
	val8 = HAL_Read8();
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);

	// Return 8-bit value read
	return val8;
}
// ############################## HOST COMMANDS ###################################
// ------------------------- Write a host command ----------------------------
void HAL_HostCmdWrite(uint8_t cmd, uint8_t param)
{
	// CS low begins the SPI transfer
	HAL_ChipSelect(1);
	// Send command
	MCU_SPIWrite8(cmd);
	// followed by parameter
	MCU_SPIWrite8(param);
	// and a dummy 00 byte
	MCU_SPIWrite8(0x00);
	// CS high terminates the SPI transfer
	HAL_ChipSelect(0);
}
// ######################### SUPPORTING FUNCTIONS ################################

// --------- Increment co-processor address offset counter --------------------
void HAL_IncCmdPointer(uint16_t commandSize)
{
	// Calculate new offset
	writeCmdPointer = (writeCmdPointer + commandSize) & (EVE_RAM_CMD_SIZE - 1);
}

// --------- Increment co-processor address offset counter --------------------
uint16_t HAL_GetCmdPointer(void)
{
	// Return new offset
	return writeCmdPointer;
}

void HAL_WriteCmdPointer(void)
{
	// and move write pointer to here
	HAL_MemWrite32(EVE_REG_CMD_WRITE, writeCmdPointer);
}

// ------ Wait for co-processor read and write pointers to be equal ------------
uint8_t HAL_WaitCmdFifoEmpty(void)
{
	uint32_t readCmdPointer;

	// Wait until the two registers match
	do
	{
		// Read the graphics processor read pointer
```

```
            readCmdPointer = HAL_MemRead32(EVE_REG_CMD_READ);

      } while ((writeCmdPointer != readCmdPointer) && (readCmdPointer != 0xFFF));


      if(readCmdPointer == 0xFFF)
      {
            // Return 0xFF if an error occurred
            return 0xFF;
      }
      else
      {
            // Return 0 if pointers became equal successfully
            return 0;
      }
}
// ------------ Check how much free space is available in CMD FIFO -------------
uint16_t HAL_CheckCmdFreeSpace(void)
{
      uint32_t readCmdPointer = 0;
      uint16_t Fullness, Freespace;

      // Check the graphics processor read pointer
      readCmdPointer = HAL_MemRead32(EVE_REG_CMD_READ);

      // Fullness is difference between MCUs current write pointer value and the
FT81x's REG_CMD_READ
      Fullness = ((writeCmdPointer - (uint16_t)readCmdPointer) & (EVE_RAM_CMD_SIZE -
1));
      // Free Space is 4K - 4 - Fullness (-4 avoids buffer wrapping round)
      Freespace = (EVE_RAM_CMD_SIZE - 4) - Fullness;

      return Freespace;
}

#endif // __linux__
```

Eve_fonts.c

```
#include <eveheader.h>
#include <stdint.h>


const uint32_t font0_offset = 0; // Taken from commmand line

#ifdef __GNUC__
const uint8_t font0[]  __attribute__((aligned(4))) =
#else // __GNUC__
const uint8_t font0[] =
#endif // __GNUC__
        /*Command Line: fnt_cvt.exe -f legacy -i DS-DIGIT.TTF -s 70 -d  0 -c setfont
-u ANSI_Nums.txt -o msp430_font*/

        /*10 characters have been converted */

            /* 148 Metric Block Begin +++  */
        /*(u'file properties ', u'format ', u'L1', u' stride ', 5, u' width ', 36,
u'height', 58)*/
            {
```

```
            /* Widths */

0,36,20,36,36,36,36,36,36,36,36,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,
            /* Format */
        1,0,0,0,
            /* Stride */
        5,0,0,0,
            /* Max Width */
        36,0,0,0,
            /* Max Height */
        58,0,0,0,
        /* Raw Data Address in Decimal: <-142> */
        114,255,255,255,
            /* 148 Metric Block End ---   */

            /*Bitmap Raw Data begin +++*/
        /*The expected raw bitmap size is 2900 Bytes */
            0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
            0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,127,255,255,192,0,63,255,255,128,1,31,255,255,16,1,143,255,254,32,1,199,255,252,9
6,1,227,255,248,224,1,

243,255,241,224,1,248,0,3,224,1,248,0,7,224,1,248,0,15,224,1,248,0,15,224,1,248,0,15,
224,1,248,0,

15,224,3,248,0,15,224,3,248,0,15,224,3,248,0,15,192,3,248,0,15,192,3,240,0,15,192,3,2
24,0,7,192,
        3,192,0,3,192,1,128,0,1,128,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,3,128,

0,3,128,7,192,0,7,128,7,224,0,31,128,7,240,0,31,128,7,240,0,31,128,7,240,0,31,128,7,2
24,0,63,

128,7,224,0,63,128,7,224,0,63,128,7,224,0,63,128,15,224,0,63,128,15,224,0,63,128,15,1
92,0,31,128,15,

143,255,143,0,15,31,255,199,0,14,63,255,231,0,12,127,255,243,0,8,255,255,249,0,1,255,
255,248,0,7,255,255,
        252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,128,0,0,0,1,128,0,0,0,3,128,0,0,0,7,128,0,0,0,7,128,0,
        0,0,15,128,0,0,0,31,128,0,0,0,63,0,0,0,0,63,0,0,0,0,63,0,0,0,0,63,0,0,0,0,
        127,0,0,0,0,127,0,0,0,0,127,0,0,0,0,127,0,0,0,0,127,0,0,0,0,127,0,0,0,0,63,0,
        0,0,0,30,0,0,0,0,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,
        0,28,0,0,0,0,62,0,0,0,0,126,0,0,0,0,254,0,0,0,0,254,0,0,0,0,254,0,0,0,0,252,
        0,0,0,0,252,0,0,0,0,252,0,0,0,0,252,0,0,0,0,252,0,0,0,1,252,0,0,0,0,252,0,0,
        0,0,124,0,0,0,0,124,0,0,0,0,60,0,0,0,0,28,0,0,0,0,12,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,0,31,255,255,16,0,15,255,254,32,0,7,255,
252,96,0,3,

255,248,224,0,3,255,241,224,0,0,0,3,224,0,0,0,7,224,0,0,0,15,224,0,0,0,15,224,0,0,0,1
5,
```

```
224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,192,0,0,0,15,192,0,0,0,15,192,0,

0,0,7,192,0,15,255,243,192,0,31,255,241,128,0,63,255,248,0,0,63,255,252,0,0,63,255,24
8,0,1,31,255,

240,0,3,143,255,224,0,7,192,0,0,0,7,224,0,0,0,7,240,0,0,0,7,240,0,0,0,7,240,0,0,0,

7,224,0,0,0,7,224,0,0,0,7,224,0,0,0,7,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,192,

0,0,0,15,143,255,128,0,15,31,255,192,0,14,63,255,224,0,12,127,255,240,0,8,255,255,248
,0,1,255,255,248,
        0,7,255,255,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,0,31,255,255,16,0,15,255,254,32,0,7,
255,252,96,

0,3,255,248,224,0,3,255,241,224,0,0,0,3,224,0,0,0,7,224,0,0,0,15,224,0,0,0,15,224,0,0
,

0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,192,0,0,0,15,192,0,0,0,15,

192,0,0,0,7,192,0,15,255,243,192,0,31,255,241,128,0,63,255,248,0,0,63,255,252,0,0,63,
255,248,0,0,

31,255,241,0,0,15,255,227,128,0,0,0,7,128,0,0,0,31,128,0,0,0,31,128,0,0,0,31,128,0,0,
0,

31,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,

0,0,0,31,128,0,15,255,143,0,0,31,255,207,0,0,63,255,231,0,0,127,255,243,0,0,255,255,2
49,0,1,255,
        255,248,0,7,255,255,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,128,0,0,32,1,192,0,0,96,1,192,0,

0,224,1,224,0,1,224,1,240,0,3,224,1,248,0,7,224,1,248,0,15,224,1,248,0,15,224,1,248,0
,15,224,

1,248,0,15,224,1,248,0,15,224,3,248,0,15,224,3,248,0,15,224,3,248,0,15,192,3,248,0,15
,192,3,240,

0,15,192,3,224,0,7,192,3,207,255,243,192,1,159,255,241,128,0,63,255,248,0,0,63,255,25
2,0,0,63,255,248,

0,0,31,255,241,0,0,15,255,227,128,0,0,0,7,128,0,0,0,31,128,0,0,0,31,128,0,0,0,31,128,
0,

0,0,31,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,
        63,128,0,0,0,63,128,0,0,0,31,0,0,0,0,15,0,0,0,0,7,0,0,0,0,3,0,0,0,0,1,0,
        0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,1,31,255,255,0,1,143,255,254
,0,1,

199,255,252,0,1,227,255,248,0,1,243,255,240,0,1,248,0,0,0,1,248,0,0,0,1,248,0,0,0,1,2
48,0,
        0,0,1,248,0,0,0,1,248,0,0,0,3,248,0,0,0,3,248,0,0,0,3,248,0,0,0,3,248,0,0,0,
```

```
3,240,0,0,0,3,224,0,0,0,3,207,255,240,0,1,159,255,240,0,0,63,255,248,0,0,63,255,252,0
,0,63,

255,248,0,0,31,255,241,0,0,15,255,227,128,0,0,0,7,128,0,0,0,31,128,0,0,0,31,128,0,0,0
,31,

128,0,0,0,31,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,

0,0,63,128,0,0,0,31,128,0,15,255,143,0,0,31,255,207,0,0,63,255,231,0,0,127,255,243,0,
0,255,255,

249,0,1,255,255,248,0,7,255,255,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,1,31,255,255,0,1,143,255
,254,

0,1,199,255,252,0,1,227,255,248,0,1,243,255,240,0,1,248,0,0,0,1,248,0,0,0,1,248,0,0,0
,1,

248,0,0,0,1,248,0,0,0,1,248,0,0,0,3,248,0,0,0,3,248,0,0,0,3,248,0,0,0,3,248,0,

0,0,3,240,0,0,0,3,224,0,0,0,3,207,255,240,0,1,159,255,240,0,0,63,255,248,0,0,63,255,2
52,0,

0,63,255,248,0,1,31,255,241,0,3,143,255,227,128,7,192,0,7,128,7,224,0,31,128,7,240,0,
31,128,7,240,

0,31,128,7,240,0,31,128,7,224,0,63,128,7,224,0,63,128,7,224,0,63,128,7,224,0,63,128,1
5,224,0,63,

128,15,224,0,63,128,15,192,0,31,128,15,143,255,143,0,15,31,255,199,0,14,63,255,231,0,
12,127,255,243,0,8,

255,255,249,0,1,255,255,248,0,7,255,255,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,0,31,255,255,16,0,15
,

255,254,32,0,7,255,252,96,0,3,255,248,224,0,3,255,241,224,0,0,0,3,224,0,0,0,7,224,0,0
,0,15,

224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,224,0,0,0,15,192,0,
        0,0,15,192,0,0,0,15,192,0,0,0,7,192,0,0,0,3,192,0,0,0,1,128,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,3,128,0,0,0,7,128,0,0,0,31,128,0,0,0,31,128,

0,0,0,31,128,0,0,0,31,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,
        0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,31,0,0,0,0,15,0,0,0,0,7,0,0,0,0,3,
        0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,1,31,255,255,16,

1,143,255,254,32,1,199,255,252,96,1,227,255,248,224,1,243,255,241,224,1,248,0,3,224,1
,248,0,7,224,1,248,
```

```
0,15,224,1,248,0,15,224,1,248,0,15,224,1,248,0,15,224,3,248,0,15,224,3,248,0,15,224,3
,248,0,15,

192,3,248,0,15,192,3,240,0,15,192,3,224,0,7,192,3,207,255,243,192,1,159,255,241,128,0
,63,255,248,0,0,

63,255,252,0,0,63,255,248,0,1,31,255,241,0,3,143,255,227,128,7,192,0,7,128,7,224,0,31
,128,7,240,0,

31,128,7,240,0,31,128,7,240,0,31,128,7,224,0,63,128,7,224,0,63,128,7,224,0,63,128,7,2
24,0,63,128,

15,224,0,63,128,15,224,0,63,128,15,192,0,31,128,15,143,255,143,0,15,31,255,199,0,14,6
3,255,231,0,12,127,

255,243,0,8,255,255,249,0,1,255,255,248,0,7,255,255,252,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,192,0,63,255,255,128,1,31,255,

255,16,1,143,255,254,32,1,199,255,252,96,1,227,255,248,224,1,243,255,241,224,1,248,0,
3,224,1,248,0,7,224,

1,248,0,15,224,1,248,0,15,224,1,248,0,15,224,1,248,0,15,224,3,248,0,15,224,3,248,0,15
,224,3,248,

0,15,192,3,248,0,15,192,3,240,0,15,192,3,224,0,7,192,3,207,255,243,192,1,159,255,241,
128,0,63,255,248,

0,0,63,255,252,0,0,63,255,248,0,0,31,255,241,0,0,15,255,227,128,0,0,0,7,128,0,0,0,31,
128,0,

0,0,31,128,0,0,0,31,128,0,0,0,31,128,0,0,0,63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,

63,128,0,0,0,63,128,0,0,0,63,128,0,0,0,31,128,0,15,255,143,0,0,31,255,207,0,0,63,255,
231,0,
        0,127,255,243,0,0,255,255,249,0,1,255,255,248,0,7,255,255,252,0,
            /*Bitmap Raw Data end ---*/
            }
;

uint32_t eve_init_fonts(void)
{
      const EVE_GPU_FONT_HEADER *font0_hdr = (const EVE_GPU_FONT_HEADER *)font0;
      uint32_t font0_size = sizeof(font0);

      EVE_LIB_WriteDataToRAMG(font0, font0_size, font0_offset);

      EVE_LIB_BeginCoProList();
      EVE_CMD_DLSTART();
      EVE_CLEAR(1,1,1);
      EVE_COLOR_RGB(255, 255, 255);
      EVE_BEGIN(EVE_BEGIN_BITMAPS);
      EVE_BITMAP_HANDLE(FONT_CUSTOM);
      EVE_BITMAP_SOURCE(font0_hdr->PointerToFontGraphicsData);
      EVE_BITMAP_LAYOUT(font0_hdr->FontBitmapFormat,
                  font0_hdr->FontLineStride, font0_hdr->FontHeightInPixels);
      EVE_BITMAP_SIZE(EVE_FILTER_NEAREST, EVE_WRAP_BORDER, EVE_WRAP_BORDER,
                  font0_hdr->FontWidthInPixels,
```

```
                    font0_hdr->FontHeightInPixels);
        EVE_CMD_SETFONT(FONT_CUSTOM, font0_offset);
        EVE_END();
        EVE_DISPLAY();
        EVE_CMD_SWAP();
        EVE_LIB_EndCoProList();
        EVE_LIB_AwaitCoProEmpty();
        return ((font0_size + font0_offset) + 16) & (~15);
}
```

Eve_calibrateplus.h

```
#ifndef _EVE_EXAMPLE_H
#define _EVE_EXAMPLE_H

#include <stdint.h>



#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

/**
 @brief Definitions of handles for custom fonts and bitmaps.
 */
//@{
#define FONT_CUSTOM 8
#define BITMAP_BRIDGETEK_LOGO 7
//@}

/**
 @brief Key for identifying if touchscreen calibration values are programmed
correctly.
 */
#define VALID_KEY_TOUCHSCREEN 0xd72f91a3

/**
 @brief Structure to hold touchscreen calibration settings.
 @details This is used to store the touchscreen calibration settings persistently
 in Flash and identify if the calibration needs to be re-performed.
 */
struct touchscreen_calibration {
        uint32_t key; // VALID_KEY_TOUCHSCREEN
        uint32_t transform[6];
};

/* Globals available within the eve_example code */
extern uint32_t eve_img_bridgetek_logo_width;
extern uint32_t eve_img_bridgetek_logo_height;

/* Functions called within the eve_example code */
void eve_calibrate(void);
uint32_t eve_init_fonts(void);
uint32_t eve_load_images(uint32_t);
uint8_t eve_read_tag(uint8_t *key);
```

```
/* Functions called from eve_example code to platform specific code */
int8_t platform_calib_init(void);
int8_t platform_calib_write(struct touchscreen_calibration *calib);
int8_t platform_calib_read(struct touchscreen_calibration *calib);

#ifdef __cplusplus
} /* extern "C" */
#endif /* __cplusplus */

#endif /* _EVE_EXAMPLE_H */
```

Eve_config.h

```
#ifndef _EVE_CONFIG_H_
#define _EVE_CONFIG_H_

#define FT800 1
#define FT801 2
#define FT811 3
#define FT812 4

#define WVGA 1

#ifndef FT8XX_TYPE
#define FT8XX_TYPE FT812
//#define FT8XX_TYPE FT800
#endif

#ifndef FT8XX_DISPLAY
#define FT8XX_DISPLAY WVGA
#endif

#undef FT81X_ENABLE
#if (FT8XX_TYPE == FT811) || (FT8XX_TYPE == FT812)
#define FT81X_ENABLE
#endif

#if FT8XX_DISPLAY == WVGA

#define EVE_DISP_WIDTH 800 // Active width of LCD display
#define EVE_DISP_HEIGHT 480 // Active height of LCD display
#define EVE_DISP_HCYCLE 928 // Total number of clocks per line
#define EVE_DISP_HOFFSET 88 // Start of active line
#define EVE_DISP_HSYNC0 0 // Start of horizontal sync pulse
#define EVE_DISP_HSYNC1 48 // End of horizontal sync pulse
#define EVE_DISP_VCYCLE 525 // Total number of lines per screen
#define EVE_DISP_VOFFSET 32 // Start of active screen
#define EVE_DISP_VSYNC0 0 // Start of vertical sync pulse
#define EVE_DISP_VSYNC1 3 // End of vertical sync pulse
#define EVE_DISP_PCLK 2 // Pixel Clock
#define EVE_DISP_SWIZZLE 0 // Define RGB output pins
#define EVE_DISP_PCLKPOL 1 // Define active edge of PCLK

//#define EVE_DISP_WIDTH 480 // Active width of LCD display
//#define EVE_DISP_HEIGHT 272 // Active height of LCD display
//#define EVE_DISP_HCYCLE 548 // Total number of clocks per line
//#define EVE_DISP_HOFFSET 43 // Start of active line
//#define EVE_DISP_HSYNC0 0 // Start of horizontal sync pulse
//#define EVE_DISP_HSYNC1 41 // End of horizontal sync pulse
```

```
//#define EVE_DISP_VCYCLE 292 // Total number of lines per screen
//#define EVE_DISP_VOFFSET 12 // Start of active screen
//#define EVE_DISP_VSYNC0 0 // Start of vertical sync pulse
//#define EVE_DISP_VSYNC1 10 // End of vertical sync pulse
//#define EVE_DISP_PCLK 5 // Pixel Clock
//#define EVE_DISP_SWIZZLE 2 // Define RGB output pins
//#define EVE_DISP_PCLKPOL 1 // Define active edge of PCLK


#endif // FT81X_WQVGA

#endif /* _EVE_CONFIG_H_ */
```

EVE.h

```
#ifndef EVE_HEADER_H
#define      EVE_HEADER_H

// for Uint8/16/32 and Int8/16/32 data types.
#include <stdint.h>
// Include the configuration for this instance.

