Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema integral para apoyo a la comunicación personal en pacientes con diversidad funcional motora

Autor: Francisco Beltrán Catalán

Tutor: Hipólito Guzmán Miranda

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

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema integral para apoyo a la comunicación personal en pacientes con diversidad funcional motora

Autor:
Francisco Beltrán Catalán

Tutor:
Hipólito Guzmán Miranda
Profesor Contratado Doctor

Dep. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Sistema integral para apoyo a la comunicación personal en pacientes con diversidad funcional motora

Autor: Francisco Beltrán Catalán

Tutor: Hipólito Guzmán Miranda

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis amigos

A mi difunto tío José Manuel

Resumen

El objetivo de este trabajo es ayudar a pacientes con problemas de movilidad reducida a comunicarse con otras personas de su entorno utilizando elementos accesibles y económicos. La motivación surge a raíz de que, habitualmente, se tiende a aprovechar los recursos de los que se dispone para hacer especial énfasis en la prevención de accidentes y enfermedades**.** Sin embargo, no siempre se pueden evitar, pues muchos de ellos se escapan del control del ser humano y los daños que pueden causar a veces resultan irreparables, dando lugar a afecciones y enfermedades como ELA (Esclerosis Lateral Amiotrófica) o parálisis, e incluso a cuadros patológicos determinados como el síndrome de enclaustramiento. Entre los síntomas de estas afecciones se encuentra la gran dificultad para establecer un medio de comunicación con el entorno, principalmente debido a las elevadas limitaciones de movimiento que tienen. Esta causa motivó a científicos e ingenieros para buscar soluciones a este problema, dando lugar a numerosas y diversas opciones que suponen un avance importante en el sector de la ingeniería biomédica. No obstante, la mayoría de los recursos ofrecidos son fruto de investigaciones y no tienen un carácter comercial, ya que un producto sanitario debe ser estudiado específicamente para cada persona. El motivo reside en que la percepción del mundo de cada paciente puede ser muy diferente y lo que para unos podría resultar cómodo, para otros podría tornarse al disgusto. En adición, aquellos elementos que se comercializan sufren la tendencia de ser excesivamente caros, ante lo cual resulta inevitable plantearse la siguiente cuestión: ¿De qué sirve crear herramientas de accesibilidad y/o de apoyo para los pacientes si, económicamente, resultan inaccesibles para la inmensa mayoría de la población? En este proyecto se plantea una solución estándar y económica que permite a este tipo de pacientes poder comunicarse utilizando herramientas básicas, rentables y conocidas, así como generar accesibilidad a unos dispositivos tan extendidos como son los smartphones.

Abstract

The aim of this work is to help patients with mobility problems to communicate with other people around them using accessible and affordable items. Usually, we tend to take advantage of the resources that are available for special emphasis on prevention of accidents and diseases. However, they cannot always be avoided, because many of them are beyond the control of human beings and the damage they can cause is sometimes irreparable, leading to afflictions such as ALS (Amyotrophic Lateral Sclerosis) or paralysis, even locked-in syndrome. One of the symptoms of these afflictions is the great difficulty in establishing a communication medium with the environment, mainly due to movement limitations. This situation motivated scientists and engineers to find solutions to this problem, resulting in numerous and various options involving major advancements in the field of biomedical engineering. However, most of the resources offered are the result of research and do not have a commercial character, as a medical device must be studied specifically for each person. The reason is that the world's perception of each patient can be very different and what for some could be comfortable, could turn others to disgust. In addition, those elements that are marketed tend to be too expensive, so I pose a question for this reason: What good is to create accessibility tools and / or support for patients if they are economically inaccessible to the vast majority of the population? This project proposes a standard and economical solution that allows these patients to communicate using basic, cheap and familiar tools and improve accessibility with widespread devices such as smartphones.

Índice

Índice de Tablas

Índice de Figuras

1 INTRODUCCIÓN

Con el paso del tiempo, la evolución de las tecnologías ha aportado a la sociedad numerosas herramientas que nos han permitido aumentar nuestra calidad de vida en prácticamente todos los aspectos técnicos conocidos, desde las comunicaciones, las industrias o la medicina hasta el deporte, la automoción o incluso el ocio en sí.

Habitualmente, se tiende a aprovechar los recursos de los que se dispone para hacer especial énfasis en la prevención de accidentes y enfermedades. De este modo, el entorno social se convierte en un medio más seguro y las probabilidades de sufrir cualquier tipo de incidente se reducen de forma considerable. Por ejemplo, existen medidas, ejercicios y material de apoyo que, utilizados adecuadamente, permiten mejorar la salud física y mental de una persona para evitar (o en su defecto, posponer) la aparición de enfermedades cuya cura aún no ha sido descubierta. Acorde a lo descrito, podría decirse que uno de los refranes más arraigados a estos hechos es que "*más vale prevenir que curar"*.

Sin embargo, no siempre se pueden evitar accidentes, pues muchos de ellos se escapan del control del ser humano y los daños que pueden causar a veces resultan irreparables. Estas víctimas no deben pasar inadvertidas ya que, aunque existan y se estudien diferentes vías para ayudarlas, hay casos en los que no se ha llegado a profundizar lo suficiente en el remedio que corresponde. A modo de ejemplo, encontramos gente con enfermedades degenerativas que, a una determinada edad, le resulta prácticamente imposible poder comunicarse con otras personas de su entorno, debido a su movilidad escasa o prácticamente nula. ¿Qué métodos existen para solventar el problema? ¿Realmente son accesibles para los pacientes? ¿Su uso es sencillo de cara al usuario? ¿Resultan rentables ante una mayoría poblacional?

De acuerdo a la situación del paciente, cada uno tendrá unas necesidades que cubrir según el tipo de síntomas que presente su enfermedad o discapacidad. En concreto, y acorde al ejemplo descrito en el párrafo anterior, uno de los problemas más destacables en este contexto es la dificultad que se presenta a la hora de establecer una comunicación con personas del entorno, pues la limitación de movimientos implica un mayor impedimento al articular palabras. Esta afección cobra especial relevancia si se tiene en cuenta que nos encontramos en el apogeo de la era tecnológica, donde la comunicación juega un papel importante en la sociedad y se sitúa como uno de los grandes pilares de las necesidades humanas actuales.

Acorde a lo comentado en esta sección, en este documento se describe una solución que aporta apoyo a la comunicación personal, un sistema para mejorar la capacidad de comunicación en los pacientes con problemas de movilidad con independencia del origen de sus limitaciones y de la extensión temporal que abarquen, con el objetivo de facilitar la comunicación aportando un recurso sencillo y suficientemente económico que, en adición, genere accesibilidad a las nuevas tecnologías.

2 ENFERMEDADES INFLUYENTES

2.1 A qué pacientes va dirigido

Para evitar posibles malentendidos o confusiones, a continuación se reflejan algunas enfermedades cuyos síntomas aportan limitaciones en este contexto, con una breve descripción adjunta de las mismas para corroborar si realmente los pacientes que la sufren necesitarían o no de estas prestaciones:

- Tetraplejia: ✔

  La tetraplejia es una lesión en la parte superior e inferior del cuerpo que limita considerablemente la movilidad. La parte superior no tiene por qué perder la movilidad al 100%; sin embargo, la zona de la cabeza se mantiene operativa y es capaz de poder mantener conversaciones y hablar sin necesidad de realizar ningún movimiento, por lo que este tipo de pacientes no necesitarían este sistema inicialmente para llevar a cabo una comunicación; sin embargo, podría verse beneficiado por la funcionalidad correspondiente a la accesibilidad. Debido a esto, los tetrapléjicos son pacientes que pueden sacar beneficios de su uso.

- Paraplejia: ✗

  Consiste en una lesión en la parte inferior del cuerpo. Situación similar a la de los tetrapléjicos, pero con menor limitación de movimientos. Este tipo de pacientes no necesitarían utilizar el sistema propuesto.

- Hemiplejia: ✔

  Produce una lesión en un hemisferio del cerebro que cursa con parálisis del brazo y pierna contrarias al hemisferio dañado, afectando significativamente a personas mayores y, en muchas ocasiones, a la mitad de la cara del paciente afectado. Este tipo de dificultades podrían sobrellevarse con una herramienta de comunicación y accesibilidad.

- Parálisis cerebral: ✔

  La parálisis cerebral es un trastorno permanente y no progresivo que causa una limitación de la actividad de la persona, atribuida a problemas en el desarrollo cerebral del feto o del niño. Los desórdenes psicomotrices de la parálisis cerebral están a menudo acompañados de problemas sensitivos, cognitivos, de comunicación y percepción, y en algunas ocasiones, de trastornos del comportamiento. Como se puede comprobar, aquí sería necesaria la herramienta que proponemos en el proyecto.

- Esclerosis múltiple: ✗

  Los síntomas más frecuentes son los siguientes: trastornos visuales, problemas de equilibrio, problemas en el habla, temblor en las manos, debilidad en los miembros, pérdida de fuerza… Normalmente, la Esclerosis Múltiple se detecta tras un primer brote de la enfermedad. Los síntomas de este primer brote son muy variados, pero entre los más fácilmente reconocibles, destacan hormigueo, debilidad, falta de coordinación (ataxia), alteraciones visuales, rigidez muscular, trastornos del habla (disartria), andar inestable, entre otros. También tiene síntomas cognitivos como problemas en la memoria o en la codificación y recuperación de la información. Por último decir que la EM produce también problemas en los aspectos emocionales tales como ansiedad o depresión.

  Las dificultades de estos pacientes son demasiado severos como para que este proyecto les resultara suficientemente útil.

- ELA / ALS (Esclerosis Lateral Amiotrófica): ✔

  Es una enfermedad exclusivamente neuromotora, con lo cual, se ven afectados la mayor parte de músculos bajo control voluntario y los pacientes pierden su fuerza para mover brazos y piernas. Las neuronas de los músculos oculares no se ven afectadas. También con el desarrollo de la enfermedad, los pacientes pueden llegar a tener problemas en la articulación de palabras e incluso en la deglución, aunque lo más grave es la disfunción de los músculos respiratorios, ya que el paciente puede llegar a fallecer por esta causa.

  Claramente, aquí también se necesitaría el uso del sistema.

- Enfermedades degenerativas que conlleven la reducción de movilidad con el paso del tiempo y, por tanto, riesgo a sufrir problemas de habla. ✔

- Pacientes varios que necesiten temporal o permanentemente de dicha herramienta (accidentes de coche, sometimiento a quimioterapias, etc.) ✔

Todas estas personas tienen el problema en común de que no pueden o les es muy complicado poder comunicarse con los de su entorno, desde familiares hasta enfermeros/as de los hospitales. Es por ello que el uso de una herramienta que pudiera cubrir una de sus necesidades más básicas se convierte en algo interesante de estudiar y desarrollar.

2.2 Necesidades del paciente

Finalmente, se concluye que estos pacientes poseen necesidades importantes en común tales como:

- Comunicación mediante sistema electrónico debido a grandes incapacidades de movilidad y dificultad para hablar o vocalizar correctamente.
- Aportar la posibilidad de utilizar smartphones para fomentar el ocio y aprovechar al máximo las herramientas disponibles y empleadas para generar accesibilidad a las nuevas tecnologías.

3 ESTUDIO DEL ESTADO DEL ARTE

Acorde a lo descrito en la sección anterior, se procede a indicar diferentes herramientas, medios y mecanismos, comerciales y de investigación, con las que se está trabajando en pos de ayudar a los pacientes a poder utilizar un medio de comunicación.

3.1 Programas de tecnología asistida para ayudar a la comunicación y a la participación de ocio de las personas con esclerosis lateral amiotrófica: Dos estudios de caso único

En este artículo (Lancioni et al., 2012 [14]) se estudian dos casos concretos, donde los pacientes sufren esclerosis lateral amiotrófica y se aplican programas de tecnología asistida para la comunicación. Para el primero, se buscó dar la posibilidad de escribir y enviar mensajes de texto, así como establecer videollamadas con sus hijos; para el segundo, se buscó el que fuera partícipe de diferentes actividades de ocio, además de poder realizar solicitudes relacionadas con necesidades básicas.

El nuevo programa de tecnología asistida para el primer paciente involucró a dos sistemas relacionados con la informática (que sirvieron para la mensajería de texto y videollamada), dos monitores (es decir, un monitor principal y un monitor secundario), un microinterruptor óptico, y una interfaz de conexión del microinterruptor con los sistemas informáticos . El microinterruptor óptico era idéntico al utilizado para un programa más antiguo basado en la tecnología a la que daba uso anteriormente y consistía en un diodo emisor de luz infrarroja con una pequeña unidad de detección de luz infrarroja fija a un lado de la cabeza del paciente, que se activaba mediante una mínima y ligera inclinación de la cabeza. El microinterruptor también era adecuado para utilizar el parpadeo como método de respuesta, pero el paciente prefirió escoger la primera opción.

Para el segundo caso, los elementos utilizados fueron prácticamente los mismos (dos sistemas relacionados con la informática que sirvieron para la participación de ocio, elaboración de solicitudes y para la mensajería de texto, un monitor, un microinterruptor óptico y una interfaz que conectaba el microinterruptor con los sistemas informáticos). El microinterruptor óptico es exactamente el mismo que el empleado para el primer paciente, aunque en este caso se implementó debajo de la barbilla, utilizando dicha zona para comunicarse.

3.2 Dispositivo de ortografía para pacientes con parálisis

Birbaumer et al. (1999 [3]) desarrollaron un nuevo medio de comunicación para pacientes con parálisis completa que utiliza los potenciales corticales lentos (SCPs) del electroencefalograma para accionar un dispositivo de ortografía electrónico. Se conformó el control voluntario de los SCP en dos pacientes de Locked-in (síndrome de enclaustramiento) con esclerosis lateral amiotrófica avanzada que eran capaces de aprender a utilizar un dispositivo de ortografía mediante el control de sus respuestas cerebrales. El dispositivo de ortografía maneja un cursor en una pantalla de vídeo, lo que permite a los sujetos seleccionar las letras del alfabeto.

3.3 Una interfaz hombre-máquina utilizando la simetría entre los ojos para detectar la dirección de la mirada

Existen casos de parálisis muy grave donde la capacidad de una persona para controlar el movimiento se limita al uso de los músculos de alrededor de los ojos, el movimiento de éstos o el uso del parpadeo, siendo el único medio que poseen para poder comunicarse. Las interfaces que ayudan en la comunicación son a menudo intrusivas, requieren hardware especial o se basan en iluminación de infrarrojos activa. Por ello, Magee, Betke, Gips, Scott y Waber (2008 [16]) desarrollaron un sistema de interfaz de comunicación no intrusiva llamada EyeKeys, que se ejecuta en un equipo a nivel de consumidor con la entrada de vídeo desde una cámara de bus serie universal barato, el cual funciona sin necesidad de una iluminación especial. El sistema detecta y escanea el rostro de la persona que utiliza la correlación de plantilla a múltiples escalas. La simetría entre los ojos izquierdo y derecho se explota para detectar si la persona está mirando a la cámara o al lado izquierdo o derecho. La dirección del ojo detectado a continuación se puede utilizar para controlar las aplicaciones tales como los programas de ortografía o incluso algunos juegos.

3.4 WISION–Sistema de Interfaz inalámbrica de la Interpretación de símbolos oculares en personas con enfermedades neuromusculares

En este trabajo, Ramesh, Asok Nair y Menon (2011 [17]) propusieron el diseño de un sistema que permitiera a las personas seriamente discapacitadas comunicarse usando sus señales EOG (electrooculográficas), EMG (electromiográficas) y EEG (electroencefalográficas). El enfoque típico es modificar los dispositivos finales para interpretar estas señales y hacerlas funcionar en consecuencia. Sin embargo, en este caso se utiliza un Sistema Intérprete en el que estas señales son decodificadas e interconectadas con los dispositivos existentes. La ventaja es que los dispositivos finales no tienen que ser cambiados para el usuario, ya que la lógica requerida para la interpretación está dentro del Sistema intérprete. Por lo tanto, actúa como una interfaz a través de la cual el usuario puede controlar múltiples dispositivos al mismo tiempo. Estos sistemas pueden capturar y procesar señales de EOG , EMG y EEG y transmitirlos al sistema intérprete a través de un módulo Bluetooth de baja energía. Esta metodología ayudará a los usuarios a interactuar con varios dispositivos con un número limitado de símbolos.

3.5 Un microinterruptor sensible a la voz para un hombre con esclerosis lateral amiotrófica y alteración motora generalizada

El objetivo de este estudio es evaluar un micro de voz sensible para un hombre de 67 años de edad con esclerosis lateral amiotrófica, quien tuvo continuas dificultades para utilizar un microinterruptor óptico a través de los movimientos pequeños de mentón (Lancioni et al., 2013 [15]). Para ello, el paciente utiliza los microinterruptores en combinación con un programa asistido por ordenador para dos cosas: seleccionar canciones y videos de acceso por preferencia o hacer peticiones, y manipular un sistema de mensajería para comunicarse con su esposa y sus hijos. Para evaluar el desempeño del paciente con los dos microinterruptores, se alternan sesiones con un microinterruptor u otro.

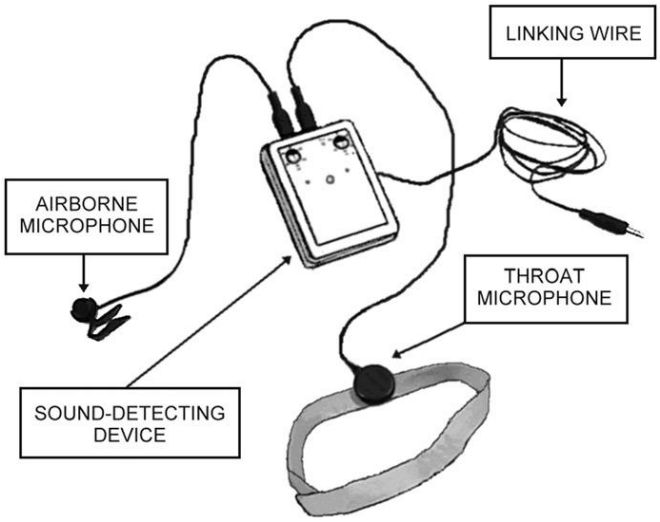

Figura 1. Sistema con microinterruptor utilizado en el estudio.

El nuevo microinterruptor consistió en un dispositivo de detección de sonido con un micrófono en la garganta (que tuvo lugar en la laringe de Paul con una simple banda para el cuello) y un micrófono en el aire (colocado bajo su barbilla con un pequeño gancho). Los dos micrófonos utilizan canales de entrada separados cuyos umbrales de activación podían ser regulados de manera independiente. Este microinterruptor, que era una versión mejorada de una prueba realiza por Lancioni et al. con una persona con discapacidad múltiple, permitió a Paul elegir entre las opciones disponibles del programa con una pequeña emisión de voz de unos 55 decibelios. La activación del sistema se produce sólo si ambos micrófonos se activan simultáneamente. Esto es un requisito para asegurar que no se producen falsos positivos debido a la activación accidental de uno de los micrófonos (es decir, la activación accidental del micrófono de garganta a través de un movimiento de la cabeza-cuello distónica o la activación accidental del micrófono de aire a través de voz no intencionado, ruido ambiental…). El rendimiento del hombre con el nuevo microinterruptor era aparentemente más eficiente (es decir, que en general podría activar el nuevo microinterruptor con un solo intento, mientras que necesitaba una media de alrededor de dos intentos para cada activación del microinterruptor óptico). El hombre también ha desarrollado una clara preferencia por el nuevo microinterruptor.

Un microinterruptor que se puede utilizar de manera eficiente y sin esfuerzo es de importancia crítica para una persona con ALS.

3.6 Interfaz hombre-máquina para potenciar la comunicación facial no verbal basada en señales EEG relacionadas con las emociones específicas

A diferencia de la tecnología de asistencia para la comunicación verbal, la interfaz cerebro-máquina (en inglés BMI o Brain Machine Interface) o cerebro-computador (en inglés BCI o Brain Computer Interface) no se ha establecido como una herramienta de comunicación no verbal para la esclerosis lateral amiotrófica (ELA). La comunicación “face to face” permite el acceso a una información emocional considerable, pero las personas que sufren de trastornos neurológicos, como la ELA o el autismo, no pueden expresar sus emociones o comunicar sus sentimientos negativos con tanta facilidad. Aunque las emociones pueden ser inferidas al ver las expresiones faciales, la predicción emocional para las caras neutrales requiere un juicio avanzado. Para hacer frente a este problema, en este estudio se intentó decodificar reacciones emocionales condicionadas a los estímulos a raíz de una cara neutral (Kashihara, 2014 [11]). Esta idea fue motivada por la suposición de que, si las señales del electroencefalograma

(EEG) se pueden utilizar para detectar las respuestas emocionales de los pacientes a partir de caras inexpresivas específicas, los resultados podrían ser incorporados en el diseño y desarrollo de herramientas de comunicación no verbales basados en BCI/IMC, por lo que se desarrolló un método eficiente para detectar respuestas emocionales negativas implícitas a partir de caras específicas mediante el uso de señales de EEG. Un método de clasificación basado en una máquina de vectores como soporte permite la fácil clasificación de rostros neutrales que desencadenan dichas emociones individuales específicas, de forma que con esta clasificación, una cara en un ordenador se transforma en un semblante triste o disgustado.

