Proyecto Fin de Carrera
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Sistema de bajo coste para la monitorización del consumo eléctrico en instalaciones eléctricas monofásicas.

Autor: Elías Marín Domínguez
Tutor: Juan Carlos del Pino López.

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

Sevilla, 2019

Proyecto Fin de Carrera
Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

Sistema de bajo coste para la monitorización del consumo eléctrico en instalaciones eléctricas monofásicas.

Autor:
Elías Marín Domínguez

Tutor:
Juan Carlos del Pino López
Profesor titular

Dpto. de Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Sistema de bajo coste para la monitorización del consumo eléctrico en instalaciones eléctricas monofásicas.

Autor: Elías Marín Domínguez

Tutor: Juan Carlos del Pino López

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mi familia

A mis maestros

Agradecimientos

A mis padres, por la confianza puesta en mí y el apoyo en todos estos años.

A mis abuelos, por el amor y cariño.

A mis hermanos.

A Chele, por la convivencia en estos últimos años.

A mi tutor y profesor, Juan Carlos del Pino López, por darme la oportunidad de desarrollar su idea y por enseñar de esa forma tan clara y original.

A tí, por haberme acompañado hasta aquí, en una etapa tan decisiva para mí. Por ser parte de la motivación que me empujaba.

Y, por último, a mí, por ser como soy y haber llegado hasta aquí.

Elías Marín Domínguez

Sevilla, 2019

Resumen

Siguiendo la tendencia actual de conectar todo dispositivo-objeto cotidiano a internet (IOT), se ha desarrollado un sistema que instalado en el cuadro eléctrico de la vivienda-nave, con una serie de sensores y con tecnología wifi, para poder consultar en remoto el estado del consumo de la vivienda. El sistema tiene implementado un algoritmo de ahorro de dinero, que compara el consumo en euros de la compañía contratada frente a la competencia. Además de una serie de extras adicionales, como serían el poder actuar en remoto algunos de los circuitos de la vivienda e incluso análisis del uso de la potencia contratada, sugeriendo subidas o bajadas de potencia. Algunos de estos extras han sido implementados, otros por falta de tiempo están únicamente planteados.

Como se comenta, este sistema, se encarga de llevar un recuento de consumo no solo con la comercializadora con la que se tiene contratada, sino con una serie de comercializadoras que se encuentran en una base de datos propia, con el objetivo de proponer cambios de compañía eléctrica por ahorro en dinero.

También, se encarga de llevar un estudio del uso de la potencia que se tiene contratada, proponiendo también bajadas o subidas de potencia.

De momento el sistema está adaptado únicamente para el mercado libre. Se cuenta con una base de datos de elaboración propia, de las principales compañías eléctricas (de momento solo Fenie Energia y Endesa, pero fácilmente ampliables), con los precios asociados a cada tipo de tarifa.

La APP se podría adaptar muy fácilmente para poder lanzar notificaciones al usuario cada vez que se exceda un consumo límite impuesto, así como notificar, cuando se entre en las horas baratas, de manera que el usuario pueda desviar todo el consumo a estas horas.

Se podría incluso tener no solo un sensor de tensión y de corriente para el consumo en global, sino tener medidas independientes de cada circuito de la instalación. Incluso, si se quiere de la posibilidad de cortar-cerrar estos circuitos en remoto, se podría adaptar muy fácilmente el sistema.

Este aparato de bajo consumo eléctrico consta de una placa de desarrollo junto con dos sensores, que iría al cuadro de luz del cliente.

Además de esto, se tiene un servidor Linux, en concreto una máquina virtual alquilada a la infraestructura de Google y editada personalizada a medida. Este servidor es el que gobierna a cada dispositivo instalado en casa de usuario, se encarga de realizar copia de seguridad de los datos de los sensores, gestiona la app Blynk.

Dentro de este servidor se encuentra la base de datos montada, con tablas para los precios de las distintas comercializadoras y los datos de los sensores de cada sistema.

Esta plataforma de Google Cloud ofrece numerosos servicios basados en la nube, en este caso se está alquilando un “servidor”-máquina virtual, configurada con Debian9 como SO, 25GB HDD, 1Vcpu, 2GB RAM…

El alquiler de la máquina supone un coste de 3 cent/hora.

A modo de resumen, se tiene un servidor, con las características deseadas dentro de la infraestructura de Google.

Una ventaja importante a tener en cuenta al usar este tipo de servicios en la nube, es la estabilidad y fiabilidad

de la máquina. Al hacer uso de la infraestructura de Google.

Una vez se crea la instancia, se suministra la dirección IP externa para poder controlar en remoto a la máquina.

En ella, sobre Debian 9, he montado el stack LAMP, servidor BLYNK (como está basado en Java, también Java 8).

LAMP → Linux: Debian9

Apache

MYSQL 5.6, servidor base de datos

PHP 7

Índice

ÍNDICE DE TABLAS E ILUSTRACIONES

ÍNDICE DE FIGURAS

Notación

BLYNK	Plataforma IOT usada para el sistema
IOT	The internet of things
APP	APPlication
GCP	Google Cloud Platform
LAMP	Linux-Apache-MYSQL-PHP
MCU	Microcontrolador
MPU	Microprocesador
PHP	Hypertext Preprocessor
GET	Método HTML para envio de datos
IDE	Integrated development environment
PCB	Printed Circuit Board
RPI	Raspberry Pi
WIFI	

1 INTRODUCCIÓN

En la actualidad, se aprecia cada vez más fácilmente, la tendencia a conectar todo tipo de dispositivos cotidianos a internet, ya sea buscando comodidad (todo controlado desde el Smartphone) o productividad, con tal de aprovechar todos los datos que nos brinda internet, que no son pocos.

Desde bombillas gestionadas por una APP móvil a asistentes virtuales capaces de atender nuestras preguntas a través de comandos de voz.

¿Por qué no darle más posibilidades a un cuadro de luz de vivienda, con tal de volverlo más "inteligente"?

Y si lo enfocamos para que no solo sea más cómodo y rápido llevar un control del consumo en euros en luz en tiempo real, sino que se preocupe también, de lanzar sugerencias de ahorro.

El esquema de trabajo general del sistema sería el siguiente:

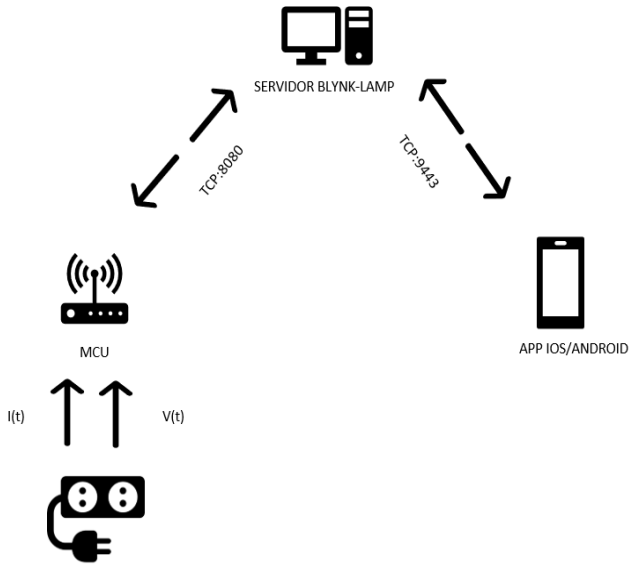


Ilustración 1. Esquema general del sistema

1.1. Motivación y objetivo

Este trabajo de fin de grado simplemente sigue la tendencia actual del IOT, más concretamente siguiendo el término de las Smart Homes, a la par que se presenta una solución a una preocupación tan común como es el ahorro energético, realmente, ahorro de dinero. Otro punto importante que aborda este trabajo, sería el de crear conocimiento con respecto al consumo en euros de un determinado aparato durante un tiempo concreto. Normalmente se manejan parametros del tipo kw, kwh, €/kwh. Cuando lo que realmente se busca es saber cuanto

dinero nos ha supuesto conectar cierto aparato a la luz durante un determinado tiempo. Crear conciencia del coste de consumo de cada aparato en el ámbito domestico es una tarea importante y necesaria.

La idea básica de la que se parte era montar un sistema que recogiera datos de tensión y corriente, pidiera al usuario cierta información acerca del plan de luz contratado y presentara datos de consumo en tiempo real.

Como primeros extras; llevar de alguna manera, control del uso de la potencia contratada e incluso llevar una comparativa de precios entre comercializadoras en función del consumo.

Si además a este sistema se le diera la posibilidad de tomar decisiones, del tipo abro circuito A, cierro circuito B, en función del precio de la luz, lo cual sería fácilmente implementable, se podría extender su uso a industria, donde un pequeño ahorro en luz, supondría una gran diferencia en la factura de luz. De momento el prototipo del sistema es pasivo, es decir solo realiza operaciones de lectura y envío de datos al servidor para consulta. Esta tarea de recogida de datos de sensores y almacenarlos en bases de datos, con el fin de explotar estos datos posteriormente está ampliamente extendido. Por ejemplo, se pueden usar estos datos para detectar comportamientos anómalos que no respeten el patrón acostumbrado y lanzar errores o warnings según toque. De importancia, para el diagnóstico del sistema en remoto.

Por otro lado, el disponer de una APP, muy sencilla de usar y con posibilidad de realimentación al usuario con notificaciones en función del estado del consumo en la casa es muy atractivo. Piénsese, que el usuario desde fuera de casa y en cualquier momento tiene a su disposición el consumo en euros en tiempo real de la vivienda, el consumo en kwh, los picos de potencia que se han registrado y a la hora que se han producido…

El prototipo montado, consiste en una placa de desarrollo Linkit Smart 7688 Duo, a la que se le han unido un par de sensores (en su configuración más simple), el circuito de acondicionamiento respectivo y un alimentador para la placa. Se encarga de llevar a cabo una serie de medidas y tras cierto procesado, puede enviar una serie de datos al usuario vía APP, pasando primeramente por el servidor basado en Linux montado.

Comentar que se cuenta con un servidor LAMP y un servidor BLYNK basado en Java, los cuales corren sobre la misma máquina física. El primero se encarga de recibir datos desde los sistemas instalados en casas de usuarios, además de entregar datos de tarifas de las distintas comercializadoras. El segundo se encarga de gestionar la APP móvil.

1.2. Estado del arte

La automatización o monitorización de actividades en remoto, está experimentando un crecimiento notable, con la aparición de nuevas e incontables placas de desarrollo y plataformas IOT. Actualmente la mayor parte de dispositivos conectados, que quieren poder disfrutar de todos los datos que brinda internet, usan WIFI, el principal problema que se presenta es el elevado consumo que tienen. Se están investigando distintas alternativas que necesiten menos energía.

La idea tuvo su aparición gracias a Kevin Ashton en el AutoId Center del MIT en 1999, donde se andaba trabajando con técnicas de sensores y RFID.

La flexibilidad que da una placa de desarrollo ya sea basada en microprocesador o microcontrolador, incluso actualmente muy común el uso de placas con MCU y MPU mixtas, hacen que el ámbito de aplicación sea enorme y por lo tanto el esquema de trabajo también. Ya sea una placa extrayendo datos de internet , una placa que recoge datos de sensores y los procesa y los hace accesibles desde cualquier parte de la red, o incluso realizando ambas operaciones como en el caso que nos ocupa.

Si a esto se le unen tendencias actuales del tipo machine learning, para el procesado de datos de una base de datos que proviene de un dispositivo conectado con el fin de predecir comportamientos y tendencias, su ámbito de aplicación crece aún más.

1.3. Resumen del hardware usado

El sistema global se compone:

- Parte destinada a las lecturas con procesado intermedio:

- Placa de desarrollo **Linkit Smart 7688 Duo**, cuenta con microcontrolador ATMEGA32U4 (se puede programar desde el IDE de Arduino), cuenta también con una MPU Mediatek mt7688, corriendo OpenWrt.

  Esta placa se encarga de la recogida de los valores de corriente y de voltaje de la red, entre muestra y muestra hace cálculos intermedios para que el resultado posteriormente sea más rápido de obtener. Una vez realiza la lectura completa de un periodo de red, calcula valores eficaces, así como potencia activa, reactiva y cos de phi y se lo transmite al servidor, para que el mismo gestione la APP con esos nuevos datos. Además de realizar una copia de seguridad sobre la base de datos alojada en el servidor.

- Sensor de corriente no invasivo de la serie **SCT013**, en concreto he trabajado con el SCT013-30, el cual trabaja hasta con 30 A en ac.
- **Transformador ac/ac**
- **Circuito acondicionamiento**, para adaptar el rango de medidas tanto del sct como del trafo a un rango admisible por la placa. Para el transformador además de ajustar el rango completo a los 0-3.3V de la placa, se añade un offset en dc para que la señal alterna, se mueva siempre entre valores positivos.

  Para el sensor de corriente simplemente añadir el punto medio, para que los valores de tensión sean siempre positivos.

- Lado del servidor que apoya a todo el hardware:

- **Raspberry pi 3 Model B**, corriendo Raspbian Jessie (distribución Linux). La misma se encarga del procesado de datos provenientes de la placa y de la gestión de la APP, además de la gestión de la base de datos propia.

Este servidor puede ser común para varias de estas placas, si por ejemplo se tuvieran varios proyectos, todos ellos colgarían del mismo servidor.

Como ya se ha mencionado, finalmente a modo de servidor se ha migrado desde la Raspberry a una máquina virtual dentro de un servidor perteneciente a Google con Debian 9 como sistema operativo.

2 HARDWARE

LA idea que se tenía desde el primer momento para abordar el trabajo era dividir el sistema en 3 módulos. Por un lado, una placa basada en microcontrolador que se encargara de las lecturas, una placa basada en microprocesador que se encargara de realizar procesado y envio de datos al servidor, y el mismo servidor para el control de la base de datos y la gestión de la APP.

Primeramente, se comenzaron las pruebas con la "famosa" placa esp8266, placa ampliamente usada para el IoT, con un modesto precio y unas características bien sobradas. También para las primeras pruebas se hicieron tests con un Arduino Uno.

Del esp, existía la desventaja de la poca documentación relativa al uso de su ADC interno. Tampoco fue fácil encontrar información relativa a sus TIMERs internos. Como lo que se buscaba era poder asociar un TIMER con el ADC, para tener control sobre cuando se realizaba la lectura de voltaje e intensidad, se requería programar estos periféricos a nivel de registros.

Con el arduino Uno, y su atmega328p, se consiguió programar estos dos periféricos y hacer que el tiempo de muestreo de las lecturas estuviera acordado.

Posteriormente, se optó por usar la placa de desarrollo linkit Smart 7688 duo, la cual integra, como se comentó anteriormente un MCU y un MPU, comunicados por puerto serie entre sí. Sería equivalente al Arduino Yun.

Por unos 15€, se puede obtener esta placa. Las ventajas que aporta son las siguientes:

- El MCU se puede programar usando el IDE de Arduino.
- El MPU corre un SO, OpenWrt, que brinda muchas oportunidades complementando al MCU anterior.
- Cuenta con slot SD, para ampliar la memoria interna.
- Cuenta con comunicación WIFI-ethernet.

Entonces, la MCU programada desde el entorno de arduino, se encarga de la lectura de los sensores de manera periódica, así como de enviar los datos finales a la raspberry, para que los dirija a la APP.

La MPU de la misma placa, apoya a la MCU, con su chip WIFI.

Anteriormente, se quiso cargar toda la parte de gestión de la APP dentro del MPU de la placa Linkit, pero tras varios intentos de compilación cruzada fallidos, todo debido a falta de librerías de ciertas plataformas IOT, para este SO(OpenWrt), se optó por trasladar este programa escrito en C a la RPI.

Se darán más detalles, en el siguiente punto dedicado a software.