#include "EVE_config.h"
#include "FT8xx.h"

/**
 @brief Initialise EVE API.
 @details Initialise the EVE API layer, HAL layer and MCU-specific hardware
       layer.
 */
void EVE_Init(void);

/**
 @brief EVE API: Begin coprocessor list
 @details Starts a coprocessor list. Waits for the coprocessor to be idle
       before asserting chip select.
 */
void EVE_LIB_BeginCoProList(void);

/**
 @brief EVE API: End coprocessor list
 @details Ends a coprocessor list. Deasserts chip select.
 */
void EVE_LIB_EndCoProList(void);

/**
 @brief EVE API: Waits for coprocessor list to end
 @details Will poll the coprocessor command list until it has been completed.
 */
void EVE_LIB_AwaitCoProEmpty(void);

/**
 @brief EVE API: Write a buffer to memory mapped RAM
 @details Writes a block of data via SPI to the EVE.
 @param ImgData - Pointer to start of data buffer.
 @param DataSize - Number of bytes in buffer.
 @param DestAddress - 24 bit memory mapped address on EVE.
```

```
 */
void EVE_LIB_WriteDataToRAMG(const uint8_t *ImgData, uint32_t DataSize, uint32_t
DestAddress);

/**
 @brief EVE API: Read a buffer from memory mapped RAM
 @details Reads a block of data via SPI from the EVE.
 @param ImgData - Pointer to start of receive data buffer.
 @param DataSize - Number of bytes to read (rounded up to be 32-bit aligned).
 @param DestAddress - 24 bit memory mapped address on EVE.
 */
void EVE_LIB_ReadDataFromRAMG(uint8_t *ImgData, uint32_t DataSize, uint32_t
SrcAddress);

/**
 @brief EVE API: Write a buffer to the coprocessor command memory
 @details Writes a block of data via SPI to the EVE coprocessor.
       This must be part of a coprocessor list. It will typically be called
       after a coprocessor command to provide data for the operation.
       The data will be added to the coprocessor command list therefore the
       write will block on available space in this list.
 @param ImgData - Pointer to start of data buffer.
 @param DataSize - Number of bytes in buffer.
 */
void EVE_LIB_WriteDataToCMD(const uint8_t *ImgData, uint32_t DataSize);

/**
 @brief EVE API: Write a string the coprocessor command memory
 @details Writes a string via SPI to the EVE coprocessor.
       This must be part of a coprocessor list. It will typically be called
       after a coprocessor command to provide a string for the operation.
       The data will be added to the coprocessor command list therefore the
       write will block on available space in this list.
 @param ImgData - Pointer to start of data buffer.
 @param DataSize - Number of bytes in buffer.
 */
uint16_t EVE_LIB_SendString(const char* string);

/**
 @brief EVE API: Get properties of an CMD_LOADIMAGE operation
 @details Obtains the details of an image decoded by the CMD_LOADIMAGE
       coprocessor command. The properties of the image are taken from
       the coprocessor command list.
 @param addr - Pointer to variable to receive the image start address.
 @param width - Pointer to variable to receive the image width.
 @param height - Pointer to variable to receive the image height.
 */
void EVE_LIB_GetProps(uint32_t *addr, uint32_t *width, uint32_t *height);

// Raw command interface write
void EVE_CMD(uint32_t c);
// Graphics instructions
void EVE_CLEAR_COLOR_RGB(uint8_t r, uint8_t g, uint8_t b);
void EVE_CLEAR_COLOR(uint32_t c);
void EVE_CLEAR(uint8_t c, uint8_t s, uint8_t t);
void EVE_COLOR_RGB(uint8_t r, uint8_t g, uint8_t b);
void EVE_COLOR(uint32_t c);
void EVE_VERTEX2F(int16_t x, int16_t y);
void EVE_VERTEX2II(uint16_t x, uint16_t y, uint8_t handle,uint8_t cell);
void EVE_BITMAP_HANDLE(uint8_t handle);
void EVE_BITMAP_SOURCE(uint32_t addr);
```

```
void EVE_BITMAP_LAYOUT(uint8_t format, uint16_t linestride, uint16_t height);
void EVE_BITMAP_SIZE(uint8_t filter, uint8_t wrapx, uint8_t wrapy, uint16_t width,
uint16_t height);
void EVE_CELL(uint8_t cell);
void EVE_TAG(uint8_t s);
void EVE_ALPHA_FUNC(uint8_t func, uint8_t ref);
void EVE_STENCIL_FUNC(uint8_t func, uint8_t ref, uint8_t mask);
void EVE_BLEND_FUNC(uint8_t src, uint8_t dst);
void EVE_STENCIL_OP(uint8_t sfail, uint8_t spass);
void EVE_POINT_SIZE(uint16_t size);
void EVE_LINE_WIDTH(uint16_t width);
void EVE_CLEAR_COLOR_A(uint8_t alpha);
void EVE_COLOR_A(uint8_t alpha);
void EVE_CLEAR_STENCIL(uint8_t s);
void EVE_CLEAR_TAG(uint8_t s);
void EVE_STENCIL_MASK(uint8_t mask);
void EVE_TAG_MASK(uint8_t mask);
void EVE_SCISSOR_XY(uint16_t x, uint16_t y);
void EVE_SCISSOR_SIZE(uint16_t width, uint16_t height);
void EVE_CALL(uint16_t dest);
void EVE_JUMP(uint16_t dest);
void EVE_BEGIN(uint8_t prim);
void EVE_COLOR_MASK(uint8_t r, uint8_t g, uint8_t b, uint8_t a);
void EVE_END(void);
void EVE_SAVE_CONTEXT(void);
void EVE_RESTORE_CONTEXT(void);
void EVE_RETURN(void);
void EVE_MACRO(uint8_t m);
void EVE_DISPLAY(void);

// Co-Processor Widgets
void EVE_CMD_TEXT(int16_t x, int16_t y, int16_t font, uint16_t options, const char*
string);
void EVE_CMD_BUTTON(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font,
uint16_t options, const char* string);
void EVE_CMD_KEYS(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t
options, const char* string);
void EVE_CMD_NUMBER(int16_t x, int16_t y, int16_t font, uint16_t options, int32_t n);
void EVE_CMD_LOADIDENTITY(void);
void EVE_CMD_TOGGLE(int16_t x, int16_t y, int16_t w, int16_t font, uint16_t options,
uint16_t state, const char* string);
void EVE_CMD_GAUGE(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t major,
uint16_t minor, uint16_t val, uint16_t range);
void EVE_CMD_REGREAD(uint32_t ptr, uint32_t result);
void EVE_CMD_GETPROPS(uint32_t ptr, uint32_t w, uint32_t h);
void EVE_CMD_MEMCPY(uint32_t dest, uint32_t src, uint32_t num);
void EVE_CMD_SPINNER(int16_t x, int16_t y, uint16_t style, uint16_t scale);
void EVE_CMD_BGCOLOR(uint32_t c);
void EVE_CMD_SWAP(void);
void EVE_CMD_INFLATE(uint32_t ptr);
void EVE_CMD_TRANSLATE(int32_t tx, int32_t ty);
void EVE_CMD_STOP(void);
void EVE_CMD_SLIDER(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t range);
void EVE_BITMAP_TRANSFORM_A(long a);
void EVE_BITMAP_TRANSFORM_B(long b);
void EVE_BITMAP_TRANSFORM_C(long c);
void EVE_BITMAP_TRANSFORM_D(long d);
```

```
void EVE_BITMAP_TRANSFORM_E(long e);
void EVE_BITMAP_TRANSFORM_F(long f);
void EVE_CMD_INTERRUPT(uint32_t ms);
void EVE_CMD_FGCOLOR(uint32_t c);
void EVE_CMD_ROTATE(int32_t a);
void EVE_CMD_MEMWRITE(uint32_t ptr, uint32_t num);
void EVE_CMD_SCROLLBAR(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t size, uint16_t range);
void EVE_CMD_GETMATRIX(int32_t a, int32_t b, int32_t c, int32_t d, int32_t e, int32_t
f);
void EVE_CMD_SKETCH(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t ptr,
uint16_t format);
void EVE_CMD_MEMSET(uint32_t ptr, uint32_t value, uint32_t num);
void EVE_CMD_GRADCOLOR(uint32_t c);
void EVE_CMD_BITMAP_TRANSFORM(int32_t x0, int32_t y0, int32_t x1, int32_t y1, int32_t
x2, int32_t y2, int32_t tx0, int32_t ty0, int32_t tx1, int32_t ty1, int32_t tx2,
int32_t ty2, uint16_t result);
void EVE_CMD_CALIBRATE(uint32_t result);
void EVE_CMD_SETFONT(uint32_t font, uint32_t ptr);
void EVE_CMD_LOGO(void);
void EVE_CMD_APPEND(uint32_t ptr, uint32_t num);
void EVE_CMD_MEMZERO(uint32_t ptr, uint32_t num);
void EVE_CMD_SCALE(int32_t sx, int32_t sy);
void EVE_CMD_CLOCK(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t h,
uint16_t m, uint16_t s, uint16_t ms);
void EVE_CMD_GRADIENT(int16_t x0, int16_t y0, uint32_t rgb0, int16_t x1, int16_t y1,
uint32_t rgb1);
void EVE_CMD_SETMATRIX(void);
void EVE_CMD_TRACK(int16_t x, int16_t y, int16_t w, int16_t h, int16_t tag);
void EVE_CMD_GETPTR(uint32_t result);
void EVE_CMD_PROGRESS(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options,
uint16_t val, uint16_t range);
void EVE_CMD_COLDSTART(void);
void EVE_CMD_DIAL(int16_t x, int16_t y, int16_t r, uint16_t options, uint16_t val);
void EVE_CMD_LOADIMAGE(uint32_t ptr, uint32_t options);
void EVE_CMD_DLSTART(void);
void EVE_CMD_SNAPSHOT(uint32_t ptr);
void EVE_CMD_SCREENSAVER(void);
void EVE_CMD_MEMCRC(uint32_t ptr, uint32_t num, uint32_t result);

#ifdef FT81X_ENABLE

// Graphics instructions
void EVE_VERTEX_FORMAT(uint8_t frac);
void EVE_BITMAP_LAYOUT_H(uint8_t linestride, uint8_t height);
void EVE_BITMAP_SIZE_H(uint8_t width, uint8_t height);
void EVE_PALETTE_SOURCE(uint32_t addr);
void EVE_VERTEX_TRANSLATE_X(uint32_t x);
void EVE_VERTEX_TRANSLATE_Y(uint32_t y);
void EVE_NOP(void);

// Co-Processor Widgets
void EVE_CMD_VIDEOSTART(void);
void EVE_CMD_SETROTATE(uint32_t r);
void EVE_CMD_SETFONT2(uint32_t font, uint32_t ptr, uint32_t firstchar);
void EVE_CMD_MEDIAFIFO(uint32_t ptr, uint32_t size);
void EVE_CMD_SNAPSHOT2(uint32_t fmt, uint32_t ptr, int16_t x, int16_t y, int16_t w,
int16_t h);
void EVE_CMD_INT_SWLOADIMAGE(uint32_t ptr, uint32_t options);
void EVE_CMD_CSKETCH(int16_t x, int16_t y, uint16_t w, uint16_t h, uint32_t ptr,
uint16_t format,uint16_t freq);
```

```
void EVE_CMD_ROMFONT(uint32_t font, uint32_t romslot);
void EVE_CMD_PLAYVIDEO(uint32_t options);
void EVE_CMD_SYNC(void);
void EVE_CMD_VIDEOFRAME(uint32_t dst, uint32_t ptr);
void EVE_CMD_SETBASE(uint32_t base);
void EVE_CMD_SETBITMAP(uint32_t source, uint16_t fmt, uint16_t w, uint16_t h);
void EVE_CMD_SETSCRATCH(uint32_t handle);
//##############################################################################################
############
void EVE_LEER_PANTALLA();
uint8_t EVE_LEER_TAG();
void EVE_BOTON_P(int16_t x,int16_t y,int16_t L,int16_t L2, uint8_t t);
void EVE_MATRIX_BOTON(int n);
void EVE_INICIALIZAR_NUEVA_PANTALLA();
void EVE_MOSTRAR_FIN_PANTALLA();
void EVE_LIMPIAR_PANTALLA_A(uint8_t R,uint8_t G,uint8_t B);
void EVE_COLOR_ELEMENTO();
void EVE_EDIT_MODE_L();
void EVE_EDIT_MODE_MATRIX();
void CREAR_BOTON(uint8_t origen,uint16_t x,uint16_t y,uint16_t w,uint16_t h, char
named[5],uint8_t pantallaAsocd, uint8_t tipod, uint16_t comandod);
void EVE_BOTON_P_2(uint8_t n);
void EVE_HANDLER_ACCION();
void EVE_MOSTRAR_BOTONES_GLOBAL();
void EVE_ACTUADOR(uint8_t tag);
void WEB_SUBMITER();
void EVE_ERASE(uint8_t boton);
void FLASH_STOCKER_I(uint8_t i);
void INIT_ESTRUCT_PRINC();
void FLASH_FISHER_I(uint8_t i);
void FLASH_RENEW();
void FLASH_STOCKER_C(uint8_t i);
void FLASH_FISHER_C(uint8_t i);
void MOSTRAR_IP();
void CALIB_STOCKER();
void CALIB_FISHER();

#include <stdbool.h>

struct Accion{
    uint8_t tipo;
//    1-EditMatrixmode
//    2-EditLibre
//    3-IncPantalla
//    4-DecPantalla
//    5-ComandoTeclado
//    6-Sync?...
    uint16_t comando; //Si lo hubiera
    bool mod[4];
};
struct fisico{
    uint16_t x;
    uint16_t y;
    uint16_t w;
    uint16_t h;
};

struct Boton{
```

```
    struct fisico fis;
    char name[6];
    uint8_t tagAsoc;
    uint8_t pantallaAsoc; //16 es territorio de nadie, 0 es presente en todas as
pantallas
    struct Accion BotAccion;
};

static const int8_t EveKeypUsageCodes[][2]={
                                            {0x01,0x59},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_1},
                                            {0x01,0x5A},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_2},
                                            {0x01,0x5B},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_3},
                                            {0x01,0x5C},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_4},
                                            {0x01,0x5D},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_5},
                                            {0x01,0x5E},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_6},
                                            {0x01,0x5F},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_7},
                                            {0x01,0x60},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_8},
                                            {0x01,0x61},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_9},
                                            {0x01,0x62},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_0},

{0x01|0x04,0x59},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_1},

{0x01|0x04,0x5A},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_2},

{0x01|0x04,0x5B},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_3},

{0x01|0x04,0x5C},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_4},

{0x01|0x04,0x5D},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_5},

{0x01|0x04,0x5E},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_6},

{0x01|0x04,0x5F},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_7},

{0x01|0x04,0x60},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_8},

{0x01|0x04,0x61},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_9},

{0x01|0x04,0x62},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_0},
                                            {0x01,0x54},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_SLASH},
                                            {0x01,0x55},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_STAR},
                                            {0x01,0x56},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_MINUS},
                                            {0x01,0x57},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_PLUS},
                                            {0x01,0x63},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_PERIOD},
                                            {0x01,0x67},//{HID_KEYB_LEFT_CTRL,
HID_KEYB_USAGE_KEYPAD_EQUAL},
```

```
{0x01|0x04,0x54},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_SLASH},

{0x01|0x04,0x55},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_STAR},

{0x01|0x04,0x56},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_MINUS},

{0x01|0x04,0x57},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_PLUS},

{0x01|0x04,0x63},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_PERIOD},

{0x01|0x04,0x67},//{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT,
HID_KEYB_USAGE_KEYPAD_EQUAL},
};



#endif /* FT81X_ENABLE */

#endif /* EVE_HEADER_H */
```

FT8xx.h

```
#ifndef _FT8XX_H_
#define      _FT8XX_H_

/* Definitions used for co-processor command buffer */
#define EVE_CMD_SIZE          (4)       //4 byte per coprocessor command of EVE

/* For FT801, FT811 and FT813 */
#define CTOUCH_MODE_COMPATIBILITY 1
#define CTOUCH_MODE_EXTENDED 0

#ifndef FT81X_ENABLE

#define FT800_VERSION "1.9.0".

/* For FT81x enable the switch in platform.h file */
/* Lower boundary of trimming */
#define LOW_FREQ_BOUND  47040000L//98% of 48Mhz

#define EVE_RAM_CMD              0x108000UL
#define EVE_RAM_CMD_SIZE         (4*1024L)
#define EVE_RAM_DL               0x100000UL
#define EVE_RAM_DL_SIZE          (8*1024L)
#define EVE_RAM_G                0x0UL
#define EVE_RAM_G_SIZE           (256*1024L)
#define EVE_RAM_PAL              0x102000UL
#define EVE_RAM_REG              0x102400UL

#define EVE_ROMFONT_TABLEADDRESS 0xffffcUL
#define EVE_REG_ANA_COMP         0x102570UL
```

```
#define EVE_REG_ANALOG           0x102538UL
#define EVE_REG_CLOCK            0x102408UL
#define EVE_REG_CMD_DL           0x1024ecUL
#define EVE_REG_CMD_READ         0x1024e4UL
#define EVE_REG_CMD_WRITE        0x1024e8UL
#define EVE_REG_CPURESET         0x10241cUL
#define EVE_REG_CRC              0x102568UL
#define EVE_REG_CSPREAD          0x102464UL
#define EVE_REG_CTOUCH_EXTENDED 0x1024f4UL
#define EVE_REG_CTOUCH_GESTURE 0x102538UL
#define EVE_REG_CTOUCH_IDS 0x10250cUL
#define EVE_REG_CTOUCH_TOUCH0_XY 0x102510UL
#define EVE_REG_CTOUCH_TOUCH1_XY 0x102508UL
#define EVE_REG_CTOUCH_TOUCH2_XY 0x102574UL
#define EVE_REG_CTOUCH_TOUCH3_XY 0x102578UL
#define EVE_REG_CTOUCH_TOUCH4_X 0x102538UL
#define EVE_REG_CTOUCH_TOUCH4_Y 0x10250cUL
#define EVE_REG_CYA_TOUCH        0x102534UL
#define EVE_REG_CYA0             0x1024d0UL
#define EVE_REG_CYA1             0x1024d4UL
#define EVE_REG_DATESTAMP        0x10253cUL
#define EVE_REG_DITHER           0x10245cUL
#define EVE_REG_DLSWAP           0x102450UL
#define EVE_REG_FRAMES           0x102404UL
#define EVE_REG_FREQUENCY        0x10240cUL
#define EVE_REG_GPIO             0x102490UL
#define EVE_REG_GPIO_DIR         0x10248cUL
#define EVE_REG_HCYCLE           0x102428UL
#define EVE_REG_HOFFSET          0x10242cUL
#define EVE_REG_HSIZE            0x102430UL
#define EVE_REG_HSYNC0           0x102434UL
#define EVE_REG_HSYNC1           0x102438UL
#define EVE_REG_ID               0x102400UL
#define EVE_REG_INT_EN           0x10249cUL
#define EVE_REG_INT_FLAGS        0x102498UL
#define EVE_REG_INT_MASK         0x1024a0UL
#define EVE_REG_MACRO_0          0x1024c8UL
#define EVE_REG_MACRO_1          0x1024ccUL
#define EVE_REG_OUTBITS          0x102458UL
#define EVE_REG_PCLK             0x10246cUL
#define EVE_REG_PCLK_POL         0x102468UL
#define EVE_REG_PLAY             0x102488UL
#define EVE_REG_PLAYBACK_FORMAT  0x1024b4UL
#define EVE_REG_PLAYBACK_FREQ    0x1024b0UL
#define EVE_REG_PLAYBACK_LENGTH  0x1024a8UL
#define EVE_REG_PLAYBACK_LOOP    0x1024b8UL
#define EVE_REG_PLAYBACK_PLAY    0x1024bcUL
#define EVE_REG_PLAYBACK_READPTR 0x1024acUL
#define EVE_REG_PLAYBACK_START   0x1024a4UL
#define EVE_REG_PWM_DUTY         0x1024c4UL
#define EVE_REG_PWM_HZ           0x1024c0UL
#define EVE_REG_RENDERMODE       0x102410UL
#define EVE_REG_ROMSUB_SEL       0x1024e0UL
#define EVE_REG_ROTATE           0x102454UL
#define EVE_REG_SNAPSHOT         0x102418UL
#define EVE_REG_SNAPY            0x102414UL
#define EVE_REG_SOUND            0x102484UL
#define EVE_REG_SWIZZLE          0x102460UL
#define EVE_REG_TAG              0x102478UL
#define EVE_REG_TAG_X            0x102470UL
#define EVE_REG_TAG_Y            0x102474UL
```