3.7 Interfaces hombre-máquina invasivas: un estudio de actitudes, conocimientos y métodos de recuperación de información para pacientes con parálisis

Las interfaces cerebro-máquina son una opción terapéutica emergente que pueden permitir a los pacientes paralizados obtener el control sobre los dispositivos de tecnología asistida (en inglés Assistive Technology Devices o ATD). Los enfoques de BMI se pueden clasificar en general en invasivos (basado en electrodos implantados intracranealmente) y no invasivos (basado en electrodos de la piel o sensores extracorpóreos). Las BMI invasivas tienen una relación favorable entre señal y ruido, y por lo tanto permiten la extracción de más información que el método no invasivo, aunque están asociados a los riesgos relacionados con la implantación del dispositivo de neurocirugía. Estudios recientes han investigado las actitudes de los pacientes paralizados elegibles para el BMI (Lahr et al., 2015 [13]), en especial los pacientes afectados por esclerosis lateral amiotrófica. No obstante, no está claro aún si los pacientes paralizados aceptarían la implantación intracraneal de electrodos de IMC con la premisa de mejoras de decodificación, y a qué gama de pacientes con enfermedades como la apoplejía o lesión de la médula espinal va dirigido este nuevo tipo de tratamiento. Este estudio reveló que la mayoría de los pacientes conocía y mostraba una actitud positiva hacia los enfoques de IMC invasivos. La aceptación de la tecnología IMC invasiva dependía de las mejoras que se esperan de la tecnología. Por otra parte, la encuesta reveló que para los pacientes paralizados, Internet es una fuente importante de información sobre ATD. Por ello, los sitios web adaptados a los usuarios potenciales de IMC deberían desarrollarse más para proporcionar información fiable a los pacientes, y también para ayudar a vincular a los usuarios potenciales de IMC con los investigadores involucrados en el desarrollo de la tecnología de IMC.

3.8 Comunicación *"Point-and-click"* neuronal para una persona con síndrome de enclaustramiento incompleto

Nota: "Point-and-click" o "apuntar y hacer clic" hace referencia a un ratón de pantalla diseñado para aquellas personas que no pueden hacer clic con un ratón físico.

Uno de los objetivos de la investigación en la interfaz cerebro-ordenador es desarrollar medios rápidos y fiables de comunicación para las personas con parálisis y anartria (Bacher et al., 2014 [2]). Se evaluó la capacidad de un individuo con síndrome de enclaustramiento incompleto para comunicarse mediante los nervios, apuntando y haciendo clic con el botón de control. Se desarrolló una interfaz para proporcionar un control de un cursor de ordenador en conjunto con uno de los dos teclados virtuales que se ofrecen en una pantalla. El participante utiliza esta interfaz para comunicarse cara a cara con el personal de investigación mediante el uso de conversión de texto a voz, y de forma remota mediante una aplicación de chat de Internet. Este estudio demuestra el primer uso de una interfaz cerebro-ordenador cortical para la comunicación de apuntar y hacer clic con el botón neuronal por un individuo con síndrome de enclaustramiento incompleto.

3.9 OTTAA (One Touch Talk Assistive App)

Una aplicación diseñada por estudiantes de la universidad de Córdoba (Argentina) para ayudar a personas discapacitadas y con trastornos de autismo o alzhéimer a poder comunicarse.

Figura 2. Interfaz software de OTTAA.

Tal y como describe Argento (2015 [1]), la redacción de La Voz (2015 [18]) y la Universidad Blas Pascal (2014 [10]), OTTAA ayuda a las personas a comunicarse de una forma más rápida y efectiva, logrando frases más ricas y completas en comparación con las tarjetas físicas e ilustraciones convencionales. No sólo aporta ventajas para las personas con autismo, sino para todo aquel que presente dificultades para interpretar el entorno en el que se encuentra, como por ejemplo, hora del día, clima, ubicación o agenda. OTTAA hace esta interpretación automáticamente y la pone al servicio de una mejor comunicación.

El funcionamiento actual se basa en la selección táctil de pictogramas gráficos universales creados por la ARASAAC (portal Aragonés de Sistemas Aumentativos y Alternativos de Comunicación) que representan acciones u objetos, con los cuales el usuario va construyendo una frase para luego sonorizarla por medio del sintetizador de voz que posee el dispositivo. Además, los diseñadores de la aplicación están trabajando en conjunto con la Universidad Blas Pascal para desarrollar la posibilidad de controlarla sólo con los gestos de la cara, de modo que se pudiera incluir a más personas en el conjunto de usuarios de OTTAA, como por ejemplo aquellas que padecen de ELA.

Esta aplicación se puede encontrar de forma gratuita en Play Store, la tienda oficial de Google ([26]).

3.10 Sistema con lectura encefalográfica incorporada

Se basa en una tecnología que decodifica las señales eléctricas del cerebro llamadas electroencefalografías (EEG) al emplear la interfaz Emotiv EPOC, un dispositivo diseñado originalmente como accesorio periférico para consolas de videojuegos. Esta herramienta permite al usuario controlar el desarrollo del juego con el pensamiento. También se implementó un software para convertir las señales de EEG a comandos mentales y después a emulaciones de pulsos del teclado de una computadora personal, donde se acciona la reproducción sonora de las palabras “sí” y “no”, dependiendo de la acción mental invocada por el paciente. Todo esto forma parte del comunicado que la entidad “Ciudadanos en Red” publicó en su web, siendo retransmitido por la comunidad “Mientras tanto en México” ([6]). Klose (2007 [12]) basó su estudio en un sistema similar al descrito en esta sección, por lo que se adjunta también la fuente del estudio para poder establecer comparativas.

3.11 Sistema de Interacción y Comunicación Alternativa Multi-Dispositivo por Seguimiento Ocular y Facial de Bajo Coste (INTERAAC)

Proyecto de investigación del CSIC (2016 [7]) que busca ayudar a personas con movilidad reducida y dificultad en el habla a comunicarse con el exterior. Consiste en un sistema de video-oculografía binocular que utiliza las cámaras y lentes de las que disponen los ordenadores y dispositivos móviles. La tecnología se basa en la localización de diversos puntos de la cara (seguimiento facial 3D), de este modo se hace una correcta lectura aunque el enfermo mueva la cabeza de manera constante (algo frecuente en parálisis cerebral o trastornos neurodegenerativos), tenga estrabismo o mire de perfil a la cámara.

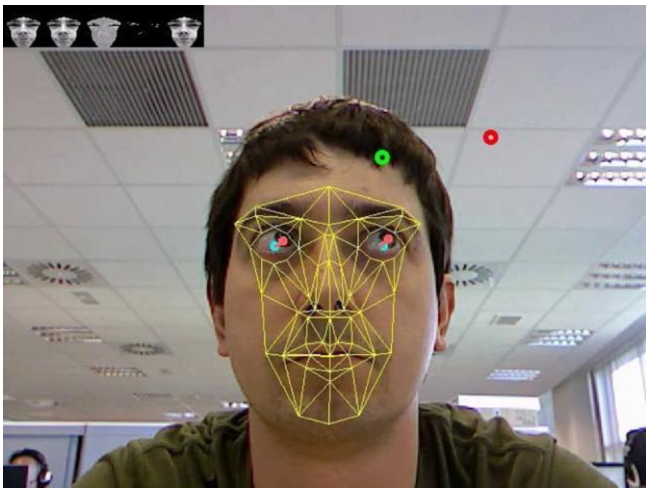

Figura 3. Prueba de escaneo de un rostro.

Así mismo, Horowitz y Brownell (2015 [9]) hacen referencia a tecnologías muy similares a las ofrecidas en este estudio, por lo que se adjuntan en la sección de Referencias bibliográficas. En el ámbito económico, el presupuesto total del proyecto asciende a casi 700.000 euros.

3.12 My Tobii: Tecnología para escribir y enviar mensajes de texto para pacientes con síndrome de enclaustramiento

Los pacientes con síndrome de enclaustramiento son personas con plena funcionalidad cognitiva, pero con incapacidad de poder realizar movimiento alguno con su cuerpo (únicamente los ojos). Se desarrolla una tecnología que permite al paciente escribir y enviar textos en un ordenador con el fin de poder facilitarle la posibilidad de establecer una comunicación con los de su entorno (Torres Cautivo, 2013 [19] y Dotinga, 2014 [8]). Su funcionamiento reside en la detección y seguimiento de los movimientos de sus ojos, de forma que sus pupilas actúan como seleccionadoras de las letras a escribir para formar oraciones. Al mirar hacia arriba, el programa ejecuta la acción seleccionada en el programa. Esta tecnología permitió a Alberto Vega (escritor, actor y dramaturgo chileno) poder escribir un libro donde refleja su vida y sus experiencias antes y después de sufrir un accidente de bicicleta que le provocaría posteriormente esta limitación. En el vídeo ofrecido por el canal de YouTube del Centro de Desarrollo de Tecnologías de Inclusión (CEDETi UC) (2016 [5]) se puede ver el análisis resumido del proceso de estudio enfocado a este escritor.

Los precios de los diferentes productos de la gama My Tobii ofrecidos por la entidad Link Assistive ([21]) se sitúan entre 445.89 € y 19 079.68 €.

3.13 Conclusiones del estudio

La inmensa mayoría de los elementos encontrados en la búsqueda del estado del arte proceden de estudios de investigación y artículos científicos, por lo que carecen de presupuestos o precios comerciales. No obstante, se destacan los sistemas OTTAA, INTERAAC y My Tobii. El primero de ellos, en términos económicos, se ofrece de forma gratuita, mientras que los otros dos requieren inversiones o costes bastante elevados para su adquisición.

En base a dichas condiciones, se comprueba que apenas se ofrecen recursos comerciales accesibles a los pacientes y los pocos productos que se encuentran a su disposición conllevan gastos de adquisición que, en muchos casos, se escapan de las posibilidades financieras de los usuarios. Asimismo, prácticamente todos los estudios realizados hasta la fecha se centraban en uno o dos pacientes concretos sometidos a pruebas, por lo que no queda reflejada ninguna clase de universalidad material lo suficientemente sencilla y práctica que puedan utilizar los pacientes.

Por tanto, para lograr que todo el mundo tenga acceso a herramientas que les permita o facilite establecer un medio de comunicación con el prójimo, es necesario buscar una solución de coste reducido, de fácil entendimiento y de uso sencillo. De esta manera, se encontraría un remedio alternativo a las opciones conocidas, cuyo punto destacable residiría en el precio y en la metodología usada.

4 SISTEMA PROPUESTO

Se propone llevar a cabo el desarrollo de un proyecto que emplee un sistema electrónico para controlar remotamente un Smartphone, con el fin de establecer una conexión con éste y utilizar los recursos reflejados en una aplicación para establecer una vía de comunicación entre el paciente y otras personas de su entorno. Para ello, se programará un cursor de ratón que será controlado por estos dispositivos externos y permitirá el manejo del teléfono sin necesidad de tocar la pantalla. A pesar del gran inconveniente de que no resulta tan sofisticado como otros sistemas propuestos, el punto clave reside en la ventaja que supone económicamente su uso, permitiendo a cualquier persona tener acceso a ella.

4.1 Componentes necesarios y herramientas utilizadas

Para la realización del trabajo se utilizará un dispositivo LaunchPad MSP-EXP430G2 de Texas Instruments, el cual permitirá facilitar la programación del microcontrolador MSP430G2553 que transmitirá la información al teléfono; se hará uso también de un joystick, cuyo objetivo será indicar la dirección que ha de tomar el puntero en su manejo; cables y una placa de prueba para testeo; y por último un módulo bluetooth HC-05 con el que se establecerá la conexión. De cara a una futura comercialización, se utilizaría una envolvente adecuada al proyecto y un botón suficientemente grande como para facilitar su uso.

El microcontrolador se programará en lenguaje C y se utilizará el software gratuito “Energia” ([23]) para tal fin. Asimismo,se hará uso del software “Android Studio” ([22]) para facilitar la programación de la aplicación android del Smartphone, para el cual se utilizará el lenguaje de programación Java.

4.2 Funcionamiento del sistema y estructura de la aplicación

La aplicación ofrecerá al usuario la posibilidad de establecer una conexión bluetooth con su dispositivo. Para mayor comodidad del portador, la conexión se realizará de forma automática al activar dicha funcionalidad. Asimismo, también dispondrá de la posibilidad de activar un cursor, el cual será manejado remotamente por el sistema electrónico.

Una vez establecida la conexión y activado el cursor, el usuario podrá hacer “click” sobre una imagen representativa de la aplicación, el cual, al ser pulsada, conducirá a una actividad con diferentes tipos de mensajes a su disposición, entre las que se encuentran sentir dolor, tener hambre o simplemente reflejar que el paciente se encuentra bien. Todo ello será controlado por el cursor, con el que también podrá navegar por el teléfono aprovechando el sistema de control propuesto.

Dicho control remoto emplea un joystick que actuaría funcionalmente como un ratón, así como un botón para poder ejercer la acción de “click”. Por simplicidad en el desarrollo, se ha aprovechado el botón que ofrece el propio LaunchPad para tal fin, con la intención de utilizar un botón más grande una vez se desarrolle el sistema.

Para que el funcionamiento sea adecuado, la conexión bluetooth y el control del ratón deberán mantenerse operativos dentro y fuera de la aplicación, de forma que el paciente pueda aprovechar las opciones que le ofrece el sistema operativo utilizando la misma herramienta con la que se comunica. Además, se facilitará la opción de poder activar el modo de funcionamiento “autoclick”, con el que no sería necesario el uso del botón.

La actividad principal de la aplicación poseerá dos botones y una imagen, manteniendo una estructura como la que sigue:

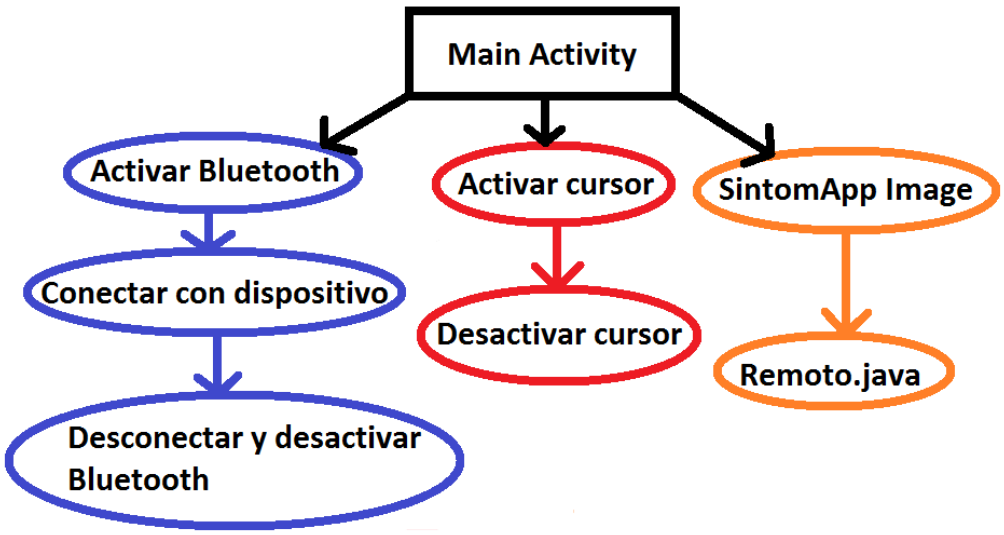

Figura 4. Estructura principal de la App

Siendo "Remoto.java" el archivo correspondiente a la actividad que incluye los diferentes mensajes de la aplicación.

Por otra parte, "Remoto.java" tendrá acceso a las siguientes opciones:

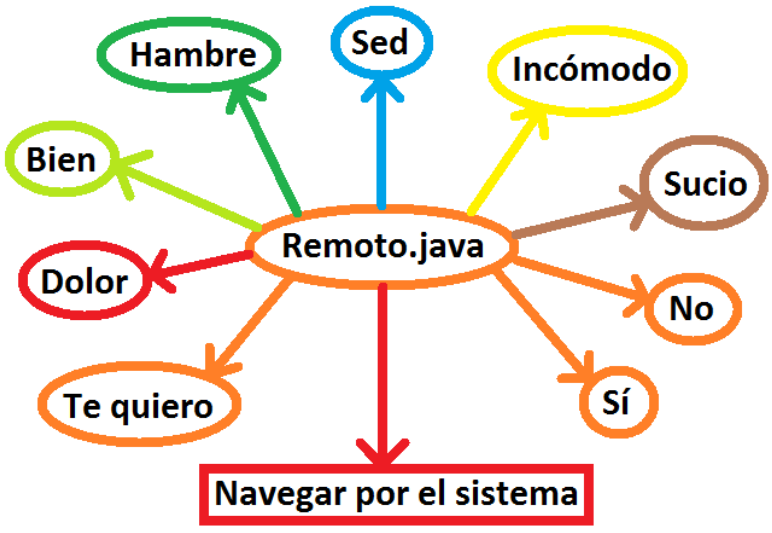

Figura 5. Estructura de la actividad correspondiente a los mensajes

Por último, en el apartado de ajustes y configuración, se mostrarán tres opciones a elegir por el usuario: El modo de "autoclick" con sonido, el cual producirá un estímulo sonoro antes de hacer click; el modo "autoclick" sin sonido, con mismo funcionamiento pero sin dicho estímulo; y el modo normal que usará el botón.

5 JUSTIFICACIÓN DE DISEÑO

Como se comentó previamente, todas las herramientas diseñadas con un fin biomédico deben ser estudiadas minuciosamente para que el paciente que las utilice se sienta cómodo con ellas, ya que probablemente deba requerir de su uso durante un tiempo prolongado de su vida. En este caso, el sistema tiene un carácter generalista y sencillo, por lo que debe ser fácil de utilizar, intuitivo y agradable a la vista para todos los pacientes que requieran de su uso.

5.1 Interfaz hombre-máquina

A nivel de interfaz hombre-máquina, el sistema requiere únicamente de un botón y un joystick, por lo que no deja lugar a dudas en cuanto a cómo deben usarse en este contexto. El botón debe ser de un tamaño lo suficientemente grande como para facilitar el uso del sistema a los pacientes; así mismo, el joystick seleccionado (ver *Figura 16*) no está escogido al azar, pues su tamaño lo hace un dispositivo de manejo sencillo acorde al prototipado del proyecto y, además, aporta la posibilidad de ser empujado con el brazo o la mano, sin requerir una sujeción directa con dedos o sin necesidad de ser agarrado.

El tamaño del cursor debe ser adecuado para que se pueda apreciar con claridad sin pecar de excesivo tamaño, pues de ser así sería un obstáculo visual y, más que una ayuda, podría resultar una molestia. Del mismo modo, la inclusión de la funcionalidad del autoclick configurable aporta variedad y buenas experiencias al usuario para poder utilizar su dispositivo ajustado a sus preferencias.

Cuantos menos elementos a manejar, más sencillo será su uso. Recordemos que los pacientes a los que va dirigido este sistema son personas con elevadas limitaciones locomotoras, por lo que no se puede tolerar dificultad en su manejo.

5.2 Comunicación entre dispositivos

Se barajaron diferentes opciones en cuanto al medio de transmisión o conexión entre el dispositivo y el Smartphone. Si se centra en las opciones más destacables, se encuentran el uso del puerto serie, el uso de WiFi y el uso de bluetooth.

La ventaja de usar el puerto serie es que la comunicación es directa, más sencilla, segura y rápida. Sin embargo, se barajó la posibilidad de que el paciente pudiera salir a la calle a dar un paseo junto a la persona responsable de su cuidado y se llegó a la conclusión de que los cables podrían resultar molestos y entorpecer el movimiento del usuario enganchándose en cualquier saliente que pudiera encontrarse. Además, la tendencia tecnológica indica el fomento del uso de tecnologías inalámbricas y la gran comodidad en su uso, por lo que esta opción fue descartada.

El uso de WiFi también pareció bastante interesante, sobre todo teniendo en cuenta la evolución tecnológica, el desarrollo de las “smart cities” y del IoT (Internet of Things). No obstante, no se optó por su uso debido a que se le forzaría al paciente a depender de diferentes puntos de acceso o, en su defecto, del uso de los datos del teléfono, hecho que resultó convincente para dejar al margen esta posibilidad.

Por ello, finalmente se acordó el uso del bluetooth, debido a que, a pesar de la problemática del consumo, la conexión entre dispositivos es única, proporciona independencia de puntos de acceso y la comunicación es inalámbrica. En adición, muchos productos del mercado siguen utilizando de forma considerable las

herramientas de bluetooth con diferentes propósitos (coches, equipos de música…), por lo que, según la evolución tecnológica, podría mejorarse el sistema para facilitar la interactividad con estos elementos.

5.3 Aplicación del smartphone

Desde el punto de vista de la aplicación, ésta deberá ser agradable a la vista, pues el usuario necesitará tenerla siempre activa si quiere comunicarse con otra persona. Por ello, se empleará una combinación de colores tal que no resulte muy molesto el uso de la misma, o dicho de otra forma, se evitará el uso de colores muy vivos para su presentación en las actividades principales de la aplicación. También se procurará no utilizar colores fríos, ya que este sistema debe verse como una herramienta positiva que motive al paciente a usarla y así mejorar su estilo de vida, de lo contrario podría dar la sensación de pesimismo incitando a pensar que su uso supone más una molestia que una ayuda para el usuario. Por ello, se recurrirá a colores cálidos.

Los diferentes mensajes a utilizar por el paciente deben ser aquellos que cubran sus necesidades básicas. En este caso, se ha considerado que las más relevantes son sentir dolor, tener hambre, tener sed, sentir incomodidad, sentirse sucio o con necesidades higiénicas, y sentirse bien. Además, también se incluyen las opciones de "Sí" y "No", así como la posibilidad de decir "Te quiero", pues en situaciones tan delicadas puede generar mucha frustración querer mostrar afecto por el prójimo y no tener posibilidad de reflejarlo de ninguna manera.

Todas estas opciones se representarán con imágenes intuitivas y colores que puedan relacionarse con estos síntomas. Dichas imágenes deberán representar la misma sensación que se desea expresar y se diseñarán de tal manera que resulten, en cualquier caso, divertidas e intuitivas. Además, al ser seleccionadas reflejarán sus mensajes por escrito y reproducirán mediante sonidos dichos textos, todo para que la comunicación resulte lo más cómoda y real posible.