Para el prototipo se monta una caja estanca con carril DIN con los siguientes componentes; entrada de

tensión con toma de enchufe macho, magnetotérmico 16A Schneider, toma de enchufe hembra para la carga a analizar, portafusibles 1A para proteger fuente de alimentación de la placa y por lo tanto la placa, también protege a transformador 230/12V y por último, sensor de corriente en una de las fases de la toma hembra para la carga a analizar.

Figura 1.Prototipo I

Figura 2.Prototipo II

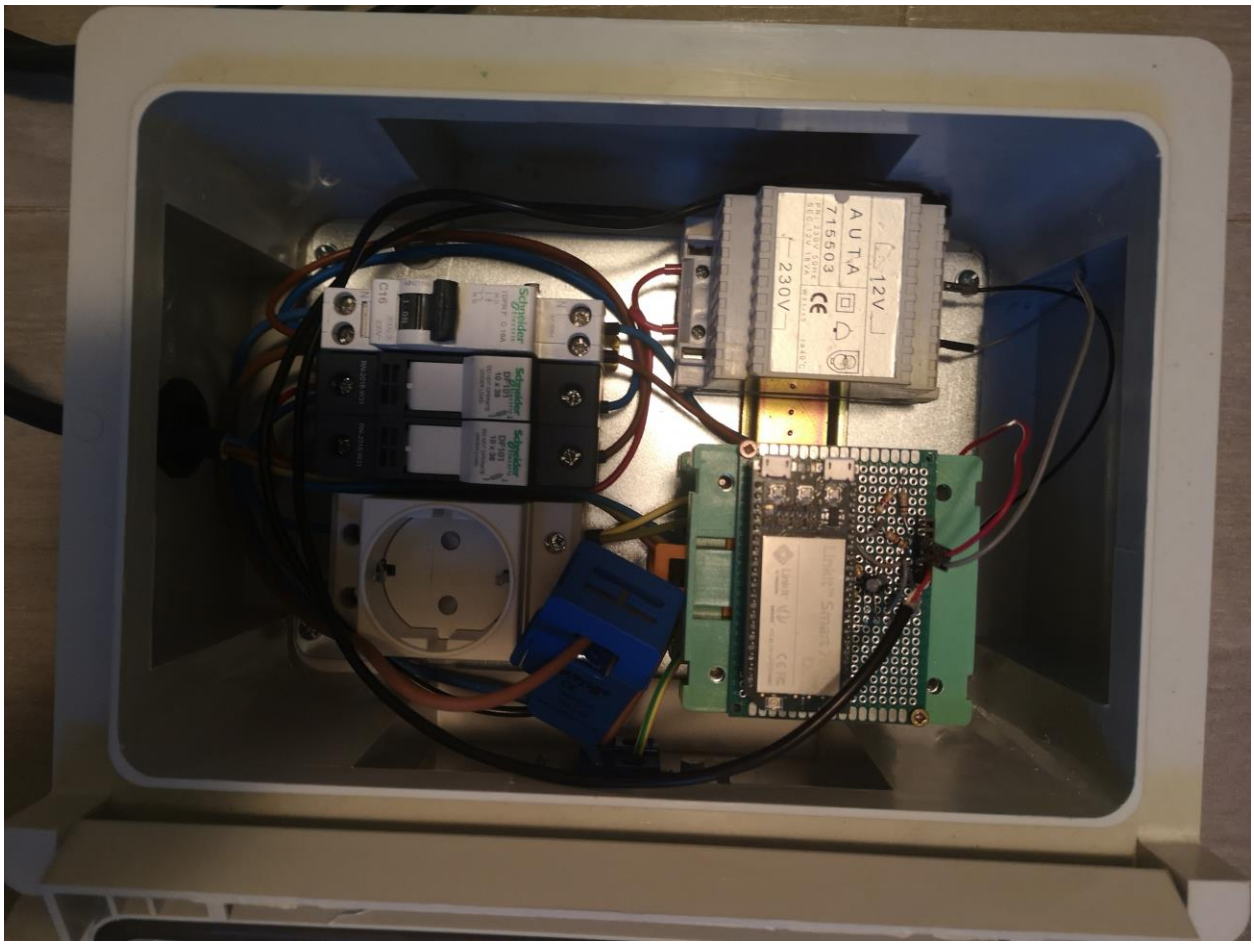


Figura 3.Prototipo III

2.1. Linkit Smart 7688 Duo

Figura 4. Placa de desarrollo Linkit Smart 7688 Duo

Como ya se adelantó anteriormente esta placa de desarrollo cuenta con una MPU apoyada por una MCU. Es una placa diseñada para el IoT. En concreto cuenta con un ATmega32U4, como MCU, el cual puede ser programado usando el ide de Arduino. La MPU, es una Mediatek MT7688AN corriendo distribución Linux, en concreto OpenWrt. Posee conectividad WIFI, numerosos periféricos internos (asociados a la MCU).

Problemas detectados durante su uso:

- De fábrica sin SO, lo cual no suele ser habitual. Se procede a descargar la imagen desde la página del fabricante y cargarla sobre la placa haciendo uso de memoria usb formateada en un formato en específico(FAT32).
- Pasar de modo estación a punto de acceso para reconfigurar la red WIFI a la que se conecta puede llegar a resultar un poco tedioso.
- Memoria de programa de ATMEGA32U4 se queda corta muy fácilmente si se hace uso extensivo de librerias.

Especificaciones técnicas y otras características:

*580MHz (MPU) 8MHZ(MCU)

*Voltaje de operación: 3.3V

*ADC →10bits(MCU), 12 canales

*Wi-Fi 802.11 b/g/n(2.4G)

*GPIO,I2C,I2S,SPI,UART,PWM, ETHERNET PORT

*32MB flash, 128MB DDR2 RAM (MPU)

*32KB flash,1KB EEPROM

*Soporta USB host y tarjetas SD, para expandir la memoria interna.

*Tamaño de la placa 60.8*26.0mm

En resumen, placa muy completa por un precio bastante modesto.

Link para la compra de la placa, [1].

2.2. Raspberry Pi3 ModelB

Figura 5.Raspberry Pi3 Model b

Este tipo de placas, bien conocidas, se engloban dentro de lo que se conoce como ordenadores de placa reducida.

Se trata de un mini pc, con unas especificaciones limitadas, si se compara con un pc doméstico, pero sobrado en potencia para las tareas a las que suele estar destinado (como servidor web, manejo de base de datos, domótica …).

En concreto para el desarrollo de este sistema, se ha usado la Pi3 Model B. Comentar que aun más reciente, la Pi3 Model B+.

La Pi3 cuenta con una CPU quad core, 1GB de RAM, chip bluetooth-WIFI, puerto ethernet, 40 pines GPIO, salida HDMI… Todo esto por un precio de unos 35€.

Estas placas se comercializan sin SO, el primer paso que hay que hacer con ellas es bajar un sistema operativo compatible, como el propietario de la fundación → Raspbian, basado en debian. Este SO debe ser montado sobre una SD apta.

Problemas detectados durante su uso:

- Se optó por una SD de 8gb que rápidamente se quedó sin memoria y empezó a funcionar de manera anómala, por ejemplo el servidor BLYNK no llegaba siquiera a dispararse al arrancar la raspberry por falta (necesita almacenar en la placa cierta información acerca de la nueva conexión entrante).
- La SD que se estaba usando era de clase 4, como el SO se carga sobre la sd lo ideal es invertir en una buena SD desde primera hora, de manera que el conjunto se comporte de mejor forma.
- Se presentaron también problemas relacionados no con la placa en si sino con la red local que se saturaba por el tráfico entrante y saliente hacia el exterior. El problema venía del plan de internet contratado que tenía una pésima velocidad de subida.

Pasando a las especificaciones técnicas más en detalle:

*Chip BCM2837: quad core cluster ARM Cortex A53

*1GB de RAM

*Chip BCM43438 LAN & BLE

*40GPIO

*4 USB

*HDMI

*MicroSD slot

Link, para la compra de la placa, [2].

2.3. Sensor SCT013

Figura 6.Sensor de Corriente SCT013

Sensor de corriente no invasivo de tipo pinza amperimétrica, para la medida de la corriente. Su principio de funcionamiento se basa en un transformador de corriente, obteniéndose una medida de corriente proporcional a la intensidad que atraviesa el circuito. Dependiendo del tipo de modelo que escojamos, podemos obtener a la salida del mismo una medida de tensión o intensidad, lo preferible a no ser que se quiera adaptar el rango de medida para ajustar-afinar la precisión del mismo es que la salida sea en tensión.

La relación de transformación que rige a este sensor de corriente:

$$\frac{I_s}{I_p} = \frac{Vp}{Vs} = \frac{Np}{Ns}$$

Donde el número de espiras del devanado primario suele ser simplemente el conductor, jugando con el número de espiras del secundario podemos conseguir que la intensidad en el secundario decrezca más o menos.

La precisión del sensor suele ser de un 1-2%. La única desventaja que se le puede achacar a este tipo de sensores, es que al ser una carga inductiva introduce un desfase que puede llegar a los 3º.

Todos los modelos tienen un precio similar, en vendedores internacionales suele rondar los 5€.

En este proyecto se ha usado el sct013-030, el cual produce una salida de hasta 1vrms para un máximo de corriente de 30Arms. Por lo tanto, con el modelo actual de sensor de corriente, se puede llegar a monitorizar una vivienda de hasta aproximadamente 7kw de potencia contratada.

Link para la compra, del modelo usado (SCT013-030), en Amazon, producto prime, [3].

2.4. Transformador AC/AC

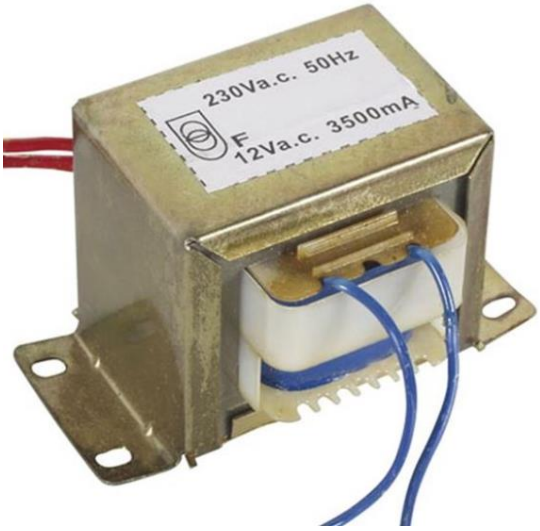


Figura 7.Transformador AC/AC

Para medir la tensión de red, se han hecho pruebas con dos tipos de transformadores de tensión. En el caso de España, la tensión de red 230Vac a 50 Hz. De nuevo aparece el problema del desfase que introducen los transformadores.

Se han usado para las pruebas un transformador 230VAC/12VAC para montar en carril DIN y un mini transformador de 230VAC/3VAC.

Como se sabe, un transformador consta de dos devanados, arrollados en torno a un núcleo magnético. Si hacemos circular por el devanado primario corriente alterna, se genera e induce un flujo magnético en el núcleo, que, al atravesar el devanado secundario, genera una tensión inducida.

Como se indica en el punto anterior, sensor de corriente sct, se juega con el número de devanados en el primario y secundario, para conseguir una relación de transformación determinada.

En general, aunque un transformador tiene pérdidas, son máquinas de elevado rendimiento, en torno al 95%.

Para un dispositivo final lo ideal sería sustituir el transformador tipo carril DIN, por un mini transformador del tipo:

Figura 8.Mini Transformador AC/AC PCB

2.5. Circuito de acondicionamiento para sensores

Para el sensor de corriente, el cual aporta una salida de tensión de 1Vac, lo único que se ha incluido es un offset DC, de 1.5V para que la tensión alterna siempre este entre valores positivos.

En concreto tras añadir este offset, la medida de tensión del sensor se mueve en este rango [0.09V,2.91V].

Incluir circuito con resistencias y condensador para los 1.6V de offset DC.

Esquema del circuito usado para conseguir este offset en DC:

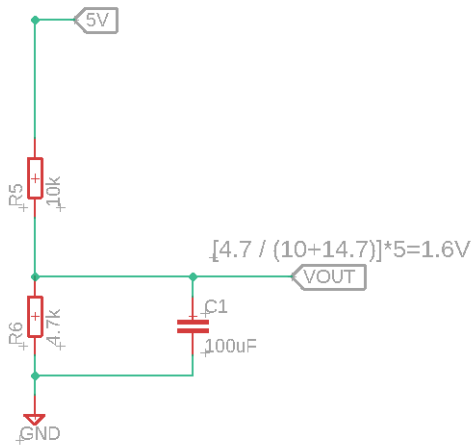


Figura 9.Circuito OFFSET DC

Para el transformador, aparte de incluir el mismo offset dc para que las medidas siempre sean positivas, se ha incluido un divisor resistivo para bajar de los 12VAC (10.2VAC realmente, según voltímetro) a 0.65VAC.

A continuación, se presenta el circuito usado a modo de divisor resistivo, para hacer la conversión de voltaje desde la salida del transformador y obtener unos valores admisibles para la placa que realiza el sensado.

Donde:

- AC1 representa uno de los dos polos de la señal alterna de salida del transformador
- AC2 el segundo polo de la señal alterna, este nodo a su vez se conecta al punto que añade el offset en DC al circuito
- Vout representa el nodo de salida para el voltaje reducido desde el transformador
- Relación de transformación usando este circuito → (1/15.7)*Vin

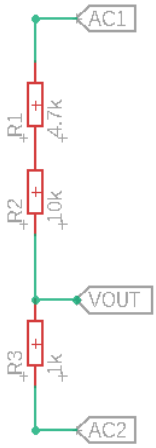


Figura 10.Circuito divisor resistivo

2.6. Lista de precios para el prototipo

A continuación se presenta una tabla excel a modo de "BOM", término usado para hacer referencia a la lista de costes de un prototipo-producto:

Como se puede apreciar en la tabla el componente que más encarece el total de la lista, es la placa RPI 3b, con un precio de entorno a los 35 euros. Aunque ha de tenerse en cuenta que cada Raspberry gobernaría no solo a una placa de sensado sino a varias de ellas. Si el número de prototipos instalados en casas de usuarios comenzara a crecer en exceso lo ideal sería trasladar el servidor BLYNK a una computadora de mayores recursos.

Entonces si descontamos el precio de la RPI del total(el cual habría que repartir en function del numero de proyectos que cuelguen de ella) se tiene un total de 37.63 euros de coste hardware.

Tabla 1. BOM- Coste componentes que componen el sistema

Item No	Design Ref	Item Description	Supplier	Manufacturer	Model	Mfg Part No	Qty	*Cost/unit	IVA	Total	Instructions / Comments
1	Rpi	Ordenador de placa reducida	Rs Compo	Raspberry	Pi3 B	136-5473	1	35,66 €	No	35,66 €	
2	Máquina Virtual	GCP, máquina virtual alquilada	Google		1vCPU, 2GB RAM, Intel Haswell,		30	1,12 €	No	33,6	
3	Linkit	Placa de desarrollo	Seeed Studio	Seeed Studio - Mediatek	Smart 7688 Duo	*	1	15,50 €	No	15,50 €	
4	SCT013-030	Sensor de corriente	Amazon	Yeeco	30A	*	1	10,99 €	No	10,99 €	
5	Transformador 220/12vac	Transformador	Farnel	Myrra	2*12V	1214589	1	4,49 €	No	4,49 €	
6	R1	10K	Digi Key	Yageo	10k	10KEBK-ND	2	0,09 €	No	0,18 €	
7	R2	4,7K	Digi Key	Yageo	4,7k	*	2	0,09 €	No	0,18 €	
8	R3	1K	Digi Key	Yageo	1k	*	1	0,09 €	No	0,09 €	
9	C1	100uF	Cetronic	*	100uF	*	1	0,20 €	No	0,20 €	
10	Cable	Cable	Amazon	*	*	*	1	1,00 €	No	1,00 €	
11	Protoboard	Protoboard	Amazon	*	*	*	1	5,00 €	No	5,00 €	
total										73,29 €	
total sin Rpi										37,63 €	
precio por mes máquina virtual alquilada a la infraestructura de google										33,6	

3 SOFTWARE

En cuanto a software se tienen de nuevo 3 partes bien diferenciadas, asociadas a cada módulo hardware, los cuales se han descrito en el punto anterior. Por un lado se encuentra el programa arduino que se encarga de las lecturas, de cierto procesado y del envío de datos hacia el servidor usando la plataforma BLYNK, la cual se detellará posteriormente. Por otro lado, el programa escrito en C que se encarga de recibir datos del servidor BLYNK, el cual se encuentra gestionando la APP, y también los que proceden del MCU. Este programa se encarga de llevar un recuento de consumo y mostrar datos en los distintos widgets de la APP. También se tiene el servidor BLYNK, montado en la RPI para el control de la APP, es Open-Source, y está basado en java. Lo único que se ha tendido que hacer con el servidor es configurarlo para adaptarlo a los requerimientos de la APP. Por último se ha montado un servidor LAMP en la RPI, para poder gestionar el acceso a la base de datos propia creada para almacenar información respectiva a las distintas comercializadoras.