```
#define EVE_REG_TAP_CRC           0x102420UL
#define EVE_REG_TAP_MASK          0x102424UL
#define EVE_REG_TOUCH_ADC_MODE    0x1024f4UL
#define EVE_REG_TOUCH_CHARGE      0x1024f8UL
#define EVE_REG_TOUCH_DIRECT_XY   0x102574UL
#define EVE_REG_TOUCH_DIRECT_Z1Z2 0x102578UL
#define EVE_REG_TOUCH_MODE        0x1024f0UL
#define EVE_REG_TOUCH_OVERSAMPLE 0x102500UL
#define EVE_REG_TOUCH_RAW_XY      0x102508UL
#define EVE_REG_TOUCH_RZ          0x10250cUL
#define EVE_REG_TOUCH_RZTHRESH    0x102504UL
#define EVE_REG_TOUCH_SCREEN_XY   0x102510UL
#define EVE_REG_TOUCH_SETTLE      0x1024fcUL
#define EVE_REG_TOUCH_TAG         0x102518UL
#define EVE_REG_TOUCH_TAG_XY      0x102514UL
#define EVE_REG_TOUCH_TRANSFORM_A 0x10251cUL
#define EVE_REG_TOUCH_TRANSFORM_B 0x102520UL
#define EVE_REG_TOUCH_TRANSFORM_C 0x102524UL
#define EVE_REG_TOUCH_TRANSFORM_D 0x102528UL
#define EVE_REG_TOUCH_TRANSFORM_E 0x10252cUL
#define EVE_REG_TOUCH_TRANSFORM_F 0x102530UL
#define EVE_REG_TRACKER           0x109000UL
#define EVE_REG_TRIM              0x10256cUL
#define EVE_REG_VCYCLE            0x10243cUL
#define EVE_REG_VOFFSET           0x102440UL
#define EVE_REG_VOL_PB            0x10247cUL
#define EVE_REG_VOL_SOUND         0x102480UL
#define EVE_REG_VSIZE             0x102444UL
#define EVE_REG_VSYNC0            0x102448UL
#define EVE_REG_VSYNC1            0x10244cUL

#define EVE_ENC_ALPHA_FUNC(func,ref) ((0x9UL << 24)|(((func) & 0x7UL) << 8)|(((ref) &
0xffUL) << 0))
#define EVE_ENC_BEGIN(prim) ((0x1fUL << 24)|(((prim) & 0xFUL) << 0))
#define EVE_ENC_BITMAP_HANDLE(handle) ((0x5UL << 24)|(((handle) & 0x1fUL) << 0))
#define EVE_ENC_BITMAP_LAYOUT(format,linestride,height) ((0x7UL << 24)|(((format) &
0x1fUL) << 19)|(((linestride) & 0x3ffUL) << 9)|(((height) & 0x1ffUL) << 0))
#define EVE_ENC_BITMAP_SIZE(filter,wrapx,wrapy,width,height) ((0x8UL <<
24)|(((filter) & 0x1UL) << 20)|(((wrapx) & 0x1UL) << 19)|(((wrapy) & 0x1UL) <<
18)|(((width) & 0x1ffUL) << 9)|(((height) & 0x1ffUL) << 0))
#define EVE_ENC_BITMAP_SOURCE(addr) ((0x1UL << 24)|(((addr) & 0xffffffUL) << 0))
#define EVE_ENC_BITMAP_TRANSFORM_A(a) ((0x15UL << 24)|((((uint32_t)(a)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_B(b) ((0x16UL << 24)|((((uint32_t)(b)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_C(c) ((0x17UL << 24)|((((uint32_t)(c)) & 0xFFFFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_D(d) ((0x18UL << 24)|((((uint32_t)(d)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_E(e) ((0x19UL << 24)|((((uint32_t)(e)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_F(f) ((0x1aUL << 24)|((((uint32_t)(f)) & 0xFFFFFFUL)
<< 0))
#define EVE_ENC_BLEND_FUNC(src,dst) ((0xbUL << 24)|(((src) & 0x7UL) << 3)|(((dst) &
0x7UL) << 0))
#define EVE_ENC_CALL(dest) ((0x1dUL << 24)|(((dest) & 0xFFFFUL) << 0))
#define EVE_ENC_CELL(cell) ((0x6UL << 24)|(((cell) & 0x7fUL) << 0))
#define EVE_ENC_CLEAR_COLOR_A(alpha) ((0xfUL << 24)|(((alpha) & 0xffUL) << 0))
```

```
#define EVE_ENC_CLEAR_COLOR_RGB(red,green,blue) ((0x2UL << 24)|(((red) & 0xffUL) <<
16)|(((green) & 0xffUL) << 8)|(((blue) & 0xffUL) << 0))
#define EVE_ENC_CLEAR_COLOR(c) ((0x2UL << 24)|(((uint32_t)(c)) & 0x00ffffffUL))
#define EVE_ENC_CLEAR_STENCIL(s) ((0x11UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_CLEAR_TAG(s) ((0x12UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_CLEAR(c,s,t) ((0x26UL << 24)|((((uint32_t)(c)) & 0x1UL) <<
2)|((((uint32_t)(s)) & 0x1UL) << 1)|((((uint32_t)(t)) & 0x1UL) << 0))
#define EVE_ENC_COLOR_A(alpha) ((0x10UL << 24)|(((alpha) & 0xffUL) << 0))
#define EVE_ENC_COLOR_MASK(r,g,b,a) ((0x20UL << 24)|((((uint32_t)(r)) & 0x1UL) <<
3)|((((uint32_t)(g)) & 0x1UL) << 2)|((((uint32_t)(b)) & 0x1UL) <<
1)|((((uint32_t)(a)) & 0x1UL) << 0))
#define EVE_ENC_COLOR_RGB(red,green,blue) ((0x4UL << 24)|(((red) & 0xffUL) <<
16)|(((green) & 0xffUL) << 8)|(((blue) & 0xffUL) << 0))
#define EVE_ENC_COLOR(c) ((0x4UL << 24)|(((uint32_t)(c)) & 0x00ffffffUL))
#define EVE_ENC_DISPLAY() ((0x0UL << 24))
#define EVE_ENC_END() ((0x21UL << 24))
#define EVE_ENC_JUMP(dest) ((0x1eUL << 24)|(((dest) & 0xFFFFUL) << 0))
#define EVE_ENC_LINE_WIDTH(width) ((0xeUL << 24)|(((width) & 0xFFFUL) << 0))
#define EVE_ENC_MACRO(m) ((0x25UL << 24)|((((uint32_t)(m)) & 0x1UL) << 0))
#define EVE_ENC_POINT_SIZE(size) ((0xdUL << 24)|(((size) & 0x1FFFUL) << 0))
#define EVE_ENC_RESTORE_CONTEXT() ((0x23UL << 24))
#define EVE_ENC_RETURN() ((0x24UL << 24))
#define EVE_ENC_SAVE_CONTEXT() ((0x22UL << 24))
#define EVE_ENC_SCISSOR_SIZE(width,height) ((0x1cUL << 24)|(((width) & 0x3ffUL) <<
10)|(((height) & 0x3ffUL) << 0))
#define EVE_ENC_SCISSOR_XY(x,y) ((0x1bUL << 24)|((((uint32_t)(x)) & 0x1ffUL) <<
9)|((((uint32_t)(y)) & 0x1ffUL) << 0))
#define EVE_ENC_STENCIL_FUNC(func,ref,mask) ((0xaUL << 24)|(((func) & 0x7UL) <<
16)|(((ref) & 0xffUL) << 8)|(((mask) & 0xffUL) << 0))
#define EVE_ENC_STENCIL_MASK(mask) ((0x13UL << 24)|(((mask) & 0xffUL) << 0))
#define EVE_ENC_STENCIL_OP(sfail,spass) ((0xcUL << 24)|(((sfail) & 0x7UL) <<
3)|(((spass) & 0x7UL) << 0))
#define EVE_ENC_TAG_MASK(mask) ((0x14UL << 24)|(((mask) & 0x1UL) << 0))
#define EVE_ENC_TAG(s) ((0x3UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_VERTEX2F(x,y) ((0x1UL << 30)|((((uint32_t)(x)) & 0xffffUL) <<
15)|((((uint32_t)(y)) & 0xffffUL) << 0))
#define EVE_ENC_VERTEX2II(x,y,handle,cell) ((0x2UL << 30)|((((uint32_t)(x)) &
0x1ffUL) << 21)|((((uint32_t)(y)) & 0x1ffUL) << 12)|(((handle) & 0x1fUL) <<
7)|(((cell) & 0x7fUL) << 0))

#define EVE_ENC_CMD_APPEND           0xffffff1eUL
#define EVE_ENC_CMD_BGCOLOR          0xffffff09UL
#define EVE_ENC_CMD_BITMAP_TRANSFORM 0xffffff21UL
#define EVE_ENC_CMD_BUTTON           0xffffff0dUL
#define EVE_ENC_CMD_CALIBRATE        0xffffff15UL
#define EVE_ENC_CMD_CLOCK            0xffffff14UL
#define EVE_ENC_CMD_COLDSTART        0xffffff32UL
#define EVE_ENC_CMD_CRC              0xffffff03UL
#define EVE_ENC_CMD_CSKETCH                 0xffffff35UL
#define EVE_ENC_CMD_DIAL             0xffffff2dUL
#define EVE_ENC_CMD_DLSTART          0xffffff00UL
#define EVE_ENC_CMD_EXECUTE          0xffffff07UL
#define EVE_ENC_CMD_FGCOLOR          0xffffff0aUL
#define EVE_ENC_CMD_GAUGE            0xffffff13UL
#define EVE_ENC_CMD_GETMATRIX        0xffffff33UL
#define EVE_ENC_CMD_GETPOINT         0xffffff08UL
#define EVE_ENC_CMD_GETPROPS         0xffffff25UL
#define EVE_ENC_CMD_GETPTR           0xffffff23UL
#define EVE_ENC_CMD_GRADCOLOR        0xffffff34UL
#define EVE_ENC_CMD_GRADIENT         0xffffff0bUL
#define EVE_ENC_CMD_HAMMERAUX        0xffffff04UL
```

```
#define EVE_ENC_CMD_IDCT              0xffffff06UL
#define EVE_ENC_CMD_INFLATE           0xffffff22UL
#define EVE_ENC_CMD_INTERRUPT         0xffffff02UL
#define EVE_ENC_CMD_KEYS              0xffffff0eUL
#define EVE_ENC_CMD_LOADIDENTITY      0xffffff26UL
#define EVE_ENC_CMD_LOADIMAGE         0xffffff24UL
#define EVE_ENC_CMD_LOGO              0xffffff31UL
#define EVE_ENC_CMD_MARCH             0xffffff05UL
#define EVE_ENC_CMD_MEMCPY            0xffffff1dUL
#define EVE_ENC_CMD_MEMCRC            0xffffff18UL
#define EVE_ENC_CMD_MEMSET            0xffffff1bUL
#define EVE_ENC_CMD_MEMWRITE          0xffffff1aUL
#define EVE_ENC_CMD_MEMZERO           0xffffff1cUL
#define EVE_ENC_CMD_NUMBER            0xffffff2eUL
#define EVE_ENC_CMD_PROGRESS          0xffffff0fUL
#define EVE_ENC_CMD_REGREAD           0xffffff19UL
#define EVE_ENC_CMD_ROTATE            0xffffff29UL
#define EVE_ENC_CMD_SCALE             0xffffff28UL
#define EVE_ENC_CMD_SCREENSAVER       0xffffff2fUL
#define EVE_ENC_CMD_SCROLLBAR         0xffffff11UL
#define EVE_ENC_CMD_SETFONT           0xffffff2bUL
#define EVE_ENC_CMD_SETMATRIX         0xffffff2aUL
#define EVE_ENC_CMD_SKETCH            0xffffff30UL
#define EVE_ENC_CMD_SLIDER            0xffffff10UL
#define EVE_ENC_CMD_SNAPSHOT          0xffffff1fUL
#define EVE_ENC_CMD_SPINNER           0xffffff16UL
#define EVE_ENC_CMD_STOP              0xffffff17UL
#define EVE_ENC_CMD_SWAP              0xffffff01UL
#define EVE_ENC_CMD_TEXT              0xffffff0cUL
#define EVE_ENC_CMD_TOGGLE            0xffffff12UL
#define EVE_ENC_CMD_TOUCH_TRANSFORM   0xffffff20UL
#define EVE_ENC_CMD_TRACK             0xffffff2cUL
#define EVE_ENC_CMD_TRANSLATE         0xffffff27UL
```

```
#define EVE_BEGIN_BITMAPS               0x1UL
#define EVE_BEGIN_EDGE_STRIP_A          0x7UL
#define EVE_BEGIN_EDGE_STRIP_B          0x8UL
#define EVE_BEGIN_EDGE_STRIP_L          0x6UL
#define EVE_BEGIN_EDGE_STRIP_R          0x5UL
#define EVE_BEGIN_LINE_STRIP            0x4UL
#define EVE_BEGIN_LINES                 0x3UL
#define EVE_BEGIN_POINTS               0x2UL
#define EVE_BEGIN_RECTS                 0x9UL
#define EVE_BLEND_DST_ALPHA             0x3UL
#define EVE_BLEND_ONE                   0x1UL
#define EVE_BLEND_ONE_MINUS_DST_ALPHA   0x5UL
#define EVE_BLEND_ONE_MINUS_SRC_ALPHA   0x4UL
#define EVE_BLEND_SRC_ALPHA             0x2UL
#define EVE_BLEND_ZERO                  0x0UL
#define EVE_DLSWAP_DONE          0x0UL
#define EVE_DLSWAP_FRAME         0x2UL
#define EVE_DLSWAP_LINE          0x1UL
#define EVE_FILTER_BILINEAR              0x1UL
#define EVE_FILTER_NEAREST               0x0UL
#define EVE_FORMAT_ARGB1555              0x0UL
#define EVE_FORMAT_ARGB2                 0x5UL
#define EVE_FORMAT_ARGB4                 0x6UL
#define EVE_FORMAT_BARGRAPH              0xbUL
```

```
#define EVE_FORMAT_L1                   0x1UL
#define EVE_FORMAT_L4                   0x2UL
#define EVE_FORMAT_L8                   0x3UL
#define EVE_FORMAT_PALETTED             0x8UL
#define EVE_FORMAT_RGB332               0x4UL
#define EVE_FORMAT_RGB565               0x7UL
#define EVE_FORMAT_TEXT8X8              0x9UL
#define EVE_FORMAT_TEXTVGA              0xaUL
#define EVE_INT_CMDEMPTY         0x20UL
#define EVE_INT_CMDFLAG          0x40UL
#define EVE_INT_CONVCOMPLETE     0x80UL
#define EVE_INT_PLAYBACK         0x10UL
#define EVE_INT_SOUND            0x8UL
#define EVE_INT_SWAP             0x1UL
#define EVE_INT_TAG              0x4UL
#define EVE_INT_TOUCH            0x2UL
#define EVE_LINEAR_SAMPLES       0x0UL
#define EVE_OPT_CENTER           0x600UL
#define EVE_OPT_CENTERX          0x200UL
#define EVE_OPT_CENTERY          0x400UL
#define EVE_OPT_FLAT             0x100UL
#define EVE_OPT_MONO             0x1UL
#define EVE_OPT_NOBACK           0x1000UL
#define EVE_OPT_NODL             0x2UL
#define EVE_OPT_NOHANDS          0xc000UL
#define EVE_OPT_NOHM             0x4000UL
#define EVE_OPT_NOPOINTER        0x4000UL
#define EVE_OPT_NOSECS           0x8000UL
#define EVE_OPT_NOTICKS          0x2000UL
#define EVE_OPT_RIGHTX           0x800UL
#define EVE_OPT_SIGNED           0x100UL
#define EVE_STENCIL_DECR                0x4UL
#define EVE_STENCIL_INCR                0x3UL
#define EVE_STENCIL_INVERT              0x5UL
#define EVE_STENCIL_KEEP                0x1UL
#define EVE_STENCIL_REPLACE             0x2UL
#define EVE_STENCIL_ZERO                0x0UL
#define EVE_TEST_ALWAYS               0x7UL
#define EVE_TEST_EQUAL                0x5UL
#define EVE_TEST_GEQUAL               0x4UL
#define EVE_TEST_GREATER              0x3UL
#define EVE_TEST_LEQUAL               0x2UL
#define EVE_TEST_LESS                 0x1UL
#define EVE_TEST_NEVER                0x0UL
#define EVE_TEST_NOTEQUAL             0x6UL
#define EVE_TOUCHMODE_CONTINUOUS 0x3UL
#define EVE_TOUCHMODE_FRAME      0x2UL
#define EVE_TOUCHMODE_OFF        0x0UL
#define EVE_TOUCHMODE_ONESHOT    0x1UL
#define EVE_ULAW_SAMPLES         0x1UL
#define EVE_VOL_ZERO                0x0UL
#define EVE_WRAP_BORDER              0x0UL
#define EVE_WRAP_REPEAT              0x1UL
#define EVE_ADC_DIFFERENTIAL     0x1UL
#define EVE_ADC_SINGLE_ENDED     0x0UL
#define EVE_ADPCM_SAMPLES        0x2UL

#else

#define FT81X_VERSION "1.0.4"
```

```
/* For FT801 enable the switch in platform.h file */
/* Lower boundary of trimming */
#define LOW_FREQ_BOUND  58800000L//98% of 60Mhz

#define EVE_RAM_CMD              0x308000UL
#define EVE_RAM_CMD_SIZE         (4*1024L)
#define EVE_RAM_DL               0x300000UL
#define EVE_RAM_DL_SIZE          (8*1024L)
#define EVE_RAM_G                0x0UL
#define EVE_RAM_G_SIZE           (1024*1024L)
#define EVE_RAM_REG              0x302000UL
#define EVE_RAM_ROMSUB           0x30a000UL