Por último, también habrá un botón en la aplicación que permita dejarla en segundo plano para así navegar por el Smartphone y que el paciente pueda interactuar con su dispositivo, generando accesibilidad y proporcionando la oportunidad de ofrecer ocio al usuario.

5.4 Ventajas desde el ámbito social

- La tecnología empleada en el sistema es barata, por lo que entra en juego el factor Low Cost y la accesibilidad para personas que no pueden permitirse otras alternativas mucho más costosas.
- Los dispositivos empleados son de tamaño reducido, lo que conlleva a un diseño más compacto, ligero y cómodo para los pacientes.
- Ahora los pacientes con movilidad reducida podrán navegar por su Smartphone o tablet, independientemente de si pueden o no pueden articular palabras decentemente. Antes ya se podía, pero mediante tecnologías demasiado caras.
- Actualmente, casi todo el mundo tiene a su disposición un dispositivo móvil con el que poder interactuar a la hora de utilizar el sistema electrónico propuesto, por lo que el coste sería mucho más reducido al no necesitar incluir algo semejante en el pack. En caso de necesidad, podrían utilizarse tablets proporcionadas por los hospitales (junto a esta tecnología) debido a su adecuado tamaño (más visual que un teléfono) y las opciones a la hora de adquirirlo de forma gratuita y/o con precios muy bajos.
- La única modificación que se aplicaría al teléfono sería la instalación de la App y la activación de los permisos necesarios para utilizar la conexión Bluetooth y manejar el cursor.

- El paciente no utiliza, en este contexto, herramientas invasivas, ni se sometería a sobreesfuerzos, y le daría más libertad a la hora de buscar entretenimiento.
- Facilita la comunicación con cuidadores, tanto familiares como enfermeros. Es una herramienta que podrían llevarse a casa (préstamo del centro hospitalario, como se suele hacer con las sillas de ruedas).

6 SINTOMAPP: MENSAJES DE LA APLICACIÓN

Atendiendo a los criterios de diseño especificados en el apartado anterior, se procede a comenzar con el diseño de la actividad de la aplicación correspondiente a los mensajes, tal y como se especifica en la *Figura 5*. "Android Studio" posee una herramienta de simulación para comprobar cómo se mostraría la aplicación en un modelo de teléfono y versión de android concretos (a elegir por el programador), facilitando considerablemente el desarrollo del mismo. En este caso, las pruebas se realizarán sobre un Huawei P8 Lite con una versión de Android 5.0.1.

El primer paso es diseñar el logotipo de la aplicación y los representativos de cada mensaje para así ubicarlos en la actividad. Estas imágenes se encontrarán en la carpeta "drawable" dentro del directorio "res", en el proyecto de la aplicación Android, donde también estarán los archivos xml correspondientes a los diseños y funcionalidades de la aplicación. Las imágenes resultantes se muestran en la siguiente figura:

Figura 6. Logotipos

A continuación, se procede a generar los audios que sonarán al pulsar cada imagen. Para lograrlo, se recurrirá al software gratuito "Loquendo" ([24]), el cual permite reproducir textos mediante voces artificiales. Los archivos resultantes se encontrarán dentro del directorio "res", en el interior de una carpeta nombrada "raw". Éstos se asociarán a cada imagen mediante el fichero "SoundManager.java" situado dentro del directorio "java".

Finalmente, se deberán crear diferentes archivos que permitan la representación y reproducción de todo lo realizado hasta ahora. Dentro del directorio "res" se encontrará una carpeta denominada "layout", donde se ubicarán los archivos xml y darán forma a la aplicación. Para cada imagen, que actuará como si fuera un botón, se creará otro archivo xml que aplique los efectos apropiados para cada caso (por ejemplo, la inserción del texto o de la imagen), siendo en cada caso un "activity" y un "content". El primero asocia el contenido del segundo a la clase correspondiente a su actividad. Realmente no es necesario crear dos ficheros xml, pero de esta manera se puede alterar el contenido de "content" sin modificar el fichero "activity" xml en sí. En base a eso, será necesario crear tantas actividades java como sean necesarias para abordar todas estas opciones. Las actividades se ubicarán en el directorio "main", dentro de una sucesión de carpetas que conducirán a las actividades. Los archivos alojados ahí se programarán en java y son los encargados de utilizar los recursos citados previamente para que la aplicación funcione tal y como el programador indique. Todos los códigos se encuentran en la sección de *Anexos*.

Para ofrecer un mayor dinamismo y comodidad al usuario, las imágenes y los textos ofrecidos al pulsar las diferentes opciones parpadearán suavemente y, transcurridos 5 segundos, volverá a mostrarse la actividad principal. De este modo, el paciente se ahorraría volver atrás manualmente. Es importante recordar que los colores deben resultar cálidos y acogedores, sin permitir que pudieran volverse molestos para el paciente. En este sentido, se empleará un color anaranjado suave.

En la ventana de los mensajes se dispondrá de una sección de "ajustes del sistema" para indicar el modo de funcionamiento deseado por el usuario: el modo normal, el modo de autoclick con sonido y el autoclick sin sonido. La mayor parte de la programación de esta parte pertenece al microcontrolador, por lo que se detallará más adelante.

Por último, una vez programados todos los archivos, utilizamos las diferentes herramientas de Android Studio para probar la aplicación. En este caso, se probará directamente en el teléfono para facilitar un visionado y una comprobación directos. Al instalarse la aplicación, se puede comprobar que aparece entre las aplicaciones del teléfono con su correspondiente logotipo.

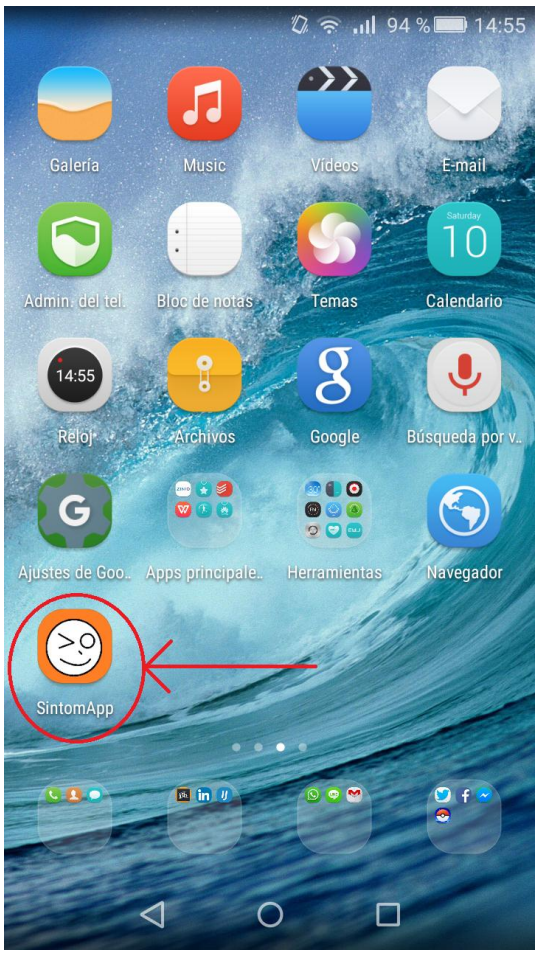

Figura 7. Aplicación instalada

Así mismo, si accedemos a la aplicación encontraremos las diferentes imágenes ubicadas según lo previsto y el botón "Navegar por el sistema" aún sin programar:

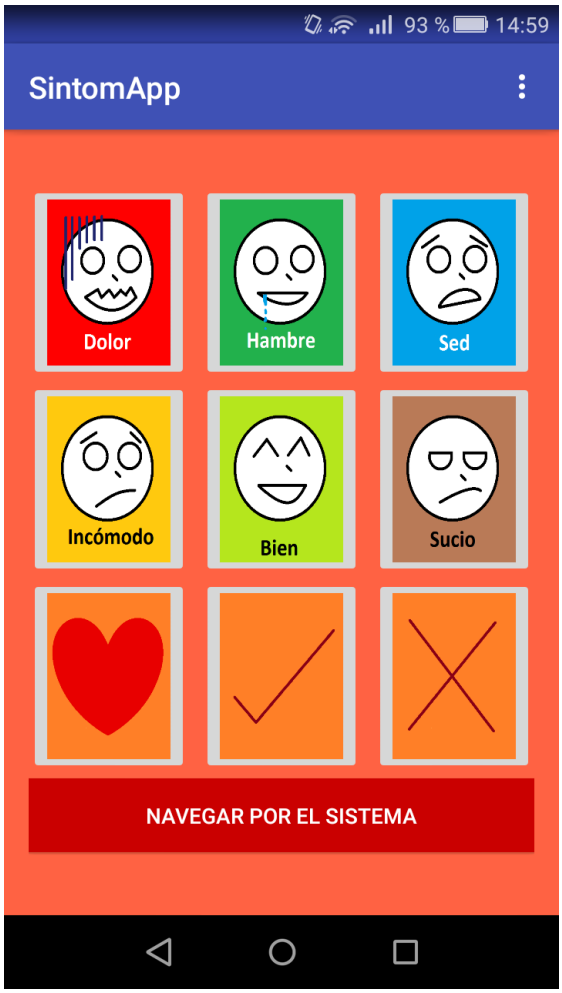

Figura 8. Vista previa de "Remoto.java"

A continuación, comprobamos que al pulsar cada una de las imágenes se accede a las diferentes actividades asociadas a las mismas. A modo de ejemplo, se ilustra a continuación la opción "Bien".

Figura 9. Actividad "Bien.java"

Por otra parte, basta con asociar al botón “Navegar por el sistema” el siguiente código para que cumpla con su cometido, esto es, volver al Home del teléfono para poder interactuar con el resto de aplicaciones:

```
bHide = (Button) findViewById(R.id.Button_Control);
bHide.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent newActivity = new Intent(Intent.ACTION_MAIN);
        newActivity.addCategory(Intent.CATEGORY_HOME);
        newActivity.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(newActivity);
    }
});
```

Finalmente, comprobamos que, al pulsar el apartado de ajustes, se muestran las opciones deseadas.

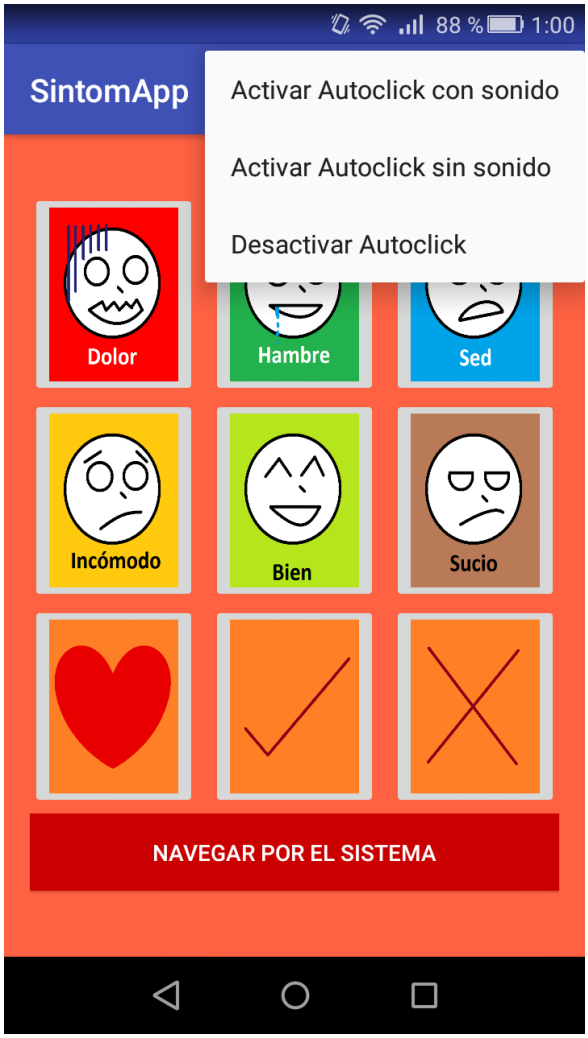

Figura 10. Opciones de la configuración del sistema

7 SINTOMAPP: VENTANA PRINCIPAL

Tal y como se indicó en anteriores apartados, la ventana principal de la aplicación contendrá una imagen del logotipo de la app que irá acompañada de dos botones: uno para establecer la conexión bluetooth y el otro para activar al cursor.

El color de fondo será el mismo que el utilizado en el *Capítulo 6* y los botones serán, respectivamente, azul y rojo para la conexión bluetooth y la activación del puntero. Siguiendo el mismo procedimiento utilizado previamente, se puede obtener fácilmente el siguiente resultado:

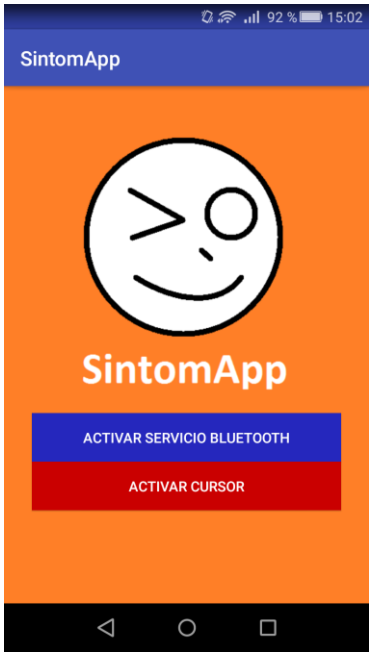

Figura 11. Ventana principal

7.1 Acceso a "Remoto.java"

Siguiendo la idea prevista, se busca que, al seleccionar la imagen de SintomApp, la aplicación muestre la ventana diseñada en el anterior capítulo, es decir, "Remoto.java". Para ello, asociamos dicho elemento a la acción de ser pulsado con el fin de que, tras dicho acto, inicie la actividad "Remoto.class":

```
/** Botón para acceder a las opciones de SintomApp */

        SintomApp = (ImageView) findViewById(R.id.imageView);
        SintomApp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(MainActivity.this, Remoto.class);
                startActivity(intent);

            }
        });
```

Si compilamos y ejecutamos la aplicación, se verificará que al tocar la imagen aparecerá la ventana "Remoto", tal y como se muestra en la *Figura 8*.

7.2 Conexión Bluetooth

En esta sección se describe la metodología empleada para establecer la conexión entre el teléfono y el sistema desde el punto de vista de la aplicación android. El código correspondiente se encontrará en el fichero "ConexionBT.java" y actuará como un servicio y no una actividad, pues su contenido es puramente funcional y no requiere de vista alguna. También serán necesarios unos ficheros que permitan la conexión en segundo plano, ya que "ConexionBT.java" sólo mantendría la conexión dentro de la aplicación.

El botón situado en la ventana principal cambiará su texto según se vaya pulsando. De esta manera, con un solo elemento se podrá activar el bluetooth del teléfono, establecer la conexión con el dispositivo y desconectar y desactivar el bluetooth. Esto será posible gracias a la funcionalidad "setText" en el fichero "MainActivity.java".

Para empezar, en "ConexionBT.java" será necesario asignar un identificador único universal o universally unique identifier (UUID). Es un número de 16 bytes (128 bits) que se utiliza para crear identificadores únicos universales que permitan reconocer y distinguir un objeto dentro de un sistema, o el mismo objeto en diferentes contextos. Existen métodos sencillos para generar un identificador para nuestro software, todos ellos disponibles en páginas web habilitadas a tal fin ([25]). En nuestro caso, la que utilizaremos en nuestro código será "*00001101-0000-1000-8000-00805F9B34FB*". Esto se utilizará posteriormente para establecer el hilo de conexión con el dispositivo.

No obstante, el primer paso es lograr que realice la conexión de forma automática con el dispositivo. Para tal fin, se codifica la función "Conecta", el cual emplea la dirección del dispositivo HC-05 para establecer la conexión (en este caso *98:D3:31:40:80:AC*):

```
/** Conexión con dispositivo Bluetooth. Dirección particular
    para autoconexión: 98:D3:31:40:80:AC */

private void Conecta(){
    String address = "98:D3:31:40:80:AC";
    BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
    ConnectThread connect = new ConnectThread(device);
    connect.start();
}
```

"ConnectThread" es una función que permite abordar la conectividad a nivel de sockets, dirección y UUID.

Una vez que se establezca la conexión, el teléfono comenzará a recibir información del dispositivo (movimientos de joystick y pulsación de botón), por lo que será necesario tomar dichos mensajes y almacenarlos en una variable para que el cursor pueda utilizarlos e interpretarlos adecuadamente.

Por último, se debe asociar cada pulsación del botón «Activar servicio Bluetooth» a una acción. Utilizando la función "onStartCommand" y variables auxiliares, se consigue que, en la primera pulsación, se active la conexión bluetooth; en la segunda pulsación se conecta con el dispositivo que utilizará el usuario, cambiando el nombre a «Conectar con dispositivo»; y en la tercera se lleva a cabo la desconexión, cambiando el mensaje a «Desactivar servicio Bluetooth». No obstante, tal y como se comentó previamente, será necesario emplear funciones adicionales para que la conexión se mantenga en segundo plano. De esta forma, al salir de la aplicación, la funcionalidad del sistema no variará ni se verá obstaculizada a tales efectos. Para ello, utilizamos tres archivos adicionales situados dentro del directorio "java":

- MemoryService.java: Comprueba y notifica la cantidad de memoria disponible del sistema.
- ProgressIntentService.java: Encargado de mantener el funcionamiento de un servicio de primer plano en segundo plano.

- Constantes.java: Constantes utilizadas para el correcto funcionamiento de los dos archivos anteriores.

Con esto, sólo faltaría añadirlo dentro de la función "onStartCommand" justo en el punto donde se establece la conexión y comunicación del dispositivo con el Smartphone:

```
/** Conexión en segundo plano */

final ActivityManager.MemoryInfo memoryInfo = new
ActivityManager.MemoryInfo();
final ActivityManager activityManager =
        (ActivityManager) getSystemService(ACTIVITY_SERVICE);

Timer timer = new Timer();

timerTask = new TimerTask() {
    @Override
    public void run() {
        activityManager.getMemoryInfo(memoryInfo);
        // 1048576 MB es la cantidad máxima de memoria RAM disponible
        String availMem = memoryInfo.availMem / 1048576 + "MB";

        Log.d(TAG, availMem);

        Intent localIntent = new Intent(Constantes.ACTION_RUN_SERVICE)
                .putExtra(Constantes.EXTRA_MEMORY, availMem);

        // Emitir el intent a la actividad
        LocalBroadcastManager.
                getInstance(ConexionBT.this).sendBroadcast(localIntent);
    }
};

timer.scheduleAtFixedRate(timerTask, 0, 1000);
Conecta();
```

7.3 Programación del cursor

El primer paso consiste en buscar una imagen de un puntero y utilizarlo como si fuera el propio ratón del sistema. Éste se ubicará en la carpeta "mipmap" dentro del directorio "res". La imagen se empleará en "cursor.xml", dentro de la carpeta "layout", que será quien se encargue de la representación del mismo al pulsar el botón. Para el correcto funcionamiento del manejo del cursor, también será necesario utilizar una configuración específica para tal fin, programada en el fichero "mouse_accessibility_service_config.xml" dentro de la carpeta "xml" en el interior del directorio "res". De lo contrario, no será posible llevar a cabo el objetivo previsto.

Teniendo eso en cuenta, se procede a describir los dos ficheros principales para el manejo del cursor.

7.3.1 MouseEvent.java

Este fichero se encarga de interpretar los mensajes recibidos por el LaunchPad. Dependiendo del valor recibido, se le indicará a la función principal el movimiento o acción que debe realizar el puntero acorde a los siguientes mensajes:

```
public class MouseEvent {

    public static final int
            ARRIBA = 0,
            ABAJO = 1,
            IZQUIERDA = 2,
            DERECHA = 3,
            CLICK = 4,
            STOP = 5,
            CLICKAUTO = 6;


    public final int direccion;

    public MouseEvent(int direccion) {
        this.direccion = direccion;
    }
}
```

De esta forma:

- Si recibe un 0, se interpretará como que el cursor debe desplazarse hacia arriba.
- Si recibe un 1, se interpretará como que el cursor debe desplazarse hacia abajo.
- Si recibe un 2, se interpretará como que el cursor debe desplazarse hacia la izquierda.
- Si recibe un 3, se interpretará como que el cursor debe desplazarse hacia la derecha.
- Si recibe un 4, se interpretará como que se ha realizado la acción de "click" con el botón.
- Si recibe un 5, se interpretará como que no se está realizando ninguna acción.
- Si recibe un 6, autoclick estará activado.