Como software usado, ya se ha mencionado que se ha usado el IDE de Arduino para programar la placa de desarrollo Linkit, también Putty, usándolo como cliente SSH, para el control en remoto de la RPI y el MPU de la placa Linkit.

También citar en este punto, el uso del protocolo SCP y el software WinSCP, el cual ha sido de gran ayuda para facilitar el intercambio de archivos entre mi computadora personal y las placas con SO propio.

En cada punto, se incluirán los enlaces necesarios y que han sido de más ayuda para la descarga de software.

3.1. Sistemas Operativos

La Raspberry se comercializa sin el SO instalado. De hecho, se necesita una tarjeta microSD, para poder instalar sobre ella el SO. Esto, aun pareciendo una desventaja, aporta mucha flexibilidad. Se podrían tener distintos SO en distintas SD, y alternar entre ellos tan solo cambiando la tarjeta de memoria.

Aunque el SO oficial para la RPI es Raspbian, existen numerosas alternativas de SO de terceros, en la misma página oficial de Raspberry se incluyen enlaces de descarga de las imágenes tanto del SO oficial como de SO de terceros.

Raspbian, es una distribución de GNU/LINUX, libre y basada en Debian, se desarrolló en específico para esta placa (Raspberry) y con la idea inicial de la enseñanza informática. Esa es la esencia de esta placa y de su SO asociado.

En concreto, la versión de Raspbian que se ha usado es la 8, conocida como Jessie.

Para instalar el SO, simplemente acudiendo a la página web oficial de la fundación RPI, [4] , descargando la imagen, y usando software del tipo Win32 Disk Imager, se puede tener lista la placa para jugar, en menos de 10 minutos.

Por otro lado, para la placa **Linkit Smart 7688 duo**, hay que seguir un proceso similar, pero solo en el caso de que uno se encuentre con una imagen dañada o corrupta de fábrica. Por norma general, el fabricante proporciona la placa con el SO ya instalado. En este se caso se trata de OpenWrt.

Si se acude a la página de descargas de la placa en cuestión: [5]

Se pueden acceder a distintas versiones firmware de la placa. Así como a SDKs para hacer la compilación cruzada desde tu computadora personal, por ejemplo, se usó el SDK para ejecutar código en C en la propia MPU.

Aunque los resultados no fueron los esperados, por problemas de librerías, se tuvo que traspasar este código a la RPI, donde sí se pudo ejecutar con éxito.

En caso de que se encuentre con una placa con imagen corrupta seguir los pasos descritos en este tutorial:

[6]

Para la máquina virtual estamos usando Debian 9 Stretch como imagen del sistema. Para configurar la misma con esta imagen es tan fácil como seleccionar la imagen en el desplegable que aparece al crear la instancia en la página de Google Cloud Platform.

Ventajas que presenta usar la máquina virtual perteneciente a la plataforma de Google en la nube frente a la RPI anteriormente usada para el mismo fin:

- Máquina virtual es escalable y editable en recursos hardware, si por ejemplo necesitaras más memoria persistente o RAM puedes modificarla fácilmente. Incluso la generación de los núcleos de la máquina que alquilas.
- Conexión a red más fiable-estable que la red doméstica a la que conectarías la RPI, de hecho existieron problemas de estabilidad de red y fallos esporádicos del sistema en conjunto.
- Servicios asociados a la plataforma que pueden incluirse fácilmente. BigData, IoT, almacenamiento…

3.2. Conexión a Internet

Simplemente para la RPI conectar cable Ethernet, podría estar fuera de la red local, donde se coloque la placa para el sensado, pero dentro de la red local que se encuentre se deberían abrir los puertos respectivos para el funcionamiento de la APP BLYNK. La misma se puede conectar también via WIFI si se quiere.

Para la placa Linkit, igualmente la conexión puede ser física o sin cable. Para la conexión sin cable, simplemente, tras el primer encendido, la placa arranca en modo punto de acceso, y consultando una página web que sirve en una dirección en concreto puedes configurar la red WIFI a la que quieres que se conecte en modo estación.

3.3. Programa MCU

Para poder programar la placa Linkit (MCU ATMEGA32U4), desde el entorno Arduino simplemente limitarse a seguir estos sencillos pasos:

[7]

Entonces, tras esto la placa se encuentra lista para ser programada.

El programa .ino desarrollado para el MCU de esta placa se encarga de:

- Sensado periódico de tensión y corriente haciendo uso del ADC interno y del TIMER, de manera que en resumen, el ADC salta, cada periodo del TIMER, recogiendo una medida de tensión o corriente, según toque. Una vez el ADC finaliza la conversión (tiene el resultado de la conversión) y hasta que el timer vuelve a saltar, dedicamos este intervalo de tiempo sobrante en hacer cálculos intermedios. De manera que llevamos la suma del cuadrado de los valores instantáneos de corriente y tensión, además de un contador de medidas para finalmente poder hallar el valor RMS.

- Cálculo del cosfi. Mediante detección de cambio de signo de valor de tensión o corriente, se recogen los cruces por cero con su respectivo momento de tiempo. Esto se realiza como parte de procesado intermedio entre que el ADC devuelve dato convertido y el TIMER vuelve a indicar comienzo de siguiente conversión. Se recogen los cambios de signo tanto por flanco de subida como bajada, y posteriormente se contemplan todas las posibles combinaciones para el cálculo del cosfi.

- Envío de datos al programa de la RPI escrito en C++, mediante MQTT usando BLYNK, en concreto una herramienta presente en la plataforma conocida como Bridge, que permite el intercambio de datos entre hardware sin pasar por el servidor BLYNK. Una vez completado un ciclo de lectura de un periodo de red, se realizan los cálculos finales para obtener, P, cosfi ,Q. Tras esto enviamos los datos mediante Bridge al programa c++ que se encarga de la gestión de la APP.

- Si se quisiera poder actuar en remoto, o de manera autónoma por toma de decisiones, este sería el momento de activar alguna salida.

Comentar que para tener control sobre la periodicidad del muestreo se ha tenido que ajustar el ADC mediante el uso de sus registros internos, lo mismo se ha tenido que hacer para ajustar el TIMER a las necesidades de esta aplicación.

Detallando acerca de estas dos piezas hardware:

- **TIMER**: están destinados a temporizar eventos sin tener que recurrir a funciones del tipo delay, bloqueando tontamente el programa.

  Nos permiten lanzar interrupciones una vez se haya alcanzado el tiempo configurado sin necesidad de bloquear el programa durante ese tiempo acordado.

  Se utiliza el TIMER en modo CTC, saltando la interrupción cuando se alcance el tiempo asignado. El MCU que usa la placa Linkit posee un oscilador interno que corre a 8Mhz, pudiendo llegar a disparar una interrupción cada 125ns. Existe un registro interno que es el que indica cada cuantos ticks de 125ns debe dispararse la interrupción. Lo siguiente a tener en cuenta es el valor máximo de ticks que se le puede indicar al timer asociado al número de bits de ese timer. En este caso se recurre al TIMER1 del MCU de 16 bits. Pudiendo llegar a un máximo de ticks acordados de 2^16 -1 → 65535=Vmaxespecificar. Por último tenemos el prescaler, que se encarga de dividir la frecuencia del oscilador interno por varios factores posibles, modificando así la cadencia de los ticks del timer.

  En este caso se ha recurrido a un prescaler de 8, teniendo así un "oscilador" final de 1Mhz. Cada tick equivaldría a 1us. Y usando un compare match de 190, se tiene un timer que dispara interrupciones cada aproximadamente 190us.

  Si el periodo del timer es, redondeando, 200us, y el periodo de la onda de red es de 20ms, se tienen 100 muestras por cada periodo de red. En total, 50 muestras de corriente y 50 mustras de voltaje.

- ADC: para la lectura de señales analógicas como las señales de tensión y corriente, se requieren conversores analógicos a digitales. Hay que tener en cuenta que el MCU es un chip digital y por lo tanto trabaja solo con valores binarios. Las tareas fundamentales y básicas que cumplen estos conversores son:

- Muestrear esa señal en un tiempo acordado.
- Cuantificar ese valor muestreado de acuerdo a la resolución del ADC.
- Codificar ese dato en binario.

El adc del atmega32u4 es un adc de 8/10 bits de resolución ←→ dependiendo de la velocidad de conversión que se le solicite. Además, jugando con el voltaje de referencia, se puede conseguir un mejor ajuste para el resultado final. De nuevo como con el TIMER, se tiene el prescaler, para determinar la cadencia de los ticks y con ello el tiempo de conversión. De nuevo se usa el prescaler de 8, resultando en un tick de 1us.

En el datasheet del MCU se especifica que el ADC requiere de 25 ciclos de reloj para la primera conversión y 13 para el resto. Por lo tanto, aproximadamente el tiempo de conversión será de 25us y 13us respectivamente.

El detalle, que de esos 200 us de periodo de timer, 13 o 25 us se emplean en la conversión del dato y el resto para procesado.

El problema o inconveniente que se tiene aquí es el siguiente: estamos recogiendo muestras alternativamente de tensión y coriente, por lo que constantemente se está reprogramando el ADC, y como consecuencia el tiempo de conversión es de 25 us.

Programa MCU → ADCTIMER_BLYNK

Detalle función que inicializa el ADC del MCU ATMEGA32u4

```
//------------------------------------------------funciones inicializacion hardware atmega32u4
// inicializo ADC trigered compare match timer 1
//cuando el timer 1 venza se dispara la conversion y tengo hasta que vuelva a disparar para realizar la conversion que toque
void adc_init() {
  //con esta configuracion tiempo de conversion es 16 us
  // Voltage Reference 2.56v
  //vcc 01
  //ref aref 00
  ADMUX |= (0 << REFS1) | (1 << REFS0);
  // ADC Channel A0 (ahora mismo recoge dato del sensor de temperatura interno del chip atmega32u4)
  ADMUX |= (0 << MUX5) | (0 << MUX4) | (0 << MUX3) | (0 << MUX2) | (0 << MUX1) | (0 << MUX0);
  // Prescaler (1/8),
  ADCSRA=0;
  ADCSRA |= (0 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
  // ADC Auto Trigger Source - Timer1B Compare Interrupt
  ADCSRB |= (1 << ADTS2) | (0 << ADTS1) | (1 << ADTS0);
  //ADCSRB |= (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);
  // Enable ADC
  ADCSRA |= (1 << ADEN);
  // Enable Auto Trigger
  ADCSRA |= (1 << ADATE);
  //enbale adc interrupt
  ADCSRA |= (1<< ADIE);
  //sei();
  //ADCSRA |=(1<<ADSC);
}
```

Figura 11.Funcion código .ino inicializa el ADC

Lo primero que se hace es fijar la referencia del ADC, se tomo voltaje de referencia los 3.3V de voltaje de operación de la placa.

Posteriormente se selecciona el canal de interés para realizar la primera lectura, en este caso el A0.

Ahora se configura la velocidad de la conversión. → 16us es la velocidad de conversión programada, salvo para la primera conversión que el ADC requiere mas tiempo.

Se asocia un timer en concreto el TIMER 1 para que vaya marcando el inicio de la conversión de una nueva

muestra.

Por ultimo habilito el ADC, el auto TRIGGER, y las interrupción asociada al fin de la conversión.

Detalle función que inicializa el TIMER 1

- Contador se resetea
- Configuro el tiempo de disparo de la interrupción jugando con los prescaladores y con el valor a llegar con el contador
- Inicializo el contador
- Habilito las interrupciones del timer, en concreto del subtimer B

```
// incializo Timer1 Compare Interrupt with a given compare count number
void timer1_init(uint16_t compareMatchRegister) {
  // Reset Timer Count
  TCCR1A = 0;
  // Mode = CTC, Prescaler = 1
  TCCR1B = (1 << WGM12) | (0 << CS12) | (1 << CS11) | (0 << CS10);
//TCCR1B = (1 << WGM12) | (0 << CS12) | (1 << CS11) | (1 << CS10);
  // Initialize the counter
  TCNT1 = 0;
  // Timer compare value
  OCR1A = compareMatchRegister;
  OCR1B = compareMatchRegister;
  // Initialize the Timer1_Compare_A interrupt
 // TIMSK1 = (1 << OCIE1A);
  //TIMSK1 = (1 << OCIE1B)|(1<<OCIE1A);
  TIMSK1 = (1 << OCIE1B);
  // Enable global interrupts
  //sei();
  //t1=micros();

}
```

Figura 12.Función código .ino inicializa el TIMER

Por útimo, y como parte fundamental del programa, se tiene la **función asociada al fin de la conversión** de un dato dentro del ADC.

Para esta parte aun siendo analizada en el documento presente, se adjunta enlace para la descarga del archivo de programa completo comentado. Entonces, por extensión de la función mejor dirigirse dentro del código a →

ISR(ADC_vect){

……..

}

3.4. Programa en C++ para la RPI

Para compilar y ejecutar el programa en c++ usando la plataforma BLYNK sobre Linux, se ha seguido la siguiente guía para la puesta en marcha:

[8]

Simplemente ajustar una serie de parámetros para advertirle al compilador gcc que estamos trabajando con Linux

o Raspberry a la hora de la compilación.

Este programa se ocupa de lo siguiente:

- Se conecta al servidor BLYNK, el cual está montado en la misma RPI-Instancia Linux perteneciente a GCP.
- Extrae datos de las comercializadoras de la base de datos propia que se encuentra también en el servidor Linux. De manera que recoge los nombres y precios de las mismas, para los cálculos de consumo y sugerencias de ahorro.
- Lleva un control del consumo en kwh en función de los datos que le pasa la placa Linkit (comercializadora, potencia y tarifa contratada), además de un control del consumo en euros en función de los datos extraídos de la base de datos. No solo del consumo con la compañía contratada sino también el consumo con el resto de compañías presentes en la base de datos.
- Lleva un control del periodo y franja horaria en la que se está realizando el consumo, para escoger el precio asociado del kwh si se tiene contratada tarifa con discriminación horaria.
- Lleva un control del consumo uso de la potencia contratada, lanzando sugerencias. Es decir se detecta cuantos picos de potencia se han producido durante el mes en cuestión, además de llevar un recuento de la potencia en media que se está demandando.
- Lanza notificaciones de consumo-ahorro a la APP.
- Debería almacenar variables críticas en cada ciclo por posibles pérdidas de alimentación.
- Realmente, si se pudiera implementar en la MCU, o en la propia MPU que acompaña al MCU dentro de la placa linkit, estaríamos quitando tanta dependencia de red. Ahora mismo, el MCU envía datos a la RPI por internet al programa c++, y ahora el programa en c++ tiene que volcar estos datos a la APP. De la otra forma solo tendríamos un envío de datos por red.

  Se hicieron pruebas para esto mismo, para tener programa de sensado y gestión de APP BLYNK en el mismo MCU, pero la memoria de programa no era suficiente.