#define EVE_ROMFONT_TABLEADDRESS 0x2ffffcUL
#define EVE_REG_ANA_COMP         0x302184UL
#define EVE_REG_ANALOG           0x30216cUL
#define EVE_REG_BIST_EN          0x302174UL
#define EVE_REG_BUSYBITS         0x3020e8UL
#define EVE_REG_CLOCK            0x302008UL
#define EVE_REG_CMD_DL           0x302100UL
#define EVE_REG_CMD_READ         0x3020f8UL
#define EVE_REG_CMD_WRITE        0x3020fcUL
#define EVE_REG_CMDB_SPACE       0x302574UL
#define EVE_REG_CMDB_WRITE       0x302578UL
#define EVE_REG_CPURESET         0x302020UL
#define EVE_REG_CRC              0x302178UL
#define EVE_REG_CSPREAD          0x302068UL
#define EVE_REG_CTOUCH_EXTENDED  0x302108UL
#define EVE_REG_CTOUCH_TOUCH0_XY 0x302124UL
#define EVE_REG_CTOUCH_TOUCH1_XY 0x30211cUL
#define EVE_REG_CTOUCH_TOUCH2_XY 0x30218cUL
#define EVE_REG_CTOUCH_TOUCH3_XY 0x302190UL
#define EVE_REG_CTOUCH_TOUCH4_X  0x30216cUL
#define EVE_REG_CTOUCH_TOUCH4_Y  0x302120UL
#define EVE_REG_CYA_TOUCH        0x302168UL
#define EVE_REG_DATESTAMP        0x302564UL
#define EVE_REG_DITHER           0x302060UL
#define EVE_REG_DLSWAP           0x302054UL
#define EVE_REG_FRAMES           0x302004UL
#define EVE_REG_FREQUENCY        0x30200cUL
#define EVE_REG_GPIO             0x302094UL
#define EVE_REG_GPIO_DIR         0x302090UL
#define EVE_REG_GPIOX            0x30209cUL
#define EVE_REG_GPIOX_DIR        0x302098UL
#define EVE_REG_HCYCLE           0x30202cUL
#define EVE_REG_HOFFSET          0x302030UL
#define EVE_REG_HSIZE            0x302034UL
#define EVE_REG_HSYNC0           0x302038UL
#define EVE_REG_HSYNC1           0x30203cUL
#define EVE_REG_ID               0x302000UL
#define EVE_REG_INT_EN           0x3020acUL
#define EVE_REG_INT_FLAGS        0x3020a8UL
#define EVE_REG_INT_MASK         0x3020b0UL
#define EVE_REG_MACRO_0          0x3020d8UL
#define EVE_REG_MACRO_1          0x3020dcUL
#define EVE_REG_MEDIAFIFO_READ   0x309014UL
#define EVE_REG_MEDIAFIFO_WRITE  0x309018UL
#define EVE_REG_OUTBITS          0x30205cUL
```

```
#define EVE_REG_PATCHED_ANALOG   0x302170UL
#define EVE_REG_PATCHED_TOUCH_FAULT 0x30216cUL
#define EVE_REG_PCLK             0x302070UL
#define EVE_REG_PCLK_POL         0x30206cUL
#define EVE_REG_PLAY             0x30208cUL
#define EVE_REG_PLAYBACK_FORMAT  0x3020c4UL
#define EVE_REG_PLAYBACK_FREQ    0x3020c0UL
#define EVE_REG_PLAYBACK_LENGTH  0x3020b8UL
#define EVE_REG_PLAYBACK_LOOP    0x3020c8UL
#define EVE_REG_PLAYBACK_PLAY    0x3020ccUL
#define EVE_REG_PLAYBACK_READPTR 0x3020bcUL
#define EVE_REG_PLAYBACK_START   0x3020b4UL
#define EVE_REG_PWM_DUTY         0x3020d4UL
#define EVE_REG_PWM_HZ           0x3020d0UL
#define EVE_REG_RENDERMODE       0x302010UL
#define EVE_REG_ROMSUB_SEL       0x3020f0UL
#define EVE_REG_ROTATE           0x302058UL
#define EVE_REG_SNAPFORMAT       0x30201cUL
#define EVE_REG_SNAPSHOT         0x302018UL
#define EVE_REG_SNAPY            0x302014UL
#define EVE_REG_SOUND            0x302088UL
#define EVE_REG_SPI_EARLY_TX     0x30217cUL
#define EVE_REG_SPI_WIDTH        0x302188UL
#define EVE_REG_SWIZZLE          0x302064UL
#define EVE_REG_TAG              0x30207cUL
#define EVE_REG_TAG_X            0x302074UL
#define EVE_REG_TAG_Y            0x302078UL
#define EVE_REG_TAP_CRC          0x302024UL
#define EVE_REG_TAP_MASK         0x302028UL
#define EVE_REG_TOUCH_ADC_MODE   0x302108UL
#define EVE_REG_TOUCH_CHARGE     0x30210cUL
#define EVE_REG_TOUCH_DIRECT_XY  0x30218cUL
#define EVE_REG_TOUCH_DIRECT_Z1Z2 0x302190UL
#define EVE_REG_TOUCH_FAULT      0x302170UL
#define EVE_REG_TOUCH_MODE       0x302104UL
#define EVE_REG_TOUCH_OVERSAMPLE 0x302114UL
#define EVE_REG_TOUCH_RAW_XY     0x30211cUL
#define EVE_REG_TOUCH_RZ         0x302120UL
#define EVE_REG_TOUCH_RZTHRESH   0x302118UL
#define EVE_REG_TOUCH_SCREEN_XY  0x302124UL
#define EVE_REG_TOUCH_SETTLE     0x302110UL
#define EVE_REG_TOUCH_TAG        0x30212cUL
#define EVE_REG_TOUCH_TAG_XY     0x302128UL
#define EVE_REG_TOUCH_TAG1       0x302134UL
#define EVE_REG_TOUCH_TAG1_XY    0x302130UL
#define EVE_REG_TOUCH_TAG2       0x30213cUL
#define EVE_REG_TOUCH_TAG2_XY    0x302138UL
#define EVE_REG_TOUCH_TAG3       0x302144UL
#define EVE_REG_TOUCH_TAG3_XY    0x302140UL
#define EVE_REG_TOUCH_TAG4       0x30214cUL
#define EVE_REG_TOUCH_TAG4_XY    0x302148UL
#define EVE_REG_TOUCH_TRANSFORM_A 0x302150UL
#define EVE_REG_TOUCH_TRANSFORM_B 0x302154UL
#define EVE_REG_TOUCH_TRANSFORM_C 0x302158UL
#define EVE_REG_TOUCH_TRANSFORM_D 0x30215cUL
#define EVE_REG_TOUCH_TRANSFORM_E 0x302160UL
#define EVE_REG_TOUCH_TRANSFORM_F 0x302164UL
#define EVE_REG_TRACKER          0x309000UL
#define EVE_REG_TRACKER_1        0x309004UL
#define EVE_REG_TRACKER_2        0x309008UL
#define EVE_REG_TRACKER_3        0x30900cUL
```

```
#define EVE_REG_TRACKER_4        0x309010UL
#define EVE_REG_TRIM             0x302180UL
#define EVE_REG_VCYCLE           0x302040UL
#define EVE_REG_VOFFSET          0x302044UL
#define EVE_REG_VOL_PB           0x302080UL
#define EVE_REG_VOL_SOUND        0x302084UL
#define EVE_REG_VSIZE            0x302048UL
#define EVE_REG_VSYNC0           0x30204cUL
#define EVE_REG_VSYNC1           0x302050UL

#define EVE_ENC_ALPHA_FUNC(func,ref) ((0x9UL << 24)|(((func) & 0x7UL) << 8)|(((ref) &
0xffUL) << 0))
#define EVE_ENC_BEGIN(prim) ((0x1fUL << 24)|(((prim)&15UL) << 0))
#define EVE_ENC_BITMAP_HANDLE(handle) ((0x5UL << 24)|(((handle) & 0x1fUL) << 0))
#define EVE_ENC_BITMAP_LAYOUT_H(linestride,height) ((0x28UL << 24)|(((linestride) &
0x3UL) << 2)|(((height) & 0x3UL) << 0))
#define EVE_ENC_BITMAP_LAYOUT(format,linestride,height) ((0x7UL << 24)|(((format) &
0x1fUL) << 19)|(((linestride) & 0x3ffUL) << 9)|(((height) & 0x1ffUL) << 0))
#define EVE_ENC_BITMAP_SIZE_H(width,height) ((0x29UL << 24)|(((width) & 0x3UL) <<
2)|(((height) & 0x3UL) << 0))
#define EVE_ENC_BITMAP_SIZE(filter,wrapx,wrapy,width,height) ((0x8UL <<
24)|(((filter) & 0x1UL) << 20)|(((wrapx) & 0x1UL) << 19)|(((wrapy) & 0x1UL) <<
18)|(((width) & 0x1ffUL) << 9)|(((height) & 0x1ffUL) << 0))
#define EVE_ENC_BITMAP_SOURCE(addr) ((0x1UL << 24)|(((addr) & 0x3FFFFFUL) << 0))
#define EVE_ENC_BITMAP_TRANSFORM_A(a) ((0x15UL << 24)|((((uint32_t)(a)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_B(b) ((0x16UL << 24)|((((uint32_t)(b)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_C(c) ((0x17UL << 24)|((((uint32_t)(c)) & 0xFFFFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_D(d) ((0x18UL << 24)|((((uint32_t)(d)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_E(e) ((0x19UL << 24)|((((uint32_t)(e)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_BITMAP_TRANSFORM_F(f) ((0x1aUL << 24)|((((uint32_t)(f)) & 0xFFFFFFUL)
<< 0))
#define EVE_ENC_BLEND_FUNC(src,dst) ((0xbUL << 24)|(((src) & 0x7UL) << 3)|(((dst) &
0x7UL) << 0))
#define EVE_ENC_CALL(dest) ((0x1dUL << 24)|(((dest) & 0xFFFFUL) << 0))
#define EVE_ENC_CELL(cell) ((0x6UL << 24)|(((cell) & 0x7fUL) << 0))
#define EVE_ENC_CLEAR_COLOR_A(alpha) ((0xfUL << 24)|(((alpha) & 0xffUL) << 0))
#define EVE_ENC_CLEAR_COLOR_RGB(red,green,blue) ((0x2UL << 24)|(((red) & 0xffUL) <<
16)|(((green) & 0xffUL) << 8)|(((blue) & 0xffUL) << 0))
#define EVE_ENC_CLEAR_COLOR(c) ((0x2UL << 24)|(((uint32_t)(c)) & 0x00ffffffUL))
#define EVE_ENC_CLEAR_STENCIL(s) ((0x11UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_CLEAR_TAG(s) ((0x12UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_CLEAR(c,s,t) ((0x26UL << 24)|((((uint32_t)(c)) & 0x1UL) <<
2)|((((uint32_t)(s)) & 0x1UL) << 1)|((((uint32_t)(t)) & 0x1UL) << 0))
#define EVE_ENC_COLOR_A(alpha) ((0x10UL << 24)|(((alpha) & 0xffUL) << 0))
#define EVE_ENC_COLOR_MASK(r,g,b,a) ((0x20UL << 24)|((((uint32_t)(r)) & 0x1UL) <<
3)|((((uint32_t)(g)) & 0x1UL) << 2)|((((uint32_t)(b)) & 0x1UL) <<
1)|((((uint32_t)(a)) & 0x1UL) << 0))
#define EVE_ENC_COLOR_RGB(red,green,blue) ((0x4UL << 24)|(((red) & 0xffUL) <<
16)|(((green) & 0xffUL) << 8)|(((blue) & 0xffUL) << 0))
#define EVE_ENC_COLOR(c) ((0x4UL << 24)|(((uint32_t)(c)) & 0x00ffffffUL))
#define EVE_ENC_DISPLAY() ((0x0UL << 24))
#define EVE_ENC_END() ((0x21UL << 24))
#define EVE_ENC_JUMP(dest) ((0x1eUL << 24)|(((dest) & 0xFFFFUL) << 0))
```

```
#define EVE_ENC_LINE_WIDTH(width) ((0xeUL << 24)|(((width) & 0xFFFUL) << 0))
#define EVE_ENC_MACRO(m) ((0x25UL << 24)|((((uint32_t)(m)) & 0x1UL) << 0))
#define EVE_ENC_NOP() ((0x2dUL << 24))
#define EVE_ENC_PALETTE_SOURCE(addr) ((0x2aUL << 24)|(((addr) & 0x3FFFFFUL) << 0))
#define EVE_ENC_POINT_SIZE(size) ((0xdUL << 24)|(((size) & 0x1FFFUL) << 0))
#define EVE_ENC_RESTORE_CONTEXT() ((0x23UL << 24))
#define EVE_ENC_RETURN() ((0x24UL << 24))
#define EVE_ENC_SAVE_CONTEXT() ((0x22UL << 24))
#define EVE_ENC_SCISSOR_SIZE(width,height) ((0x1cUL << 24)|(((width) & 0xFFFUL) <<
12)|(((height) & 0xFFFUL) << 0))
#define EVE_ENC_SCISSOR_XY(x,y) ((0x1bUL << 24)|((((uint32_t)(x)) & 0x7FFUL) <<
11)|((((uint32_t)(y)) & 0x7FFUL) << 0))
#define EVE_ENC_STENCIL_FUNC(func,ref,mask) ((0xaUL << 24)|(((func) & 0x7UL) <<
16)|(((ref) & 0xffUL) << 8)|(((mask) & 0xffUL) << 0))
#define EVE_ENC_STENCIL_MASK(mask) ((0x13UL << 24)|(((mask) & 0xffUL) << 0))
#define EVE_ENC_STENCIL_OP(sfail,spass) ((0xcUL << 24)|(((sfail) & 0x7UL) <<
3)|(((spass) & 0x7UL) << 0))
#define EVE_ENC_TAG_MASK(mask) ((0x14UL << 24)|(((mask) & 0x1UL) << 0))
#define EVE_ENC_TAG(s) ((0x3UL << 24)|((((uint32_t)(s)) & 0xffUL) << 0))
#define EVE_ENC_VERTEX_FORMAT(frac) ((0x27UL << 24)|(((frac) & 0x7UL) << 0))
#define EVE_ENC_VERTEX_TRANSLATE_X(x) ((0x2bUL << 24)|((((uint32_t)(x)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_VERTEX_TRANSLATE_Y(y) ((0x2cUL << 24)|((((uint32_t)(y)) & 0x1FFFFUL)
<< 0))
#define EVE_ENC_VERTEX2F(x,y) ((0x1UL << 30)|((((uint32_t)(x)) & 0xffffUL) <<
15)|((((uint32_t)(y)) & 0xffffUL) << 0))
#define EVE_ENC_VERTEX2II(x,y,handle,cell) ((0x2UL << 30)|((((uint32_t)(x)) &
0x1ffUL) << 21)|((((uint32_t)(y)) & 0x1ffUL) << 12)|(((handle) & 0x1fUL) <<
7)|(((cell) & 0x7fUL) << 0))

#define EVE_ENC_CMD_APPEND           0xffffff1eUL
#define EVE_ENC_CMD_BGCOLOR          0xffffff09UL
#define EVE_ENC_CMD_BITMAP_TRANSFORM 0xffffff21UL
#define EVE_ENC_CMD_BUTTON           0xffffff0dUL
#define EVE_ENC_CMD_CALIBRATE        0xffffff15UL
#define EVE_ENC_CMD_CLOCK            0xffffff14UL
#define EVE_ENC_CMD_COLDSTART        0xffffff32UL
#define EVE_ENC_CMD_CRC              0xffffff03UL
#define EVE_ENC_CMD_CSKETCH             0xffffff35UL
#define EVE_ENC_CMD_DIAL             0xffffff2dUL
#define EVE_ENC_CMD_DLSTART          0xffffff00UL
#define EVE_ENC_CMD_EXECUTE          0xffffff07UL
#define EVE_ENC_CMD_FGCOLOR          0xffffff0aUL
#define EVE_ENC_CMD_GAUGE            0xffffff13UL
#define EVE_ENC_CMD_GETMATRIX        0xffffff33UL
#define EVE_ENC_CMD_GETPOINT         0xffffff08UL
#define EVE_ENC_CMD_GETPROPS         0xffffff25UL
#define EVE_ENC_CMD_GETPTR           0xffffff23UL
#define EVE_ENC_CMD_GRADCOLOR        0xffffff34UL
#define EVE_ENC_CMD_GRADIENT         0xffffff0bUL
#define EVE_ENC_CMD_HAMMERAUX        0xffffff04UL
#define EVE_ENC_CMD_IDCT_DELETED     0xffffff06UL
#define EVE_ENC_CMD_INFLATE          0xffffff22UL
#define EVE_ENC_CMD_INT_RAMSHARED    0xffffff3dUL
#define EVE_ENC_CMD_INT_SWLOADIMAGE  0xffffff3eUL
#define EVE_ENC_CMD_INTERRUPT        0xffffff02UL
#define EVE_ENC_CMD_KEYS             0xffffff0eUL
#define EVE_ENC_CMD_LOADIDENTITY     0xffffff26UL
#define EVE_ENC_CMD_LOADIMAGE        0xffffff24UL
#define EVE_ENC_CMD_LOGO             0xffffff31UL
#define EVE_ENC_CMD_MARCH            0xffffff05UL
```

```
#define EVE_ENC_CMD_MEDIAFIFO        0xffffff39UL
#define EVE_ENC_CMD_MEMCPY           0xffffff1dUL
#define EVE_ENC_CMD_MEMCRC           0xffffff18UL
#define EVE_ENC_CMD_MEMSET           0xffffff1bUL
#define EVE_ENC_CMD_MEMWRITE         0xffffff1aUL
#define EVE_ENC_CMD_MEMZERO          0xffffff1cUL
#define EVE_ENC_CMD_NUMBER           0xffffff2eUL
#define EVE_ENC_CMD_PLAYVIDEO        0xffffff3aUL
#define EVE_ENC_CMD_PROGRESS         0xffffff0fUL
#define EVE_ENC_CMD_REGREAD          0xffffff19UL
#define EVE_ENC_CMD_ROMFONT          0xffffff3fUL
#define EVE_ENC_CMD_ROTATE           0xffffff29UL
#define EVE_ENC_CMD_SCALE            0xffffff28UL
#define EVE_ENC_CMD_SCREENSAVER      0xffffff2fUL
#define EVE_ENC_CMD_SCROLLBAR        0xffffff11UL
#define EVE_ENC_CMD_SETBASE          0xffffff38UL
#define EVE_ENC_CMD_SETBITMAP        0xffffff43UL
#define EVE_ENC_CMD_SETFONT          0xffffff2bUL
#define EVE_ENC_CMD_SETFONT2         0xffffff3bUL
#define EVE_ENC_CMD_SETMATRIX        0xffffff2aUL
#define EVE_ENC_CMD_SETROTATE        0xffffff36UL
#define EVE_ENC_CMD_SETSCRATCH       0xffffff3cUL
#define EVE_ENC_CMD_SKETCH           0xffffff30UL
#define EVE_ENC_CMD_SLIDER           0xffffff10UL
#define EVE_ENC_CMD_SNAPSHOT         0xffffff1fUL
#define EVE_ENC_CMD_SNAPSHOT2        0xffffff37UL
#define EVE_ENC_CMD_SPINNER          0xffffff16UL
#define EVE_ENC_CMD_STOP             0xffffff17UL
#define EVE_ENC_CMD_SWAP             0xffffff01UL
#define EVE_ENC_CMD_SYNC             0xffffff42UL
#define EVE_ENC_CMD_TEXT             0xffffff0cUL
#define EVE_ENC_CMD_TOGGLE           0xffffff12UL
#define EVE_ENC_CMD_TOUCH_TRANSFORM  0xffffff20UL
#define EVE_ENC_CMD_TRACK            0xffffff2cUL
#define EVE_ENC_CMD_TRANSLATE        0xffffff27UL
#define EVE_ENC_CMD_VIDEOFRAME       0xffffff41UL
#define EVE_ENC_CMD_VIDEOSTART       0xffffff40UL