7.3.2 MouseAccessibilityService.java

Al iniciarse el servicio, una función se encargará de representar por pantalla el cursor en una determinada posición utilizando la variable "cursorView"; al mismo tiempo, para controlar el manejo del cursor se utilizará la variable "cursorLayout".

```
/** Se genera el puntero; cursorView pone la imagen del ratón; cursorLayout
se ocupa del
            movimiento del mismo y su ubicación. */

        cursorView = View.inflate(getBaseContext(), R.layout.cursor, null);
        cursorLayout = new LayoutParams(
                LayoutParams.WRAP_CONTENT,
                LayoutParams.WRAP_CONTENT,
                LayoutParams.TYPE_SYSTEM_ERROR,
                LayoutParams.FLAG_DISMISS_KEYGUARD |
LayoutParams.FLAG_NOT_FOCUSABLE | LayoutParams.FLAG_NOT_TOUCHABLE |
LayoutParams.FLAG_LAYOUT_IN_SCREEN,
                PixelFormat.TRANSLUCENT);
        cursorLayout.gravity = Gravity.TOP | Gravity.LEFT;
        cursorLayout.x = 200;
        cursorLayout.y = 200;
```

Paralelamente, será necesario conocer el estado de la ventana activa del Smartphone para asociar el click con lo que se ubique justo debajo del cursor. Para ello, se utilizará la interfaz WindowManager, que permite tomar

información de la pantalla del dispositivo. A partir de ahora, se puede proceder a llevar a cabo la interpretación de los mensajes recibidos por el dispositivo externo. Será necesario dar un margen de tiempo para que el sistema pueda procesarlos sin bloquearse.

```
/** Toma servicio del "window manager" */

        windowManager = (WindowManager) getSystemService(WINDOW_SERVICE);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {

                    /** Leemos la información recibida por el Launchpad */

                    MoveM = ConexionBT.mensaje;

                    /** Interpretamos los datos recibidos */

                    if (MoveM != null) {

                        String message = new String(MoveM);
                        final int event = Integer.parseInt(message);
                        new Handler(getMainLooper()).post(new Runnable() {
                            @Override
                            public void run() {
                                onMouseMove(new MouseEvent(event));
                            }

                        });
                    }

                    /** Damos un pequeño margen de tiempo para su
procesamiento */

                    try {
                        doLongOperation();
                    } catch (final Exception ex) {
                        Log.i("---", "Exception in thread");
                    }
                }
            }
        }).start();
    }
```

Según el evento recibido, se llevará a cabo una determinada acción u otra. Los movimientos serán constantes y de una velocidad moderada por simplicidad de diseño y manejo para los pacientes. Las variables auxiliares y los eventos de STOP permiten que no se produzcan rebotes en la pulsación del botón y así el sistema pueda realizar la acción sin problemas.

```
/** Movimiento del ratón y activación de click */

    public void onMouseMove(final MouseEvent event) {

        switch (event.direccion) {
            case MouseEvent.IZQUIERDA:
                cursorLayout.x = cursorLayout.x - 2;
                break;
            case MouseEvent.DERECHA:
                cursorLayout.x = cursorLayout.x + 2;
```

```
                break;
            case MouseEvent.ARRIBA:
                cursorLayout.y = cursorLayout.y - 2;
                break;
            case MouseEvent.ABAJO:
                cursorLayout.y = cursorLayout.y + 2;
                break;
            case MouseEvent.CLICK:
                if(aux == 0) {
                    click();
                    aux = 1;
                }
                break;
            case MouseEvent.STOP:
                if(aux == 1) {
                    aux = 0;
                }
                break;
            case MouseEvent.CLICKAUTO:
                if(aux2 == 0){
                    aux2 = 1;
                    sound.play(sms);
                }
                break;
            default:
                break;
        }
        windowManager.updateViewLayout(cursorView, cursorLayout);
    }
```

Por último, el efecto de click toma los valores de posición de "cursorLayout" en sus ejes x e y para buscar la ubicación del puntero. Mediante otras funciones especificadas y programadas en el fichero, utiliza la información de la ventana activa y lleva a cabo una búsqueda de la posición del ratón descartando las situaciones más lejanas hasta dar con el foco de la pulsación.

Una vez encontrada dicha zona, se procede a realizar la acción de click para acceder al contenido del elemento que seleccionara el usuario.

```
/** Efecto de click */

     private void click() {
         Log.d(TAG, String.format("Click [%d, %d]", cursorLayout.x,
cursorLayout.y));
         AccessibilityNodeInfo nodeInfo = getRootInActiveWindow();

        if (nodeInfo == null) {
            System.out.println("No se detecta ventana");
            return;
        }
        AccessibilityNodeInfo nearestNodeToMouse =
findSmallestNodeAtPoint(nodeInfo, cursorLayout.x, cursorLayout.y + 50);
        if (nearestNodeToMouse != null) {
            logNodeHierachy(nearestNodeToMouse, 0);

nearestNodeToMouse.performAction(AccessibilityNodeInfo.ACTION_CLICK);
        }
        nodeInfo.recycle();
    }
```

7.4 AndroidManifest.xml

Para finalizar, será necesario que se modifique un fichero importante en todo proyecto android, que es "AndroidManifest.xml" situado en la carpeta "manifest". En este archivo se deben especificar todas las funcionalidades y características de las clases empleadas en el proyecto, así como la habilitación de todos los permisos necesarios para que el funcionamiento sea el adecuado. En este caso, los permisos necesarios son los siguientes:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.communicationsupport.kikobc.myapplication">
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

- Los permisos BLUETOOTH y BLUETOOTH_ADMIN son los necesarios para poder manipular la conectividad bluetooth del teléfono.
- El permiso SYSTEM_ALERT_WINDOW permite el correcto funcionamiento del cursor tomando información de la ventana activa del teléfono.

Reinstalamos la aplicación y comprobamos que los resultados son los esperados.

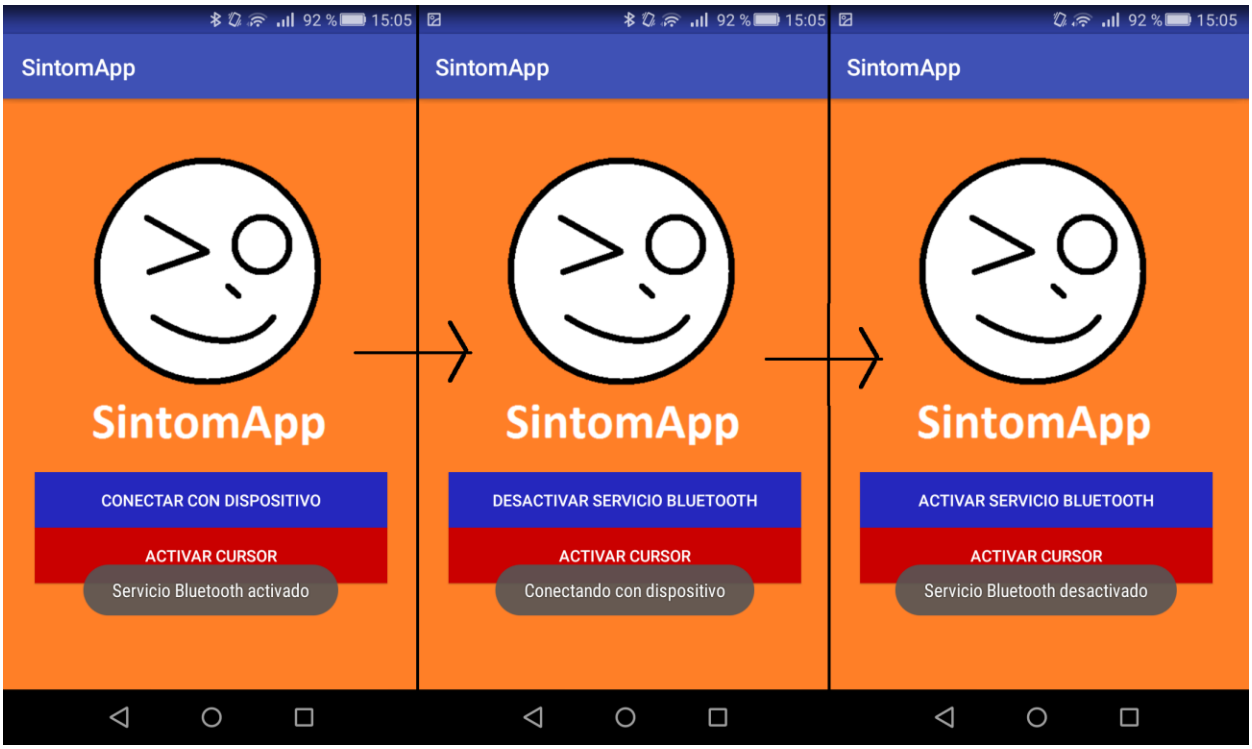

Figura 12. Comprobación conectividad bluetooth

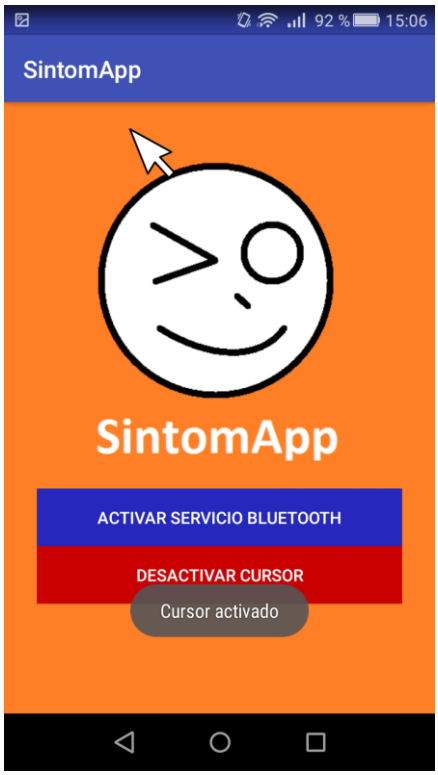

Figura 13. Comprobación de la aparición del ratón

Con todo esto, el proyecto podría estar listo para generar la apk del programa. No obstante, aún queda por programar el microcontrolador que controlará el teléfono mediante esta aplicación, por lo que aún se debe comprobar si el control del cursor es efectivo.

8 PROGRAMACIÓN DEL LAUNCHPAD

El LaunchPad es, a efectos prácticos, un arduino de Texas Instruments. Es por ello que su funcionalidad es muy similar a la de estos dispositivos y su desarrollo suele seguir la misma estructura. Programado en lenguaje C, se utilizan dos funciones principales diferentes para su manejo: "setup" y "loop", ambas de tipo void. La función "setup" se encarga de inicializar el sistema, las variables y las conexiones; por otra parte, "loop" es una rutina que se repite indefinidamente mientras el dispositivo se mantenga operativo.

8.1 Montaje del sistema

Para comenzar, resulta necesario realizar el conexionado de los dispositivos adecuadamente para alcanzar el objetivo. Para ello, ha de conocerse la estructura de los diferentes dispositivos. En primer lugar, para el Launchpad se tiene la siguiente distribución de pines:

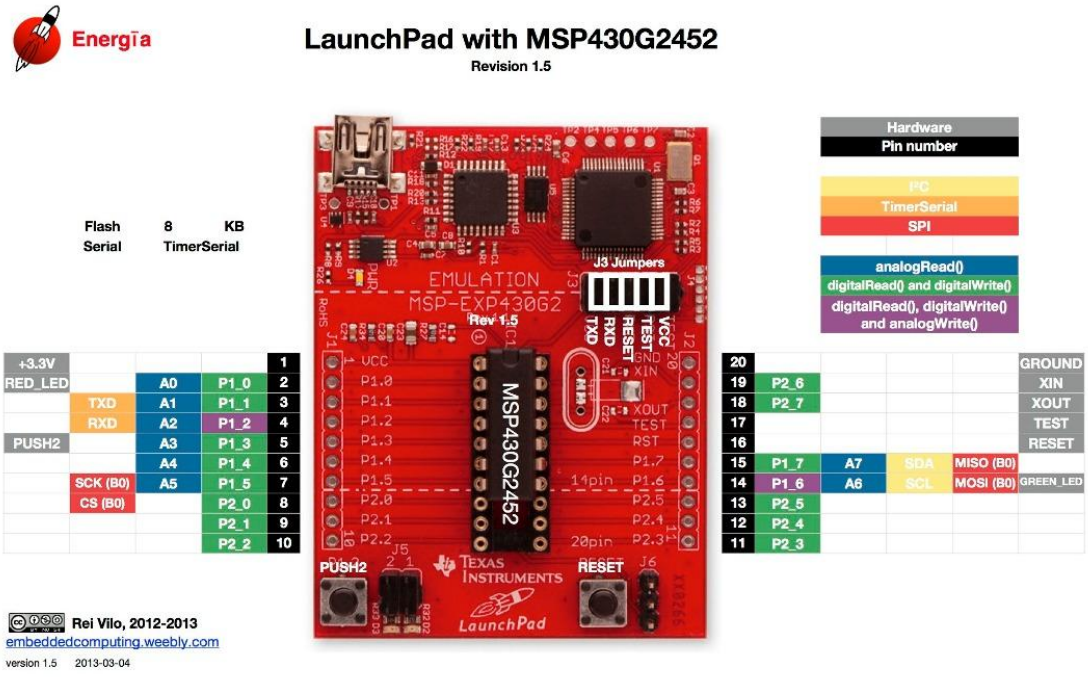

Figura 14. LaunchPad utilizado en el proyecto

Para el módulo bluetooth HC-05:

Figura 15. Módulo Bluetooth HC-05

Y para el joystick:

Figura 16. Joystick

Ahora que se conoce la distribución de pines de los dispositivos, se procede a realizar el conexionado del sistema.

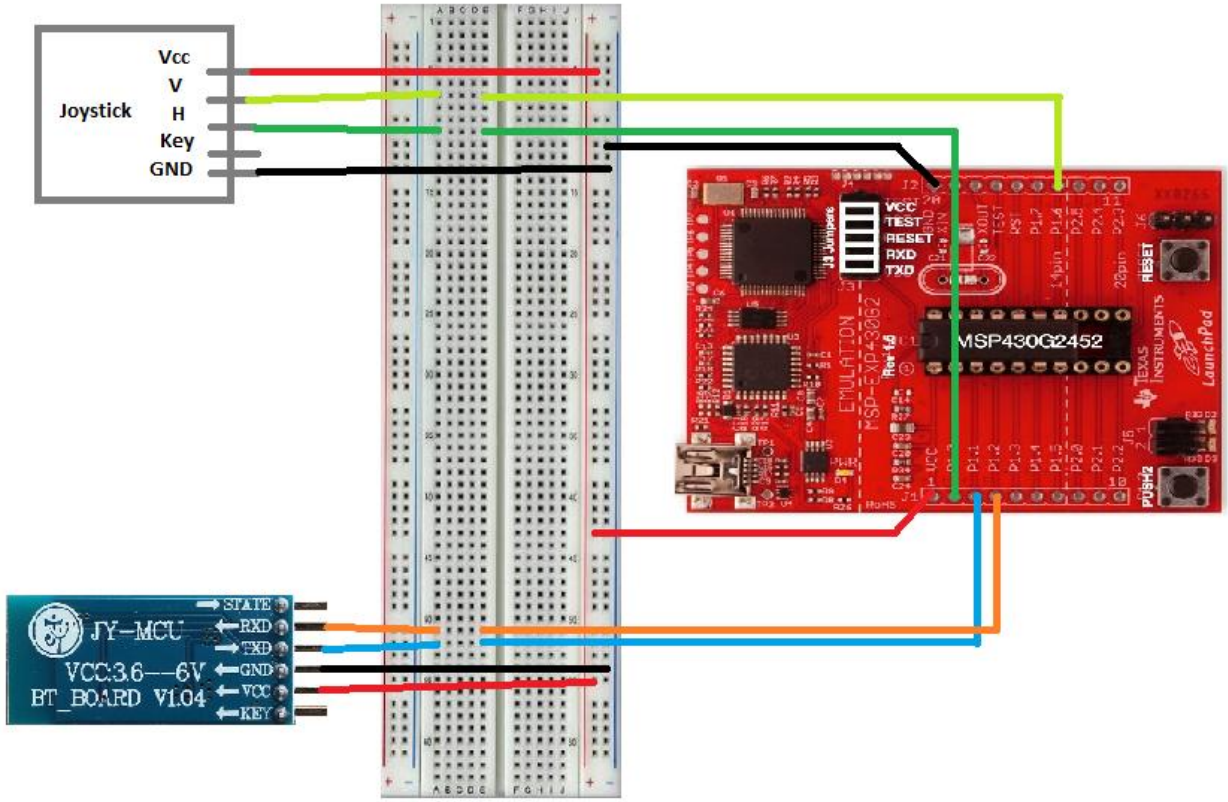

Figura 17. Conexionado del sistema

8.2 Definición de variables y asignación de pines

Los pines utilizados en el LaunchPad y las variables empleadas son las siguientes:

```
#include <msp430.h>

// Definimos los pines que vamos a usar.

#define PINJOYSTICK_V  P1_6      // Movimiento vertical Joystick
#define PINJOYSTICK_H  P1_0      // Movimiento horizontal Joystick
#define PINLED         GREEN_LED // LED verde
#define PINLED2        RED_LED   // LED rojo
#define VCC            1         // Pin de VCC

// Variables auxiliares necesarias:

int joystickValueV = 0;  // Posición vertical del joystick
int joystickValueH = 0;  // Posición horizontal del joystick
int pwmValueV = 0;       // Valor escalado de la posición vertical
int pwmValueH = 0;       // Valor escalado de la posición horizontal
int Botoncito = 0;       // Su valor dependerá de si se pulsa el botón o no
int buttonState = 0;     // Variable para leer el estado del botón
const int Boton = PUSH2; // Botón físico que se utilizará para simular el click del cursor.
volatile boolean debounce = false;
char autoclick = '0';   // Su valor indica el modo de funcionamiento
static int auxiliar = 0; // Variable auxiliar para el timer
```

- El uso de los LEDs no es necesario, pero facilita la comprobación del funcionamiento del programa al manipular el joystick. El objetivo es que uno de los LEDs se encienda o se apaque progresivamente según la posición del joystick en cada uno de sus ejes.
- Los pines asociados a los movimientos vertical y horizontal del joystick están relacionados con los LEDs, por lo que para realizar tal comprobación se deben escoger esos.
- Aunque no se vea reflejado en el código, también se usa el pin de GND, solo que no resulta necesario iniciarlizarlo.
- Las variables "joystickValueV" y "joystickValueH" recogerán el valor de tensión que proporcione cada eje del joystick. Recordemos que un joystick no es más que un potenciómetro con dos variables acordes a los ejes x e y. Según dichos valores, se transmitirá una determinada información, es decir, el cursor se desplazará hacia un lado u otro.
- "pwmValueV" y "pwmValueH" escalan el valor de "joystickValue" para la variación de intensidad luminosa de los leds, pues la lectura de valores abarca desde 0 hasta 1023, mientras que la escritura es entre 0 y 255.
- "Botoncito" tomará un valor u otro dependiendo del estado de "buttonState", el cual valdrá "HIGH" o "LOW" dependiendo de si se pulsa el botón o no. Dichos valores se definen en la función «Void setup()», donde se inicializan las variables. La variable "Boton" tendrá asignado el valor "PUSH2", que es el que está asociado al botón del LaunchPad.
- "debounce" es una variable booleana que se utiliza en una función externa dedicada a evitar el efecto rebote en la pulsación de los botones.
- Las variables "autoclick" y "auxiliar", así como la librería "msp430.h", están involucradas con el funcionamiento del Timer que permitirá gestionar el modo autoclick.

8.3 Funciones utilizadas

8.3.1 Void setup()

Esta función inicializa y asocia el valor de los pines a las variables, indica si son pines de entrada o salida e inicializa el servicio de comunicación mediante la función «serial.begin()» y, en este caso, relaciona la pulsación del botón del LaunchPad con una función externa para controlar el efecto de los rebotes. El motivo por el que se usa «serial.begin()» para la conexión, es que el microcontrolador utiliza el puerto serie para comunicarse con el módulo bluetooth, el cual iniciará la transmisión por dicho medio. Por ello, no es necesario utilizar otro tipo de funciones y/o librerías para su uso. También se inicializan en esta función los registros y configuraciones necesarias para el correcto manejo del Timer.

8.3.2 Void loop()

El bucle infinito. Esta función se ejecuta una y otra vez mientras el programa esté en funcionamiento. En cada iteración, se utiliza la función "analogRead" para leer el estado de los ejes del joystick y del botón. Si el botón está pulsado, "Botoncito" valdrá '4' (el número asociado a la acción de click en la aplicación, en el fichero "*MouseEvent.java*"), en caso contrario su valor será '0'. Si el botón se pulsó, se envía el dato correspondiente mientras que se ejecuta la función externa antirrebotes.

Los valores que aportan los ejes del joystick son positivos, por lo que se les restará un número igual a su valor en estado de reposo para ubicarlo inicialmente en 0. El motivo por el que se hace esto es porque las coordenadas del centro de la pantalla del teléfono se encuentran en (0,0), por lo que centrar el valor resulta necesario.

Según el valor asociado a las posiciones del joystick se transmitirá un resultado u otro para que la aplicación pueda interpretarlo. Para ello, se establece el umbral numérico de ±30 para que no sea demasiado sensible al movimiento del joystick, ya que podría entorpecer el control del cursor. Recordemos que, aunque en estado de reposo tome el valor 0 para cada eje, realmente se están midiendo valores de tensión analógicos, por lo que es necesario aportar un margen de seguridad para que el cursor no se desplace por sí solo.

También se hará uso de "Serial.read()" para leer la información que transmita el teléfono. Esto servirá para ajustar el modo de funcionamiento del sistema.

Finalmente, se deja transcurrir un breve periodo de tiempo (unos 100ms) utilizando la función "delay" para que la transmisión de información se produzca adecuadamente.

Si el usuario no hace nada se transmite un '5', que es el valor asociado a STOP en la aplicación. Es necesario incluir este estado, ya que de lo contrario se transmitiría siempre el último valor enviado. Dicho en otras palabras, sin ese estado, al desplazar el puntero a la derecha un poco, en la realidad se movería de forma indefinida hasta que el usuario cambiara de dirección.

8.3.3 Void func ()

Cambia el valor de la variable "debounce" para evitar los rebotes en la función anterior.

8.3.4 Void timerA0ISR ()

Función correspondiente al Timer. Utiliza el valor de "autoclick" para ajustar un modo de funcionamiento u otro, considerando entre otras variables los valores aportados por el joystick. Si el modo autoclick con sonido está activado, se transmitirá un "6" (valor asociado a la reproducción del sonido) y poco después un "4" (la acción de click), reseteando los valores del contador y de otras variables implicadas en la función.