  Tampoco se pudo usar la MPU que acompaña a la MCU en la placa por fallo en compilación cruzada debido a problemas de librerias con BLYNK y openWrt.

A la hora de detallar el programa main.cpp, simplemente mostrar fragmentos de código de los puntos más clave del programa, a parte de adjuntar un enlace de descarga del mismo:

Interior del bucle loop:

```
423  void loop()
424  {
425    kw=rand() % 10 + 1;
426    kwh=rand() % 10 + 1;
427    Blynk.run();
428    //para el mes en el que estoy necesito saber si horario verano e invierno
429    //solo si tengo discriminacion horaria??
430    //el periodo del dia en el que estoy
431
432
433
434    //si tengo todos los datos necesarios del usuario
435    if(ok[0] && ok[1] && ok[2]){
436      //nueva confifuracion
437      //aqui tendria que calcular el precio fijo por potencia
438      //considero 30 dias el mes
439      for(int j=0;j<ncom;j++){
440        precio[j]=precios[j][0]*30*pmax;
441        cout<<"precio por potencia al mes:";
442        cout<<precio[j]<<endl;
443      }
444
445      Blynk.virtualWrite(V8,precio[com]);
446
447      kwhcont=0;
448      contpmax=0;
449      kwmedia=0;
450      c=0;
451      flag2=1;
452      ok[0]=0;
453      ok[1]=0;
454      ok[2]=0;
455    }
456
```

Figura 13.Función loop() del código gestiona la APP dentro de la raspberry pi c++ (I)

En esta primera parte del código se comprueba si se tienen los datos del usuario: comercializadora, tarifa y potencia contratada (de ahí lo del vector ok de longitud 3). Si se tienen los datos del usuario se calcula el precio por potencia para ese mes para cada una de las distintas comercializadoras. También se inicializan una serie de variales para el periodo en cuestión, como:

- Kwhcont: para el recuento total de consumo en ese periodo
- Contpmax: para el recuento de picos de potencia máxima
- C:
- Flag2: variable bandera para indicar al programa que ya se ha configurado la APP con los datos de los usuarios.

Tras hacer esta comprobación, se comprueba si se tiene la APP configurada y si ha llegado un dato nuevo desde la placa de sensado.

Tras esto se llama a la función horper(), la cual chequea si es el inicio de un nuevo periodo de facturación y contabiliza gastos del usuario en función de la hora que sea (lleva el recuento con su comercializadora y con las demás de la base de datos para sugerir cambios de tarifa para ahorrar dinero).

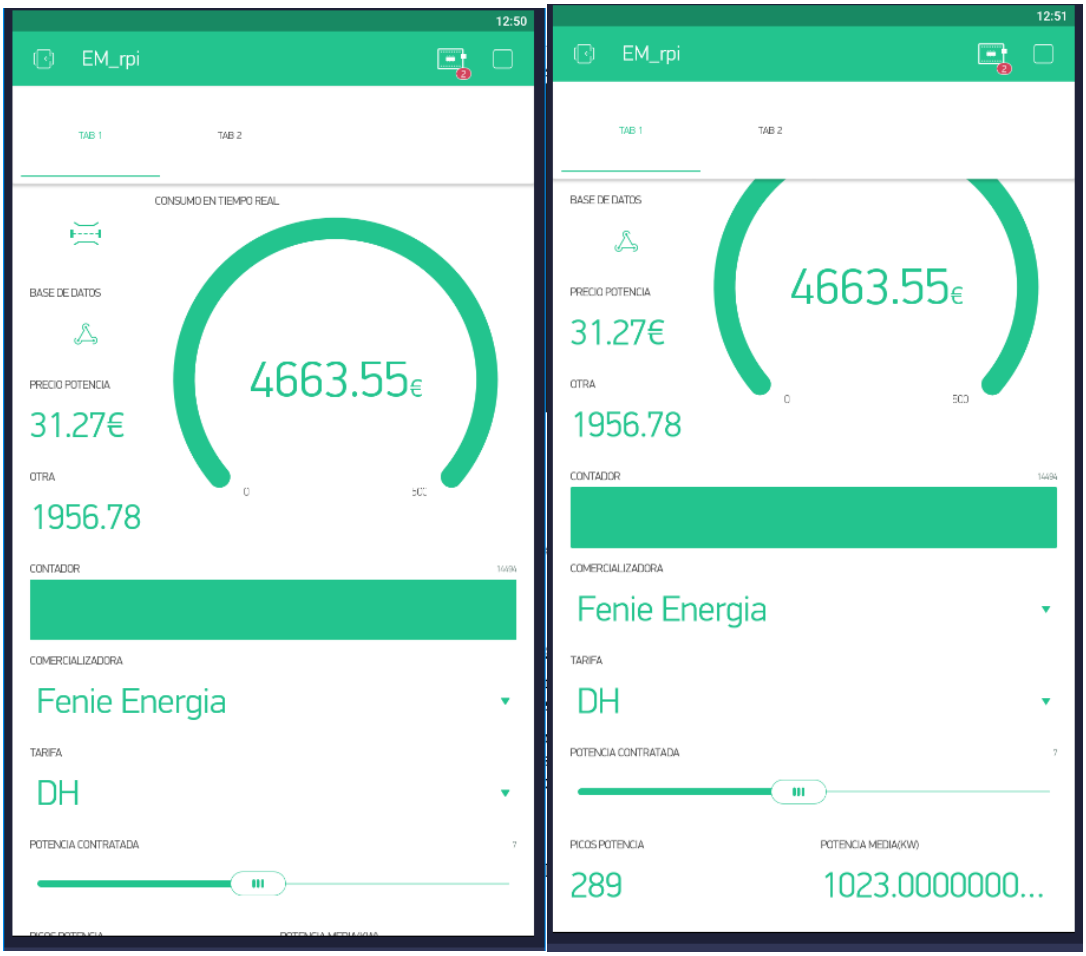
Se actualiza el recuento de consumo en kwh, se chequea si en este periodo de sensado y procesado se ha alcanzado un pico de potencia, se va llevando un recuento de la media de potencia y se actualizan los widgets respectivos con los datos obtenidos hacia la APP.

```
458     if (flag2 && flagdatos){
459       /*float aux=horper();
460       cout<<aux<<endl;
461       precio=precio+aux*kwh;
462       cout<<precio<<endl;
463 
464       Blynk.virtualWrite(V7,precio);*/
465       //calculo de precios en t real
466       horper();
467       kwhcont=kwhcont+kwh;
468       Blynk.virtualWrite(V1,kwhcont);
469       //recuento de la potencia media del mes
470       //contador de picos de consumo y recojo el momento del dia
471       if(kw>=pmax)contpmax++;
472       kwmedia=kwmedia+kw;
473       c++;
474       Blynk.virtualWrite(V10,contpmax);
475       Blynk.virtualWrite(V11,kwmedia/c);
476       cout<<kw<<endl;
477       flagdatos=0;
478     }
479     delay(5000);
480     cout<<"nuevo periodo"<<endl;
481     //Blynk.virtualWrite(V0,HIGH);
482     //Blynk.virtualWrite(V0,1);
483 
484   }
485 
```

Figura 14.Función loop() del código gestiona la APP dentro de la raspberry pi c++ (II)

3.5. Plataforma IoT BLYNK

Figura 15.APP BLYNK Android (I)



Las fotos de arriba se corresponden con la página principal de la APP BLYNK, widgets que la conforman:

- Consumo en tiempo real en euros del mes en cuestión
- Precio por potencia contratada ese mes
- Precio por consumo en euros de un competidor seleccionado (otra comercializadora)
- Contador de consumo en kwh del mes
- Seleccionable tipo desplegable para el tipo de tarifa →
    - SDH : sin discriminación horaria
    - DH: con discriminacion horaria
    - DH-Supervalle: con discriminacion horaria supervalle

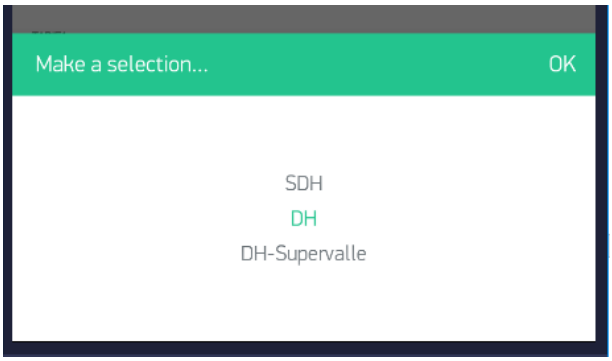


Figura 16.APP BLYNK Android (II)

- Slider para seleccionar la potencia contratada
- Seleccionable tipo desplegable para seleccionar una de las comercializadoras disponibles.

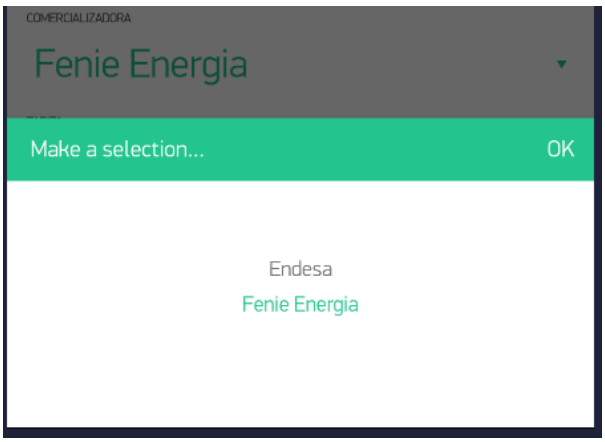


Figura 17.APP BLYNK Android (III)

- Indicador de picos de potencia
- Indicador de media de potencia en ese period de factuación

Por otro lado, en el menú secundario nos encontramos con lo siguiente

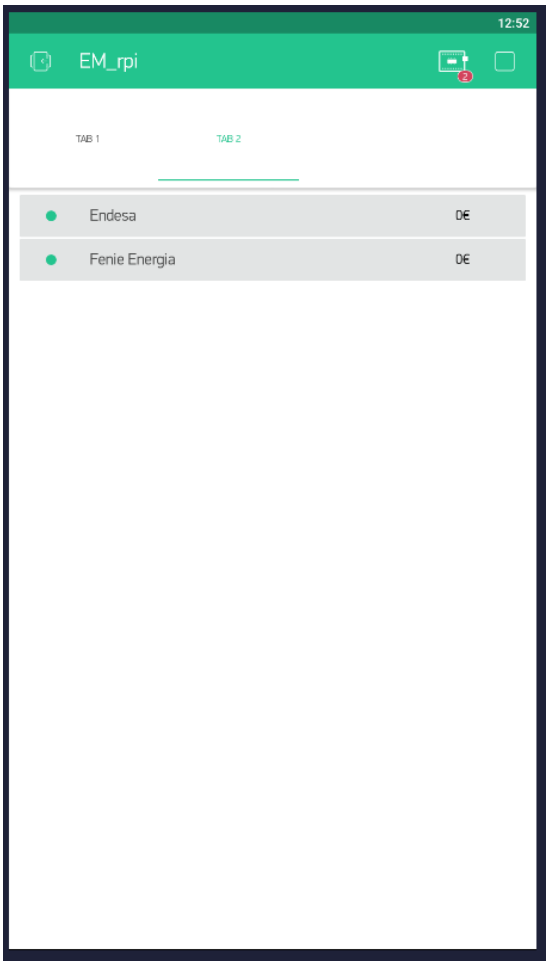


Figura 18.APP BLYNK Android (IV)

Aquí se va llevando el recuento del consumo en euros no solo con tu comercializadora sino también con todas las que componen la base de datos.

Hoy en día existen numerosas plataformas orientadas al IoT. En general estas plataformas mínimamente se encargan de comunicarse con el hardware (MCU), recogiendo datos desde el microcontrolador o lanzándole órdenes, además de guardar estos datos en la nube. Algunas pueden añadir más servicios del tipo, tratamiento Big Data.

Plataformas conocidas serían: Thingspeak, Adafruit IO, Thinger…

BLYNK, al igual que la mayoría de estas plataformas se encarga del control en remoto del hardware, visualización de datos asociados a los sensores, almacenamiento de datos en la nube, incluso proporcionan el paquete necesario para montar el servidor local BLYNK en hardware propio.

Esta es una de las grandes ventajas que aporta BLYNK, y es que liberan el servidor, para montar el servidor en local y no tener que depender de servidores propios de la empresa basados en la nube. Además de esta manera, muchas características propias del servidor puedes modificarlas y adaptarlas de manera más específica a la aplicación, como por ejemplo:

- Puertos asociados a cada canal de comunicación.
- Montar la base de datos en local, para posteriormente aplicar Big Data.
- Controlar parámetros de comunicación, de tiempo de espera, trafico de mensajes máximo…

En este caso se ha montado el servidor BLYNK, como ya se ha mencionado dentro de una RPI pi 3.

Un proyecto elaborado con BLYNK se compone de 3 partes fundamentales:

- La APP, para IOS y Android. En ella se tiene que configurar el servidor asociado para que la APP funcione, se tiene que componer la APP usando distintos tipos de widgets, configurarlos y asociarlos a distintos pines virtuales para que luego desde el programa MCU se puedan controlar.
- El servidor BLYNK, que puede ser local o usando servidores propios de la compañía.
- Hardware final, el cual va a ser finalmente gobernado por la APP. Este hardware tiene que contar con un programa basado en librerías que se nos proporcionan.

Entonces cuando desde la APP se actué un botón, esta acción llega al servidor, se procesa y reenvía la orden al hardware para que realice la acción final. O viceversa, si se manda un dato desde el hardware hacia la APP, almacenándose ese dato en la base de datos a su paso por el servidor.

Pasos a seguir para poder usar la plataforma:

- Descarga de la librería para el hardware en cuestión
- Descarga de la APP móvil para IOS o Android, y plantear los widgets que van a ser usados, asi como asociarlos a los pines virtuales que toque para hacer referencia a ellos posteriormente desde el programa hardware
- Descarga del servidor BLYNK si se quiere explotar al máximo las ventajas que esta plataforma ofrece e instalarlo dentro del hardware respectivo.

  Para hacerlo más robusto lo ideal es hacer que el servidor se dispare automáticamente tras el inicio del ordenador o placa donde se haya alojado. Por ejemplo tras la pérdida de alimentación y recuperarla, el servidor arrancaría automáticamente.

  Para esto se ha usado el administrador de procesos Cron.

  También es necesario que el servidor tenga asociada una ip estática, así como hacer "port forwarding" para redirigir todas las peticiones en relación al servidor BLYNK hacia la maquina que lo aloja.

  Por ejmplo para la comunicación del servidor con el hardware se ha usado el puerto 8082, para la comunicación con la APP BLYNK se ha usado el 9443, y para el control en remoto de la RPI via ssh, incluso desde fuera de la red local se ha usado el puerto 22.