#define EVE_BEGIN_BITMAPS              0x1UL
#define EVE_BEGIN_EDGE_STRIP_A         0x7UL
#define EVE_BEGIN_EDGE_STRIP_B         0x8UL
#define EVE_BEGIN_EDGE_STRIP_L         0x6UL
#define EVE_BEGIN_EDGE_STRIP_R         0x5UL
#define EVE_BEGIN_LINE_STRIP           0x4UL
#define EVE_BEGIN_LINES                0x3UL
#define EVE_BEGIN_POINTS              0x2UL
#define EVE_BEGIN_RECTS                0x9UL
#define EVE_BLEND_DST_ALPHA            0x3UL
#define EVE_BLEND_ONE                  0x1UL
#define EVE_BLEND_ONE_MINUS_DST_ALPHA  0x5UL
#define EVE_BLEND_ONE_MINUS_SRC_ALPHA  0x4UL
#define EVE_BLEND_SRC_ALPHA            0x2UL
#define EVE_BLEND_ZERO                 0x0UL
#define EVE_DLSWAP_DONE          0x0UL
#define EVE_DLSWAP_FRAME         0x2UL
#define EVE_DLSWAP_LINE          0x1UL
#define EVE_FILTER_BILINEAR             0x1UL
#define EVE_FILTER_NEAREST              0x0UL
```

```
#define EVE_FORMAT_ARGB1555             0x0UL
#define EVE_FORMAT_ARGB2                0x5UL
#define EVE_FORMAT_ARGB4                0x6UL
#define EVE_FORMAT_BARGRAPH             0xbUL
#define EVE_FORMAT_L1                   0x1UL
#define EVE_FORMAT_L2                   0x11UL
#define EVE_FORMAT_L4                   0x2UL
#define EVE_FORMAT_L8                   0x3UL
#define EVE_FORMAT_PALETTED             0x8UL
#define EVE_FORMAT_PALETTED4444         0xfUL
#define EVE_FORMAT_PALETTED565          0xeUL
#define EVE_FORMAT_PALETTED8            0x10UL
#define EVE_FORMAT_RGB332               0x4UL
#define EVE_FORMAT_RGB565               0x7UL
#define EVE_FORMAT_TEXT8X8              0x9UL
#define EVE_FORMAT_TEXTVGA              0xaUL
#define EVE_INT_CMDEMPTY         0x20UL
#define EVE_INT_CMDFLAG          0x40UL
#define EVE_INT_CONVCOMPLETE     0x80UL
#define EVE_INT_G8               0x12UL
#define EVE_INT_L8C              0xcUL
#define EVE_INT_PLAYBACK         0x10UL
#define EVE_INT_SOUND            0x8UL
#define EVE_INT_SWAP             0x1UL
#define EVE_INT_TAG              0x4UL
#define EVE_INT_TOUCH            0x2UL
#define EVE_INT_VGA              0xdUL
#define EVE_LINEAR_SAMPLES       0x0UL
#define EVE_OPT_CENTER           0x600UL
#define EVE_OPT_CENTERX          0x200UL
#define EVE_OPT_CENTERY          0x400UL
#define EVE_OPT_FLAT             0x100UL
#define EVE_OPT_FULLSCREEN       0x8UL
#define EVE_OPT_MEDIAFIFO        0x10UL
#define EVE_OPT_MONO             0x1UL
#define EVE_OPT_NOBACK           0x1000UL
#define EVE_OPT_NODL             0x2UL
#define EVE_OPT_NOHANDS          0xc000UL
#define EVE_OPT_NOHM             0x4000UL
#define EVE_OPT_NOPOINTER        0x4000UL
#define EVE_OPT_NOSECS           0x8000UL
#define EVE_OPT_NOTEAR           0x4UL
#define EVE_OPT_NOTICKS          0x2000UL
#define EVE_OPT_RIGHTX           0x800UL
#define EVE_OPT_SIGNED           0x100UL
#define EVE_OPT_SOUND            0x20UL
#define EVE_STENCIL_DECR                0x4UL
#define EVE_STENCIL_INCR                0x3UL
#define EVE_STENCIL_INVERT              0x5UL
#define EVE_STENCIL_KEEP                0x1UL
#define EVE_STENCIL_REPLACE             0x2UL
#define EVE_STENCIL_ZERO                0x0UL
#define EVE_TEST_ALWAYS              0x7UL
#define EVE_TEST_EQUAL               0x5UL
#define EVE_TEST_GEQUAL              0x4UL
#define EVE_TEST_GREATER             0x3UL
#define EVE_TEST_LEQUAL              0x2UL
#define EVE_TEST_LESS                0x1UL
#define EVE_TEST_NEVER               0x0UL
#define EVE_TEST_NOTEQUAL            0x6UL
#define EVE_TOUCHMODE_CONTINUOUS 0x3UL
```

```
#define EVE_TOUCHMODE_FRAME      0x2UL
#define EVE_TOUCHMODE_OFF        0x0UL
#define EVE_TOUCHMODE_ONESHOT    0x1UL
#define EVE_ULAW_SAMPLES         0x1UL
#define EVE_VOL_ZERO                 0x0UL
#define EVE_WRAP_BORDER               0x0UL
#define EVE_WRAP_REPEAT               0x1UL
#define EVE_ADC_DIFFERENTIAL     0x1UL //?
#define EVE_ADC_SINGLE_ENDED     0x0UL //?
#define EVE_ADPCM_SAMPLES        0x2UL //?

#endif

#define FT_GPU_NUMCHAR_PERFONT (128)
#define FT_GPU_FONT_TABLE_SIZE (148)

/* FT81x and FT80x font table structure */
/* Font table address in ROM can be found by reading the address from 0xFFFFC
location. */
/* 16 font tables are present at the address read from location 0xFFFFC */
typedef struct
{
    /* All the values are in bytes */
    /* Width of each character font from 0 to 127 */
    uint8_t  FontWidth[FT_GPU_NUMCHAR_PERFONT];
    /* Bitmap format of font wrt bitmap formats supported by FT800 - L1, L4, L8 */
    uint32_t FontBitmapFormat;
    /* Font line stride in FT800 ROM */
    uint32_t FontLineStride;
    /* Font width in pixels */
    uint32_t FontWidthInPixels;
    /* Font height in pixels */
    uint32_t FontHeightInPixels;
    /* Pointer to font graphics raw data */
    uint32_t PointerToFontGraphicsData;
} EVE_GPU_FONT_HEADER;

/* Nothing beyond this */

#endif /* _FT8XX_H_ */
```

HAL.h

```
#ifndef HAL_HEADER_H
#define      HAL_HEADER_H

#include <stdint.h> // for Uint8/16/32 and Int8/16/32 data types

#include "FT8xx.h" // Register and command definitions for FT8xx

/**
 @brief Initialise EVE HAL Layer.
 @details Power cycle and start the EVE display in a controlled manner.
       This will call the MCU-specific initialisation routine and check for
       the presence of a supported FT8xx device on the SPI bus.
 */
```

```
void HAL_EVE_Init(void);

/**
 @brief Chip Select Control
 @details Abstract the low-level MCU chip select control to either
       enable or disable. A call to this function to enable chip select
       will result in a logic low on the SPI CS line.
 @param enable - Non-zero to enable chip select
                          Zero to deselect.
 */
void HAL_ChipSelect(int8_t enable);

/**
 @brief Power Down Control
 @details Abstract the low-level MCU power down control line to
       control the EVE display. A call to this function to enable
       power down will result in a logig low on the PD line.
 @param enable - Non-zero to enable power down
                          Zero to disable (normal operating state of EVE).
 */
void HAL_PowerDown(int8_t enable);

/**
 @brief Increment command memory write pointer
 @details The command memory write pointer stores the current
       location where coprocessor commands are written. This is
       kept internally in the HAL and is NOT written to the
       REG_CMD_WRITE register on EVE until the HAL_WriteCmdPointer
       function is called. This allows multiple commands (up-to
       the size of the command memory) to be stored and executed
       when required.
 @param commandSize - The number of bytes to advance the command
       memory write pointer. This will wrap at the end of the
       command memory.
 */
void HAL_IncCmdPointer(uint16_t commandSize);

/**
 @brief Get the current command memory write pointer
 @details Obtains the current command memory write pointer from
       the value stored internally in the HAL. It is not the value
       read from the REG_CMD_WRITE register.
 @returns Command memory write pointer
 */
uint16_t HAL_GetCmdPointer(void);

/**
 @brief Commits the current command memory write pointer
 @details Stores the internal HAL command memory write pointer to
       the REG_CMD_WRITE register. This will start the coprocessor
       working through items in the display list.
 */
void HAL_WriteCmdPointer(void);

/**
 @brief Wait for display list to complete
 @details Polls the REG_CMD_READ register until it matches the
       current command memory write pointer. This will indicate that
       the coprocessor has completed working through the items in the
       display list.
 @returns Zero for normal completion or 0xff for an error condition.
```

```
 */
uint8_t HAL_WaitCmdFifoEmpty(void);

/**
 @brief Calculate free space in the command memory.
 @details Works out how many bytes of command memory is available
       to receive display list instructions. There will always be
       4 bytes (one display list entry) reserved in the command memory
       to ensure that the display list can never create a loop.
 @returns Number of free bytes.
 */
uint16_t HAL_CheckCmdFreeSpace(void);

/**
 @brief Write a 32 bit value to an EVE memory location
 @details Formats a memory space write to EVE. This can be any register
         or mapped memory on the device (display list or command list).
         This function will control chip select.
 @param address - 24 bit address on EVE
 @param val32 - value to write
 */
void HAL_MemWrite32(uint32_t address, uint32_t val32);

/**
 @brief Write a 16 bit value to an EVE memory location
 @details Formats a memory space write to EVE. This can be any register
         or mapped memory on the device (display list or command list).
         This function will control chip select.
 @param address - 24 bit address on EVE
 @param val16 - value to write
 */
void HAL_MemWrite16(uint32_t address, uint16_t val16);

/**
 @brief Write an 8 bit value to an EVE memory location
 @details Formats a memory space write to EVE. This can be any register
         or mapped memory on the device (display list or command list).
         This function will control chip select.
 @param address - 24 bit address on EVE
 @param val8 - value to write
 */
void HAL_MemWrite8(uint32_t address, uint8_t val8);

/**
 @brief Read a 32 bit value from an EVE memory location
 @details Formats a memory space read to EVE. This can be any register
         or mapped memory on the device (display list or command list).
         This function will control chip select.
 @param address - 24 bit address on EVE
 @returns value read from EVE
 */
uint32_t HAL_MemRead32(uint32_t address);

/**
 @brief Read a 16 bit value from an EVE memory location
 @details Formats a memory space read to EVE. This can be any register
         or mapped memory on the device (display list or command list).
         This function will control chip select.
```

```
 @param address - 24 bit address on EVE
 @returns value read from EVE
 */
uint16_t HAL_MemRead16(uint32_t address);

/**
 @brief Read an 8 bit value from an EVE memory location
 @details Formats a memory space read to EVE. This can be any register
          or mapped memory on the device (display list or command list).
          This function will control chip select.
 @param address - 24 bit address on EVE
 @returns value read from EVE
 */
uint8_t HAL_MemRead8(uint32_t address);

/**
 @brief Sends an address for writing to EVE
 @details Formats a memory address for writing to the EVE.
          This function will not control chip select.
 @param address - 24 bit address on EVE
 */
void HAL_SetWriteAddress(uint32_t address);

/**
 @brief Sends an address for reading to EVE
 @details Formats a memory address for read from the EVE.
          This function will not control chip select.
 @param address - 24 bit address on EVE
 */
void HAL_SetReadAddress(uint32_t address);

/**
 @brief Sends an 8 bit command to EVE
 @details Sends a 8 bit command and paramter using SPI to the EVE.
          This function will control chip select.
 @returns 8 bit value read
 */
void HAL_HostCmdWrite(uint8_t cmd, uint8_t param);

/**
 @brief Sends a block of data to EVE
 @details Sends a block of data using SPI to the EVE.
          This function will not control chip select.
 @param val32 - 32 bit value
 */
void HAL_Write(const uint8_t *buffer, uint32_t length);

/**
 @brief Sends a 32 bit value to EVE
 @details Sends a 32 bit value using SPI to the EVE.
          This function will not control chip select.
 @param val32 - 32 bit value
 */
void HAL_Write32(uint32_t val32);

/**
 @brief Sends a 16 bit value to EVE
 @details Sends a 16 bit value using SPI to the EVE.
          This function will not control chip select.
 @param val16 - 16 bit value
 */
```

```
void HAL_Write16(uint16_t val16);

/**
 @brief Sends a 8 bit value to EVE
 @details Sends a 8 bit value using SPI to the EVE.
          This function will not control chip select.
 @param val8 - 8 bit value
 */
void HAL_Write8(uint8_t val8);

/**
 @brief Reads a 32 bit value from EVE
 @details Sends a 32 bit dummy value using SPI to the EVE
        and receives the result.
          This function will not control chip select.
 @returns 32 bit value read
 */
uint32_t HAL_Read32(void);

/**
 @brief Reads a 16 bit value from EVE
 @details Sends a 16 bit dummy value using SPI to the EVE
        and receives the result.
          This function will not control chip select.
 @returns 16 bit value read
 */
uint16_t HAL_Read16(void);

/**
 @brief Reads a 8 bit value from EVE
 @details Sends a 8 bit dummy value using SPI to the EVE
        and receives the result.
          This function will not control chip select.
 @returns 8 bit value read
 */
uint8_t HAL_Read8(void);

#endif /* HAL_HEADER_H */
```

MCU.h

```
#ifndef MCU_HEADER_H
#define      MCU_HEADER_H

#include <stdint.h> // for Uint8/16/32 and Int8/16/32 data types

#include "FT8xx.h" // Register and command definitions for FT8xx

/**
 @brief MCU specific initialisation
 @details Must contain any MCU-specific initialisation. This will typically be
        setting up the SPI bus, GPIOs and operating environment requirements.
 */
void MCU_Init(void);

/**
 @brief MCU specific setup
```

```
 @details Called after the EVE has been power cycled and started. Contains
       any MCU-specific configuration options for the EVE.
 */
void MCU_Setup(void);

/**
 @brief MCU specific chip select enable
 @details This function will pull the chip select line to the EVE low to
       allow data transmission on the SPI bus.
       The EVE requires chip select to toggle frequently.
 */
void MCU_CSlow(void);

/**
 @brief MCU specific chip select deassert
 @details This function will pull the chip select line to the EVE high to
       prevent data transmission on the SPI bus.
       The EVE requires chip select to toggle frequently.
 */
void MCU_CShigh(void);

/**
 @brief MCU specific power down enable
 @details This function will pull the power down line to the EVE low to
       force the device into power down mode.
       This will be done during EVE initialisation and can be done to allow
       deep power saving.
 */
void MCU_PDlow(void);

/**
 @brief MCU specific power down disable
 @details This function will pull the power down line to the EVE high to
       enable normal operation of the EVE.
       This will be done during EVE initialisation and can be done to allow
       recovery from deep power saving.
 */
void MCU_PDhigh(void);

/**
 @brief MCU specific SPI write
 @details Performs an SPI write of the data block and discards the data
       received in response.
 @param DataToWrite - pointer to buffer to write.
 @param length - number of bytes to write.
 */
void MCU_SPIWrite(const uint8_t *DataToWrite, uint32_t length);

/**
 @brief MCU specific SPI 8 bit read
 @details Performs an SPI dummy write and returns the data received in
       response.
 @returns Data received from EVE.
 */
uint8_t MCU_SPIRead8(void);

/**
 @brief MCU specific SPI 8 bit write
 @details Performs an SPI write and discards the data received in
       response.
 @param Data to write to EVE.
```

```
 */
void MCU_SPIWrite8(uint8_t DataToWrite);

/**
 @brief MCU specific SPI 16 bit read
 @details Performs an SPI dummy write and returns the data received in
       response.
 @returns Data received from EVE.
 */
uint16_t MCU_SPIRead16(void);

/**
 @brief MCU specific SPI 16 bit write
 @details Performs an SPI write and discards the data received in
       response.
 @param Data to write to EVE.
 */
void MCU_SPIWrite16(uint16_t DataToWrite);

/**
 @brief MCU specific SPI 24 bit read
 @details Performs an SPI dummy write and returns the data received in
       response.
 @returns Data received from EVE.
 */
uint32_t MCU_SPIRead24(void);

/**
 @brief MCU specific SPI 24 bit write
 @details Performs an SPI write and discards the data received in
       response.
 @param Data to write to EVE.
 */
void MCU_SPIWrite24(uint32_t DataToWrite);

/**
 @brief MCU specific SPI 32 bit read
 @details Performs an SPI dummy write and returns the data received in
       response.
 @returns Data received from EVE.
 */
uint32_t MCU_SPIRead32(void);

/**
 @brief MCU specific SPI 32 bit write
 @details Performs an SPI write and discards the data received in
       response.
 @param Data to write to EVE.
 */
void MCU_SPIWrite32(uint32_t DataToWrite);

/**
 @brief MCU specific 20 ms delay
 @details Cause the MCU to idle or otherwise delay for a minimum of
       20 milliseconds. This is used during initialisation to perform a
       power down of the EVE for a controlled minimum period of time.
 */
void MCU_Delay_20ms(void);
```

```
/**
 @brief MCU specific 500 ms delay
 @details Cause the MCU to idle or otherwise delay for a minimum of
       500 milliseconds. This is used during initialisation to perform a
       power down of the EVE for a controlled minimum period of time.
 */
void MCU_Delay_500ms(void);

/**
 @brief MCU specific byte swapping routines
 @details EVE addresses from the HAL_SetReadAddress and HAL_SetWriteAddress
       are sent in big-endian format. However, data for registers or memory
       mapped areas are in little-endian format.
 */
//@{
uint16_t MCU_htobe16(uint16_t h);
uint32_t MCU_htobe32(uint32_t h);
uint16_t MCU_htole16(uint16_t h);
uint32_t MCU_htole32(uint32_t h);
uint16_t MCU_be16toh(uint16_t h);
uint32_t MCU_be32toh(uint32_t h);
uint16_t MCU_le16toh(uint16_t h);
uint32_t MCU_le32toh(uint32_t h);
//@}

#endif /* MCU_HEADER_H */
```

usbkeyboard

usb_keyb_structs.c

```
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <usbkeyboardheader.h>

#include "driverlib2.h"
#include <utils/lwiplib.h>



const uint8_t g_pui8LangDescriptor[] =
{
    4,
    USB_DTYPE_STRING,
    USBShort(USB_LANG_ES_MODERN)
};

const uint8_t g_pui8ManufacturerString[] =
{
    (17 + 1) * 2,
    USB_DTYPE_STRING,
```