Si el modo de funcionamiento es el de autoclick sin sonido, el procedimiento será exactamente el mismo, pero sin considerar el primer envío descrito previamente. En caso de estar desactivado, la función no hará nada. Por último, si en el transcurso se mueve el joystick el contador se resetea y las variables utilizadas recuperan su valor original.

9 PRUEBA EXPERIMENTAL DEL PROYECTO

A continuación se procede a mostrar el funcionamiento del sistema completo. El primer paso consiste en enlazar los dos dispositivos mediante conexión bluetooth y, posteriormente, acceder a los ajustes del teléfono para activar las herramientas de accesibilidad necesarias para permitir su funcionamiento. Acto seguido, se procederá a iniciar la aplicación y probar si la funcionalidad es la correcta.

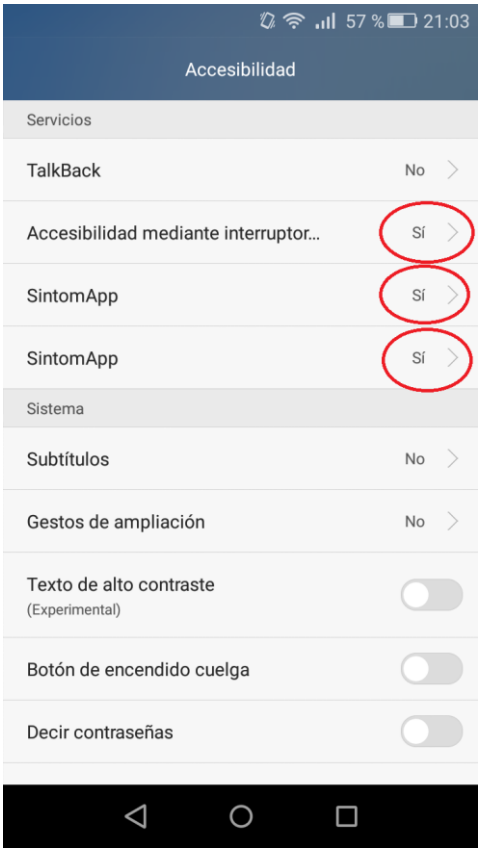

Figura 18. Sección de accesibilidad en los ajustes del sistema

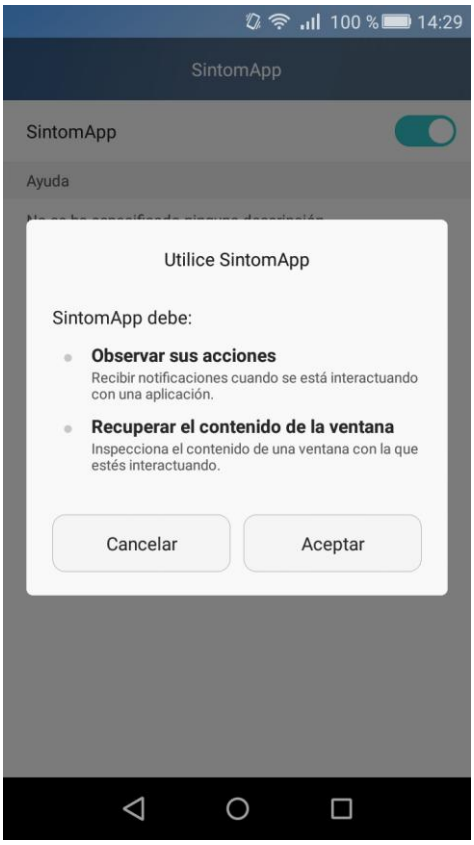

Figura 19. Activación de permisos de accesibilidad de SintomApp

Una vez que todo esté a punto, iniciamos la conexión bluetooth y activamos el cursor de la aplicación. Inicialmente, el LED rojo del módulo bluetooth parpadeará de forma constante hasta establecer una conexión con un dispositivo, en cuyo caso se apagará y cada dos segundos aproximadamente parpadeará un par de veces.

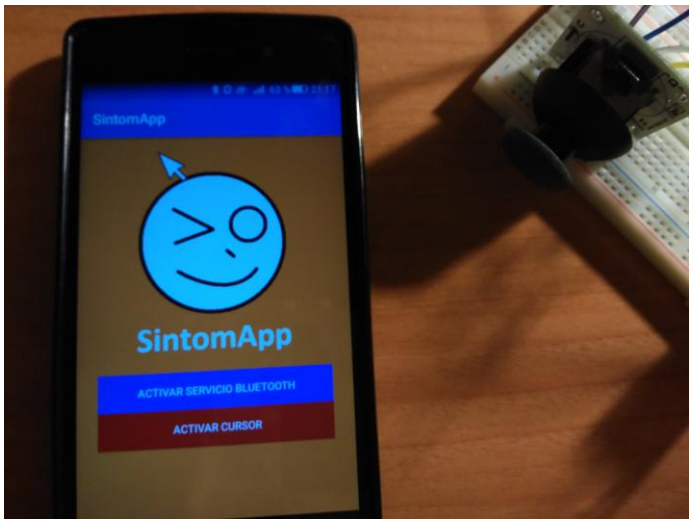

Figura 20. Prueba de funcionamiento. Conexión.

Si se intenta manipular el joystick, se comprobará que el cursor se mueve en la dirección que indique el dispositivo.

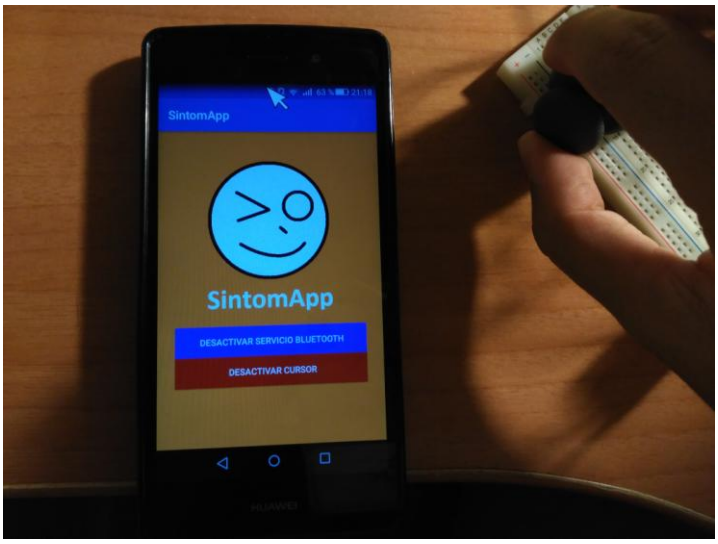

Figura 21. Movimiento de cursor. Desplazamiento ascendente.

A continuación, se hará uso del botón del LaunchPad para hacer click sobre el logotipo de SintomApp, el cual permitirá el acceso al menú de mensajes de la aplicación.

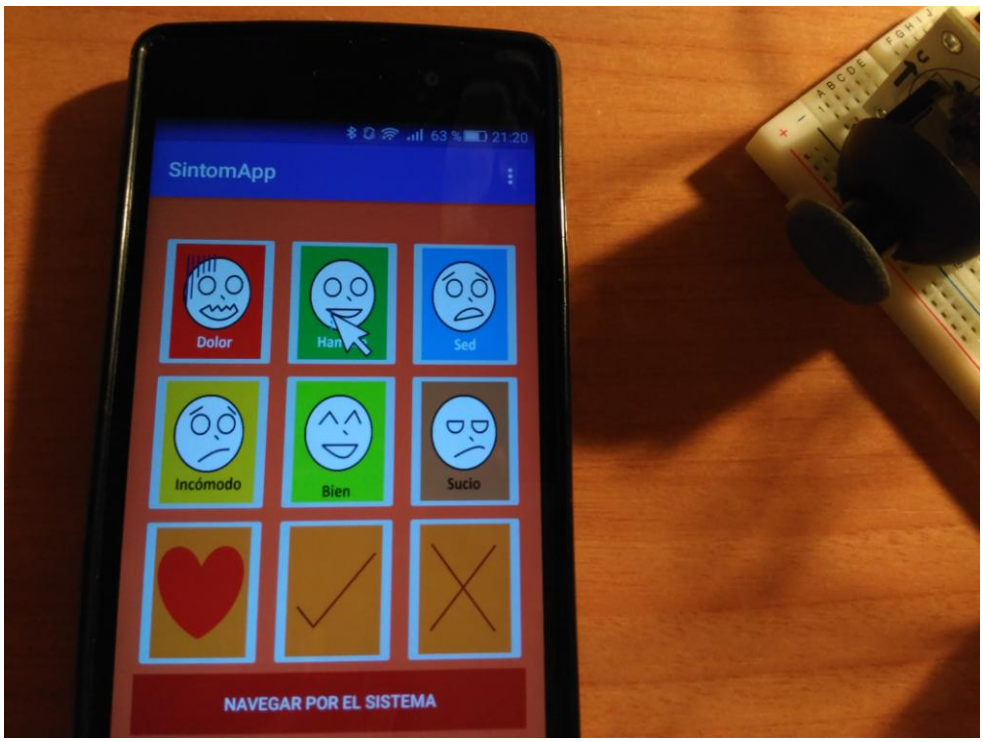

Figura 22. Funcionamiento del click dentro de la app. Acceso a menú de mensajes.

Se comprueba si el acceso a las diferentes opciones se puede llevar a cabo sin problemas. A modo de ejemplo, se muestra la opción de “Hambre”.

Figura 23. Acceso a mensajes. Hambre.

Transcurridos 5 segundos, la aplicación vuelve a mostrar el menú de opciones. Se comprueba de esta manera que el sistema funciona y es posible emplear esta herramienta para facilitar la comunicación de pacientes con movilidad reducida. No obstante, aún falta por comprobar si el funcionamiento sigue siendo el adecuado fuera de la aplicación.

Situamos el cursor sobre el botón "Navegar por el sistema" y hacemos click presionando el botón del LaunchPad.

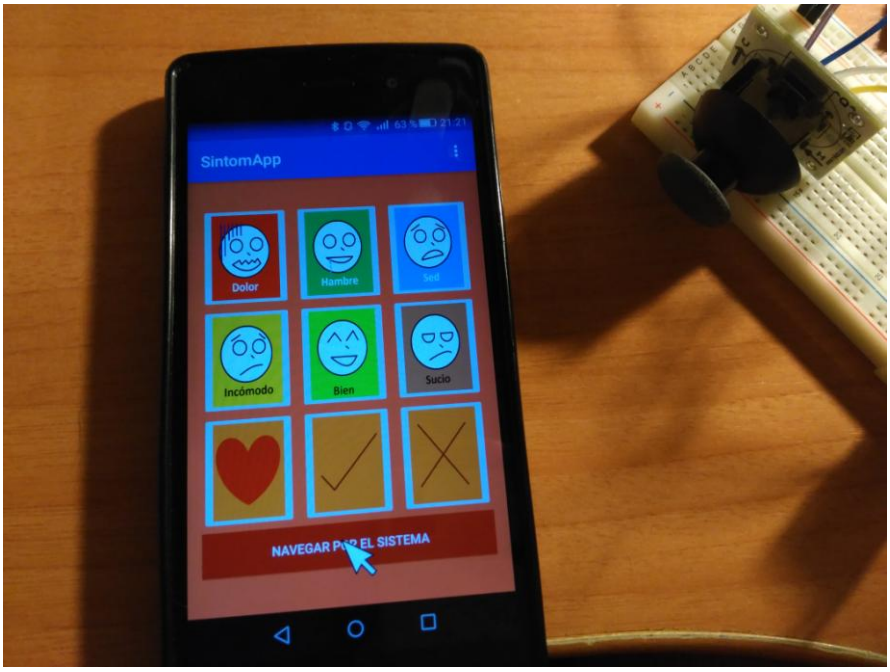

Figura 24. Prueba de navegación por el sistema (1)

Figura 25. Prueba de navegación por el sistema (2)

Desplazamos el cursor sobre otra aplicación. A modo de ejemplo, se intentará acceder a los archivos del sistema. Situamos el cursor sobre el icono correspondiente y presionamos el botón de click.

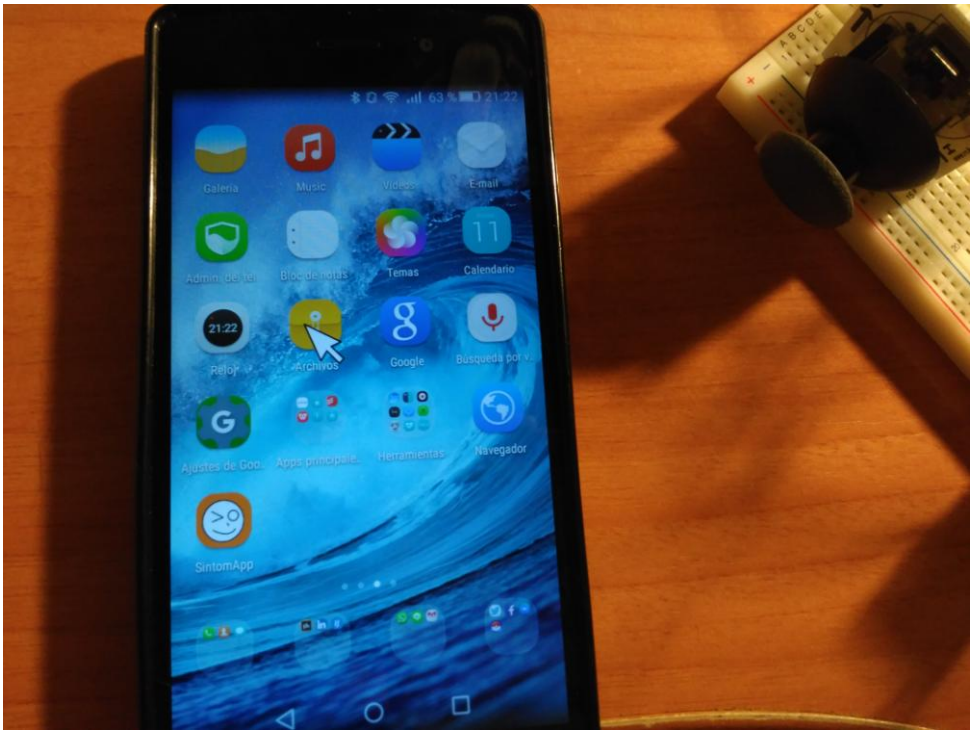

Figura 26. Acceso a aplicaciones ajenas (1)

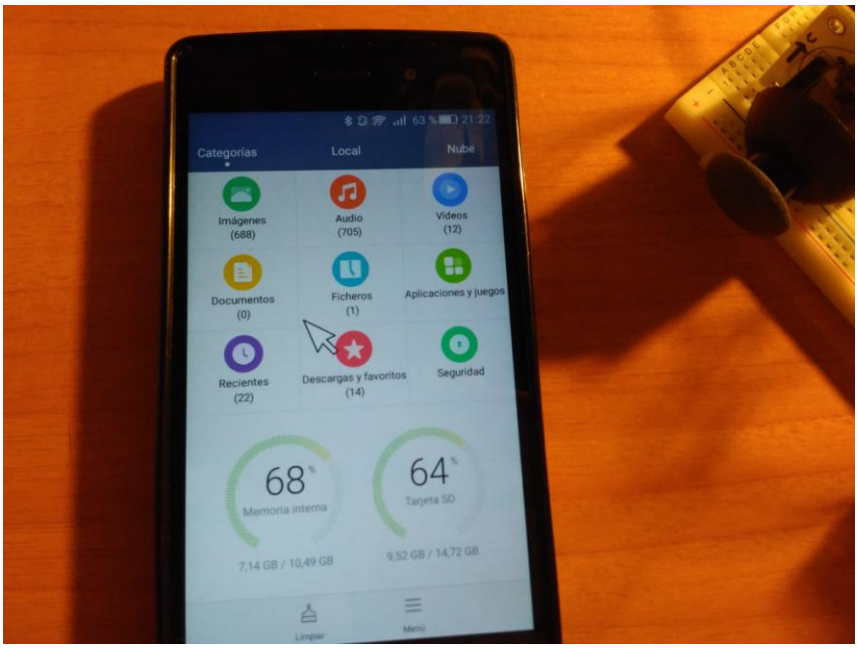

Figura 27. Acceso a aplicaciones ajenas (2)

Una vez iniciado el programa, probamos si el manejo de la aplicación resulta efectiva. Desplazamos el cursor hacia la pestaña “local” y presionamos el botón.

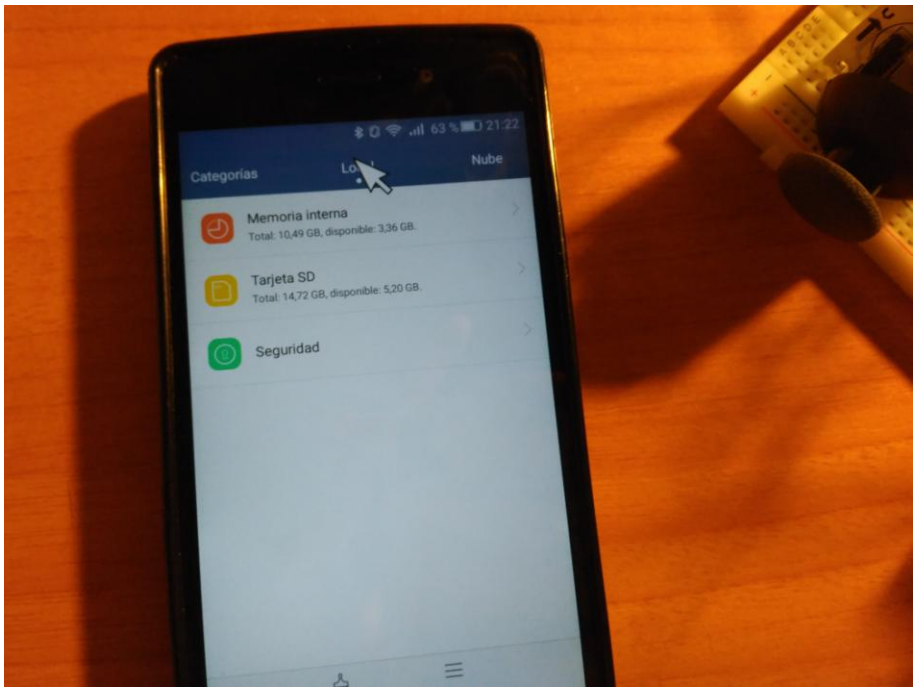

Figura 28. Acceso a aplicaciones ajenas (3)

En vista de los resultados obtenidos, el sistema funciona y no se producen errores en el manejo de la herramienta propuesta en este proyecto.

10 PRESUPUESTOS Y CONCLUSIONES

A continuación se detallan los presupuestos del material necesario para la realización del proyecto. Dependiendo del enfoque dado, los precios podrán variar ligeramente, por ello se ha considerado oportuno analizar cada caso y concretar los resultados en diferentes tablas, de forma que se puedan barajar las diferentes posibilidades ante diferentes situaciones.

10.1 Presupuestos

La primera tabla muestra el presupuesto de los componentes utilizados en el proyecto excluyendo la placa de prueba y los cables, pues se considera que ante el desarrollo del sistema en una PCB no sería necesario adquirir tales materiales.

Tabla 10–1 Presupuesto del proyecto

Componentes necesarios	Precio	Distribuidor
LaunchPad MSP430	16.24 €	Amazon
Módulo Bluetooth HC-05	4.32 €	Taobao
Joystick	2.79 €	Taobao
Botón de gran tamaño	5.31 €	Adafruit
Envolvente (madera)	3.49 €	Creavea
TOTAL	32.15 €	

La segunda tabla tiene un enfoque comercial. Generalmente, este tipo de herramientas no suelen tener fines comerciales debido a los elevados precios que suelen adjuntar, así como la reducida demanda del producto motivada, entre otras cosas, por dicho precio. Sin embargo, este sistema tiene un precio considerablemente barato en comparación a otros productos, por lo que, aunque sea a pequeña escala, se ha considerado adecuada la aportación de tal información.

Para su comercialización, lo único que necesitaría sería un sistema de baterías portables con su correspondiente cargador, por lo que la suma total sólo se diferencia en la inclusión de dicho producto.

Tabla 10–2 Presupuesto del proyecto comercial

Componentes necesarios	Precio	Distribuidor
LaunchPad MSP430	16.24 €	Amazon
Módulo Bluetooth HC-05	4.32 €	Taobao
Joystick	2.79 €	Taobao
Baterías + Cargador	15.24 €	Amazon
Botón de gran tamaño	5.31 €	Adafruit
Envolvente (madera)	3.49 €	Creavea
TOTAL	47.39 €	

No obstante, no hay que olvidar que el Launchpad es una herramienta que facilita la programación de un microcontrolador. En otras palabras, se podría sustituir directamente por un microcontrolador, reduciendo considerablemente los costes. Para asegurar una mayor coherencia con el proyecto descrito y los materiales empleados, el microcontrolador utilizado en los presupuestos será de la misma compañía que el LaunchPad (Texas Instruments); teniendo eso en cuenta, su precio total es el mostrado en la siguiente tabla.

Tabla 10–3 Presupuesto del proyecto comercial sin Launchpad

Componentes necesarios	Precio	Distribuidor
Microcontrolador	1.56 €	Mouser
Módulo Bluetooth HC-05	4.32 €	Taobao
Joystick	2.79 €	Taobao
Botón de gran tamaño	5.31 €	Adafruit
Baterías + Cargador	15.24 €	Amazon
Envolvente (madera)	3.49 €	Creavea
TOTAL	32.71 €	

10.2 Trabajos futuros

Como bien se ha reflejado en varias ocasiones, este proyecto presenta limitaciones y sus prestaciones no son perfectas. Es por ello que resulta interesante remarcar las opciones que convendría mejorar o estudiar posibles mejoras para ofrecer soluciones más eficaces.