  Captura asociada a los puertos del router para abrir la comunicación hacia el exterior de la red:

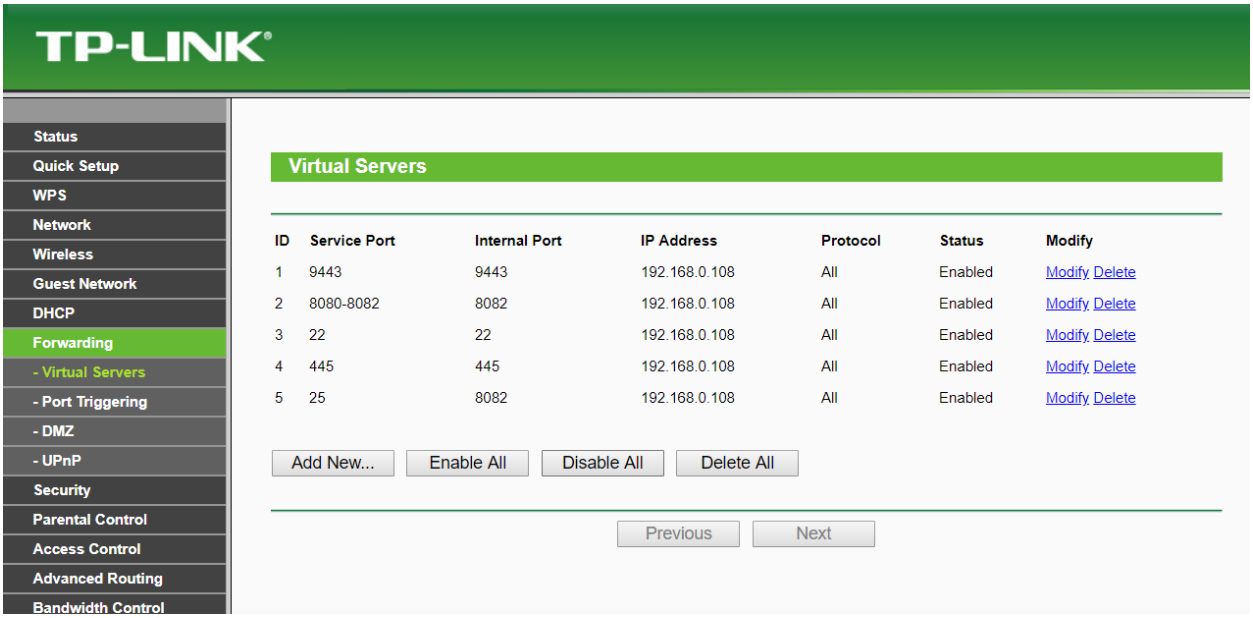


Figura 19.Port Forwarding

Cabe destacar el uso de un rango de puertos externos asociados a un único puerto interno. Como por ejemplo el [8080,8082] → 8082. En ciertas redes locales se detectó que el firewall del router bloqueaba la comunicación asociada al puerto 8082, sin embargo usando puertos más usuales como el 8080, si dejaba realizarla. Es como si se le hicera pensar al router primero donde esta instalado el harware de sensado que realiza comunicaciones usando el puerto 8080, cuando en realidad se esta usando el 8082 dentro del router donde se encuentra el servidor. De esta manera el hardware instalado en las casas de usuarios podría usar cualquier puerto dentro del rango [8080-8082].

Como documento de guía se ha seguido el siguiente enlace: [9]

3.6. Plataforma Google Cloud

Plataforma similar a las que ofrecen Microsoft (Azure) y Amazon (Amazon Web Services), que basadas en la nube reúnen una serie de servicios de distinto ámbito.

Destacan:

- Compute engine, basado en máquinas virtuales que usan la infraestructura de Google, escalables y altamente editables-configurables tanto en "hardware", como en imagen que corren, se alquilan por tiempo de uso y en función también de la configuración usada.

  En cuanto al hardware puedes seleccionar el tamaño y tecnología de disco duro, RAM, CPU's, GPU…
- Cloud Storage, almacenamiento basado en la nube de todo tipo de datos.
    - Cloud SQL.
    - Persisten Disk, usado en máquinas virtuales.
- Análisis de datos, plataforma de big data para análisis de datos
- Servicios de inteligencia artificial
    - Aprendizaje automático
    - Análisis de video
    - Análisis de imagen
    - Análisis de texto
    - Reconocimiento de voz

3.7. Puesta a punto de la máquina virtual dentro de GCP

Primero se necesita una cuenta de Google, posteriormente hay que enlazar esta cuenta con la plataforma de Google cloud. Se adjunta enlace: [10]

En el momento de escribir el presente documento, Google ofrece una prueba gratuita al inscribirte, con un crédito de 300$, con caducidad un año, para que pruebes la plataforma.

El siguiente paso acceder a la pestaña de Compute Engine, crear instancia, apareciendo el siguiente menú de configuración…

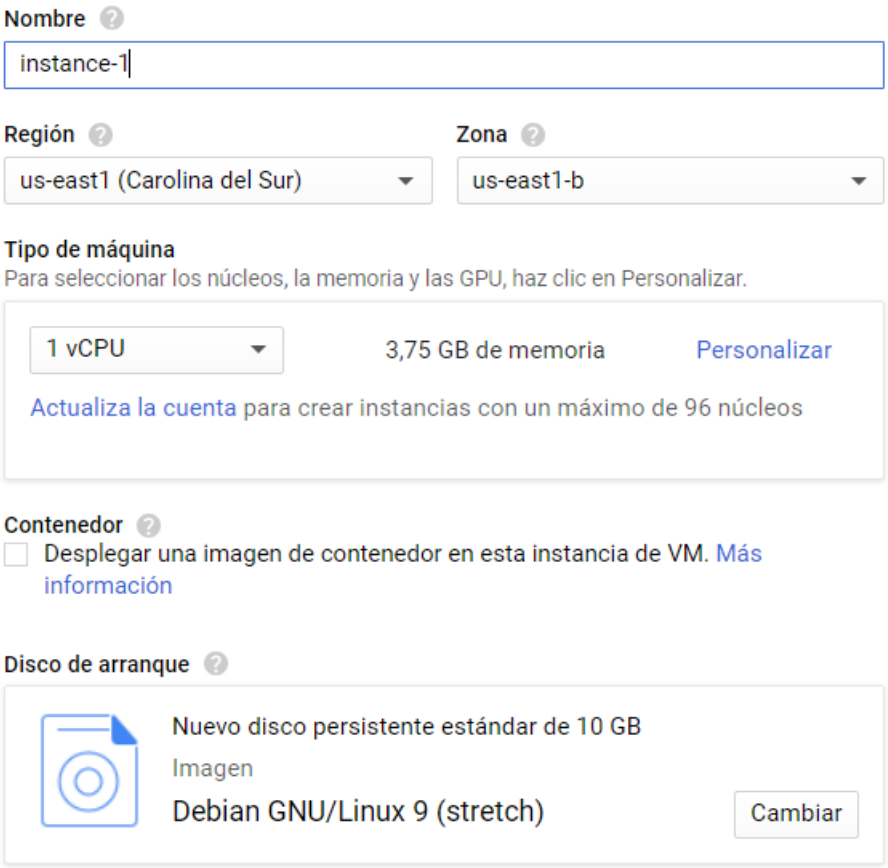


Figura 20.Configuración máquina virtual I

Donde hay que seleccionar la región en la que queremos ejecutar los recursos.

También la configuración de la máquina:

- Número de núcleos virtuales
- RAM
- Generación del procesador de la máquina que ejecuta la VM
- Habilitar GPU y tipo de de la misma en caso de que aplique

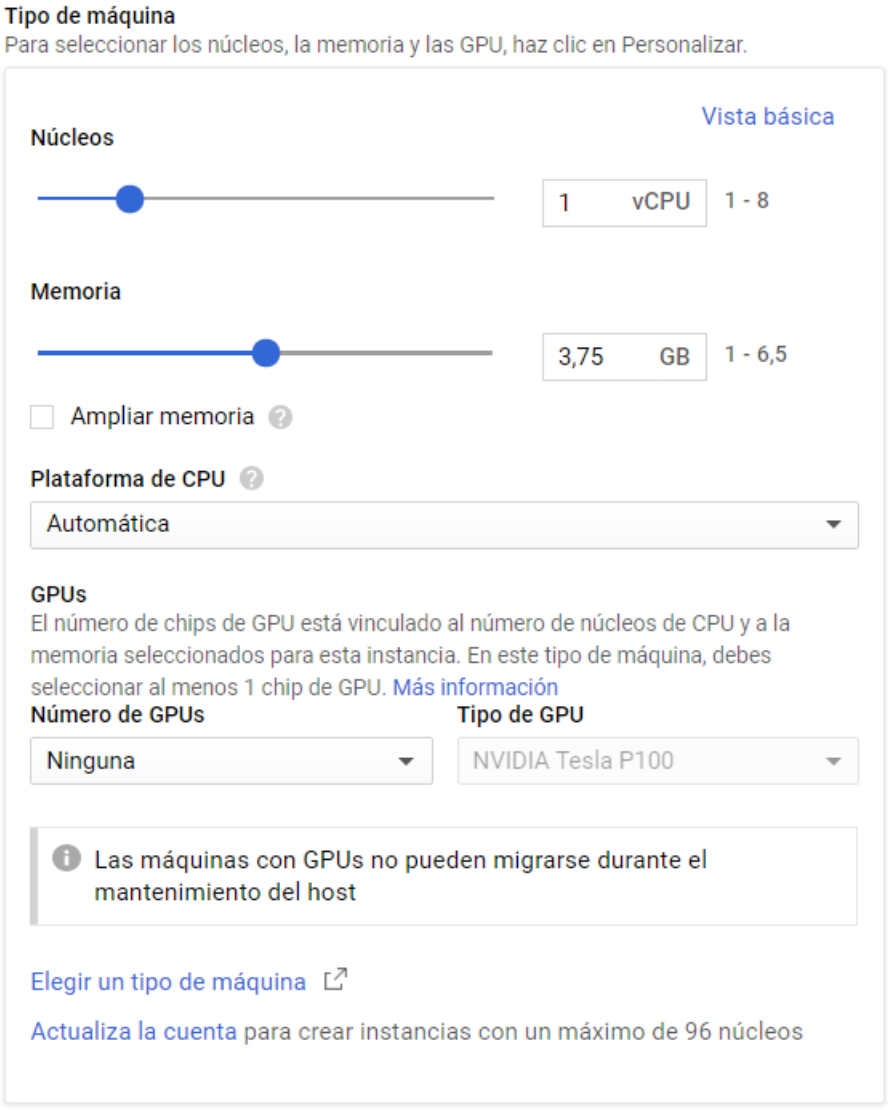


Figura 21.Configuración máquina virtual II

Por último, antes de crear la instancia, seleccionar la imagen que queremos que ejecute la VM, así como la tecnología y capacidad del disco duro.

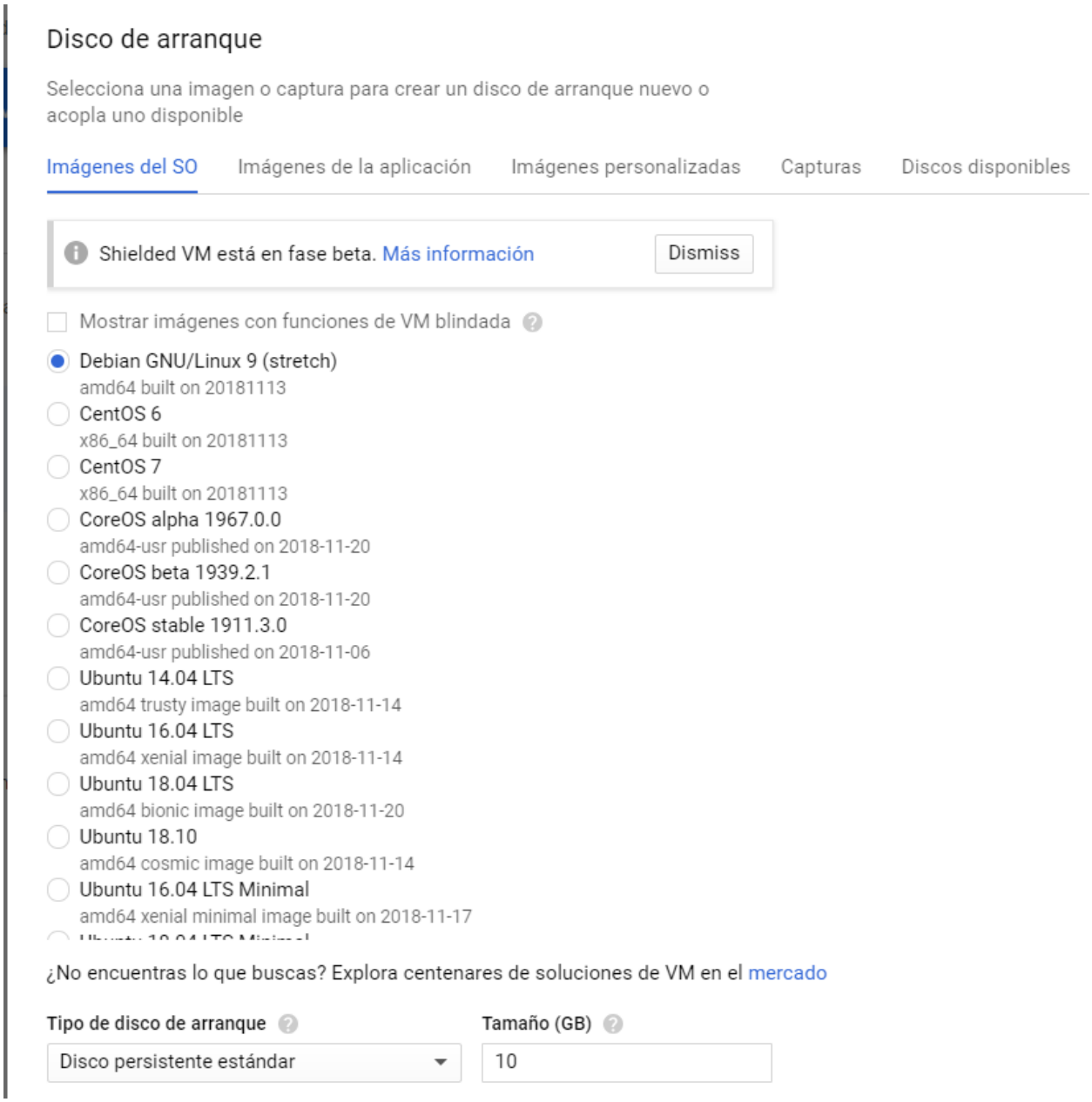
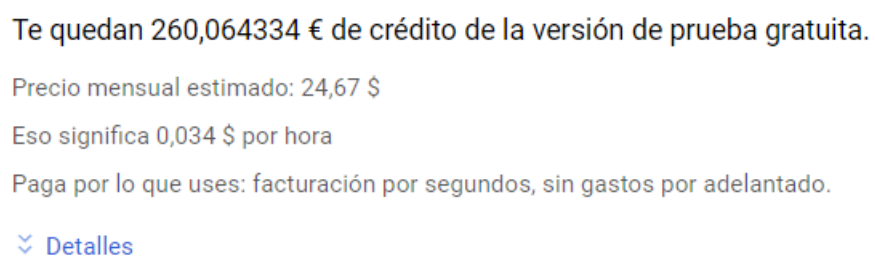


Figura 22.Configuración máquina virtual III

Una vez se haya configurado la máquina la plataforma muestra una estimación del coste por alquiler de la máquina, que en mi caso queda a unos 3 céntimos por hora de alquiler.



Tras crear la instancia se proporciona la IP externa de la máquina para acceder a ella en remote usando SSH, la plataforma ofrece un shell online para controlar en remoto la misma. Si se quiere controlar la máquina usando la shell propia del pc personal o por ejemplo putty, hay que generar una clave y referenciarla dentro de la plataforma.

Usando servicios como el de no-ip, podemos escoger un nombre de dominio y referenciarlo a la ip externa de nuestra máquina virtual. Entonces, tras esto el nombre de dominio equivale a usar la ip externa de nuestra máquina virtual.

En este caso hemos usado el nombre de dominio newfarm.hopto.org, usando el plan gratuito de esta empresa. El inconveniente, que transcurrido un mes hay que volver a activar el nombre de dominio.

3.8. Base de datos MYSQL

Se ha montado una base de datos propia como se ha mencionado anteriormente, que almacena los precios asociados a las distintas comercializadoras.