```
    'G', 0, 'u', 0, 'z', 0, 'M', 0, 'a', 0, 'n', 0, 'z', 0, ' ', 0, 'I', 0,
    'n', 0, 'd', 0, 'u', 0, 's', 0, 't', 0, 'r', 0, 'y', 0, ' ', 0,
};

const uint8_t g_pui8ProductString[] =
{
    (16 + 1) * 2,
    USB_DTYPE_STRING,
    'E', 0, 'V', 0, 'E', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0, 'o', 0,
    'a', 0, 'r', 0, 'd', 0, ' ', 0, 'O', 0, 'N', 0, 'E', 0
};

const uint8_t g_pui8SeriailNumberString[] =
{
    (8 + 1) * 2,
    USB_DTYPE_STRING,
    '1', 0, '2', 0, '3', 0, '4', 0, '5', 0, '6', 0, '7', 0, '8', 0
};

const uint8_t g_pui8HIDInterfaceString[] =
{
    (22 + 1) * 2,
    USB_DTYPE_STRING,
    'H', 0, 'I', 0, 'D', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0,
    'o', 0, 'a', 0, 'r', 0, 'd', 0, ' ', 0, 'I', 0, 'n', 0, 't', 0,
    'e', 0, 'r', 0, 'f', 0, 'a', 0, 'c', 0, 'e', 0
};

const uint8_t g_pui8ConfigString[] =
{
    (26 + 1) * 2,
    USB_DTYPE_STRING,
    'H', 0, 'I', 0, 'D', 0, ' ', 0, 'K', 0, 'e', 0, 'y', 0, 'b', 0,
    'o', 0, 'a', 0, 'r', 0, 'd', 0, ' ', 0, 'C', 0, 'o', 0, 'n', 0,
    'f', 0, 'i', 0, 'g', 0, 'u', 0, 'r', 0, 'a', 0, 't', 0, 'i', 0,
    'o', 0, 'n', 0
};

const uint8_t * const g_ppui8StringDescriptors[] =
{
    g_pui8LangDescriptor,
    g_pui8ManufacturerString,
    g_pui8ProductString,
    g_pui8SeriailNumberString,
    g_pui8HIDInterfaceString,
    g_pui8ConfigString
};

#define NUM_STRING_DESCRIPTORS (sizeof(g_ppui8StringDescriptors) /            \
                                sizeof(uint8_t *))
tUSBDHIDKeyboardDevice g_sKeyboardDevice =
{
    USB_VID_TI_1CBE,
    USB_PID_KEYBOARD,
    500,
    USB_CONF_ATTR_SELF_PWR | USB_CONF_ATTR_RWAKE,
    KeyboardHandler,
    (void *)&g_sKeyboardDevice,
```

```
    g_ppui8StringDescriptors,
    NUM_STRING_DESCRIPTORS
};
//*****************************************************************************
#######
static const int8_t g_ppi8KeyUsageCodes[][2] =
{
    { 0, HID_KEYB_USAGE_SPACE },                        //   0x20
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_1 },          // ! 0x21
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_FQUOTE },     // " 0x22
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_3 },          // # 0x23
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_4 },          // $ 0x24
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_5 },          // % 0x25
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_7 },          // & 0x26
    { 0, HID_KEYB_USAGE_FQUOTE },                       // ' 0x27
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_9 },          // ( 0x28
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_0 },          // ) 0x29
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_8 },          // * 0x2a
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_EQUAL },      // + 0x2b
    { 0, HID_KEYB_USAGE_COMMA },                        // , 0x2c
    { 0, HID_KEYB_USAGE_MINUS },                        // - 0x2d
    { 0, HID_KEYB_USAGE_PERIOD },                       // . 0x2e
    { 0, HID_KEYB_USAGE_FSLASH },                       // / 0x2f
    { 0, HID_KEYB_USAGE_0 },                            // 0 0x30
    { 0, HID_KEYB_USAGE_1 },                            // 1 0x31
    { 0, HID_KEYB_USAGE_2 },                            // 2 0x32
    { 0, HID_KEYB_USAGE_3 },                            // 3 0x33
    { 0, HID_KEYB_USAGE_4 },                            // 4 0x34
    { 0, HID_KEYB_USAGE_5 },                            // 5 0x35
    { 0, HID_KEYB_USAGE_6 },                            // 6 0x36
    { 0, HID_KEYB_USAGE_7 },                            // 7 0x37
    { 0, HID_KEYB_USAGE_8 },                            // 8 0x38
    { 0, HID_KEYB_USAGE_9 },                            // 9 0x39
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_SEMICOLON },  // : 0x3a
    { 0, HID_KEYB_USAGE_SEMICOLON },                    // ; 0x3b
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_COMMA },      // < 0x3c
    { 0, HID_KEYB_USAGE_EQUAL },                        // = 0x3d
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_PERIOD },     // > 0x3e
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_FSLASH },     // ? 0x3f
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_2 },          // @ 0x40
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_A },          // A 0x41
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_B },          // B 0x42
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_C },          // C 0x43
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_D },          // D 0x44
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_E },          // E 0x45
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_F },          // F 0x46
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_G },          // G 0x47
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_H },          // H 0x48
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_I },          // I 0x49
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_J },          // J 0x4a
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_K },          // K 0x4b
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_L },          // L 0x4c
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_M },          // M 0x4d
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_N },          // N 0x4e
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_O },          // O 0x4f
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_P },          // P 0x50
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_Q },          // Q 0x51
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_R },          // R 0x52
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_S },          // S 0x53
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_T },          // T 0x54
    { HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_U },          // U 0x55
```

```
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_V },          // V 0x56
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_W },          // W 0x57
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_X },          // X 0x58
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_Y },          // Y 0x59
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_Z },          // Z 0x5a
{ 0, HID_KEYB_USAGE_LBRACKET },                     // [ 0x5b
{ 0, HID_KEYB_USAGE_BSLASH },                       // \ 0x5c
{ 0, HID_KEYB_USAGE_RBRACKET },                     // ] 0x5d
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_6 },          // ^ 0x5e
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_MINUS },      // _ 0x5f
{ 0, HID_KEYB_USAGE_BQUOTE },                       // ` 0x60
{ 0, HID_KEYB_USAGE_A },                            // a 0x61
{ 0, HID_KEYB_USAGE_B },                            // b 0x62
{ 0, HID_KEYB_USAGE_C },                            // c 0x63
{ 0, HID_KEYB_USAGE_D },                            // d 0x64
{ 0, HID_KEYB_USAGE_E },                            // e 0x65
{ 0, HID_KEYB_USAGE_F },                            // f 0x66
{ 0, HID_KEYB_USAGE_G },                            // g 0x67
{ 0, HID_KEYB_USAGE_H },                            // h 0x68
{ 0, HID_KEYB_USAGE_I },                            // i 0x69
{ 0, HID_KEYB_USAGE_J },                            // j 0x6a
{ 0, HID_KEYB_USAGE_K },                            // k 0x6b
{ 0, HID_KEYB_USAGE_L },                            // l 0x6c
{ 0, HID_KEYB_USAGE_M },                            // m 0x6d
{ 0, HID_KEYB_USAGE_N },                            // n 0x6e
{ 0, HID_KEYB_USAGE_O },                            // o 0x6f
{ 0, HID_KEYB_USAGE_P },                            // p 0x70
{ 0, HID_KEYB_USAGE_Q },                            // q 0x71
{ 0, HID_KEYB_USAGE_R },                            // r 0x72
{ 0, HID_KEYB_USAGE_S },                            // s 0x73
{ 0, HID_KEYB_USAGE_T },                            // t 0x74
{ 0, HID_KEYB_USAGE_U },                            // u 0x75
{ 0, HID_KEYB_USAGE_V },                            // v 0x76
{ 0, HID_KEYB_USAGE_W },                            // w 0x77
{ 0, HID_KEYB_USAGE_X },                            // x 0x78
{ 0, HID_KEYB_USAGE_Y },                            // y 0x79
{ 0, HID_KEYB_USAGE_Z },                            // z 0x7a
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_LBRACKET },   // { 0x7b
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_BSLASH },     // | 0x7c
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_RBRACKET },   // } 0x7d
{ HID_KEYB_LEFT_SHIFT, HID_KEYB_USAGE_BQUOTE },     // ~ 0x7e
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_1},                          // ~ 0x7f
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_2},                          // ~ 0x80
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_3},                          // ~ 0x81
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_4},                          // ~ 0x82
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_5},                          // ~ 0x83
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_6},                          // ~ 0x84
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_7},                          // ~ 0x85
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_8},                          // ~ 0x86
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_9},                          // ~ 0x87
{HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_0},                          // ~ 0x88
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_1},        // ~ 0x89
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_2},        // ~ 0x8a
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_3},        // ~ 0x8b
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_4},        // ~ 0x8c
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_5},        // ~ 0x8d
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_6},        // ~ 0x8e
{HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_7},        // ~ 0x8f
```

```
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_8},        // ~ 0x90
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_9},        // ~ 0x91
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_0},        // ~ 0x92
    {HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_SLASH},                      // ~ 0x93
    {HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_STAR},                       // ~ 0x94
    {HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_PLUS},                       // ~ 0x96
    {HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_PERIOD},                     // ~ 0x97
    {HID_KEYB_LEFT_CTRL, HID_KEYB_USAGE_KEYPAD_EQUAL},                      // ~ 0x98
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_SLASH},    // ~ 0x99
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_STAR},     // ~ 0x9a
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_MINUS},    // ~ 0x9b
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_PLUS},     // ~ 0x9c
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_PERIOD},   // ~ 0x9d
    {HID_KEYB_LEFT_CTRL|HID_KEYB_LEFT_ALT, HID_KEYB_USAGE_KEYPAD_EQUAL},    // ~ 0x9e
    //
    // Add characters outside of 0x20-0x7e here to avoid breaking the table
    // lookup calculations.
    //
    { 0, HID_KEYB_USAGE_ENTER },                        // LF 0x0A
};

#define SYSTICKS_PER_SECOND     100

volatile bool g_bConnected = false;

volatile bool g_bSuspended = false;

volatile uint32_t g_ui32SysTickCount;

#define MAX_SEND_DELAY          50

volatile bool g_bDisplayUpdateRequired;

volatile enum
{

    STATE_UNCONFIGURED,

    STATE_IDLE,

    STATE_SENDING
}
g_eKeyboardState = STATE_UNCONFIGURED;

uint32_t KeyboardHandler(void *pvCBData, uint32_t ui32Event, uint32_t ui32MsgData,
                 void *pvMsgData)
{
    switch (ui32Event)
    {

        case USB_EVENT_CONNECTED:
        {
            g_bConnected = true;
            g_bSuspended = false;
            break;
        }

        case USB_EVENT_DISCONNECTED:
        {
            g_bConnected = false;
            break;
```

```
        }

        case USB_EVENT_TX_COMPLETE:
        {

            g_eKeyboardState = STATE_IDLE;
            break;
        }

        case USB_EVENT_SUSPEND:
        {
            g_bSuspended = true;
            break;
        }

        case USB_EVENT_RESUME:
        {
            g_bSuspended = false;
            break;
        }

        case USBD_HID_KEYB_EVENT_SET_LEDS:
        {

            MAP_GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,
                             (ui32MsgData & HID_KEYB_CAPS_LOCK) ? GPIO_PIN_0 :
                             0);

            break;
        }

        default:
        {
            break;
        }
    }

    return(0);
}

bool WaitForSendIdle(uint_fast32_t ui32TimeoutTicks)
{
    uint32_t ui32Start;
    uint32_t ui32Now;
    uint32_t ui32Elapsed;

    ui32Start = g_ui32SysTickCount;
    ui32Elapsed = 0;

    while(ui32Elapsed < ui32TimeoutTicks)
    {

        if(g_eKeyboardState == STATE_IDLE)
        {
            return(true);
        }

        ui32Now = g_ui32SysTickCount;
```

```
        ui32Elapsed = ((ui32Start < ui32Now) ? (ui32Now - ui32Start) :
                    (((uint32_t)0xFFFFFFFF - ui32Start) + ui32Now + 1));
    }

    return(false);
}

void SendString(char *pcStr)
{
    uint32_t ui32Char;

    while(*pcStr)
    {

        ui32Char = *pcStr++;

        if((ui32Char < ' ') || (ui32Char > '~'))
        {

            if (ui32Char != '\n')
            {
            continue;
            }
        }

        if (ui32Char == '\n'){
            ui32Char = 0x5f;
        }
        else{

        ui32Char -= ' ';
        }

        g_eKeyboardState = STATE_SENDING;
        if(USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice,
g_ppi8KeyUsageCodes[ui32Char][0], g_ppi8KeyUsageCodes[ui32Char][1], true) !=
KEYB_SUCCESS){
            return;
        }

        if(!WaitForSendIdle(MAX_SEND_DELAY))
        {
            g_bConnected = 0;
            return;
        }

        g_eKeyboardState = STATE_SENDING;
        if(USBDHIDKeyboardKeyStateChange((void *)&g_sKeyboardDevice,0,
g_ppi8KeyUsageCodes[ui32Char][1],false) != KEYB_SUCCESS){
            return;
        }

        if(!WaitForSendIdle(MAX_SEND_DELAY)){
            g_bConnected = 0;
            return;
        }
    }
}

void SysTickIntHandler(void){
```

```
    g_ui32SysTickCount++;
    lwIPTimer(SYSTICKMS); //de Html
}

extern uint32_t ui32SysClock;
void Inicia_Teclado(){

    uint32_t ui32PLLRate;

    PinoutSet(true, true);

    g_bConnected = false;
    g_bSuspended = false;

    USBStackModeSet(0, eUSBModeDevice, 0);

    SysCtlVCOGet(SYSCTL_XTAL_25MHZ, &ui32PLLRate);
    USBDCDFeatureSet(0, USBLIB_FEATURE_CPUCLK, &ui32SysClock);
    USBDCDFeatureSet(0, USBLIB_FEATURE_USBPLL, &ui32PLLRate);

    USBDHIDKeyboardInit(0, &g_sKeyboardDevice);

    MAP_SysTickPeriodSet(ui32SysClock / SYSTICKS_PER_SECOND);
    MAP_SysTickIntEnable();
    MAP_SysTickEnable();

}
```

usb_keyb_structs.h

```
#ifndef __USB_KEYB_STRUCTS_H__
#define __USB_KEYB_STRUCTS_H__

//*****************************************************************************
//
// If building with a C++ compiler, make all of the definitions in this header
// have a C binding.
//
//*****************************************************************************
#ifdef __cplusplus
extern "C"
{
#endif

#ifndef SYSTICKHZ
#define SYSTICKHZ               100
#define SYSTICKMS               (1000 / SYSTICKHZ)
#endif

extern tUSBDHIDKeyboardDevice g_sKeyboardDevice;

uint32_t KeyboardHandler(void *pvCBData, uint32_t ui32Event, uint32_t ui32MsgData,
                         void *pvMsgData);
bool WaitForSendIdle(uint_fast32_t ui32TimeoutTicks);
void SendString(char *pcStr);
void SysTickIntHandler(void);
void Inicia_Teclado();
```

```
static const int8_t g_ppi8KeyUsageCodes[][2];

#define HID_KEYB_USAGE_KEYPAD_EQUAL 0x67

//*****************************************************************************
//
// Mark the end of the C bindings section for C++ compilers.
//
//*****************************************************************************
#ifdef __cplusplus
}
#endif

#endif // __USB_KEYB_STRUCTS_H__
```

ethernet

io_fs.c

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>

#include "driverlib2.h"

#include "evethernet.h"

//*****************************************************************************
//
// Include the web file system data for this application.  This file is
// generated by the makefsfile utility, using the following command:
//
//     ../../../../tools/bin/makefsfile -i fs -o io_fsdata.h -r -h -q
//
// If any changes are made to the static content of the web pages served by the
// application, this script must be used to regenerate io_fsdata.h in order
// for those changes to be picked up by the web server.
//
//C:\ti\TivaWare_C_Series-2.2.0.295\tools\bin\makefsfile -i fs -o io_fsdata.h -r -h -
q
//
//*****************************************************************************
#include "io_fsdata.h"


extern char auxTecString[36][5];
extern char auxRedString[10][5];
extern char auxGrnString[10][5];
extern char auxBluString[10][5];
extern char teclado[36][1];

int ip;

//*****************************************************************************
//
// Open a file and return a handle to the file, if found.  Otherwise,
// return NULL.  This function also looks for special file names used to
```

```
// provide specific status information or to control various subsystems.
// These file names are used by the JavaScript on the "IO Control Demo 1"
// example web page.
//
//*****************************************************************************
struct fs_file *
fs_open(const char *pcName)
{
    const struct fsdata_file *psTree;
    struct fs_file *psFile = NULL;

    psFile = mem_malloc(sizeof(struct fs_file));
    if(psFile == NULL)
    {
        return(NULL);
    }

    if(ustrncmp(pcName, "/pag_minus", 10) == 0)
    {
        static char pcBuf[2];

        io_change_numPag(true,pcBuf, 2);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/pag_plus", 9) == 0)
    {
        static char pcBuf[2];

        io_change_numPag(false,pcBuf, 2);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/Bot_minus", 10) == 0)
    {
        static char pcBuf[3];

        io_change_numVirTec(true,pcBuf, 7);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }
```

```
else if(ustrncmp(pcName, "/Bot_plus", 9) == 0)
{
    static char pcBuf[3];

    io_change_numVirTec(false,pcBuf, 7);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/ModTecCtrl", 11) == 0)
{
    static char pcBuf[5];

    io_change_ModTec(1,pcBuf, 15);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/ModTecAlt", 10) == 0)
{
    static char pcBuf[4];

    io_change_ModTec(2,pcBuf, 15);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/ModTecShift", 12) == 0)
{
    static char pcBuf[6];

    io_change_ModTec(3,pcBuf, 15);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/ModTecNone", 11) == 0)
{
    static char pcBuf[5];
```

```
    io_change_ModTec(4,pcBuf, 15);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/SubmitButton", 13) == 0)
{
    io_submit_state();
    //            psFile->data = pcBuf;
    //            psFile->len = strlen(pcBuf);
    //            psFile->index = psFile->len;
    //            psFile->pextension = NULL;
    //
    //            return(psFile);
}

else if(ustrncmp(pcName, "/WindBut", 8) == 0)
{
    static char pcBuf[6];

    io_change_windbut(pcBuf, 6);
    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

ip=0;
while(ustrncmp(pcName, auxTecString[ip], 5) != 0 && ip<36){
    ip++;
}

if(ustrncmp(pcName, auxTecString[ip], 5) == 0 && ip<36){
    static char pcBuf[2];

    io_change_Tec(ip,pcBuf, 2);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

ip=0;
while(ustrncmp(pcName, auxRedString[ip], 5) != 0 && ip<10){
    ip++;
}

if(ustrncmp(pcName, auxRedString[ip], 5) == 0 && ip<10){
```

```
        static char pcBuf[4];

        io_change_RedVal(ip,pcBuf, 4);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    ip=0;
    while(ustrncmp(pcName, auxGrnString[ip], 5) != 0 && ip<10){
        ip++;
    }

    if(ustrncmp(pcName, auxGrnString[ip], 5) == 0 && ip<10){
        static char pcBuf[4];

        io_change_GrnVal(ip,pcBuf, 4);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    ip=0;
    while(ustrncmp(pcName, auxBluString[ip], 5) != 0 && ip<10){
        ip++;
    }

    if(ustrncmp(pcName, auxBluString[ip], 5) == 0 && ip<10){
        static char pcBuf[4];

        io_change_BluVal(ip,pcBuf, 4);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }


////////////////////////////////////////////////////////////////////////////////

    if(ustrncmp(pcName, "/numPagState", 12) == 0)
    {
        static char pcBuf[2];

        io_set_numPagState(pcBuf, 2);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
```

```
        return(psFile);
    }

    else if(ustrncmp(pcName, "/NumBotState", 12) == 0)
    {
        static char pcBuf[3];

        io_set_NumBotState(pcBuf, 3);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/ModTecState", 12) == 0)
    {
        static char pcBuf[15];

        io_set_ModTecState(pcBuf, 15);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/TecState", 9) == 0)
    {
        static char pcBuf[2];

        io_set_TecState(pcBuf, 2);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/RedState", 9) == 0)
    {
        static char pcBuf[4];

        io_set_RedState(pcBuf, 4);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/GrnState", 9) == 0)
    {
        static char pcBuf[4];
```

```
    io_set_GrnState(pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}

else if(ustrncmp(pcName, "/BluState", 9) == 0)
{
    static char pcBuf[4];

    io_set_BluState(pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}
//
// If I can't find it there, look in the rest of the main psFile system
//
else
{
    //
    // Initialize the psFile system tree pointer to the root of the linked
    // list.
    //
    psTree = FS_ROOT;

    //
    // Begin processing the linked list, looking for the requested file name.
    //
    while(NULL != psTree)
    {
        //
        // Compare the requested file "pcName" to the file name in the
        // current node.
        //
        if(ustrncmp(pcName, (char *)psTree->name, psTree->len) == 0)
        {
            //
            // Fill in the data pointer and length values from the
            // linked list node.
            //
            psFile->data = (char *)psTree->data;
            psFile->len = psTree->len;

            //
            // For now, we setup the read index to the end of the file,
            // indicating that all data has been read.
            //
            psFile->index = psTree->len;

            //
            // We are not using any file system extensions in this
            // application, so set the pointer to NULL.
            //
```

```
                psFile->pextension = NULL;

                //
                // Exit the loop and return the file system pointer.
                //
                break;
            }

            //
            // If we get here, we did not find the file at this node of the
            // linked list.  Get the next element in the list.
            //
            psTree = psTree->next;
        }
    }

    //
    // If we didn't find the file, ptTee will be NULL.  Make sure we
    // return a NULL pointer if this happens.
    //
    if(NULL == psTree)
    {
        mem_free(psFile);
        psFile = NULL;
    }

    //
    // Return the file system pointer.
    //
    return(psFile);
}

//*****************************************************************************
//
// Close an opened file designated by the handle.
//
//*****************************************************************************
void
fs_close(struct fs_file *psFile)
{
    //
    // Free the main psFile system object.
    //
    mem_free(psFile);
}

//*****************************************************************************
//
// Read the next chunk of data from the file.  Return the iCount of data
// that was read.  Return 0 if no data is currently available.  Return
// a -1 if at the end of file.
//
//*****************************************************************************
int
fs_read(struct fs_file *psFile, char *pcBuffer, int iCount)
{
    int iAvailable;
```

```
    //
    // Check to see if a command (pextension = 1).
    //
    if(psFile->pextension == (void *)1)
    {
        //
        // Nothing to do for this file type.
        //
        psFile->pextension = NULL;
        return(-1);
    }

    //
    // Check to see if more data is available.
    //
    if(psFile->len == psFile->index)
    {
        //
        // There is no remaining data.  Return a -1 for EOF indication.
        //
        return(-1);
    }

    //
    // Determine how much data we can copy.  The minimum of the 'iCount'
    // parameter or the available data in the file system buffer.
    //
    iAvailable = psFile->len - psFile->index;
    if(iAvailable > iCount)
    {
        iAvailable = iCount;
    }

    //
    // Copy the data.
    //
    memcpy(pcBuffer, psFile->data + iAvailable, iAvailable);
    psFile->index += iAvailable;

    //
    // Return the count of data that we copied.
    //
    return(iAvailable);
}

//*****************************************************************************
//
// Determine the number of bytes left to read from the file.
//
//*****************************************************************************
int
fs_bytes_left(struct fs_file *psFile)
{
    //
    // Return the number of bytes left to be read from this file.
    //
    return(psFile->len - psFile->index);
}
```