- Actualmente, las opciones de mensajería implantadas son suficientes, pero no por ello dejan de ser reducidas. Implementar un método más versátil (y complejo a nivel de diseño) como reconstrucciones de frases (planteado y desarrollado en algunas aplicaciones del estado del arte reflejadas en este documento) podría ser interesante.

- Añadir nuevas funcionalidades en cuanto a manipulación del sistema para facilitar su uso y mejorar la experiencia del usuario también es algo importante. La propuesta realizada es eficaz y de coste reducido, pero no es la única alternativa. Hay que estudiar otras opciones.
- La comunicación bluetooth, a pesar de su eficacia y las ventajas que aporta, sigue sufriendo el gran inconveniente del elevado nivel de consumo (50mA) considerando que el LaunchPad posee un microcontrolador de ultra bajo consumo (230 µA en modo activo, 0.5 µA en modo standby y 0.1 µA en OFF). Si se usaran dos pilas AA como alimentación (1.5 V y capacidad típica de 1100-3000 mAh cada una), se tendría aproximada e idealmente que la duración de descarga de batería sería de 57.36 horas como mucho. Si otros medios de comunicación aportan más ventajas en este sentido, se podrían estudiar los diferentes casos.
- Actualmente hay muchos dispositivos que llevan incorporados sistemas bluetooth. Sería interesante añadir funcionalidades en cuanto a interactividad con estos elementos y generar de esta manera una mayor accesibilidad generalizada.
- La sencillez de la aplicación hace que sea de uso intuitivo para el paciente, pero peca de funcionalidades reducidas muy centradas en su principal objetivo: proporcionar accesibilidad y aportar ayudas para establecer una comunicación. La mejora de la App puede aportar satisfacción y diversidad de ajustes y personalización a los usuarios, por lo que es necesario estudiar nuevas opciones para incluir en la misma.
- Actualmente, el manejo del cursor es mejorable. No es posible aportar la funcionalidad de volver a la ventana anterior sin "rootear" el teléfono; la información que toma la aplicación de la ventana para situar el cursor y hacer el efecto de click no incluye elementos flotantes (hace click al elemento que se encuentre justo debajo de los mismos); tampoco se ha incorporado la opción de desplazar las ventanas hacia la izquierda o la derecha, por lo que sólo se puede interactuar con el menú donde se ubique la aplicación. Sigue habiendo limitaciones y es necesario corregir los errores y mejorar la experiencia del paciente.
- Una posible mejora sería la de implantar un método de autoconexión permanente, sin necesidad de recurrir a nuevas búsquedas o reconexiones manuales.
- Las versiones de Android limitan las funcionalidades de las aplicaciones. En concreto, para versiones 5 de Android la aplicación funciona con normalidad; no obstante, para la versión 6 surgen problemas con la activación del cursor (la aplicación se cierra al llevar a cabo tal acto). Esto es debido a que la gestión de permisos se realiza de forma diferente en cada versión del sistema operativo, por lo que no es posible, en primera estancia, activar el cursor, ya que el permiso requerido está considerado como peligroso o de riesgo. Habría que reprogramar esa sección añadiendo dos o tres nuevas funciones que permitieran utilizarlo. También sería interesante estudiar si la aplicación pudiera realizarse de forma menos dependiente del sistema operativo para evitar este tipo de incidentes.

10.3 Conclusiones

En vista de los resultados, se puede comprobar que no es necesario recurrir a grandes inversiones para aportar prestaciones biomédicas, que no es necesario buscar soluciones de elevado coste para solventar problemas y que las desventajas que se puedan encontrar en la solución propuesta se ven contrarrestadas con las ventajas que suponen. Claramente, y atendiendo a las opciones halladas en la búsqueda del estado del arte, puede que esta no sea la solución más cómoda, confortable o sofisticada, pero proporciona elementos muy positivos. No permitirá construir frases o escribir libros, pero cubre las necesidades básicas de los pacientes en términos de comunicación interpersonal y añade una vía de accesibilidad a uno de los dispositivos más utilizados de todo el mundo, que son los smartphones. Y lo más importante es que esta solución, a diferencia de otras opciones estudiadas y ofrecidas durante estos años, resulta rentable para prácticamente toda la población de cualquier clase social, permitiendo que cualquiera que necesitase de estas prestaciones pudiera tener acceso a ellas sin dilemas económicos.

Dicho esto, se concluye con que se cumple el objetivo de este proyecto, que es aportar una solución de bajo coste pero suficientemente eficaz como para estar a la altura de las expectativas, siendo accesible para cualquier persona que requiera de su uso.

REFERENCIAS BIBLIOGRÁFICAS

[1] Argento, A. (2015). OTTAA: un puente para las personas con discapacidad en la comunicación desde Córdoba para todo el mundo. *Por Igual Más*. Retrieved from http://www.porigualmas.org/articles/1123/ottaa-un-puente-para-las-personas-con-discapacidad-en-la-comunicaci-n-desde-c-rdoba-para-todo-el-mundo

[2] Bacher, D., Jarosiewicz, B., Masse, N., Stavisky, S., Simeral, J., & Newell, K. et al. (2014). Neural Point-and-Click Communication by a Person With Incomplete Locked-In Syndrome. *Neurorehabilitation And Neural Repair*, *29*(5), 462-471. http://dx.doi.org/10.1177/1545968314554624

[3] Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., & Kübler, A. et al. (1999). A spelling device for the paralysed. *Nature*, *398*(6725), 297-298. http://dx.doi.org/10.1038/18581

[4] Brownell, C. (2015). Eye-tracking technology allows paralyzed people to control PCs, tablets. *Financial Post*, p. http://business.financialpost.com/. Retrieved from http://business.financialpost.com/fp-tech-desk/how-mind-controlled-tablet-technology-could-help-people-with-paralysis-communicate?__lsa=815a-132c

[5] Centro de Desarrollo de Tecnologías de Inclusión (CEDETi UC),. (2016). *T13: Alberto Vega, la tecnología y el síndrome de enclaustramiento*. Retrieved from https://www.youtube.com/watch?v=jkyrUugOwNY

[6] Ciudadanos en Red,. (2015). Mexicano crea sistema para comunicarse con personas con parálisis cerebral. *Mientras tanto en México*. Retrieved from http://www.mientrastantoenmexico.mx/27531/2015/11/21/mexicano-crea-sistema-para-comunicarse-con-personas-con-paralisis-cerebral/

[7] CSIC,. (2016). *Un proyecto para controlar ordenadores mediante la mirada y el reconocimiento facial recibe casi 700.000 euros* (p. http://www.csic.es/). Madrid: Departamento de Comunicación.

[8] Dotinga, R. (2014). Technology Helps 'Locked-In' Stroke Patient Communicate. *Healthday*, p. https://consumer.healthday.com/. Retrieved from https://consumer.healthday.com/cognitive-health-information-26/lou-gehrig-s-disease-als-news-1/technology-helps-locked-in-stroke-patient-communicate-692508.html

[9] Horowitz, B. (2015). Eye-tracking technology allows paralyzed people to control PCs, tablets. *DELL*, p. https://powermore.dell.com/. Retrieved from https://powermore.dell.com/business/eye-tracking-technology-allows-paralyzed-people-to-control-pcs-tablets/

[10] *Jóvenes empresarios que desarrollan tecnología para la inclusión - Universidad Blas Pascal*. (2014). *Universidad Blas Pascal*. Retrieved 5 June 2016, from http://www.ubp.edu.ar/novedades/jovenes-empresarios-que-desarrollan-tecnologia-para-la-inclusion/

[11] Kashihara, K. (2014). A brain-computer interface for potential non-verbal facial communication based on EEG signals related to specific emotions. *Front. Neurosci.*, *8*. http://dx.doi.org/10.3389/fnins.2014.00244

[12] Klose, C. (2007). Connections that Count: Brain-Computer Interface Enables the Profoundly Paralyzed to Communicate. *National Institute Of Health*, (Volume 2 Number 3 Pages 20 - 21), https://www.nlm.nih.gov/. Retrieved from https://www.nlm.nih.gov/medlineplus/magazine/issues/summer07/articles/summer07pg20-21.html

[13] Lahr, J., Schwartz, C., Heimbach, B., Aertsen, A., Rickert, J., & Ball, T. (2015). Invasive brain–machine interfaces: a survey of paralyzed patients' attitudes, knowledge and methods of information retrieval. *J. Neural Eng.*, *12*(4), 043001. http://dx.doi.org/10.1088/1741-2560/12/4/043001

[14] Lancioni, G., Singh, N., O'Reilly, M., Sigafoos, J., Ferlisi, G., & Ferrarese, G. et al. (2012). Technology-aided programs for assisting communication and leisure engagement of persons with amyotrophic lateral sclerosis: Two single-case studies. *Research In Developmental Disabilities*, *33*(5), 1605-1614. http://dx.doi.org/10.1016/j.ridd.2012.03.028

[15] Lancioni, G., Singh, N., O'Reilly, M., Sigafoos, J., Ferlisi, G., & Ferrarese, G. et al. (2013). A voice-sensitive microswitch for a man with amyotrophic lateral sclerosis and pervasive motor impairment. *Disability And Rehabilitation: Assistive Technology*, *9*(3), 260-263. http://dx.doi.org/10.3109/17483107.2013.785037

[16] Magee, J., Betke, M., Gips, J., Scott, M., & Waber, B. (2008). A Human–Computer Interface Using Symmetry Between Eyes to Detect Gaze Direction. *IEEE Transactions On Systems, Man, And Cybernetics - Part A: Systems And Humans*, *38*(6), 1248-1261. http://dx.doi.org/10.1109/tsmca.2008.2003466

[17] Ramesh, M., Asok Nair, A., & Menon, K. (2011). WISION- Wireless Interface System for Interpretation of Ocular Symbols from People with Neuromuscular Diseases. *2011 7Th International Conference On Wireless Communications, Networking And Mobile Computing*, 1 - 5. http://dx.doi.org/10.1109/wicom.2011.6040615

[18] Redacción LAVOZ,. (2015). Jóvenes cordobeses desarrollan aplicación de comunicación para personas con autismo. *La Voz*, p. http://www.lavoz.com.ar/. Retrieved from http://www.lavoz.com.ar/ciudadanos/jovenes-cordobeses-desarrollan-aplicacion-de-comunicacion-para-personas-con-autismo

[19] Torres Cautivo, X. (2013). Cuando la tecnología conmueve. *Terra*, p. https://noticias.terra.es/. Retrieved from http://noticias.terra.es/tecnologia/cuando-la-tecnologia-conmueve,6de65c6ab52a0410VgnVCM4000009bcceb0aRCRD.html

[20] Revelo, J. (2015). Tutorial para crear un servicio en Android. Hermosa Programación. http://www.hermosaprogramacion.com/. Retrieved from http://www.hermosaprogramacion.com/2015/07/tutorial-para-crear-un-servicio-en-android/

[21] Link Assistive Technology. (2016). *Products*. http://linkassistive.com/. Retrieved from http://linkassistive.com/products/

[22] Up to Down. Android Studio. https://android-studio.uptodown.com/. Retrieved from https://android-studio.uptodown.com/windows

[23] Energia. http://energia.nu/. Retrieved from http://energia.nu/download/

[24] SodelsCot. http://www.sodels.com/. Retrieved from http://www.sodels.com/loquendo.htm

[25] UUID Generator. Online UUID Generator. Retrieved from https://www.uuidgenerator.net/

[26] Google. Google Play. https://play.google.com/store. Retrieved from https://play.google.com/store/apps/details?id=com.stonefacesoft.ottaa&hl=es

Anexo A: Código aplicación Android

- Código "Bien.java".

```
/*********************************************************************************
 *  ////////////////////////////////////                                         *
 *  // Autor: Francisco Beltrán        //                                        *
 *  // País: España                    //                                        *
 *  // Fecha finalización: 18/08/2016  //                                        *
 *  ////////////////////////////////////                                         *
 *                                                                               *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.                   *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                             *
 *                                                                               *
 * «Sistema integral para apoyo a la comunicación personal                       *
 * en pacientes con diversidad funcional motora: SintomApp».                     *
 *                                                                               *
 * Bien.java: Código correspondiente al botón «Bien».                            *
 *                                                                               *
 * El paciente podrá indicar que se encuentra bien.                              *
 *                                                                               *
*********************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;

import com.communicationsupport.kikobc.myapplication.R;

public class Bien extends AppCompatActivity {

    ImageView foto;
    TextView texto;
    Animation transparencia;
    private long ms=0;
    private long splashTime=5000;
    private boolean splashActive = true;
    private boolean paused=false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bien);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        /** La imagen actuará como un botón */

        foto = (ImageView) findViewById(R.id.foto);
        texto = (TextView) findViewById(R.id.texto);
```

```
        /** Efecto de transparencia / parpadeo */

        transparencia = AnimationUtils.loadAnimation(this,
R.anim.transparentar);

        /** Activar animación. */

        foto.startAnimation(transparencia);
        texto.startAnimation(transparencia);

        /** Tras 5 segundos, vuelve a la actividad "Remoto.java" */

        Thread mythread = new Thread() {
            public void run() {
                try {
                    while (splashActive && ms < splashTime) {
                        if(!paused)
                            ms=ms+100;
                        sleep(100);
                    }
                } catch(Exception e) {}
                finally {
                    Intent intent = new Intent(Bien.this, Remoto.class);
                    startActivity(intent);
                }
            }
        };
        mythread.start();
    }
}
```

Los ficheros referentes a los mensajes de la aplicación (bien, dolor, hambre, sed, incomodo, sucio, tq, si y no) contienen exactamente el mismo código, con la salvedad de que el contenido tomado por los archivos xml son distintos. Por ello, se refleja únicamente "Bien.java", para evitar redundancia de código.

- Código "ConexionBT.java".

```
/*************************************************************************
 *  ////////////////////////////////////                                 *
 *  // Autor: Francisco Beltrán         //                               *
 *  // País: España                     //                               *
 *  // Fecha finalización: 18/08/2016   //                               *
 *  ////////////////////////////////////                                 *
 *                                                                       *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.           *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                     *
 *                                                                       *
 * «Sistema integral para apoyo a la comunicación personal               *
 * en pacientes con diversidad funcional motora: SintomApp».             *
 *                                                                       *
 * ConexionBT.java: Código correspondiente a la conexión Bluetooth entre *
 * dispositivos.                                                         *
 *                                                                       *
 * El paciente podrá realizar afirmaciones o aportar respuestas afirmativas.*
 *                                                                       *
*************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.app.ActivityManager;
```

```
import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;

public class ConexionBT extends Service {

    /** UUID necesario para realizar una conexión única de la app. */

    private static final UUID SerialPortServiceClass_UUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private boolean mAllowInsecureConnections;
    private static final String TAG = ConexionBT.class.getName();
    private BluetoothAdapter mBluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
    private ConnectedThread connected;
    private boolean isConnected = false;
    public static String mensaje;
    int aux = 0;
    TimerTask timerTask;


    public ConexionBT() {
        mAllowInsecureConnections=true;
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {

        if(aux == 0) {
            mBluetoothAdapter.enable();
            Toast.makeText(this, "Servicio Bluetooth activado",
Toast.LENGTH_SHORT).show();
            aux = 1;
        }else if(aux == 1){
            Toast.makeText(this, "Conectando con dispositivo",
Toast.LENGTH_SHORT).show();
            aux = 2;

            /** Conexión en segundo plano */

            final ActivityManager.MemoryInfo memoryInfo = new
ActivityManager.MemoryInfo();
            final ActivityManager activityManager =
```

```
                    (ActivityManager) getSystemService(ACTIVITY_SERVICE);

            Timer timer = new Timer();

            timerTask = new TimerTask() {
                @Override
                public void run() {
                    activityManager.getMemoryInfo(memoryInfo);
                    // 1048576 MB es la cantidad máxima de memoria RAM
disponible
                    String availMem = memoryInfo.availMem / 1048576 + "MB";

                    Log.d(TAG, availMem);

                    Intent localIntent = new
Intent(Constantes.ACTION_RUN_SERVICE)
                            .putExtra(Constantes.EXTRA_MEMORY, availMem);

                    // Emitir el intent a la actividad
                    LocalBroadcastManager.

getInstance(ConexionBT.this).sendBroadcast(localIntent);
                }
            };

            timer.scheduleAtFixedRate(timerTask, 0, 1000);
            Conecta();
        }else if(aux == 2){
            mBluetoothAdapter.disable();
            Toast.makeText(this, "Servicio Bluetooth desactivado",
Toast.LENGTH_SHORT).show();
            aux = 0;
        }
        return START_STICKY;
    }


    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }


    public void onDestroy(){
        super.onDestroy();
        timerTask.cancel();
        Intent localIntent = new Intent(Constantes.ACTION_MEMORY_EXIT);

        /** Emitir el intent a la actividad */

        LocalBroadcastManager.
                getInstance(ConexionBT.this).sendBroadcast(localIntent);

        Log.d(TAG, "Servicio destruido...");
    }

    /** Conexión con dispositivo Bluetooth. Dirección particular
        para autoconexión: 98:D3:31:40:80:AC */

    private void Conecta(){
        String address = "98:D3:31:40:80:AC";
```

```
        BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
        ConnectThread connect = new ConnectThread(device);
        connect.start();
    }

    /** Siguiente proceso de conexión. */

    private class ConnectThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final BluetoothDevice mmDevice;

        public ConnectThread(BluetoothDevice device) {

            /** Objetos temporales para asignar a mmSocket */

            BluetoothSocket tmp = null;
            mmDevice = device;

            /** Consigue un socket bluetooth para conectar con el
dispositivo. */

            try {

                Method m = device.getClass().getMethod("createRfcommSocket",
new Class[] {int.class});
                tmp = (BluetoothSocket) m.invoke(device, 1);

                if ( mAllowInsecureConnections ) {
                    Method method;

                    method =
device.getClass().getMethod("createRfcommSocket", new Class[] { int.class }
);
                    tmp = (BluetoothSocket) method.invoke(device, 1);
                }
                else
                    tmp = device.createRfcommSocketToServiceRecord(
SerialPortServiceClass_UUID );


            } catch (Exception e) {
                Log.e(TAG, "create() failed", e);
            }
            mmSocket = tmp;
        }

        public void run() {

            /** Cancelar descubrimiento de dispositivos, ya que ralentizaría
la conexión */
            mBluetoothAdapter.cancelDiscovery();

            try {

                /** Conecta el dispositivo mediante el socket. Se bloqueará
                    hasta que lo consiga o hasta que salte una excepción. */

                boolean b = mmSocket.isConnected(); System.out.println(b);
                mmSocket.connect();
                isConnected = true;
               // Toast.makeText(ConexionBT.this, "Conexión establecida",
Toast.LENGTH_SHORT).show();
```

```
            } catch (IOException connectException) {

                /** Si resulta imposible conectar, cierra el socket y sale.
*/

                System.out.println("Imposible de conectar.");
                try {
                    mmSocket.close();
                } catch (IOException closeException) { }
                return;
            }

            connected = new ConnectedThread(mmSocket);
            connected.run();
        }

        /** Cancela una conexión en proceso y cierra el socket */

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
        }
    }

    /** Transmisión de mensajes */

    private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private int ClickRemoto;

        public ConnectedThread(BluetoothSocket socket) {

            /** Leemos la información recibida */

            mmSocket = socket;
            InputStream tmpIn = null;
            try {
                tmpIn = socket.getInputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
        }

        public void run() {
            /** Buffer para la cadena de información recibida */

            byte[] buffer = new byte[1];

            /** Escucha la entrada hasta que salte alguna excepción. */

            while (true) {
                try {
                    ClickRemoto = Remoto.Autoclick;
                    if (ClickRemoto == 1){

mmSocket.getOutputStream().write("1".toString().getBytes());

                    }
                    else if (ClickRemoto == 2){
```

```
mmSocket.getOutputStream().write("2".toString().getBytes());
                    }
                    else if (ClickRemoto == 0){

mmSocket.getOutputStream().write("0".toString().getBytes());
                    }

                    /** Lee la cadena de entrada */

                    mmInStream.read(buffer);
                    mensaje = new String(buffer, "UTF-8");

                } catch (IOException e) {
                    break;
                }
            }
        }

        /** Llamada desde la actividad Remoto para cerrar la conexión */

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
        }
    }

}
```