Entonces para el mercado libre, para una potencia contratada de hasta 10 kw, tenemos por ejemplo:

comercializadora	ekwdia	ekwh	ekwhp	ekwhv	ekwhpsv	ekwhvsv	ekwhsv
Endesa	0.114323	0.166955	0.144026	0.064832	0	0	0
Fenie Energia	0.104229	0.149513	0.16796	0.092912	0.172421	0.10292	0.082916

Figura 23.Tabla de datos MYSQL

Figura 17. Tabla de datos MYSQL

De izquierda a derecha:

- Nombre de la comercializadora.
- Precio por potencia contratada por día.
- Precio por consumo con tarifa sin discriminación horaria.
- Precios por consumo con tarifa con discriminación horaria.
- Precios por consumo con tarifa con discirminacion horaria supervalle.

La idea es tener una tabla asociada a cada rango de potencia, para cada comercializadora. Además de ampliar el número de comercializadoras. Ahora mismo el mantenimiento de esta base de datos es manual, es decir cada año cuando cambien los precios de cada compañía habrá que editar esta tabla.

La idea es que estos datos se consulten y actualicen de manera autónoma. Se necesitaría:

- Programa que extraiga datos de las distintas páginas webs de las comercializadoras
- Programa que modifique las tablas MYSQL de acuerdo a lo anterior

Figura 24.Rango de Potencias a contratar dentro de Fenie Energia

3.9. Servidor LAMP

Tutorial seguido para instalar todas las dependencias: [11] y [12]

Para poder servir los datos de la base de datos hacia el MCU o MPU para que realice los cálculos de consumo en tiempo real, se monta un servidor LAMP.

LAMP es un término usado para hacer referencia al sistema de infraestructura de internet que usa las herramientas

- Linux, el sistema operativo de la máquina que lo alberga. En este caso usamos Raspbian, que ya mencionamos en el primer punto de este apartado.
- Apache, el servidor web como tal.
- MYSQL o MariaDB como gestor de base de datos. En cuanto a versiones usadas, estamos usando MYSQL 5.5 en Raspberry y 5.6 dentro de la máquina virtual alquilada a Google.
- PHP, Python… como lenguajes de programación. En raspberry usamos PHP5, y en la máquina de Google estamos usando la evolución PHP7

Este servidor montado dentro de la RPI a modo de complemento al servidor local BLYNK, se encarga de devolver los datos a todo cliente que realice la petición. Datos asociados a las tablas MYSQL, como serían los precios por tarifa y comercializadora.

Además se tiene un código PHP que se encarga de recoger los datos de una petición GET y almacenarlos dentro de la base de datos y tabla respectivos. Este código está pensado para almacenar los datos de los sensores, y otros datos que se consideren de interés para ser explotados posteriormente aplicando BigData.

Ejemplo de consulta con respuesta del servidor:

Consulta: http://89.39.22.36/MYSQL2.PHP

Respuesta:

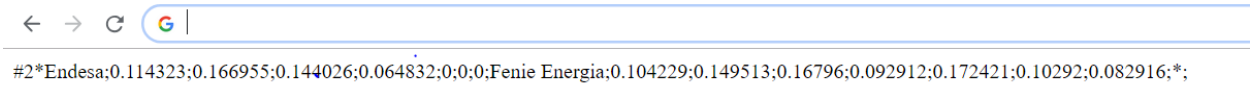


Figura 25.Test petición GET al servidor

Como se aprecia en la imagen anterior, los datos se separan usando como delimitador una coma. También previo al envío de datos masivo, se indica el número de comercializadoras que componen los datos que voy a enviar.

Los métodos GET y POST son dos métodos de envío de datos propios de HTML. Discrepan en la forma de enviar los datos, mientras que el método GET envía los datos usando la URL, el método POST los envía de manera oculta.

Ejemplo → http://89.39.22.36/MYSQL3.PHP?sistema=1&potenciacontratada=2&compañia=1&consumo=500

Se distinguen varias partes en la URL anterior:

- Direccion web en sí → http://89.39.22.36/MYSQL3.PHP
- El símbolo ? indica donde empiezan los datos
- Después aparecen parejas de datos con su nombre y su valor asociado

- Para separar parejas de datos se usa el símbolo &

En el caso del método post aunque los mismos datos serán enviados hacia la dirección que se desee, no se podrán ver en la URL.

Ambos métodos son ampliamente utilizados, cada método tiene sus ventajas e inconvenientes y no se puede decir que uno sea mejor que el otro.

Observaciones:

- GET:
    - Lleva los datos de forma visible al cliente
    - Los datos son visibles por la URL
- POST:
    - Datos ocultos

4 RESULTADOS

En cuanto a los resultados finales obtenidos y las características del sistema hay que hablar por un lado de los datos referentes a los sensores (velocidad de lectura, veracidad de los mismos…), las suposiciones y consideraciones tenidas en software (como por ejemplo el tiempo durante el cual se considera que el consumo de potencia es el mismo).

En general, el sistema se comporta de forma estable, tanto a nivel de estabilidad WIFI como a nivel de resultados de medidas.

4.1. Muestreo de tension y corriente

Como se comentó en el apartado 3.3, el sensado consiste en ir recogiendo muestras alternativas de tension y corriente en un intervalo de aproximadamente 200 us. Cada ciclo básico de sensado tiene un periodo de 200 us, 25 us supone la conversion del dato por parte del ADC y el resto se emplea para cálculos intermedios:

- Suma cuadrática de valores instantáneos de tension o corriente según toque.
- Detectar cruces por cero de tension y corriente para el cálculo del desfase y posteriormente, el cálculo del cosfi.
- Recoger los valores pico para obtener un segundo valor de RMS.

Este periodo de sensado equivale a una frecuencia de muestreo de 5khz, si tenemos en cuenta que son muestras intercaladas de tensión y corriente tenemos una frecuencia real de 2.5khz.

Finalmente se contrastan los datos obtenidos de tensión y corriente con valores obtenidos con un polímetro certificado:

- Para voltaje
- Para corriente

Si ahora comparamos el valor obtenido para el cosfi, potencia activa y reactiva con los datos proporcionados por un analizador de red tenemos:

4.2. Consideraciones para el software

Características y consideraciones referentes a la programación:

- Tiempo entre periodo de lectura y siguiente donde considero que la potencia consumida se ha mantenido constante: 5s
- Dias en media de un mes para el gasto por potencia contratada: 30 días.
- Considero que las cargas conectadas a mi Sistema son lineales y por tanto ondas de tensión y corriente son sinoidales, no considero armónicos.

- Pequeño ajuste por software para corregir el error del cosfi, relacionado con los 200us de desfase entre las muestras de tensión y corriente.
- Programa que se encarga de gestionar la APP móvil y los datos desde el MCU están en el mismo servidor. Lo ideal que el MCU pudiera albergar el sensado y gestión de la APP.

4.3. Consideraciones para el hardware

Consideraciones y características referentes al hardware:

- Uso sensores de corriente de hasta 30Arms, luego el máximo de potencia contratada admisible sería de 7KW.
- Solo un sensor de corriente → circuito general
- El sensor de corriente desvirtúa la medida en torno a los 3º por ser una carga de tipo inductiva.
- Igualmente ocurre con el transformador que desvirtúa la medida.

5 EXTRAS A IMPLEMENTAR

Como extras para el sistema (algunos de los siguientes puntos ya se han mencionado anteriormente en su punto respectivo):

- Incluir varios sensores de corriente para hacer medidas de consumo por circuito de la instalación y no solo del circuito general. El software está preparado para que sea modular por lo que únicamente estaríamos limitados por el numero de entradas analógicas del controlador y por la tasa de muestreo que se quiera. El adc del MCU en cuestión cuenta con 12 canales ADC por lo que se podría montar un sistema de hasta 11 circuitos, ya que necesitamos una entrada para la lectura del voltaje.
- Ahora mismo el Sistema está preparado para funcionar con Mercado libre con una potencia contratada menor de 10Kw. Fácilmente se podría preparar el sistema para que también se pudiera tener control del consumo en viviendas de más de esta potencia. Simplemente incluyendo otra tabla de precios en la base de datos similar en estructura a la que hay de potencia <= 10kw.

  También se podría preparar al Sistema como se comentó en puntos anteriores para que funcione dentro del Mercado regulado. Para ello el servidor debería ser capaz de extraer de una página web el precio del kwh. Esta funcionalidad daría mucho juego ya que si este precio temporal comenzara a guardarse en una base de datos y se le aplicara tratamiento de datos (BigData) podríamos empezar a predecir comportamientos y adelantarnos a ciertos valores futuros.

- Lo ideal sería que el programa que ahora mismo cuelga de la RPI para comunicarse con el servidor BLYNK y con el hardware, para llevar el control del consumo estuviese dentro de la placa encargada del sensado para evitar tanta dependencia de la red.
- Todos los datos que se mandan al servidor BLYNK podrían ser almacenados en una base de datos para aplicar un tratamiento de datos(BigData).
- Migrar de la placa de prototipado a una PCB, donde se sitúen el alojamiento de la placa Linkit, el circuito de acondicionamiento, conectores tipo “click” para la fácil integración de los sensores. Lo ideal sería o bien preparar distintos modelos de pcb con más o menos salidas de sensores o dejarlas todas accesibles para que se usen las que se requieran.
- Migrar a trifásica para un entorno industrial. La idea para el sensado sería similar a la de hacer lectura de circuitos por separado de una instalación eléctrica monofásica.
- Como responder ante la falla de la conexión a internet o pérdida de alimentación. Lo ideal que ante la pérdida de alimentación se pudiera notificar al usuario vía sms. Para esto habría que incluir un modulo GSM con una tarjeta sim para que salte la notificación cada vez que se corte la luz vía sms, este modulo se encuentra por un precio en torno a los 15 euros.