io.c

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>

#include "driverlib2.h"

#include "evethernet.h"

//*****************************************************************************
//
// Include the web file system data for this application.  This file is
// generated by the makefsfile utility, using the following command:
//
//     ../../../../tools/bin/makefsfile -i fs -o io_fsdata.h -r -h -q
//
// If any changes are made to the static content of the web pages served by the
// application, this script must be used to regenerate io_fsdata.h in order
// for those changes to be picked up by the web server.
//
//C:\ti\TivaWare_C_Series-2.2.0.295\tools\bin\makefsfile -i fs -o io_fsdata.h -r -h -
q
//
//*****************************************************************************
#include "io_fsdata.h"


extern char auxTecString[36][5];
extern char auxRedString[10][5];
extern char auxGrnString[10][5];
extern char auxBluString[10][5];
extern char teclado[36][1];

int ip;

//*****************************************************************************
//
// Open a file and return a handle to the file, if found.  Otherwise,
// return NULL.  This function also looks for special file names used to
// provide specific status information or to control various subsystems.
// These file names are used by the JavaScript on the "IO Control Demo 1"
// example web page.
//
//*****************************************************************************
struct fs_file *
fs_open(const char *pcName)
{
    const struct fsdata_file *psTree;
    struct fs_file *psFile = NULL;

    psFile = mem_malloc(sizeof(struct fs_file));
    if(psFile == NULL)
    {
        return(NULL);
    }

    if(ustrncmp(pcName, "/pag_minus", 10) == 0)
```

```
{
    static char pcBuf[2];

    io_change_numPag(true,pcBuf, 2);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/pag_plus", 9) == 0)
{
    static char pcBuf[2];

    io_change_numPag(false,pcBuf, 2);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/Bot_minus", 10) == 0)
{
    static char pcBuf[3];

    io_change_numVirTec(true,pcBuf, 7);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/Bot_plus", 9) == 0)
{
    static char pcBuf[3];

    io_change_numVirTec(false,pcBuf, 7);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

else if(ustrncmp(pcName, "/ModTecCtrl", 11) == 0)
{
    static char pcBuf[5];

    io_change_ModTec(1,pcBuf, 15);
```

```
        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/ModTecAlt", 10) == 0)
    {
        static char pcBuf[4];

        io_change_ModTec(2,pcBuf, 15);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/ModTecShift", 12) == 0)
    {
        static char pcBuf[6];

        io_change_ModTec(3,pcBuf, 15);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/ModTecNone", 11) == 0)
    {
        static char pcBuf[5];

        io_change_ModTec(4,pcBuf, 15);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    else if(ustrncmp(pcName, "/SubmitButton", 13) == 0)
    {
        io_submit_state();
        //            psFile->data = pcBuf;
        //            psFile->len = strlen(pcBuf);
        //            psFile->index = psFile->len;
        //            psFile->pextension = NULL;
        //
```

```
    //            return(psFile);
}

else if(ustrncmp(pcName, "/WindBut", 8) == 0)
{
    static char pcBuf[6];

    io_change_windbut(pcBuf, 6);
    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

ip=0;
while(ustrncmp(pcName, auxTecString[ip], 5) != 0 && ip<36){
    ip++;
}

if(ustrncmp(pcName, auxTecString[ip], 5) == 0 && ip<36){
    static char pcBuf[2];

    io_change_Tec(ip,pcBuf, 2);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

ip=0;
while(ustrncmp(pcName, auxRedString[ip], 5) != 0 && ip<10){
    ip++;
}

if(ustrncmp(pcName, auxRedString[ip], 5) == 0 && ip<10){
    static char pcBuf[4];

    io_change_RedVal(ip,pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;

    return(psFile);
}

ip=0;
while(ustrncmp(pcName, auxGrnString[ip], 5) != 0 && ip<10){
    ip++;
}

if(ustrncmp(pcName, auxGrnString[ip], 5) == 0 && ip<10){
    static char pcBuf[4];

    io_change_GrnVal(ip,pcBuf, 4);
```

```
        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }

    ip=0;
    while(ustrncmp(pcName, auxBluString[ip], 5) != 0 && ip<10){
        ip++;
    }

    if(ustrncmp(pcName, auxBluString[ip], 5) == 0 && ip<10){
        static char pcBuf[4];

        io_change_BluVal(ip,pcBuf, 4);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;

        return(psFile);
    }


////////////////////////////////////////////////////////////////////////////////

    if(ustrncmp(pcName, "/numPagState", 12) == 0)
    {
        static char pcBuf[2];

        io_set_numPagState(pcBuf, 2);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/NumBotState", 12) == 0)
    {
        static char pcBuf[3];

        io_set_NumBotState(pcBuf, 3);

        psFile->data = pcBuf;
        psFile->len = strlen(pcBuf);
        psFile->index = psFile->len;
        psFile->pextension = NULL;
        return(psFile);
    }

    else if(ustrncmp(pcName, "/ModTecState", 12) == 0)
    {
```

```
    static char pcBuf[15];

    io_set_ModTecState(pcBuf, 15);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}

else if(ustrncmp(pcName, "/TecState", 9) == 0)
{
    static char pcBuf[2];

    io_set_TecState(pcBuf, 2);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}

else if(ustrncmp(pcName, "/RedState", 9) == 0)
{
    static char pcBuf[4];

    io_set_RedState(pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}

else if(ustrncmp(pcName, "/GrnState", 9) == 0)
{
    static char pcBuf[4];

    io_set_GrnState(pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
}

else if(ustrncmp(pcName, "/BluState", 9) == 0)
{
    static char pcBuf[4];

    io_set_BluState(pcBuf, 4);

    psFile->data = pcBuf;
    psFile->len = strlen(pcBuf);
    psFile->index = psFile->len;
    psFile->pextension = NULL;
    return(psFile);
```

```
}
//
// If I can't find it there, look in the rest of the main psFile system
//
else
{
    //
    // Initialize the psFile system tree pointer to the root of the linked
    // list.
    //
    psTree = FS_ROOT;

    //
    // Begin processing the linked list, looking for the requested file name.
    //
    while(NULL != psTree)
    {
        //
        // Compare the requested file "pcName" to the file name in the
        // current node.
        //
        if(ustrncmp(pcName, (char *)psTree->name, psTree->len) == 0)
        {
            //
            // Fill in the data pointer and length values from the
            // linked list node.
            //
            psFile->data = (char *)psTree->data;
            psFile->len = psTree->len;

            //
            // For now, we setup the read index to the end of the file,
            // indicating that all data has been read.
            //
            psFile->index = psTree->len;

            //
            // We are not using any file system extensions in this
            // application, so set the pointer to NULL.
            //
            psFile->pextension = NULL;

            //
            // Exit the loop and return the file system pointer.
            //
            break;
        }

        //
        // If we get here, we did not find the file at this node of the
        // linked list.  Get the next element in the list.
        //
        psTree = psTree->next;
    }
}

//
// If we didn't find the file, ptTee will be NULL.  Make sure we
```

```
    // return a NULL pointer if this happens.
    //
    if(NULL == psTree)
    {
        mem_free(psFile);
        psFile = NULL;
    }

    //
    // Return the file system pointer.
    //
    return(psFile);
}

//*****************************************************************************
//
// Close an opened file designated by the handle.
//
//*****************************************************************************
void
fs_close(struct fs_file *psFile)
{
    //
    // Free the main psFile system object.
    //
    mem_free(psFile);
}

//*****************************************************************************
//
// Read the next chunk of data from the file.  Return the iCount of data
// that was read.  Return 0 if no data is currently available.  Return
// a -1 if at the end of file.
//
//*****************************************************************************
int
fs_read(struct fs_file *psFile, char *pcBuffer, int iCount)
{
    int iAvailable;

    //
    // Check to see if a command (pextension = 1).
    //
    if(psFile->pextension == (void *)1)
    {
        //
        // Nothing to do for this file type.
        //
        psFile->pextension = NULL;
        return(-1);
    }

    //
    // Check to see if more data is available.
    //
    if(psFile->len == psFile->index)
    {
        //
        // There is no remaining data.  Return a -1 for EOF indication.
        //
        return(-1);
```

```
    }

    //
    // Determine how much data we can copy.  The minimum of the 'iCount'
    // parameter or the available data in the file system buffer.
    //
    iAvailable = psFile->len - psFile->index;
    if(iAvailable > iCount)
    {
        iAvailable = iCount;
    }

    //
    // Copy the data.
    //
    memcpy(pcBuffer, psFile->data + iAvailable, iAvailable);
    psFile->index += iAvailable;

    //
    // Return the count of data that we copied.
    //
    return(iAvailable);
}

//*****************************************************************************
//
// Determine the number of bytes left to read from the file.
//
//*****************************************************************************
int
fs_bytes_left(struct fs_file *psFile)
{
    //
    // Return the number of bytes left to be read from this file.
    //
    return(psFile->len - psFile->index);
}
```

io.h

```
#ifndef __IO_H__
#define __IO_H__

#ifdef __cplusplus
extern "C"
{
#endif

extern volatile unsigned long g_ulAnimSpeed;

void io_change_numPag(bool PlusMin,char * pcBuf, int iBufLen);
void io_set_numPagState(char * pcBuf, int iBufLen);
void io_change_numVirTec(bool PlusMin,char * pcBuf, int iBufLen);
void io_set_NumBotState(char * pcBuf, int iBufLen);
void io_change_ModTec(uint8_t Tecla,char * pcBuf, int iBufLen);
void io_set_ModTecState(char * pcBuf, int iBufLen);
void io_change_Tec(uint8_t ip,char * pcBuf, int iBufLen);
```

```
void io_set_TecState(char * pcBuf, int iBufLen);
void io_change_RedVal(uint8_t ip,char * pcBuf, int iBufLen);
void io_set_RedState(char * pcBuf, int iBufLen);
void io_change_GrnVal(uint8_t ip,char * pcBuf, int iBufLen);
void io_set_GrnState(char * pcBuf, int iBufLen);
void io_change_BluVal(uint8_t ip,char * pcBuf, int iBufLen);
void io_set_BluState(char * pcBuf, int iBufLen);
void io_submit_state();
void io_change_windbut(char * pcBuf, int iBufLen);
void io_set_windbut(char * pcBuf, int iBufLen);
void AnimTimerIntHandler(void);
void lwIPHostTimerHandler(void);
void Init_Ethernet_Server();
void CHECK_FLAGS();
void Init_Cad_Comms();

#ifdef __cplusplus
}
#endif

#endif // __IO_H__
```

fs

index.htm

```
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>
  <html>
    <head>
      <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
      <title>EK-TM4C1294XL + EVE</title>
      <link rel="stylesheet" type="text/css" href="styles.css"/>
      <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"/>
      <script src="javascript.js" language="JavaScript1.2" charset="utf-8"></script>
      <script    src="javascript_load.js"    language="JavaScript1.2"    charset="utf-
8"></script>
    </head>
    <body>
      <div id="heading">
        <table width="100%">
          <tr>
            <!--<td>
              <a id="heading_ti" class="1" target="_" href="http://www.ti.com">
                <img src="ti.jpg" alt="Texas Instruments" width=20em>
              </a>
            </td>
            <td>-->
              <div id="heading_h1">
                Demostración de tecnología
              </div>
              <div id="heading_h2">
                EK-TM4C1294XL + ME813A-WH50C
              </div>
            </td>
            <!--<td>
                <a id="heading_chip" target="_" href="https://ftdichip.com/">
                <img src="chip.png" alt="FTDI Chip">
              </a>
            </td>-->
          </tr>
        </table>
        <hr><hr>
```

```
      </div>
      <div id="menu">
        <ul>
          <li><a id="principal">Pagina Principal</a></li>
          <li><a id="instrucciones">Instrucciones</a></li>
          <li><a id="editorPag">Editor de Pantalla</a></li>
          <li><a id="editorBot">Editor de Boton</a></li>
          <!--<li><a     target="_"     href="http://www.ti.com/tool/ek-tm4c1294xl">EK-
TM4C1294XL Product Page</a></li>
          <li><a target="_" href="http://www.ti.com/tm4c">TM4C Series TM4C129x Family
Product Page</a></li>-->
          <li><a id="info">Info de sistemas</a></li>
        </ul>
        <!--<table>
          <tr>
            <td>
              <img id="placa" src="ek-tm4c1294xl.jpg" alt="EK-TM4C1294XL">
            </td>
            <td>
              <img id="pantalla" src="pantalla.png" alt="ME813A-WH50CL">
            </td>
          </tr>
        </table>-->
      </div>
      <div id="content">
      </div>
      <div id="footing">
        <hr><hr>
        Pablo Manuel Guzmán Manzanares.   Escuela Tecnica Superior de Ingeniería,
Universidad de Sevilla.
      </div>
    </body>
  </html>
```

info.htm

```
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>
  <html>
  <head>
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>EK-TM4C1294XL Evaluation Kit</title>
    <link rel="stylesheet" type="text/css" href="styles.css"/>
    <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"/>
  </head>
  <body>
    <p id="content_heading">Info de sistemas</p>
    <div id="content_body">Click en las imagenes para acceder a la información del
hardware:</div>
    <div id="CampoEntrada">
      <a href="https://www.ti.com/tool/EK-TM4C1294XL" target="_blank">
        <img alt="EK-TM4C1294XL" src="../fs/ek-tm4c1294xl.jpg">
      </a>
    </div>
    <div id="CampoEntrada">
      <a         href="https://brtchip.com/wp-content/uploads/EVE2/DS_ME813A-WH50C.pdf"
target="_blank">
        <img alt="EK-TM4C1294XL" src="../fs/pantalla.png">
      </a>
    </div>
  </body>
  </html>
```

instrucciones.htm

```
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>
  <html>
  <head>
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>Instrucciones</title>
    <link rel="stylesheet" type="text/css" href="styles.css"/>
    <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"/>
  </head>
  <body>
    <p id="content_heading">Instrucciones</p>
    <div id="content_body">
      Estas son las instrucciones para el uso de las características de personalización
que proprociona la Web:
    </div>
    <div id="CampoEntrada">
      <p>Pantalla principal y general:</p>
      <img  alt="Pagina  principal"  src="../fs/Instr1.png"  style="border:  3px  solid;
color: #DB222A;">
      <br><br>
                                                                                <font
color="orange">NARANJA:</font> Barra principal de navegación. Contiene:<br>
      o
                                                                                Págin
a Principal, la que se vé actualmente.<br>
      o
                                                                                Instr
ucciones, página actual.<br>
      o
                                                                                Edito
res, se explicarán a continuación.<br>
      o
                                                                                Págin
a de referencia con links a los fabricantes y recursos<br><br>
                                                                                <font
color=#00ff00>VERDE</font>: cuerpo principal de la página y donde se irán mostrando el
contenido relacionado con las pestañas seleccionadas.<br><br>
                                                                                <font
color="purple">MORADO</font>: editores de características de la pantalla<br><br>

    </div>
    <div id="CampoEntrada">
      <p>Personalización de pantalla:</p>
      <img  alt="Pagina  principal"  src="../fs/Instr2.png"  style="border:  3px  solid;
color: #DB222A;">
      <br><br>
                                                                                <font
color="purple">Selector de Página a modificar</font>: se incrementa el número con los
botones del campo<br><br>
                                                                                <font
color="black">Selector fondo/elemento</font><br><br>
                                                                                <font
color="red">Teclado  para  ROJO</font>,  se  introduce  el  número  deseado  en  el  orden
centenas, decenas y unidades<br><br>
                                                                                <font
color=#00ff00>Teclado para VERDE</font>, igual a rojo<br><br>
                                                                                <font
color="blue">Teclado para AZUL</font>, igual a rojo<br>