- Código “Remoto.java”.

```
/*****************************************************************************
 *  ////////////////////////////////////                                     *
 *  // Autor: Francisco Beltrán          //                                  *
 *  // País: España                      //                                  *
 *  // Fecha finalización: 18/08/2016    //                                  *
 *  ////////////////////////////////////                                     *
 *                                                                           *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.               *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                         *
 *                                                                           *
 * «Sistema integral para apoyo a la comunicación personal                   *
 * en pacientes con diversidad funcional motora: SintomApp».                 *
 *                                                                           *
 * Remoto.java: Inicia la actividad correspondiente al control remoto del    *
 * teléfono.                                                                 *
 *                                                                           *
*****************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.content.Intent;
import android.media.AudioManager;
import android.support.v7.widget.Toolbar;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageButton;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.communicationsupport.kikobc.myapplication.R;
```

```
public class Remoto extends AppCompatActivity {

    ImageButton boton_dolor, boton_hambre, boton_incomodo, boton_sed;
    ImageButton boton_sucio, boton_bien, boton_tq, boton_si, boton_no;
    Button bHide;
    SoundManager sound;
    int bien0, dolor0, hambre0, incomodo0, no0, sed0, si0, sucio0, tq0;
    public static int Autoclick;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_remoto);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDefaultDisplayHomeAsUpEnabled(true);

/** Parte de sonido de la App */

        /** Creamos una instancia de SoundManager */

        sound = new SoundManager(getApplicationContext());

        /** Ajustamos el volumen al teléfono */

        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);

        /** Lee los sonidos que figuran en el directorio res/raw */

        bien0 = sound.load(R.raw.bien);
        dolor0 = sound.load(R.raw.dolor);
        hambre0 = sound.load(R.raw.hambre);
        incomodo0 = sound.load(R.raw.incomodo);
        no0 = sound.load(R.raw.no);
        sed0 = sound.load(R.raw.sed);
        si0 = sound.load(R.raw.si);
        sucio0 = sound.load(R.raw.sucio);
        tq0 = sound.load(R.raw.tq);


/** Botones */

        boton_dolor = (ImageButton) findViewById(R.id.Boton_Dolor);
        boton_dolor.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                /** Intent sirve para cambiar de un activity a otro.
                 Sound.play() sirve para reproducir el sonido. */

                Intent intent = new Intent(Remoto.this, Dolor.class);
                startActivity(intent); //Para lanzarlo
                sound.play(dolor0);
            }
        });

        boton_hambre = (ImageButton) findViewById(R.id.Boton_Hambre);
        boton_hambre.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            Intent intent = new Intent(Remoto.this, Hambre.class);
            startActivity(intent); //Para lanzarlo
            sound.play(hambre0);
        }
    });

    boton_incomodo = (ImageButton) findViewById(R.id.Boton_Incomodo);
    boton_incomodo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(Remoto.this, Incomodo.class);
            startActivity(intent); //Para lanzarlo
            sound.play(incomodo0);
        }
    });

    boton_sed = (ImageButton) findViewById(R.id.Boton_Sed);
    boton_sed.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(Remoto.this, Sed.class);
            startActivity(intent); //Para lanzarlo
            sound.play(sed0);
        }
    });

    boton_sucio = (ImageButton) findViewById(R.id.Boton_Sucio);
    boton_sucio.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(Remoto.this, Sucio.class);
            startActivity(intent); //Para lanzarlo
            sound.play(sucio0);
        }
    });

    boton_bien = (ImageButton) findViewById(R.id.Boton_Bien);
    boton_bien.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(Remoto.this, Bien.class);
            startActivity(intent); //Para lanzarlo
            sound.play(bien0);
        }
    });

    boton_tq = (ImageButton) findViewById(R.id.Boton_TQ);
    boton_tq.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(Remoto.this, TQ.class);
            startActivity(intent); //Para lanzarlo
            sound.play(tq0);
        }
    });

    boton_si = (ImageButton) findViewById(R.id.Boton_Si);
    boton_si.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(Remoto.this, Si.class);
                startActivity(intent); //Para lanzarlo
                sound.play(si0);
            }
        });

        boton_no = (ImageButton) findViewById(R.id.Boton_No);
        boton_no.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(Remoto.this, No.class);
                startActivity(intent); //Para lanzarlo
                sound.play(no0);
            }
        });


        bHide = (Button) findViewById(R.id.Button_Control);
        bHide.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent newActivity = new Intent(Intent.ACTION_MAIN);
                newActivity.addCategory(Intent.CATEGORY_HOME);
                newActivity.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(newActivity);
            }
        });
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        int id = item.getItemId();

        if (id == R.id.action_autoclick1){

/** Autoclick con sonido */

            Autoclick = 2;
            return true;
        }
        else if (id == R.id.action_autoclick2){

/** Autoclick sin sonido */

            Autoclick = 1;
            return true;
        }
        else if (id == R.id.action_autoclick3){

/** Autoclick desactivado */
```

```
            Autoclick = 0;
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

- Código "SoundManager.java".

```
/*****************************************************************************
 *  ////////////////////////////////////                                     *
 *  // Autor: Francisco Beltrán        //                                     *
 *  // País: España                    //                                     *
 *  // Fecha finalización: 18/08/2016  //                                     *
 *  ////////////////////////////////////                                     *
 *                                                                           *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.               *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                         *
 *                                                                           *
 * «Sistema integral para apoyo a la comunicación personal                   *
 * en pacientes con diversidad funcional motora: SintomApp».                 *
 *                                                                           *
 * SoundManager.java: Código correspondiente al manejo de los sonidos y      *
 * mensajes sonoros.                                                         *
 *                                                                           *
*****************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.content.Context;
import android.media.AudioManager;
import android.media.SoundPool;

public class SoundManager {

    private Context pContext;
    private SoundPool sndPool;
    private float rate = 1.0f;
    private float leftVolume = 1.0f;
    private float rightVolume = 1.0f;


    /** La clase SoundPool administra y ejecuta todos los recursos de audio
de la aplicación.
        Nuestro constructor determina la configuración de audio del contexto
de nuestra app. */

    public SoundManager(Context appContext)
    {
        sndPool = new SoundPool(16, AudioManager.STREAM_MUSIC, 100);
        pContext = appContext;
    }

    /** Obtiene el sonido y retorna el id del mismo */

    public int load(int idSonido)
    {
        return sndPool.load(pContext, idSonido, 1);
    }

    /**Ejecuta el sonido, toma como parámetro el id del sonido a ejecutar. */

    public void play(int idSonido)
```

```
    {
        sndPool.play(idSonido, leftVolume, rightVolume, 1, 0, rate);
    }

    /** Libera memoria de todos los objetos del sndPool que ya no son
requeridos. */

    public void unloadAll()
    {
        sndPool.release();
    }
}
```

- Código “MouseEvent.java”.

```
/*****************************************************************************
 *  ////////////////////////////////////                                     *
 *  // Autor: Francisco Beltrán        //                                     *
 *  // País: España                    //                                     *
 *  // Fecha finalización: 18/08/2016  //                                     *
 *  ////////////////////////////////////                                     *
 *                                                                           *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.               *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                         *
 *                                                                           *
 * «Sistema integral para apoyo a la comunicación personal                   *
 * en pacientes con diversidad funcional motora: SintomApp».                 *
 *                                                                           *
 * MouseEvent.java: Recoge la correspondencia de acciones del joystick.      *
 *                                                                           *
*****************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

public class MouseEvent {

    public static final int
            ARRIBA = 0,
            ABAJO = 1,
            IZQUIERDA = 2,
            DERECHA = 3,
            CLICK = 4,
            STOP = 5,
            CLICKAUTO = 6;

    public final int direccion;

    public MouseEvent(int direccion) {
        this.direccion = direccion;
    }
}
```

- Código “MouseAccessibilityService.java”.

```
/*****************************************************************************
 *  ////////////////////////////////////                                     *
 *  // Autor: Francisco Beltrán        //                                     *
 *  // País: España                    //                                     *
 *  // Fecha finalización: 18/08/2016  //                                     *
 *  ////////////////////////////////////                                     *
 *                                                                           *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.               *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                         *
 *                                                                           *
```

```
 * «Sistema integral para apoyo a la comunicación personal                    *
 * en pacientes con diversidad funcional motora: SintomApp».                  *
 *                                                                            *
 * MouseAccessibilityService.java: Código correspondiente al control del      *
 * cursor.                                                                    *
 *                                                                            *
*****************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.accessibilityservice.AccessibilityService;
import android.content.Intent;
import android.graphics.PixelFormat;
import android.graphics.Rect;
import android.os.Handler;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.view.WindowManager;
import android.view.WindowManager.LayoutParams;
import android.view.accessibility.AccessibilityEvent;
import android.view.accessibility.AccessibilityNodeInfo;
import android.widget.Toast;

import com.communicationsupport.kikobc.myapplication.R;


public class MouseAccessibilityService extends AccessibilityService {

    private static final String TAG =
MouseAccessibilityService.class.getName();

    private View cursorView;
    private LayoutParams cursorLayout;
    private WindowManager windowManager;
    private static int aux = 0;
    private static int aux2 = 0;
    String MoveM = ConexionBT.mensaje;
    SoundManager sound;
    int sms;


    private static void logNodeHierachy(AccessibilityNodeInfo nodeInfo, int
depth) {
        Rect bounds = new Rect();    // Creamos un rectangulo
        nodeInfo.getBoundsInScreen(bounds); //Lo situamos en la pantalla

        StringBuilder sb = new StringBuilder(); // Constructor de cadenas
        if (depth > 0) {
            for (int i=0; i<depth; i++) {
                sb.append("   ");
            }
            sb.append("\u2514 ");    // añade al final de la cadena esa
información.
        }
        sb.append(nodeInfo.getClassName());
        sb.append(" (" + nodeInfo.getChildCount() +  ")");  // Toma los
números del "Child"
        sb.append(" " + bounds.toString()); // Representación en formato
cadena.
        if (nodeInfo.getText() != null) {
            sb.append(" - \"" + nodeInfo.getText() + "\""); // Tomamos
información en formato texto.
```

```
        }
        Log.v(TAG, sb.toString());  // TAG fuente para hacer Log;
sb.toString() el mensaje a loggear

        for (int i=0; i<nodeInfo.getChildCount(); i++) {
            AccessibilityNodeInfo childNode = nodeInfo.getChild(i);
            if (childNode != null) {
                logNodeHierachy(childNode, depth + 1);
            }
        }
    }

    private static AccessibilityNodeInfo
findSmallestNodeAtPoint(AccessibilityNodeInfo sourceNode, int x, int y) {
        Rect bounds = new Rect();       // Rectangulo
        sourceNode.getBoundsInScreen(bounds);   //Se inserta en la ubicacion
del nodo en la pantalla

        if (!bounds.contains(x, y)) {       // sin ejes no se va a ninguna
parte xD
            return null;
        }

        for (int i=0; i<sourceNode.getChildCount(); i++) {
            AccessibilityNodeInfo nearestSmaller =
findSmallestNodeAtPoint(sourceNode.getChild(i), x, y);
            if (nearestSmaller != null) {   // se busca el punto más cercano
a los ejes x,y
                return nearestSmaller;
            }
        }
        return sourceNode;
    }

    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
    }

    @Override
    public void onInterrupt() {
        Log.d(TAG, "onInterrupt");
    }

    @Override
    public void onCreate() {
        super.onCreate();

        /** Creamos una instancia de SoundManager */

        sound = new SoundManager(getApplicationContext());


        /** Lee los sonidos que figuran en el directorio res/raw */

        sms = sound.load(R.raw.sms);

/** Se genera el puntero; cursorView pone la imagen del ratón; cursorLayout
se ocupa del
            movimiento del mismo y su ubicación. */

        cursorView = View.inflate(getBaseContext(), R.layout.cursor, null);
        cursorLayout = new LayoutParams(
```

```
                LayoutParams.WRAP_CONTENT,
                LayoutParams.WRAP_CONTENT,
                LayoutParams.TYPE_SYSTEM_ERROR,
                LayoutParams.FLAG_DISMISS_KEYGUARD |
LayoutParams.FLAG_NOT_FOCUSABLE | LayoutParams.FLAG_NOT_TOUCHABLE |
LayoutParams.FLAG_LAYOUT_IN_SCREEN,
                PixelFormat.TRANSLUCENT);
        cursorLayout.gravity = Gravity.TOP | Gravity.LEFT;
        cursorLayout.x = 200;
        cursorLayout.y = 200;

/** Toma servicio del "window manager" */

        windowManager = (WindowManager) getSystemService(WINDOW_SERVICE);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {

                    /** Leemos la información recibida por el Launchpad */

                    MoveM = ConexionBT.mensaje;

                    /** Interpretamos los datos recibidos */

                    if (MoveM != null) {

                        String message = new String(MoveM);
                        final int event = Integer.parseInt(message);
                        new Handler(getMainLooper()).post(new Runnable() {
                            @Override
                            public void run() {
                                onMouseMove(new MouseEvent(event));
                            }

                        });
                    }

                    /** Damos un pequeño margen de tiempo para su
procesamiento */

                    try {
                        doLongOperation();
                    } catch (final Exception ex) {
                        Log.i("---", "Exception in thread");
                    }
                }
            }
        }).start();
    }

/** Tiempo de espera para procesar la información */

    protected void doLongOperation() {
        try {
            Thread.sleep(5);
        } catch (InterruptedException e) {
        }
    }

    @Override
    public void onDestroy() {
```

```
        super.onDestroy();

        if (windowManager != null && cursorView != null) {
            windowManager.removeView(cursorView);
        }
    }


/** Efecto de click */

     private void click() {
         Log.d(TAG, String.format("Click [%d, %d]", cursorLayout.x,
cursorLayout.y));
         AccessibilityNodeInfo nodeInfo = getRootInActiveWindow();

        if (nodeInfo == null) {
            System.out.println("No se detecta ventana");
            return;
        }
        AccessibilityNodeInfo nearestNodeToMouse =
findSmallestNodeAtPoint(nodeInfo, cursorLayout.x, cursorLayout.y + 50);
        if (nearestNodeToMouse != null) {
            logNodeHierachy(nearestNodeToMouse, 0);

nearestNodeToMouse.performAction(AccessibilityNodeInfo.ACTION_CLICK);
        }
        nodeInfo.recycle();
    }


/** Movimiento del ratón y activación de click */

    public void onMouseMove(final MouseEvent event) {

        switch (event.direccion) {
            case MouseEvent.IZQUIERDA:
                cursorLayout.x = cursorLayout.x - 2;
                break;
            case MouseEvent.DERECHA:
                cursorLayout.x = cursorLayout.x + 2;
                break;
            case MouseEvent.ARRIBA:
                cursorLayout.y = cursorLayout.y - 2;
                break;
            case MouseEvent.ABAJO:
                cursorLayout.y = cursorLayout.y + 2;
                break;
            case MouseEvent.CLICK:
                if(aux == 0) {
                    click();
                    aux = 1;
                    aux2 = 0;
                }
                break;
            case MouseEvent.STOP:
                if(aux == 1) {
                    aux = 0;
                    aux2 = 0;
                }
                break;
            case MouseEvent.CLICKAUTO:
                if(aux2 == 0){
```

```
                    aux2 = 1;
                    sound.play(sms);
                }
                break;
            default:
                break;
        }
        windowManager.updateViewLayout(cursorView, cursorLayout);
    }

    /** Al comienzo del servicio... */

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        if(aux == 0) {
            windowManager.addView(cursorView, cursorLayout);
            Toast.makeText(this, "Cursor activado",
Toast.LENGTH_SHORT).show();
            aux = 1;
        }else if(aux == 1){
            windowManager.removeView(cursorView);
            Toast.makeText(this, "Cursor desactivado",
Toast.LENGTH_SHORT).show();
            aux = 0;
        }
        return START_STICKY;
    }
}
```

- Código “Constantes.java”.

```
/*********************************************************************
 * Constantes.java: Código que recoge constantes utilizadas en       *
 * «MemoryService.java» y «ProgressIntentService.java».              *
*********************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

public class Constantes {

    /** Constantes para {@link MemoryService} */

    public static final String ACTION_RUN_SERVICE =
"com.communicationsupport.kikobc.myapplication.Activities.action.RUN_SERVICE"
;
    public static final String ACTION_MEMORY_EXIT =
"com.communicationsupport.kikobc.myapplication.Activities.action.MEMORY_EXIT"
;
    public static final String EXTRA_MEMORY =
"com.communicationsupport.kikobc.myapplication.Activities.extra.MEMORY";


    /** Constantes para {@link ProgressIntentService} */

    public static final String ACTION_RUN_ISERVICE =
"com.communicationsupport.kikobc.myapplication.Activities.action.RUN_INTENT_S
ERVICE";
    public static final String ACTION_PROGRESS_EXIT =
"com.communicationsupport.kikobc.myapplication.Activities.action.PROGRESS_EXI
T";
    public static final String EXTRA_PROGRESS =
"com.communicationsupport.kikobc.myapplication.Activities.extra.PROGRESS";

}
```

- Código “MemoryService.java”.

```
/*****************************************************************************
 * MemoryService.java: Permite, junto a «ProgressIntentService.java»,       *
 * mantener la conexión Bluetooth en segundo plano incluso fuera de la      *
 * aplicación.                                                               *
*****************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.app.ActivityManager;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;

import java.util.Timer;
import java.util.TimerTask;

/**
 * Un {@link Service} que notifica la cantidad de memoria disponible en el
sistema
 */

public class MemoryService extends Service {
    private static final String TAG = MemoryService.class.getSimpleName();
    TimerTask timerTask;

    public MemoryService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }


    @Override
    public void onCreate() {
        Log.d(TAG, "Servicio creado...");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "Servicio iniciado...");


        final ActivityManager.MemoryInfo memoryInfo = new
ActivityManager.MemoryInfo();
        final ActivityManager activityManager =
                (ActivityManager) getSystemService(ACTIVITY_SERVICE);


        Timer timer = new Timer();

        timerTask = new TimerTask() {
            @Override
            public void run() {
                activityManager.getMemoryInfo(memoryInfo);
                String availMem = memoryInfo.availMem / 1048576 + "MB";

                Log.d(TAG, availMem);
```

```
                Intent localIntent = new
Intent(Constantes.ACTION_RUN_SERVICE)
                        .putExtra(Constantes.EXTRA_MEMORY, availMem);

                /** Emitir el intent a la actividad */

                LocalBroadcastManager.

getInstance(MemoryService.this).sendBroadcast(localIntent);
            }
        };

        timer.scheduleAtFixedRate(timerTask, 0, 1000);

        return START_NOT_STICKY;
    }

    @Override
    public void onDestroy() {
        timerTask.cancel();
        Intent localIntent = new Intent(Constantes.ACTION_MEMORY_EXIT);

        /** Emitir el intent a la actividad */

        LocalBroadcastManager.
                getInstance(MemoryService.this).sendBroadcast(localIntent);

        Log.d(TAG, "Servicio destruido...");
    }


}
```

- Código “ProgressIntentService.java”.

```
/*************************************************************************
 * ProgressIntentService.java: Permite, junto a «MemoryService.java»,    *
 * mantener la conexión Bluetooth en segundo plano incluso fuera de la   *
 * aplicación.                                                           *
*************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.app.IntentService;
import android.content.Intent;
import android.support.v4.app.NotificationCompat;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import android.widget.Toast;

/**
 * Un {@link IntentService} que simula un proceso en primer plano
 * <p>
 */
public class ProgressIntentService extends IntentService {
    private static final String TAG =
ProgressIntentService.class.getSimpleName();


    public ProgressIntentService() {
        super("ProgressIntentService");
    }
```

```
    @Override
    protected void onHandleIntent(Intent intent) {
        if (intent != null) {
            final String action = intent.getAction();
            if (Constantes.ACTION_RUN_ISERVICE.equals(action)) {
                handleActionRun();
            }
        }
    }

    /**
     * Maneja la acción de ejecución del servicio
     */
    private void handleActionRun() {
        try {

            /** Se construye la notificación */

            NotificationCompat.Builder builder = new
NotificationCompat.Builder(this)
                    .setSmallIcon(android.R.drawable.stat_sys_download_done)
                    .setContentTitle("Servicio en segundo plano")
                    .setContentText("Procesando...");

            /** Bucle de simulación */

            for (int i = 1; i <= 10; i++) {

                Log.d(TAG, i + ""); // Logueo

                /** Poner en primer plano */

                builder.setProgress(10, i, false);
                startForeground(1, builder.build());

                Intent localIntent = new
Intent(Constantes.ACTION_RUN_ISERVICE)
                        .putExtra(Constantes.EXTRA_PROGRESS, i);

                /** Emisión de {@code localIntent} */


LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);

                /** Retardo de 1 segundo en la iteración */

                Thread.sleep(1000);
            }

            /** Quitar de primer plano */

            stopForeground(true);

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, "Servicio destruido...",
Toast.LENGTH_SHORT).show();
```

```
        /** Emisión para avisar que se terminó el servicio */

        Intent localIntent = new Intent(Constantes.ACTION_PROGRESS_EXIT);
        LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
    }
}
```

- Código "MainActivity.java".