6 ANEXO

- Código sensado y envío de datos al servidor

```
//ultima modificacion 12/01/19
/*
27/08/2018

Programa completo para la recogida de datos desde los sensores y envio al mpu
para procesado y lanzar a app blynk

Usando linkit smart 7688 duo, mcu atmega32u4 mpu mediatek 7688

Mejoras:
-Varias pasadas para el calculo del cos fi, luego hacer la media
-Guardar variables en epprom por posible perdida de alimentacion
-Deshabilitar facilmente la depuracion con un define Serial **hecho**
-freno del programa en 5s


*/

//calibracion con un pulsador desde la app seria util -- offset
//corrige por software error Vrms


//28/11/18
/*
 * v2 --> recoge el maximo y el minimo, vp+ vp- hacer una media y hayar el
rms asociado
 * v3 --> para calculo del cosfi, no realizo una simple resta entre los dos
valores sino que hago un estudio mas exhaustivo de la diferencia de tiempos
en cada lectura
*/
float auxVmax=0.0;
float auxVmin=0.0;
float auxImax=0.0, auxImin=0.0;
unsigned long tpVmax,tpVmin,tpImax,tpImin;



#include "Arduino.h"   // replace this by "WProgram.h" when your IDE is older
then 1.0

#define DEBUG false // flag to turn on/off debugging
#define Serial if(DEBUG)Serial

//#define BLYNK_PRINT Serial
```

```
//**librerias usadas**
#include <TimeLib.h>
#include <Bridge.h>
#include <BlynkSimpleYun.h>


//token asociado al hardware desde la app
char auth[] = "310ee9ac730e4828b100f3f0fbb4343e";

//widget bridge para enviar datos entre distinto hardware usando protocolo
blynk
WidgetBridge bridge1(V1);

//asociar token del hardware destino para el envio de datos
BLYNK_CONNECTED() {
  bridge1.setAuthToken("a89a726f2bad4e22be7a64e95b7b5c86");
}


//**variables globales y constantes de programa**
//8/10/2018  timer 1 dispara cada 170us
int flagdebug=0;
//unsigned long c[80];
//unsigned long v[80];

//int ct=0;

//uint16_t compareRegister = 160;//120us
uint16_t compareRegister = 190;//120us
//uint16_t compareRegister = 60;//120us
//unsigned long t1;



//flag aux para sincronizar el main loop con la interrupcion
volatile boolean flag = false, flag1=false;

//-------------constantes de programa----------------
const float pi=3.14;


//resolucion del adc
const float res=3.3/1023.0;
//frecuencia angular de la onda de red
const float w=2*pi*50;
//conversion de rad a grado
const float radagrado=360.0/(2*pi);
//factor de reduccion usado para la medida de la tension, trafo mas div
resistivo
//mini trafo const float factor=228.4/0.25;
```

```
//trafo carril din
const float factor=229.5/0.945;
const float sct=30.0;
//constantes para tomar como referencia por los offset usados para la medida
de corriente y tension
int cteI=559;
int cteV=310;
const int offset=505;
//usado para el calculo del coseno de fi, usando tecnica de cruces por cero
int detect=0;
float cosfi,senfi;

uint16_t voltage;
unsigned long t1,t2,t3,t4=0;
unsigned long dift=0,maximo,minimo;

//aray para las muestras de corriente, solo para las pruebas experimentales
float current[80],voltagge[80];
float auxx[50];
int auxxx;
int auxxxVa=0;
int auxxxIa[3]={0,0,0};
//long times[160],times2[160];
int i=0;

//variables de apoyo para la deteccion de cruces por cero
float aux=0.0,auxaI=0.0,auxaV=0.0;
//variable que recoge lectura siguiente programada, canal 0 corriente, canal
1 tension
int echa=0;
int cont=0;
int analogval;

//para ir haciendo operaciones intermedias para el calculo final del rms
float sum1=0,sum2=0.0;
//contadores para llevar cuenta de medidas de tension y corriente
int l=0, j=0;
int aux1=0;

//array para almacenar cruces por cero de subida y bajada de onda de tension
y de corriente
unsigned long cpcs[2],cpcb[2],cpcsV[2],cpcbV[2];
//indices asociados
int s=0,b=0,sV=0,bV=0;

float S,P,Q;
float Irms=0.0,Vrms=0.0;
int cont2=0;
```

```
//********funciones inicializacion hardware-perifericos atmega32u4********

// inicializo ADC trigered compare match timer 1
// cuando el timer dispare arranca la conversion del adc
// una vez dispare el adc por conversion finalizada tengo hasta que este
lista la siguiente conversion para realizar calculos intermedios
void adc_init() {
  //con esta configuracion tiempo de conversion es 24 us
  // Voltage Reference 2.56v
  //vcc 01
  //ref aref 00
  ADMUX |= (0 << REFS1) | (1 << REFS0);
  // ADC Channel A0 (ahora mismo recoge dato del sensor de temperatura
interno del chip atmega32u4)
  ADMUX |= (0 << MUX5) | (0 << MUX4) | (0 << MUX3) | (0 << MUX2) | (0 <<
MUX1) | (0 << MUX0);
  // Prescaler (1/8),
  ADCSRA=0;
  ADCSRA |= (0 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
  // ADC Auto Trigger Source - Timer1B Compare Interrupt
  ADCSRB |= (1 << ADTS2) | (0 << ADTS1) | (1 << ADTS0);
  //ADCSRB |= (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);
  // Enable ADC
  ADCSRA |= (1 << ADEN);
  // Enable Auto Trigger
  ADCSRA |= (1 << ADATE);
  //enbale adc interrupt
  ADCSRA |= (1<< ADIE);
  //sei();
  //ADCSRA |=(1<<ADSC);
}


// incializo Timer1 Compare Interrupt with a given compare count number
void timer1_init(uint16_t compareMatchRegister) {
  // Reset Timer Count
  TCCR1A = 0;
  // Mode = CTC, Prescaler = 1
  TCCR1B = (1 << WGM12) | (0 << CS12) | (1 << CS11) | (0 << CS10);
//TCCR1B = (1 << WGM12) | (0 << CS12) | (1 << CS11) | (1 << CS10);
  // Initialize the counter
  TCNT1 = 0;
  // Timer compare value
  OCR1A = compareMatchRegister;
  OCR1B = compareMatchRegister;
  // Initialize the Timer1_Compare_A interrupt
```

```
 // TIMSK1 = (1 << OCIE1A);
  //TIMSK1 = (1 << OCIE1B)|(1<<OCIE1A);
  TIMSK1 = (1 << OCIE1B);
  // Enable global interrupts
  //sei();
  //t1=micros();

}


//********rutinas de interrupcion********
ISR(TIMER1_COMPB_vect) {
//  Serial.print("ok");
//  Serial.println(micros()-t3);
//  t3=micros();

  //v[ct]=micros();

}


// rutina de interrupcion asociada a la finalizacion de la conversion que
estaba en curso
ISR(ADC_vect){


  //159
  //recojo 160 muestras en total, 80 de corriente y 80 de tension
  if (i<=159){
  // Done reading
  //flag= 1;
  //conv lista adc0


  //**adc0 lectura de corriente**
  if (echa==0){
  //apunto hacia el siguiente canal, canal asociado a tension
  ADMUX |= (0 << MUX5) | (0 << MUX4) | (0 << MUX3) | (0 << MUX2) | (0 <<
MUX1) | (1 << MUX0);
  echa=1;
  //t1=micros();
  auxxx=ADC;

  //  if (abs(auxxx-auxxxIa[0])<4 && abs(auxxx-auxxxIa[1])<4 && abs(auxxx-
auxxxIa[2])<4 ){//505-508-512
  //    auxxx=offset;
  //  }
  aux=(float)(auxxx-offset)*res*sct;
  //Serial.println(ADC-cteI);
  sum1+=aux*aux;
```

```
  current[j]=auxxx;
  //c[j]=micros();
  j++;

  //detectar los pasos por cero guardando dato anterior
  //subida
  if((auxaI<0.0 && aux>0.0)){
    cpcs[s]=micros();
    //cpcs[1][s]=i;
    s++;

  }

  //bajada
  else if(auxaI>0.0 && aux<0.0){
    cpcb[b]=micros();
    //cpcb[1][b]=i;
    b++;
  }

  //  //el segundo procesado para obtener valor rms
  //  if(aux>auxImax){
  //    auxImax=aux;
  //    //tpImax=micros();
  //  }
  //  else if(aux<auxImin){
  //    auxVmin=aux;
  //    //tpImin=micros();
  //  }

  auxaI=aux;
  //  auxxxIa[1]=auxxxIa[2];
  //  auxxxIa[2]=auxxxIa[3];
  //  auxxxIa[3]=auxxx;
  //


  }



  //adc1 para la lectura de voltaje
  else {
    //ADMUX &= (1 << REFS1) | (1 << REFS0)|(0 << MUX3) | (0 << MUX2) | (0 <<
MUX1) | (0 << MUX0);
    ADMUX &= (0 << REFS1) | (1 << REFS0)|(0 << MUX5) | (0 << MUX4) | (0 <<
MUX3) | (0 << MUX2) | (0 << MUX1) | (0 << MUX0);
    echa=0;
    auxxx=ADC;
    //    if (abs(auxxx-auxxxVa)<1){//505-508-512
```

```
    //      auxxx=offset;
    //    }
    aux=(float)(auxxx-offset)*res*factor;//voltaje asociado

    sum2+=aux*aux;
    voltagge[l]=auxxx;

    l++;


  if((auxaV<0.0 && aux>0.0)){
    cpcsV[sV]=micros();
    //cpcs[1][s]=i;
    sV++;

  }
  else if(auxaV>0.0 && aux<0.0){
    cpcbV[bV]=micros();
    //cpcb[1][b]=i;
    bV++;
  }

  //el segundo procesado para obtener valor rms
  if(aux>auxVmax){
    auxVmax=aux;
//    tpVmax=micros();
  }
  else if(aux<auxVmin){
    auxVmin=aux;
//    tpVmin=micros();
  }


  auxaV=aux;
//  auxxxVa=auxxx;


  }

  }


  //fin de ciclo de lectura desde los sensores, deshabilito el ADC
  else {
    //Serial.println(i);
    //Serial.println(micros()-t2);
    //t3=micros();
//    TCCR1A=0;
//    TCCR1B=0;
//    TIMSK1 = (0 << OCIE1B);
```

```
    ADCSRA &= (0 << ADEN);
    flag=1;
    /*Serial.println("ok adc:");
    Serial.println(t3-t2);
    Serial.println(j);
    Serial.println(l);
    */
  }

  i++;

  //t3=micros();
  //Serial.println("----:");

  /*if (i==101){
  t3=micros();
  }*/
  //flag1=1;

  // Not needed because free-running mode is enabled.
  // Set ADSC in ADCSRA (0x7A) to start another ADC conversion
  // ADCSRA |= B01000000;
}




//funcion inicializacion
void setup() {
  //para depurar programa por puerto serie
  Serial.begin(9600);
  //realizo la conex con blynk, usando servidor local --> servidor basado en
la nube de google
  Blynk.begin(auth, "newfarm.hopto.org", 8080);
  //cli();

  //**Initialize ADC
  adc_init();
  //**Initialize the timer
  timer1_init(compareRegister);
  //timer2_init(compareRegister2);
  //**habilito las interrupciones
  sei();
  Serial.println("arrancamos");
  //t3=micros();
  //t1=micros();

}
```

```
//bucle loop, parte de procesado de datos finales mas envio hacia el harware
asociado a traves de blynk
void loop() {
 Blynk.run();

 if (flag){
   Serial.println(micros()-t4);
   Serial.print("Contador Corriente:");
   Serial.println(j);
   Serial.print("Contador Voltaje:");
   Serial.println(l);

   //debug
    for (int i=0;i<j;i++){
      Serial.println("corriente:");
      Serial.println(current[i]);
      Serial.println("voltaje:");
      Serial.println(voltagge[i]);
//      if (i>=1){
//        Serial.println(c[i]-c[i-1]);
//        Serial.print("time:");
//        Serial.println(v[i]);
//      }
      //Serial.println(c[i]);
      //Serial.println(voltagge[i]);
      //Serial.println(j);
      //Serial.println(auxx[i]);
      //Serial.println(times[i]);
      //Serial.println(times2[i]);

    }
    for(i=0;i<s;i++){
      //Serial.println(cpcs[i]);
    }
    for(i=0;i<b;i++){
      //Serial.println(cpcb[i]);
    }




    //operaciones finales para envio al mpu
    //calculo para resultado final para las medidas
    Irms=sqrt(sum1/j);
    if (Irms<0.3){
      Irms=0.0;
    }
    Serial.print("Irms:");
    Serial.println(Irms);
    float Vrms2;
```

```
    Vrms2=(auxVmax+ abs(auxVmin))/2;
    Vrms2=Vrms2/sqrt(2);
    Vrms=sqrt(sum2/l);
    if (Vrms<220 || Vrms2<220){
      Vrms=0.0;
      Vrms2=0.0;
    }
    Serial.print("Vrms:");
    Serial.println(Vrms);
    Serial.println(Vrms2);

    if (Irms!=0.0 && Vrms!=0.0){
      Serial.println("calculo el cosfi");
      //      Serial.println("max");
      //      Serial.println(tpVmax);
      //      Serial.println(tpImax);
      //      dift=abs(long(tpVmax)-long(tpImax));
      //      //20ms es un periodo de onda completo
      //      //1/4T 5ms==5000us
      //      if (dift<5000){
      //        cosfi=cos(w*dift*1000000*radagrado);
      //        detect=1;
      //      }
      //      if (detect==0){
      //        Serial.println("max");
      //        Serial.println(tpVmin);
      //        Serial.println(tpImin);
      //        dift=abs(long(tpVmin)-long(tpImin));
      //        //20ms es un periodo de onda completo
      //        //1/4T 5ms==5000us
      //        if (dift<5000){
      //          cosfi=cos(w*dift*1000000*radagrado);
      //          detect=1;
      //      }
      //
      //      }


      //primera pasada para detectar el fdp, en cpc,flanco de sibida.
      for (i=0;i<s&&!detect;i++){
        for (j=0;j<sV&&!detect;j++){
          maximo=max(cpcs[i],cpcsV[j]);
          minimo=min(cpcs[i],cpcsV[j]);
          dift=maximo-minimo;
          //30000 es la dif t max admisible ha habido overflow al usar la
funcion micros
          if (dift>30000)dift=4294967295-maximo+minimo;
          Serial.println(dift);
          //dift=abs(0-512);
```

```
          //si la diferencia de cruces es menor en tiempo que 1/4 ciclo de
red, procedemos al calculo del desfase
          if (dift<5000){
            cosfi=cos(w*dift/1000000);
            senfi=sin(w*dift/1000000);
            detect=1;
            Serial.println(dift);
            Serial.println(cosfi);
          }
        }
      }

      //2º pasada para detectar el fdb, en cpc, flanco de bajada
      for (i=0;i<b&&!detect;i++){
        for (j=0;j<bV&&!detect;j++){
          maximo=max(cpcb[i],cpcbV[j]);
          minimo=min(cpcb[i],cpcbV[j]);
          dift=maximo-minimo;
          if (dift>30000)dift=4294967295-maximo+minimo;
          Serial.println(dift);
          //si la diferencia de cruces es menor en tiempo que medio ciclo de
red, procedemos al calculo del desfase
          if (dift<5000){
            cosfi=cos(w*dift/1000000);
            senfi=sin(w*dift/1000000);
            detect=1;
            Serial.println(dift);
            Serial.println(cosfi);
          }
        }
      }
      Serial.println("detect:");
      Serial.println(detect);
      Serial.println(cosfi);
      Serial.println(senfi);
//      Serial.println(s);
//      Serial.println(b);
//      Serial.println(bV);
//      Serial.println(sV);
    }
    else{
      Serial.println("no calculo el cosfi");
    }
    //puedo hacer una segunda pasada esta vez con el flanco de bajada


    //envio de datos a la raspi via comunicacion serie
    //tengo que enviar Irms,Vrms,P, cosfi, t podemos cogerlo en la raspi.
    //P,Q,cosfi
```

```
    typedef union {
    float val;
    uint8_t bytes[4];
    } floatval;

    floatval Pu,Qu,cosfiu;
    S=Irms*Vrms;
    P=S*cosfi;
    Q=S*senfi;
//    P=10000;
//    Q=330;
//    cosfi=1;
    Serial.print("Potencia Real:");
    Serial.println(P);
    Serial.print("Potencia Reactiva:");
    Serial.println(Q);

    //envio datos a la rpi para que los procese y los cuelge a la app
    //P, cosfi, Q
    bridge1.virtualWrite(V11, P,Q,cosfi);
    //time_t t=now();
    //Serial.println(t);
    Pu.val=P;
    Qu.val=Q;
    cosfiu.val=cosfi;

    /*Serial.write(Pu.bytes,4);
    Serial.write(Qu.bytes,4);
    Serial.write(cosfiu.bytes,4);
   */




    //reseteamos variables para un nuevo ciclo de recogida de datos mas envio
    detect=0;
    flag=0;
    sum1=0.0;
    sum2=0.0;
    auxaI=0.0;
    auxaV=0.0;
    auxxxVa=0;
    auxxxIa[1]=0;
    auxxxIa[2]=0;
    auxxxIa[3]=0;
    auxVmax=0.0;
    auxVmin=0.0;
```

```
    i=0;
    j=0;
    l=0;
    s=0;
    sV=0;
    b=0;
    bV=0;

    delay(5000);
    //inhabilito las int
    t4=micros();
    cli();
    adc_init();
    // Initialize the timer
    timer1_init(compareRegister);
    sei();
    Serial.println("nuevo ciclo");




  }
  //30s y de nuevo periodo de lectura
  //supongo que durante esos 30 s el consumo ha sido el de la lectura que se
viene a continuacion




}
-
```