    </div>
    <div id="CampoEntrada">
      <p>Editor de comando de botones:</p>
      <img  alt="Pagina  principal"  src="../fs/Instr3.png"  style="border:  3px  solid;
color: #DB222A;">
  <br><br>
```

```
  □                                                                                <font
color="purple">Selector de Página a modificar</font>: se incrementa el número con los
botones del campo.<br><br>
  □                                                                                <font
color="black">Selector de tecla</font>: se incrementa y decrementa con los botones
asignados al campo.<br><br>
  □                                                                                <font
color="red">Selector de modificador para tecla</font>: se seleccionan los modificadores
con las teclas de nombres correspondientes<br><br>
  □                                                                                <font
color=#00ff00>Selector de Tecla</font>: se selecciona la tecla principal deseada para
el comando custom pinchando sobre la tecla del teclado virtual deseada<br><br>
  □                                                                                <font
color="blue">Actualización</font>: pulsar este botón actualiza los valores del botón
seleccionado en el dispositivo<br>

    </div>
  </body>
  </html>
```

principal.htm

```
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>
  <html>
  <head>
  <title>EK-TM4C1294XL Evaluation Kit</title>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
  <link rel="stylesheet" type="text/css" href="styles.css"/>
  <link rel="shortcut icon" type="image/x-icon" href="favicon.ico"/>
  </head>
  <body>
  <p id="content_heading">Trabajo fin de grado: Demostración de tecnología a traves de
proyecto</p>
  <hr>
  <br>
  <div id="content_body">Esta página es la implementación de un servidor web local dentro
de la TTM4C1294XL. Se usará con el fin de darle
  una posibilidad de personalización al usuario, permitiendole modificar los comandos
almacenados en las teclas virtuales. También se añade
  la posibilidad de personalización gráfica de las pantallas, pudiendole cambiar los
colores de ésta.</div>


  </body>
  </html>
  editorBot.htm
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013 Texas Instruments Incorporated. All rights reserved. -->
  <html>

  <head>
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>I/O Control Demo 1</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
    <link rel="shortcut icon" type="image/x-icon" href="favicon.ico" />
  </head>

  <body>
    <p id="content_heading">Editor de Botones</p>
    <hr>
    <br>
    <div id="content_body">Esta página se utiliza para modificar las combinaciones de
teclas asignadas a las teclas virtuales de la pantalla táctil
```

```
      <div id="CampoEntrada">
        <p>Seleccione la pantalla de los botones que desea modificar:<a
id="numPagState">-</a></p>
        <input id="pag_minus" value="<--" onclick="edNumPag('minus');" type="button">
        <input id="pag_plus" value="-->" onclick="edNumPag('plus');" type="button">
        <p></p>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione la tecla a modificar:<a id="NumBotState">-</a></p>
        <input id="bot_minus" value="-" onclick="edNumBot('minus');" type="button">
        <input id="bot_plus" value="+" onclick="edNumBot('plus');" type="button">
        <p></p>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione el modificador de tecla:<a id="ModTecState">-</a></p>
        <input id="modCtrl" value="CTRL" onclick="edModTec('Ctrl');" type="button">
        <input id="modAlt" value="ALT" onclick="edModTec('Alt');" type="button">
        <input id="modShift" value="SHIFT" onclick="edModTec('Shift');" type="button">
        <input id="modNone" value="NONE" onclick="edModTec('None');" type="button">
        <p></p>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione la tecla principal:<a id="TecState">-</a></p>
        <div id=DivTecNum>
          <input id="Tec4" value="1" onclick="edTec('1');" type="button">
          <input id="Tec4" value="2" onclick="edTec('2');" type="button">
          <input id="Tec4" value="3" onclick="edTec('3');" type="button">
          <input id="Tec4" value="4" onclick="edTec('4');" type="button">
          <input id="Tec4" value="5" onclick="edTec('5');" type="button">
          <input id="Tec4" value="6" onclick="edTec('6');" type="button">
          <input id="Tec4" value="7" onclick="edTec('7');" type="button">
          <input id="Tec4" value="8" onclick="edTec('8');" type="button">
          <input id="Tec4" value="9" onclick="edTec('9');" type="button">
          <input id="Tec4" value="0" onclick="edTec('0');" type="button">
        </div>
        <div id=DivTec1>
          <input id="Tec1" value="Q" onclick="edTec('Q');" type="button">
          <input id="Tec1" value="W" onclick="edTec('W');" type="button">
          <input id="Tec1" value="E" onclick="edTec('E');" type="button">
          <input id="Tec1" value="R" onclick="edTec('R');" type="button">
          <input id="Tec1" value="T" onclick="edTec('T');" type="button">
          <input id="Tec1" value="Y" onclick="edTec('Y');" type="button">
          <input id="Tec1" value="U" onclick="edTec('U');" type="button">
          <input id="Tec1" value="I" onclick="edTec('I');" type="button">
          <input id="Tec1" value="O" onclick="edTec('O');" type="button">
          <input id="Tec1" value="P" onclick="edTec('P');" type="button">
        </div>
        <div id=DivTec2>
          <input id="Tec2" value="A" onclick="edTec('A');" type="button">
          <input id="Tec2" value="S" onclick="edTec('S');" type="button">
          <input id="Tec2" value="D" onclick="edTec('D');" type="button">
          <input id="Tec2" value="F" onclick="edTec('F');" type="button">
          <input id="Tec2" value="G" onclick="edTec('G');" type="button">
          <input id="Tec2" value="H" onclick="edTec('H');" type="button">
          <input id="Tec2" value="J" onclick="edTec('J');" type="button">
          <input id="Tec2" value="K" onclick="edTec('K');" type="button">
          <input id="Tec2" value="L" onclick="edTec('L');" type="button">
        </div>
        <div id=DivTec3>
          <input id="Tec3" value="Z" onclick="edTec('Z');" type="button">
          <input id="Tec3" value="X" onclick="edTec('X');" type="button">
          <input id="Tec3" value="C" onclick="edTec('C');" type="button">
          <input id="Tec3" value="V" onclick="edTec('V');" type="button">
          <input id="Tec3" value="B" onclick="edTec('B');" type="button">
          <input id="Tec3" value="N" onclick="edTec('N');" type="button">
          <input id="Tec3" value="M" onclick="edTec('M');" type="button">
        </div>
        <p></p>
      </div>
```

```
      <div      id="CampoEntrada"><input      id="SubmitButton"      value="Actualizar"
onclick="SubBut();" type="button"></div>

    </div>
  </body>

  </html>
```

editorPag.htm

```
  <!DOCTYPE HTML>
  <!-- Copyright (c) 2013 Texas Instruments Incorporated. All rights reserved. -->
  <html>

  <head>
    <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
    <title>I/O Control Demo 1</title>
    <link rel="stylesheet" type="text/css" href="styles.css" />
    <link rel="shortcut icon" type="image/x-icon" href="favicon.ico" />
  </head>

  <body>
    <p id="content_heading">Editor de Páginas</p>
    <hr>
    <br>
    <div id="content_body">Esta página se utiliza para diseñar y personalizar las
pantallas de trabajo
      <div id="CampoEntrada">
        <p>Seleccione la pantalla que desea modificar:<a id="numPagState">-</a></p>
        <input     id="pag_minus_pg"     value="<--"     onclick="edNumPagP('minus');"
type="button">
        <input id="pag_plus_pg" value="-->" onclick="edNumPagP('plus');" type="button">
        <p></p>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione si desea trabajar sobre el fondo o los botones:<a
id="WindButState">-</a></p>
        <input    id="Windows/Button"    value="Fondo/Botones"    onclick="WindBut();"
type="button">
        <p></p>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione la tonalidad de Rojo:</p><a id="RedState">-</a>
        <div id=DivTecNum>
          <input id="Tec4" value="7" onclick="edRed('7');" type="button">
          <input id="Tec4" value="8" onclick="edRed('8');" type="button">
          <input id="Tec4" value="9" onclick="edRed('9');" type="button">
          <div></div>
          <input id="Tec4" value="4" onclick="edRed('4');" type="button">
          <input id="Tec4" value="5" onclick="edRed('5');" type="button">
          <input id="Tec4" value="6" onclick="edRed('6');" type="button">
          <div></div>
          <input id="Tec4" value="1" onclick="edRed('1');" type="button">
          <input id="Tec4" value="2" onclick="edRed('2');" type="button">
          <input id="Tec4" value="3" onclick="edRed('3');" type="button">
          <div></div>
          <input id="Tec4" value="0" onclick="edRed('0');" type="button">
        </div>
      </div>
      <div id="CampoEntrada">
        <p>Seleccione la tonalidad de Verde:</p><a id="GrnState">-</a>
        <div id=DivTecNum>
          <input id="Tec4" value="7" onclick="edGrn('7');" type="button">
          <input id="Tec4" value="8" onclick="edGrn('8');" type="button">
          <input id="Tec4" value="9" onclick="edGrn('9');" type="button">
```

```
        <div></div>
        <input id="Tec4" value="4" onclick="edGrn('4');" type="button">
        <input id="Tec4" value="5" onclick="edGrn('5');" type="button">
        <input id="Tec4" value="6" onclick="edGrn('6');" type="button">
        <div></div>
        <input id="Tec4" value="1" onclick="edGrn('1');" type="button">
        <input id="Tec4" value="2" onclick="edGrn('2');" type="button">
        <input id="Tec4" value="3" onclick="edGrn('3');" type="button">
        <div></div>
        <input id="Tec4" value="0" onclick="edGrn('0');" type="button">
      </div>
    </div>
    <div id="CampoEntrada">
      <p>Seleccione la tonalidad de Azul:</p><a id="BluState">-</a>
      <div id=DivTecNum>
        <input id="Tec4" value="7" onclick="edBlu('7');" type="button">
        <input id="Tec4" value="8" onclick="edBlu('8');" type="button">
        <input id="Tec4" value="9" onclick="edBlu('9');" type="button">
        <div></div>
        <input id="Tec4" value="4" onclick="edBlu('4');" type="button">
        <input id="Tec4" value="5" onclick="edBlu('5');" type="button">
        <input id="Tec4" value="6" onclick="edBlu('6');" type="button">
        <div></div>
        <input id="Tec4" value="1" onclick="edBlu('1');" type="button">
        <input id="Tec4" value="2" onclick="edBlu('2');" type="button">
        <input id="Tec4" value="3" onclick="edBlu('3');" type="button">
        <div></div>
        <input id="Tec4" value="0" onclick="edBlu('0');" type="button">
      </div>
    </div>
    <p></p>



  </body>

  </html>
```

javascript.js

```
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>

  window.onload = function()
  {
    document.getElementById('principal').onclick = loadPrincipal;
    document.getElementById('instrucciones').onclick = loadInstrucciones;
    document.getElementById('info').onclick = loadInfo;
    document.getElementById('editorBot').onclick = loadEditorBot;
    document.getElementById('editorPag').onclick = loadEditorPag;
  }

  function loadPrincipal()
  {
    loadPage("principal.htm");
    return false;
  }

  function loadInstrucciones()
  {
    loadPage("instrucciones.htm");
    return false;
  }

  function loadInfo()
  {
```

```
  loadPage("info.htm");
  return false;
}

function loadEditorBot()
{
  loadPage("editorBot.htm");
  totalGet("numPagState");
  totalGet("NumBotState");
  totalGet("ModTecState");
  totalGet("TecState");
  return false;
}

function loadEditorPag()
{
  loadPage("editorPag.htm");
  totalGet("numPagState");
  totalGet("RedState");
  totalGet("GrnState");
  totalGet("BluState");
  return false;
}

function SetFormDefaults()
{
  document.iocontrol.LEDOn.checked = ls;
  document.iocontrol.CTRLon.checked = cs;
  document.iocontrol.speed_percent.value = sp;
}

function edNumPag(a)
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("numPagState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/pag_' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
}

function edNumPagP(a)
{
  var req = false;
```

```
  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("numPagState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/pag_' + a + '_pg?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
}

function edNumBot(a)
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("NumBotState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Bot_' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function edModTec(a)
{
  var req = false;

  function ToggleComplete()
  {
```

```
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("ModTecState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/ModTec' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function edTec(a)
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("TecState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Tec' + a + '?id';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
}

function edRed(a)
{
  var req = false;

  function ToggleComplete()
  {
```

```
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("RedState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Red' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function edGrn(a)
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("GrnState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Grn' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function WindBut()
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
```

```
      if(req.status == 200)
      {
        document.getElementById("WindButState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  if(req)
  {
    req.open("GET", "/WindBut" + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function edBlu(a)
{
  var req = false;

  function ToggleComplete()
  {
    if(req.readyState == 4)
    {
      if(req.status == 200)
      {
        document.getElementById("BluState").innerHTML = "<div>" +
        req.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }
  var aux = '/Blu' + a + '?';
  if(req)
  {
    req.open("GET", aux + Math.random(), true);
    req.onreadystatechange = ToggleComplete;
    req.send(null);
  }

}

function SubBut()
{
  var req = false;

  if(window.XMLHttpRequest)
  {
    req = new XMLHttpRequest();
  }
```

```
  else if(window.ActiveXObject)
  {
    req = new ActiveXObject("Microsoft.XMLHTTP");
  }

  if(req)
  {
    req.open("GET", "/SubmitButton" + Math.random(), true);
    //req.onreadystatechange = ToggleComplete;
    req.send(null);
  }
}

function totalGet(a)
{
  var led = false;
  var aux = '/' + a;
  function ledComplete()
  {
    if(led.readyState == 4)
    {
      if(led.status == 200)
      {
        document.getElementById(a).innerHTML = "<div>" +
        led.responseText + "</div>";
      }
    }
  }

  if(window.XMLHttpRequest)
  {
    led = new XMLHttpRequest();
  }
  else if(window.ActiveXObject)
  {
    led = new ActiveXObject("Microsoft.XMLHTTP");
  }
  if(led)
  {
    led.open("GET", aux + Math.random(), true);
    led.onreadystatechange = ledComplete;
    led.send(null);
  }
}

function loadPage(page)
{
  if(window.XMLHttpRequest)
  {
    xmlhttp = new XMLHttpRequest();
  }
  else
  {
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }

  xmlhttp.open("GET", page, true);
  xmlhttp.setRequestHeader("Content-type",
  "application/x-www-form-urlencoded");
  xmlhttp.send();

  xmlhttp.onreadystatechange = function ()
  {
    if((xmlhttp.readyState == 4) && (xmlhttp.status == 200))
    {
      document.getElementById("content").innerHTML = xmlhttp.responseText;
    }
  }
}
```

javascript_load.js

```
  <!-- Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. --
>

  window.onload = function()
  {
      document.getElementById('principal').onclick = loadPrincipal;
      document.getElementById('instrucciones').onclick = loadInstrucciones;
      document.getElementById('info').onclick = loadInfo;
      document.getElementById('editorBot').onclick = loadEditorBot;
      document.getElementById('editorPag').onclick = loadEditorPag;

      loadPage("principal.htm");
  }
```

style.ccs

```
  /* Copyright (c) 2013-2020 Texas Instruments Incorporated.  All rights reserved. */

  /* Styling for the entire document. */
  body
  {
      background-color: #BFDBF7;
      color: #021927;
      font-family: Ariel,Helvetica,sans-serif;
  }

  /* Styling for the heading div. */
  #heading
  {
      clear: both;
      margin-bottom: 10px;
  }

  #heading_h1
  {
      font-size: 2em;
      font-weight: bold;
      text-align: center;
      width: 100%;
  }

  #heading_h2
  {
      font-size: 1.25em;
      font-weight: bold;
      text-align: center;
      width: 100%;
  }

  #heading hr
  {
      border-bottom: 1px solid #DB222A;
      border-top: 1px solid #DB222A;
  }

  #heading a img
  {
      border: 0;
      width: 20em;
  }


  /* Styling for the menu div. */
```

```
#menu ul
{
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #021927;
    border-radius: 35px 0px 35px 0px;
    -moz-border-radius: 35px 0px 35px 0px;
    -webkit-border-radius: 35px 0px 35px 0px;
    border: 2px solid #1F7A8C;
}

#menu li
{
    float: left;
}

#menu a
{
    background-color: #021927;
    color: #BFDBF7;
    display: block;
    font-weight: bold;
    text-align: center;
    text-decoration: none;
    width: 200px;
    padding:20px;

}

#menu a:hover
{
    background-color: #A31621;
}

#menu img{
    border: 0;
    width: 10em;
}

/* Styling for the content div. */
#content
{
    position: relative;
    margin:0;
    padding:0;
    min-height: 100%;
}
#CampoEntrada p
{
    position: relative;
    color:#021927;
    margin:0;
    padding:0;
    min-height: 100%;
}

#DivTecNum
{
    position: relative;
    margin: 0;
    border-radius: 8px;
    padding: 4px 4px;
    min-height: 100%;

}
#DivTec1
{
    position: relative;
```

```
    margin-left:7px;
    border-radius: 8px;
    padding: 4px 4px;
    min-height: 100%;
}
#DivTec2
{
    position: relative;
    margin-left:16px;
    border-radius: 8px;
    padding: 4px 4px;
    min-height: 100%;
}
#DivTec3
{
    position: relative;
    margin-left:27px;
    border-radius: 8px;
    padding: 4px 4px;
    min-height: 100%;
}

#CampoEntrada
{
  font-size: 1em;
  font-weight: bold;
  color: #BFDBF7;
  background: #1F7A8C;
  margin: auto;
  overflow: hidden;
  padding: 5px;
  border: 4px solid #BFDBF7;
}

#TecState, #ModTecState, #NumBotState
{
    position: relative;
    font-weight: bold;
    text-align: center;
    min-height: 100%;
    color: #A31621;
}
#numPagState
{
    position: relative;
    font-weight: bold;
    text-align: center;
    min-height: 100%;
    color: #BFDBF7;
}

#RedState
{
    position: relative;
    font-weight: bold;
    text-align: center;
    min-height: 100%;
    color: #DB222A;
}
#GrnState
{
    position: relative;
    font-weight: bold;
    text-align: center;
    min-height: 100%;
    color: #00FF00;
}
```

```
#BluState
{
    position: relative;
    font-weight: bold;
    text-align: center;
    min-height: 100%;
    color: #0219FF;
}

#content_heading
{
    font-size: 1.5em;
    font-weight: bold;
    text-align: center;
    color: #ffffff;
    background: #A31621;
    margin: 0 0 25px;
    overflow: hidden;
    padding: 20px;
    border-radius: 0px 35px 0px 35px;
    -moz-border-radius: 0px 35px 0px 35px;
    -webkit-border-radius: 0px 35px 0px 35px;
    border: 2px solid #1F7A8C;
}

/* Styling for the footing div. */
#footing
{
    clear: both;
    font-size: .75em;
    font-style: italic;
    text-align: right;
    position: relative;
    bottom: 0;
    left: 0;
    padding: 0;
    width: 100%;
    height: 2.5rem;            /* Footer height */
}

#footing hr
{
    border-bottom: 1px solid #ff0000;
    border-top: 1px solid #ff0000;
}
```