```
/*******************************************************************************
 *  ////////////////////////////////////                                       *
 *  // Autor: Francisco Beltrán        //                                       *
 *  // País: España                    //                                       *
 *  // Fecha finalización: 18/08/2016  //                                       *
 *  ////////////////////////////////////                                       *
 *                                                                             *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.                 *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                           *
 *                                                                             *
 * «Sistema integral para apoyo a la comunicación personal                     *
 * en pacientes con diversidad funcional motora: SintomApp».                   *
 *                                                                             *
 * MainActivity.java: Código correspondiente a la actividad principal de la    *
 * App.                                                                        *
 *                                                                             *
*******************************************************************************/

package com.communicationsupport.kikobc.myapplication.Activities;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import com.communicationsupport.kikobc.myapplication.R;


public class MainActivity extends AppCompatActivity {

    Button blutu, puntero;
    ImageView SintomApp;
    private static int auxi = 0;
    private static int auxi2 = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDefaultDisplayHomeAsUpEnabled(true);

/** Botón de activación de Bluetooth */

        blutu = (Button) findViewById(R.id.BtnBluetooth);
        blutu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                /**Definimos el campo de texto */

                TextView campo_texto = (TextView)
findViewById(R.id.BtnBluetooth);
                if (auxi2 == 0) {

                    /**Cambiamos su contenido */

                    campo_texto.setText("Conectar con dispositivo");
                    auxi2 = 1;

                } else if (auxi2 == 1) {

                    /**Cambiamos su contenido */

                    campo_texto.setText("Desactivar Servicio Bluetooth");
                    auxi2 = 2;
                    Intent intentMemoryService = new Intent(
                            getApplicationContext(), MemoryService.class);

                    /** Iniciar servicio */

                    startService(intentMemoryService);

                } else if (auxi2 == 2) {

                    /**Cambiamos su contenido */

                    campo_texto.setText("Activar Servicio Bluetooth");
                    auxi2 = 0;
                    Intent intentMemoryService = new Intent(
                            getApplicationContext(), MemoryService.class);

                    /** Detener servicio */

                    stopService(intentMemoryService);
                }

                Intent intent = new Intent(MainActivity.this,
ConexionBT.class);
                startService(intent);
            }
        });

/** Botón para activación del cursor. */

        puntero = (Button) findViewById(R.id.Button_Pointer);
        puntero.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                /** Definimos el campo de texto */

                TextView campo_texto = (TextView)
findViewById(R.id.Button_Pointer);
                if (auxi == 0) {

                    /** Cambiamos su contenido */

                    campo_texto.setText("Desactivar Cursor");
                    auxi = 1;
```

```
                } else if (auxi == 1) {

                    /** Cambiamos su contenido */

                    campo_texto.setText("Activar Cursor");
                    auxi = 0;
                }

                Intent intent = new Intent(MainActivity.this,
MouseAccessibilityService.class);
                startService(intent);

            }
        });


/** Botón para acceder a las opciones de SintomApp */

        SintomApp = (ImageView) findViewById(R.id.imageView);
        SintomApp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(MainActivity.this, Remoto.class);
                startActivity(intent);

            }
        });

    }

}
```

- Código "AndroidManifest.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.communicationsupport.kikobc.myapplication">

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/app"
        android:label="SintomApp"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".Activities.MainActivity"
            android:label="SintomApp"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
        <service
            android:name=".Activities.MouseAccessibilityService"

android:accessibilityEventTypes="typeContextClicked|typeViewClicked"
            android:accessibilityFeedbackType="feedbackAllMask"
            android:accessibilityFlags="flagRequestFilterKeyEvents"
            android:canRequestFilterKeyEvents="true"
            android:canRetrieveWindowContent="true"
            android:enabled="true"
            android:exported="false"
            android:notificationTimeout="50"

android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
            <intent-filter>
                <action
android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>

            <meta-data
                android:name="android.accessibilityservice"
                android:resource="@xml/mouse_accessibility_service_config" />
        </service>

        <activity
            android:name=".Activities.Remoto"
            android:label="@string/title_activity_remoto"
            android:theme="@style/AppTheme.NoActionBar"></activity>

        <service
            android:name=".Activities.ConexionBT"
            android:enabled="true"
            android:exported="false"

android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
            <intent-filter>
                <action
android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>

            <meta-data
                android:name="android.accessibilityservice"
                android:resource="@xml/mouse_accessibility_service_config" />
        </service>
        <service
            android:name=".Activities.MemoryService"
            android:enabled="true"
            android:exported="true"></service>
        <service
            android:name=".Activities.ProgressIntentService"
            android:exported="false"></service>

        <activity
            android:name=".Activities.Dolor"
            android:label="@string/title_activity_dolor"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Bien"
            android:label="@string/title_activity_bien"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Hambre"
```

```
            android:label="@string/title_activity_hambre"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Incomodo"
            android:label="@string/title_activity_incomodo"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.No"
            android:label="@string/title_activity_no"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Sed"
            android:label="@string/title_activity_sed"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Si"
            android:label="@string/title_activity_si"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.Sucio"
            android:label="@string/title_activity_sucio"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".Activities.TQ"
            android:label="@string/title_activity_tq"
            android:theme="@style/AppTheme.NoActionBar" />
    </application>

</manifest>
```

- Código “transparentar.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/linear_interpolator">
    <alpha
        android:fromAlpha="0.2"
        android:toAlpha="1.0"
        android:repeatMode="reverse"
        android:repeatCount="4"
        android:duration="1250"/>
</set>
```

- Código “mouse_accessibility_service_config.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<accessibility-service
xmlns:android="http://schemas.android.com/apk/res/android"

android:accessibilityEventTypes="typeWindowStateChanged|typeWindowContentChan
ged"
    android:accessibilityFlags="flagDefault"
    android:accessibilityFeedbackType="feedbackGeneric"
    android:notificationTimeout="50"
    android:canRetrieveWindowContent="true" />
```

- Código “styles.xml”.

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
```

```
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>

    <style name="AppTheme.AppBarOverlay"
parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light"
/>

</resources>
```

- Código “colors.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

- Código “dimens.xml”.

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="fab_margin">16dp</dimen>
    <dimen name="padding_small">8dp</dimen>
    <dimen name="padding_medium">8dp</dimen>
    <dimen name="padding_large">16dp</dimen>

</resources>
```

- Código “strings.xml”.

```
<resources>
    <string name="app_name">My Application</string>
    <string name="action_settings">Settings</string>
    <string name="menu_settings">Settings</string>
    <string name="none_paired">Ningún dispositivo ha sido pareado</string>
    <string name="Hello">¡Bienvenido!</string>

    <string name="EstadoCursor">Activar Cursor</string>

    <string name="DesactivarBluetooth">Desactivar Bluetooth</string>
    <string name="ActivarBluetooth">Buscar dispositivos</string>

    <string name="Boton_Dolor">Dolor</string>
    <string name="Boton_Hambre">Hambre</string>
    <string name="Boton_Sed">Sed</string>
    <string name="Boton_Incomodo">Incómodo</string>
    <string name="Boton_Bien">Bien</string>
    <string name="Boton_Sucio">Sucio</string>
    <string name="Boton_Si">Sí</string>
    <string name="Boton_No">No</string>
    <string name="Boton_TQ">TQ</string>

    <string name="title_activity_dolor">Dolor</string>
    <string name="title_activity_hambre">Hambre</string>
    <string name="title_activity_incomodo">Incómodo</string>
```

```
    <string name="title_activity_sed">Sed</string>
    <string name="title_activity_sucio">Sucio</string>
    <string name="title_activity_bien">Bien</string>
    <string name="title_activity_si">Sí</string>
    <string name="title_activity_no">No</string>
    <string name="title_activity_tq">Te quiero</string>
    <string name="title_activity_remoto">SintomApp</string>
    <string name="title_activity_settings">Settings</string>

</resources>
```

- Código “cursor.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/cursorView"
        android:src="@mipmap/mouse_pointer"
        tools:context=".MouseAccessibilityService"/>

</LinearLayout>
```

- Código “menu_main.xml”

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".Activities.MainActivity" >


    <item android:id="@+id/action_autoclick1"
        android:title="Activar Autoclick con sonido"
        app:showAsAction="withText" />

    <item android:id="@+id/action_autoclick2"
        android:title="Activar Autoclick sin sonido"
        app:showAsAction="withText" />

    <item android:id="@+id/action_autoclick3"
        android:title="Desactivar Autoclick"
        app:showAsAction="withText" />

</menu>
```

- Código “activity_bien.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
tools:context="com.communicationsupport.kikobc.myapplication.Activities.Bien"
>
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />
    </android.support.design.widget.AppBarLayout>
    <include layout="@layout/content_bien" />

</android.support.design.widget.CoordinatorLayout>
```

Debido a la redundancia de código, en lugar de incluir el código de todos los "activity" de cada mensaje, se indica que existen otros archivos de nombres "activity_main.xml", "activity_remoto.xml", "activity_dolor.xml", "activity_hambre.xml", "activity_sed.xml", "activity_incomodo.xml", "activity_sucio.xml", "activity_tq.xml", "activity_si.xml" y "activity_no.xml" cuyo contenido es exactamente el mismo, salvo por el archivo del que toman los contenidos. En el caso de "activity_bien.xml", el contenido se ubica en "content_bien.xml".

- Código "content_bien.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Bien"
    tools:showIn="@layout/activity_bien"
    android:background="#b5e61d">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/bien"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/texto"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Me encuentro bien. Gracias."
        android:id="@+id/texto"
        android:textColor="#000000"
        android:textSize="40sp"
        android:layout_marginBottom="61dp"
        android:paddingLeft="44dp"
```

```
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

- Código "content_dolor.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Dolor
"
    tools:showIn="@layout/activity_dolor"
    android:background="#ff0000">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/dolor"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_above="@+id/texto"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="No me encuentro bien. Hay una zona que me duele."
        android:id="@+id/texto"
        android:layout_marginBottom="41dp"
        android:textColor="#ffffff"
        android:textSize="45sp"
        android:paddingLeft="16dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />

</RelativeLayout>
```

- Código "content_hambre.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Hambr
e"
```

```
    tools:showIn="@layout/activity_hambre"
    android:background="#22b14c">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/hambre"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Tengo hambre, ¿podría darme algo de comer?"
        android:id="@+id/texto"
        android:layout_marginBottom="65dp"
        android:textColor="#ffffff"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:paddingLeft="32dp" />
</RelativeLayout>
```

- Código “content_sed.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Sed"
    tools:showIn="@layout/activity_sed"
    android:background="#00a2e8">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/sed"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Tengo sed. Me gustaría tomar un vaso de agua."
        android:id="@+id/texto"
        android:layout_marginBottom="73dp"
        android:paddingLeft="16dp"
```

```
        android:textColor="#ffffff"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

- Código "content_incomodo.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Incom
odo"
    tools:showIn="@layout/activity_incomodo"
    android:background="#ffc90e">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/incomodo"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/texto"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ahora mismo me siento incómodo."
        android:id="@+id/texto"
        android:paddingLeft="14dp"
        android:textColor="#000000"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_marginBottom="61dp" />
</RelativeLayout>
```

- Código "content_sucio.xml".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Sucio
```

```
"
    tools:showIn="@layout/activity_sucio"
    android:background="#b97a57">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/sucio"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Me siento sucio. Me gustaría asearme."
        android:id="@+id/texto"
        android:layout_marginBottom="49dp"
        android:paddingLeft="24dp"
        android:textColor="#000000"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

- Código “content_tq.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.TQ"
    tools:showIn="@layout/activity_tq"
    android:background="#ff702b">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/tq"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Te quiero."
        android:id="@+id/texto"
        android:layout_marginBottom="53dp"
```

```
        android:paddingLeft="16dp"
        android:textColor="#880015"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

- Código “content_si.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.Si"
    tools:showIn="@layout/activity_si"
    android:background="#ff702b">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/si"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Sí."
        android:id="@+id/texto"
        android:layout_marginBottom="45dp"
        android:paddingLeft="16dp"
        android:textColor="#880015"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

- Código “content_no.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="com.communicationsupport.kikobc.myapplication.Activities.No"
```

```
    tools:showIn="@layout/activity_no"
    android:background="#ff702b">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/foto"
        android:src="@drawable/no"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_above="@+id/texto" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="No."
        android:id="@+id/texto"
        android:layout_marginBottom="51dp"
        android:paddingLeft="16dp"
        android:textColor="#880015"
        android:textSize="40sp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

- Código “content_main.xml”.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#ff7f27"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".Activities.MainActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:id="@+id/linearLayout">

    </LinearLayout>

    <Button
        android:layout_width="305dp"
        android:layout_height="wrap_content"
        android:text="@string/EstadoCursor"
        android:id="@+id/Button_Pointer"
        android:textColor="#ffffff"
        android:background="#ca0000"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="76dp" />
```

```
    <Button
        android:id="@+id/BtnBluetooth"
        android:layout_width="305dp"
        android:layout_height="wrap_content"
        android:text="Activar Servicio Bluetooth"
        android:background="#2527bd"
        android:textColor="#ffffff"
        android:layout_above="@+id/Button_Pointer"
        android:layout_alignStart="@+id/Button_Pointer" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/app2"
        android:layout_alignParentTop="true"
        android:layout_alignStart="@+id/BtnBluetooth"
        android:layout_alignEnd="@+id/BtnBluetooth"
        android:layout_above="@+id/BtnBluetooth" />
</RelativeLayout>
```

- Código “content_remoto.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:showIn="@layout/activity_remoto"
    tools:context=".Activities.Remoto"
    android:background="#ff6243">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Navegar por el sistema"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="25dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:id="@+id/Button_Control"
        android:textColor="#ffffff"
        android:background="#ca0000" />


    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Dolor"
        android:src="@drawable/dolor"
        android:contentDescription="Boton de dolor"
        android:layout_above="@+id/Boton_Incomodo"
        android:layout_alignParentStart="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Hambre"
        android:src="@drawable/hambre"
```

```
        android:contentDescription="Boton de hambre"
        android:layout_alignTop="@+id/Boton_Dolor"
        android:layout_alignStart="@+id/Boton_Si" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Sed"
        android:src="@drawable/sed"
        android:contentDescription="Boton de sed"
        android:layout_alignTop="@+id/Boton_Hambre"
        android:layout_alignStart="@+id/Boton_Sucio" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Incomodo"
        android:src="@drawable/incomodo"
        android:contentDescription="Boton de incomodo"
        android:layout_centerVertical="true"
        android:layout_alignParentStart="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Bien"
        android:src="@drawable/bien"
        android:contentDescription="Boton de bien"
        android:layout_alignTop="@+id/Boton_Incomodo"
        android:layout_alignStart="@+id/Boton_Hambre" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Sucio"
        android:src="@drawable/sucio"
        android:contentDescription="Boton de sucio"
        android:layout_alignTop="@+id/Boton_Bien"
        android:layout_alignParentEnd="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_TQ"
        android:src="@drawable/tq"
        android:contentDescription="Boton de te quiero"
        android:layout_below="@+id/Boton_Incomodo"
        android:layout_alignParentStart="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_Si"
        android:src="@drawable/si"
        android:contentDescription="Boton de si"
        android:layout_alignBottom="@+id/Boton_TQ"
        android:layout_centerHorizontal="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Boton_No"
```

```
        android:src="@drawable/no"
        android:contentDescription="Boton de no"
        android:layout_alignTop="@+id/Boton_Si"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Anexo B: Código LaunchPad

- Código "SintomAppLaunchPad.ino".

```
/*************************************************************************************
 * ////////////////////////////////////////                                          *
 * //  Autor: Francisco Beltrán    //                                                *
 * //  País: España                //                                                *
 * //  Fecha: 02/08/2016           //                                                *
 * ////////////////////////////////////////                                          *
 *                                                                                   *
 * Grado en Ingeniería de las Tecnologías de Telecomunicación.                       *
 * Trabajo de Fin de Grado - Universidad de Sevilla.                                 *
 *                                                                                   *
 * «Sistema integral para apoyo a la comunicación personal                           *
 * en pacientes con diversidad funcional motora: SintomApp».                         *
 *                                                                                   *
 * Control remoto de un Smartphone mediante conexión Bluetooth.                      *
 *                                                                                   *
 * Programación de movimiento de Joystick y pulsación de botón transmitidos          *
 * a un teléfono para su manejo telemático. El joystick es, a efectos prácticos,     *
 * un potenciómetro cuya posición determina el nivel de tensión según los ejes       *
 * vertical y horizontal.                                                            *
 *                                                                                   *
 *************************************************************************************/

// Librería para utilizar el Timer
#include <msp430.h>

// Definimos los pines que vamos a usar.
#define PINJOYSTICK_V  P1_6     // Movimiento vertical Joystick
#define PINJOYSTICK_H  P1_0     // Movimiento horizontal Joystick
#define PINLED         GREEN_LED // LED verde
#define PINLED2        RED_LED   // LED rojo
#define VCC            1         // Pin de VCC

// Variables auxiliares necesarias:
int joystickValueV = 0;  // Posición vertical del joystick
int joystickValueH = 0;  // Posición horizontal del joystick
int pwmValueV = 0;       // Valor escalado de la posición vertical
int pwmValueH = 0;       // Valor escalado de la posición horizontal
int Botoncito = 0;       // Su valor dependerá de si se pulsa el botón o no
int buttonState = 0;     // Variable para leer el estado del botón
```

```
const int Boton = PUSH2; // Botón físico que se utilizará para simular el click del cursor.
volatile boolean debounce = false;
char autoclick = '0';   // Su valor indica el modo de funcionamiento
static int auxiliar = 0; // Variable auxiliar para el timer

/************** Inicialización del programa ****************/
void setup() {

// Pin VCC de salida. Ajustamos su valor para el correcto funcionamiento.
  pinMode(VCC,OUTPUT);
  digitalWrite(VCC,LOW);
  digitalWrite(VCC,HIGH);

// Inicializamos el pin del botón como entrada con pull-up.
// No necesitamos poner PINJOYSTICK como entrada ya que
// la función analogRead() coloca el pin en un estado especial,
// conectado al ADC. Los LEDs se usan como salida.
  pinMode(Boton, INPUT_PULLUP);
  pinMode(PINJOYSTICK_V, INPUT);
  pinMode(PINJOYSTICK_H, INPUT);
  pinMode(PINLED, OUTPUT);
  pinMode(PINLED2, OUTPUT);

// Inicializamos el puerto serie conectado al módulo Bluetooth a 9600bps.
   Serial.begin(9600);

// Asignamos al pin PUSH2 una función que se ejecutará
// cuando se detecte un flanco de subida:
  attachInterrupt(Boton, func, FALLING);

  // Configuración
  // Bits 15-10: No se usan
  // Bits 9-8: Velocidad del reloj (16MHz)
  // Bits 7-6: Divisor de entrada: 8
  // Bits 5-4: Modo control: Cuenta hasta el valor de TACCRO y resetea
  // Bit 3: No se usa
  // Bits 2: TACLR : Limpia el timer para su inicialización
  // Bit 1: Habilita interrupciones de TA0
```

```
  // Bit 0: Bandera de interrupción : inicializa a cero

  TA0CTL=0b0000001011010010;
  TACCR0=2000; // inicializa TACCR0 = 2000 para generar 1ms (16MHz) con un divisor de 8
  TACCTL0=BIT4; // Habilita la interrupción cuando TAR = TACCR0
}

/*************** Bucle infinito ****************/
void loop() {

// Lectura de los valores del joystick
  joystickValueV = analogRead(PINJOYSTICK_V);
  joystickValueH = analogRead(PINJOYSTICK_H);

// Escalamos su valor para el pwm, ya que la función analogWrite() acepta
// un valor máximo de 255 y analogRead nos devuelve un entero entre 0 y 1023.
  pwmValueV = joystickValueV/4;
  pwmValueH = joystickValueH/4;

// Enviamos el valor de tensión a los leds. No es
// del todo necesario, pero está bien visualmente.
  analogWrite(PINLED, pwmValueV);
  analogWrite(PINLED2, pwmValueH);

// Activación de Autoclick
  if (Serial.available() > 0) {
// Lectura del byte de entrada para el modo de funcionamiento
    autoclick = Serial.read();
  }

// Leemos el estado del botón.
  buttonState = digitalRead(Boton);

// Comprobamos si el botón está pulsado.
// Si no lo está, buttonState estará como HIGH (pull-up).
  if (buttonState == HIGH) {
    Botoncito = 0;
  } else {
```

```
    Botoncito = 4;
  }

/****** Transmisión de movimiento. *****/

// Tomamos los valores del joystick y los centramos en 0.
  pwmValueH = (joystickValueH/4) - 118;
  pwmValueV = (joystickValueV/4) - 117;

// Si se pulsa el botón, se interpretará como un click.
// Antirrebotes del pulsador para interrupciones:
  if (debounce == true && Botoncito == 4) {
    delay(200); // retraso de 10ms para evitar rebotes
    if (digitalRead(Boton) != LOW){ // comprobamos si sigue pulsado
      debounce = false;
      Serial.print(Botoncito);
      Botoncito = 0;
    }
  }

// Al mover el joystick se desplazará el cursor.
// Movimiento horizontal.
  if(pwmValueH>=30){
    Serial.print(3);
  } else if(pwmValueH<=-30){
    Serial.print(2);
  }

// Movimiento vertical.
  if(pwmValueV>=30){
    Serial.print(0);
  } else if(pwmValueV<=-30){
    Serial.print(1);
  }

// La función "Serial.print()" coloca los datos en el buffer de salida y retorna.
// Por tanto, el programa seguirá ejecutándose aunque los datos aún se encuentren
// en el buffer a la espera de ser enviados. Es por ello que se colocan los delay,
```

```
// para no saturar el buffer de salida.
  delay(100);

// Si no se hace nada, se le indicará al teléfono
// que se mantenga inactivo.
  Serial.print(5);
}

/******** Rutina de atención a la interrupción para PUSH2 *********/
void func() {
  if(debounce == false) {
    debounce = true;
  }
}

/*************** Función del Timer *****************/

// La función de la dirección que sigue el estado del vector es colocado en
// la localización específica de la tabla del vector de interrupción

#pragma vector = TIMER0_A0_VECTOR
__interrupt  void timerA0ISR(void) {

// Rutina de interrupción del Timer A0
  static int msCount=0; // Cuenta milisegundos

// Si no se utiliza el joystick, el modo de autoclick con sonido está activado y "auxiliar" vale 0:
  if ((pwmValueH<30 && pwmValueH>-30)  &&  (pwmValueV<30 && pwmValueV>-30)
                && (autoclick == '2') && (auxiliar == 0)) {
    msCount++;
    if (msCount >= 1500 && msCount < 2000) {
      Serial.print(6); // Transcurridos unos segundos, se indica que el sonido se puede reproducir
    }
    else if (msCount >= 2000) {
      msCount = 0;    // Posteriormente, se hace click y se resetea el contador.
      auxiliar = 1;   // La función de "auxiliar" es impedir que haga click en cada iteración.
      Serial.print(4);
    }
```

```
  }

// Si está activo el autoclick sin sonido:
  else if ((pwmValueH<30 && pwmValueH>-30) && (pwmValueV<30 && pwmValueV>-30)
                && (autoclick == '1') && (auxiliar == 0)) {
    msCount++;
    if (msCount >= 2000){
      msCount = 0;  // Se hace click tras un par de segundos
      auxiliar = 1;
      Serial.print(4);
    }
  }

// Si el modo de funcionamiento es el corriente, no se hace nada.
  else if (autoclick == '0'){
  }

// Si se utiliza el joystick, se resetea el contador y se inicializa "auxiliar" a 0.
  else if ((pwmValueH>=30 || pwmValueH <=-30) || (pwmValueV>=30 || pwmValueV<=-30)) {
    auxiliar = 0;
    msCount = 0;
  }
}
```