- Código gestión de la APP BLYNK

```
//ultima modificacion 12/01/19
/*
Programa c++ para servidor linux --> probado y compilado en raspbian jessie
rasperry pi 3b
                                  --> probado y compilado en maquina virtual
gcp debian 9 stretch

- Recibe datos desde el mcu linkit a traves de la funcionalidad bridge
- Gestiona la app: actualiza widgets en funcion de los datos recibidos desde
la mcu
                    atiende peticiones desde la app
- Recoge datos de comercializadoras desde la base de datos montada en
servidor linux mediante una peticion get
```

```
- Lleva recuento del consumo en euros, kwh, picos de potencia, potencia
promedio ... del mes en cuestion
- Incluir mas comercializadoras
- Incluir compatibilidad con el mercado regulado
- Incluir tarifas para mayor potencia --->

*/

//v1.2
//potencia activa llega en W no en kw
/*
Mejoras previstas:
-guardar en memoria variables criticas
-

*/


//librerias usadas
#include <iostream>
#include <string>
#include <stdio.h>      /* puts, printf */
#include <time.h>       /* time_t, struct tm, time, localtime */

using namespace std;

using std::string;
using std::getline;
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT stdout

#ifdef RASPBERRY
 #include <BlynkApiWiringPi.h>
#else
 #include <BlynkApiLinux.h>
#endif
#include <BlynkSocket.h>
#include <BlynkOptionsParser.h>

static BlynkTransportSocket _blynkTransport;
BlynkSocket Blynk(_blynkTransport);
#include <BlynkWidgets.h>

//variables globales
BlynkParamAllocated items(128); // list length, in bytes


int i=0;
int com,tar,pmax;
int ok[3];
```

```
void horper(void);
float precios[2][7];
int flag=0,flag2=0,flagdatos=1;//-------------------------flagdatos a 0
int vi=-1;
float precio[2];
float kwhcont=0,kwh=0;
const int ncom=2;
float p,q,cosfi;
const int seg=5;
int contpmax=0;
float kwmedia=0,kw=0;
int c=0;


string nomcom[ncom];
//**funciones asociadas a la app blynk**
//siempre al conectar recojo datos del servidor, BD
BLYNK_CONNECTED(){
  Blynk.virtualWrite(V0,1);


}
//funcion que dispara al acceder usuario a la app
BLYNK_APP_CONNECTED(){

}


//v0 asociar a la solicitud de datos del servidor que tiene la base de datos
BLYNK_WRITE(V0)
{ size_t l;




  //printf("\n");

  //printf(param.asStr());

  //printf("\n");


  string data= param.asStr();
  string data2;
  char test[]="hola";
  int pos1,pos2;
  l=data.length();
  if (l>10){
    //primero voy a extraer el numero de comercializadoras que tengo
    pos1=data.find("*");
```

```
    data2=data.substr(pos1-1,1);
    int ncom=stoi(data2);
    cout<<ncom<<endl;
    cout<<data2<<endl;
    //substraigo numero comercializadora
    data=data.substr(pos1+1);
    for (int i=0;i<ncom;i++){
    for (int j=0;j<8;j++){
      pos1=data.find(";");
      //en este caso pos1 actua como long
      data2=data.substr(0,pos1);
      //en este caso pos1 actua como indice
      data=data.substr(pos1+1);
      cout<<data2<<endl;
      if (j==0){
        nomcom[i]=data2;
        items.add(data2.c_str());
        string val="0";
        val= val + "€";
        Blynk.virtualWrite(V12, "add", i , data2.c_str() , val.c_str());
      }
      if (j!=0){
        precios[i][j-1]=stof(data2);
      }

    }
    }
    Blynk.setProperty(V4, "labels", items);

       for (int i=0;i<ncom;i++){
      for (int j=0;j<7;j++){
        cout<<precios[i][j]<<endl;
      }
    }

  }

}

    //por defecto el primer item es el 1 no el cero
    //selector comercializadora
    BLYNK_WRITE(V4)
    {
      com=param.asInt()-1;
      ok[0]=1;
      cout<<com<<endl;
      //ok++;

    }
```

```
//selector tarifa
BLYNK_WRITE(V5)
{
  tar=param.asInt()-1;
  ok[1]=1;
  //ok++;
  cout<<tar<<endl;
}

//selector pmax contratada
BLYNK_WRITE(V6)
{
  pmax=param.asInt();
  ok[2]=1;
  //ok++;
  cout<<pmax<<endl;

}

//bridge--recoge datos cada 5s de un ciclo de lectura desde el hardware
BLYNK_WRITE(V11)
{ //bridge
  //kwh
  //cosfi
  float p,q,cosfi;
  time_t t;
  //float pmax=param[0].asFloat();
  p=param[0].asFloat();
  q=param[1].asFloat();
  cosfi=param[2].asFloat();
  kw=p/1000.0;
  kwh=kw*seg/3600;
  t=time(0);
  cout<<p<<endl;
  cout<<q<<endl;
  cout<<cosfi<<endl;
  cout<<t<<endl;
  //pmax=param[1].asFloat();
  //cout<<pmax<<endl;
  flagdatos=1;
  //necesito el valor de kwh

}




/*
pos1 = data.find("*");
```

```
    data = data.substr(pos1);
    cout <<data;
    */



    /*pos1 = data.find("*");
    //cadena lista con los datos
    int pos2 = data.find(";");
    string datam = data.substr (pos+1,pos2-pos-1);

    cout<<l;
    printf("entro");
    cout<<datam;*/




  //printf(webhookdata);

  //webhookdata.ignore(256,'*');
  //getline(webhookdata,string data,'*');
  //cout<<webhookdata;
  /*
  for(int i=0;i<webhookdata.lenght();i++){
    if(webhookdata[i]=="*"){
      j=0;
    }

  }*/


/*
BLYNK_WRITE(V1) {
  int value = param.asInt();
  printf("ok");
  if (value == 1) {
    //Serial.println("Item 1 selected");
  } else if (value == 2) {
    // If item 2 is selected, change menu items...
    BlynkParamAllocated items(128); // list length, in bytes
    items.add("Elias");
    items.add("Xele");
    items.add("Sergio");
    Blynk.setProperty(V1, "labels", items);

    // You can also use it like this:
    //Blynk.setProperty(V1, "labels", "item 1", "item 2", "item 3");

  } else {
```

```
    //Serial.println("Unknown item selected");
  }
  cout<<value;
}
*/

//esta funcion va llevando el recuento de precios no solo de tu
comercializadora y tu tarifa sino otras comercializadoras con tarifa del
mismo plan
void horper(){
  float aux;
  time_t rawtime;
  struct tm * timeinfo;

  time (&rawtime);
  timeinfo = localtime (&rawtime);
  //printf ("Current local time and date: %s", asctime(timeinfo));
  int  mes=timeinfo->tm_mon;
  int  hor=timeinfo->tm_hour;
  int  minu=timeinfo->tm_min;
  int  dia=timeinfo->tm_mday;
  mes++;//0-11
  //int  min=timeinfo->tm_min;
  //cout<<mes<<endl<<hor<<":"<<min<<endl;

  //si estamos a dia 1 empieza un nuevo periodo de facturacion
  if (dia==1 && !flag){
    flag=1;
    //actualizo precio total
    for (int j=0;j<ncom;j++){

      precio[j]=precios[j][0]*30*pmax;
      cout<<precio[j]<<endl;
    }

    contpmax=0;
    kwmedia=0;
    c=0;

    kwhcont=0;

  }
  if (dia!=1  && flag){
    flag=0;
  }

  //horario
  if (mes>3 && mes<11){
    //verano
```

```
  vi=0;
  cout<<"verano"<<endl;
}
else{
  vi=1;//invierno
  cout<<"invierno"<<endl;
}

switch(tar){
  case 0:{
    //sin discriminacion horaria
    cout<<"sin discriminacion horaria"<<endl;
    for (int j=0;j<ncom;j++){
      precio[j]=precio[j]+precios[j][1]*kwh;
    }
    //aux=precios[com][1];
  }

  case 1:{
    //DH
    if (vi==0){
      cout<<"verano"<<endl;
      if(hor>=13 && hor<23)//cout<<"punta"<<endl;
        {
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][2]*kwh;
          }
        }
        //aux=precios[com][2];
      else{
          for (int j=0;j<ncom;j++){
            cout<<precios[j][3]*kwh<<endl;

            precio[j]=precio[j]+precios[j][3]*kwh;
          }
      } //aux=precios[com][3];//cout<<"valle"<<endl;//precio[com][3];
    }
    else{
      cout<<"invierno"<<endl;
      if(hor>=12 && hor<22)//cout<<"punta"<<endl;
       {
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][2]*kwh;
          }
       }
        //aux=precios[com][2];
      else //aux=precios[com][3];//cout<<"valle"<<endl; //precio[com][3];
      {
          for (int j=0;j<ncom;j++){
```

```
            precio[j]=precio[j]+precios[j][3]*kwh;
          }
      }
    }
  }

  case 2:{
    //DHS
    if (vi==0){
      cout<<"verano"<<endl;
      if(hor>=13 && hor<23){
        cout<<"punta"<<endl;
        //aux=precios[com][4];
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][4]*kwh;
          }
      }


      else if(hor>=1 && hor<7){
        cout<<"sv"<<endl;
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][6]*kwh;
          }
        //aux=precios[com][6];
      }

      else{
        cout<<"v"<<endl;
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][5]*kwh;
          }
        //aux=precios[com][5];
      }
    }
    else{
      cout<<"invierno"<<endl;
      if(hor>=13 && hor<23){
        cout<<"punta"<<endl;
        //aux=precios[com][4];
          for (int j=0;j<ncom;j++){
            precio[j]=precio[j]+precios[j][4]*kwh;
          }
      }


      else if(hor>=1 && hor<7){
        cout<<"sv"<<endl;
          for (int j=0;j<ncom;j++){
```

```
              precio[j]=precio[j]+precios[j][6]*kwh;
            }
          //aux=precios[com][6];
        }

        else{
          cout<<"v"<<endl;
            for (int j=0;j<ncom;j++){
              precio[j]=precio[j]+precios[j][5]*kwh;
            }
          //aux=precios[com][5];
        }
      }
    }
    }
    string val;
    for (int j=0;j<ncom;j++){
      val=to_string(precio[j]);
      int p=val.find(".");
      val=val.substr(0,p+3);
      cout<<p<<endl;
      cout<<nomcom[j]<<endl;
      val=val + "€";
      Blynk.virtualWrite(V12, "update", j, nomcom[j].c_str(),   val.c_str());
    }

    //solo actualizo el precio asociado a mi comercializadora
    Blynk.virtualWrite(V7,precio[com]);

    Blynk.virtualWrite(V9,precio[0]);
    //return(aux);
}




void setup()
{
  printf("ok");
  Blynk.virtualWrite(V0, "newfarm.hopto.org/mysql2.php");
  // You can perform HTTPS requests even if your hardware alone can't handle
SSL
  // Blynk  can also fetch much bigger messages,
  // if hardware has enough RAM (set BLYNK_MAX_READBYTES to 4096)
  //Blynk.virtualWrite(V0, "https://api.sunrise-
sunset.org/json?lat=50.4495484&lng=30.5253873&date=2016-10-01");
}

void loop()
```

```
{
  //kw=rand() % 10 + 1;
  //kwh=rand() % 10 + 1;
  Blynk.run();
  //para el mes en el que estoy necesito saber si horario verano e invierno
  //solo si tengo discriminacion horaria??
  //el periodo del dia en el que estoy



  //si tengo todos los datos necesarios del usuario
  if(ok[0] && ok[1] && ok[2]){
    //nueva confifuracion
    //aqui tendria que calcular el precio fijo por potencia
    //considero 30 dias el mes
    for(int j=0;j<ncom;j++){
      precio[j]=precios[j][0]*30*pmax;
      cout<<"precio por potencia al mes:";
      cout<<precio[j]<<endl;
    }

    Blynk.virtualWrite(V8,precio[com]);

    kwhcont=0;
    contpmax=0;
    kwmedia=0;
    c=0;
    flag2=1;
    ok[0]=0;
    ok[1]=0;
    ok[2]=0;
  }


  if (flag2 && flagdatos){
    /*float aux=horper();
    cout<<aux<<endl;
    precio=precio+aux*kwh;
    cout<<precio<<endl;

    Blynk.virtualWrite(V7,precio);*/
    //calculo de precios en t real
    horper();
    kwhcont=kwhcont+kwh;
    Blynk.virtualWrite(V1,kwhcont);
    //recuento de la potencia media del mes
    //contador de picos de consumo y recojo el momento del dia
    if(kw>=pmax)contpmax++;
    kwmedia=kwmedia+kw;
    c++;
```

```
    Blynk.virtualWrite(V10,contpmax);
    Blynk.virtualWrite(V11,kwmedia/c);
    cout<<kw<<endl;
    flagdatos=0;
  }
 // delay(5000);
  cout<<"nuevo periodo"<<endl;
  //Blynk.virtualWrite(V0,HIGH);
  //Blynk.virtualWrite(V0,1);

}

int main()
{

  string mensaje;
  mensaje= "hola";
  //printf(mensaje);
  cout<<mensaje<<endl;
  Blynk.begin("a89a726f2bad4e22be7a64e95b7b5c86", "newfarm.hopto.org", 8080);
  setup();

  while(true) {
    loop();

  }

  return 0;
}
```

- Códigos PHP

- **Subida de datos hacia el servidor**

```
//codigo php usado para bajar datos desde el servidor mediante una
consulta usando el metodo get, se descargan los datos asociados a
las distintas comercializadoras
 <?php
$servername = "localhost";
$username = "root";
$password = "ee2669";
$dbname = "em";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT
comercializadora,ekwdia,ekwh,ekwhp,ekwhv,ekwhpsv,ekwhvsv,ekwhsv FROM
p10";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    echo ("#");
    echo ($result->num_rows);
    echo ("*");
    while($row = $result->fetch_assoc()) {
        //echo "id: " . $row["id"]. " - Name: " . $row["firstname"].
" " . $row["lastname"]. "<br>";

        echo ($row["comercializadora"]);
        echo (";");
        echo ($row["ekwdia"]);
        echo (";");
        echo ($row["ekwh"]);
        echo (";");
        echo ($row["ekwhp"]);
        echo (";");
        echo ($row["ekwhv"]);
        echo (";");
        echo ($row["ekwhpsv"]);
        echo (";");
        echo ($row["ekwhvsv"]);
        echo (";");
        echo ($row["ekwhsv"]);
        echo (";");


   }
} else {
    //echo "0 results";
}
echo("*");
echo(";");
$conn->close();

?>
```

- **Bajada de datos desde el servidor mediante una peticion GET**

```
//codigo php para salvar datos procedentes desde el hardware en la base de datos montada en el servidor
//esta preparada para salvar datos asociados a otra aplicacion, facilmente editable para usarla en cualquier otra app, esquema modular
<?php
//variables asociadas a la base de datos donde se almacenan los datos procedentes del esp32
$servername = "localhost";
$username = "root";
$password = "ee2669";
$dbname = "newfarm";


// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

//estructura de los datos desde el esp32
//marko,nmod,  n , t1, t2,h2,p,planta[nmod],ini[nmod],riego
//total 8 datos + nmod*2

$marko=$_GET["marko"];
$nmod=$_GET["nmod"];
$n=$_GET["n"];
echo $marko;
echo $nmod;
echo $n;

$get=array("t1","t2","h2","p");
for ($x=0;$x<2;$x++){
    for ($y=0;$y<$nmod;$y++){
    if($x==0){
        $get[]="planta".(string)$y;
        echo "ok1";
    }
    else{
        $get[]="ini".(string)$y;
        echo "ok2";
    }
    echo "*";
    }

}
$get[]="riego";
```

```
for ($x=0;$x<(5+$nmod*2);$x++){
    echo $get[$x];
}

echo $n;
//habria que hacer modular el tema de plantax e inix
$sql = "INSERT INTO marko1 (t1, t2,
h2,p,planta0,planta1,planta2,planta3,planta4,ini0,ini1,ini2,ini3,ini
4,riego)VALUES(";
for($x=0;$x<(5+$nmod*2);$x++){
    $sql.="'";
    $sql.=(string)$_GET[$get[$x]];
    $sql.="'";
    if ($x<($nmod*2+5-1)){
    $sql.=",";
    }
    echo $_GET[$get[$x]] . "<br>";
}
$sql.=")";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

- **Otros**

```
//ultima modificacion 12/01/19
//lista de comandos utiles usados en la shell de Linux, en mi caso usando
debian 9 Stretch y raspbian jessie

    cd directorio --> acceder a carpeta
    cd direccion raiz hacia directorio --> acceder a carpeta no visible desde
la ruta actual
    ls  --> listar archivos directorio actual
    mkdir nombre_carpeta --> crear directorio
    rm -Rf nombre_directorio --> eliminar todo el directorio al completo
    df --> cantidad de espacio libre o disponible en los sistemas de archivos
    mysql -u nombreusuario -p --> acceder al gestor de base de datos mysql
    nano --> editor de texto plano
    clear --> limpiar pantalla de comandos
```

```
    compilar usando gcc --> mejor seguir guia para compilar archivo c++
usando librerias blynk:
    https://community.blynk.cc/t/using-c-on-a-raspberry-pi-with-blynk/11864

    instalar mysql: Server version: 5.6.42 --> mejor seguir guia de
instalacion para debian9:
    https://www.digitalocean.com/community/tutorials/how-to-install-the-
latest-mysql-on-debian-9

    para obtener la version 5.6 en especifico:
    https://tecadmin.net/install-mysql-server-on-debian9-stretch/
    https://linuxconfig.org/how-to-install-mysql-community-server-on-debian-
9-stretch-linux


//directorios habiles dentro de la maquina virtual - instancia
    /var/www/html --> codigos php, envio de datos hacia servidor y recogida
desde el mismo
    /home/mardomelias/blynk_lib_cpp_2 --> codigo main.cpp para gestion de la
app blynk, librerias y dependencias


//guia - manual para usar mysql usando comandos
    https://www.w3schools.com/sql/

    //detalle de niveles mysql
    database --> table --> values
    ...

    //comandos basicos mas usados
    show databases; --> mostrar lista base de datos
    use nombre_database; --> seleccionar base de datos
    show tables; --> listar tablas asociadas a base de datos
    select * from nombre_table; --> seleccionar todo el contenido de la tabla
y listarlo
    insert into nombre_table (nombre_var1,nom_var2...) values (valor_v1,
valor_v2...); --> insertar dentro de los campos seleccionados valores
marcados
    describe nombre_table; --> describir tabla
    drop table nombre_table; --> borrar tabla
```

REFERENCIAS

[1] [En línea]. Available: https://www.seeedstudio.com/LinkIt-Smart-7688-Duo-p-2574.HTML.

[2] [En línea]. Available: https://www.raspberrypi.org/products/#buy-now-modal.

[3] «Amazon Prime,» [En línea]. Available: https://www.amazon.es/Yeeco-salida-transformador-corriente-alterna/dp/B016DEUEZM/ref=sr_1_2?ie=UTF8&qid=1547417370&sr=8-2&keywords=sct013.

[4] «Imagenes RPI,» [En línea]. Available: https://www.raspberrypi.org/downloads/.

[5] «Descargables Linkit Smart 7688 Duo,» [En línea]. Available: https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/downloads.

[6] «Tutorial cargar imagen manual sobre la MPU,» [En línea]. Available: https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/tutorials/firmware-and-bootloader/update-the-firmware-with-a-usb-drive.

[7] «Instalar paquete Linkit Smart 7688 Duo IDE arduino,» [En línea]. Available: https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/get-started/get-started-with-the-linkit-smart-7688-duo-development-board/install-arduino-ide-with-board-support-package.

[8] [En línea]. Available: https://community.BLYNK.cc/t/using-c-on-a-raspberry-pi-with-BLYNK/11864.

[9] «Local Server BLYNK,» [En línea]. Available: https://github.com/BLYNKkk/BLYNK-server.

[10] [En línea]. Available: https://cloud.google.com/?hl=es.

[11] «Montar Servidor LAMP RPI,» [En línea]. Available: https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-MYSQL-PHP.

[12] «Montar servidor LAMP debian9,» [En línea]. Available: https://cloud.google.com/community/tutorials/setting-up-LAMP.

GLOSARIO