Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

Reporte Final del Proyecto de Integración

Licenciatura en Ingeniería en Electrónica

Proyecto de Investigación

Wearable para la movilidad de las personas con discapacidad visual

Nombre completo del alumno: Adrian Buendía Martínez
Matricula: 2123030046

Nombre completo del asesor: María Lizbeth Gallardo López
Trimestre lectivo: 19-O
Fecha de Entrega: **05 / Marzo / 2020**

Yo, MARÍA LIZBETH GALLARDO LÓPEZ, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Ma. Lizbeth Gallardo López

Nombre y firma del asesor

Yo, ADRIAN BUENDÍA MARTÍNEZ, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Adrian Buendía Martínez

Nombre y firma del alumno

Agradecimientos

A mi madre, Doris, y a mi tía, Herlinda, mujeres con el espíritu y la determinación de volcanes en erupción.

A todas y todos los que participaron en este proyecto.

Resumen

En el presente proyecto se desarrolló un sistema capaz de detectar obstáculos para alertar al usuario con discapacidad visual, aumentando su movilidad y confianza al desplazarse. Este sistema está compuesto de un dispositivo electrónico portable (*wearable*) y una aplicación móvil (*App*) que opera bajo la plataforma Android. El *wearable* está integrado a una prenda similar a un chaleco y fue construido con base en un arreglo de 4 sensores ultrasónicos y un microcontrolador PIC16F886. El *wearable* se utiliza para determinar las distancias aproximadas entre el usuario con discapacidad visual y los obstáculos que puede encontrar a su paso. La *App* trabaja sincronizadamente con el *wearable*, recibe una serie de datos correspondientes a las medidas de la distancia entre el usuario y los obstáculos. La *App* determina si hay un obstáculo, cuando una o más de las medidas obtenidas son igual o menores a 120 cm, y alerta al usuario a través de diferentes patrones de sonido y vibración según la proximidad del obstáculo.

Tabla de contenido

Índice de Tablas

Índice de Figuras

Índice de Pseudocódigos

Apéndices

1. Introducción

El área de aplicación de este proyecto está orientado a ofrecer un sistema automatizado que brinde una alternativa de apoyo en la movilidad de las personas con discapacidad visual. Este sistema permitirá transitar y trasladarse de manera segura e independiente a la persona con discapacidad, en el ámbito familiar, social y laboral. En el corto plazo, se espera que una persona con discapacidad visual emplee, además del bastón u otro instrumento mecánico manual, un sistema automatizado que lo alerte sobre obstáculos superiores a la altura de la cadera. En el largo plazo, se espera que los instrumentos mecánicos sean desplazados por sistemas automatizados que sean portados como una prenda de vestir (*wearable*).

En este proyecto de integración se desarrolló un dispositivo electrónico portable (*wearable*) y una aplicación móvil (*App*) que opera bajo la plataforma Android. El *wearable* se construyó con base en un arreglo de 4 sensores ultrasónicos y un microcontrolador PIC16F886, y está integrado a una prenda similar a un chaleco. El *wearable* se utiliza para determinar las distancias aproximadas entre el usuario con discapacidad visual y los obstáculos que puede encontrar a su paso. La *App* trabaja sincronizadamente con el *wearable*, recibe una serie de datos correspondientes a las medidas de la distancia entre el usuario y los obstáculos. La *App* determina si hay un obstáculo, cuando una o más de las medidas obtenidas son igual o menores a 120cm, y alerta al usuario a través de diferentes patrones de sonido y vibración según la proximidad del obstáculo. Mientras el obstáculo se encuentre más cercano al usuario, el sonido y vibración del dispositivo Android será más rápido, además para identificar la dirección del obstáculo se implementaron 3 diferentes sonidos (izquierda, en medio, derecha).

De acuerdo con la OMS (Organización mundial de la Salud), en una publicación del 2014, *"La discapacidad visual puede limitar a las personas en la realización de tareas cotidianas y afectar su calidad de vida, así como sus posibilidades de interacción con el mundo circundante"*. Con base en lo anterior, este proyecto integra la tecnología electrónica y las tecnologías de la información para ayudar a las personas con discapacidad visual a sentirse emocionalmente más seguras e independientes, porque el *wearable* los apoyará, alertando sobre obstáculos, durante sus traslados a diferentes lugares.

2. Antecedentes

Para la elaboración del proyecto se realizó una búsqueda exhaustiva, a fin de encontrar proyectos semejantes que sirvieran como referencia para desarrollar tecnología que facilite la movilidad de personas con discapacidad visual. A continuación, se presenta un breve resumen de los trabajos encontrados, resaltando las similitudes y diferencias con respecto al proyecto de integración.

2.1. A wearable System for Mobility Improvement of Visually Impaired People *[1]*

La investigación del artículo se centra en localizar obstáculos, con el fin de reducir las dificultades que tienen las personas con discapacidad visual para su desplazamiento. El sistema diseñado, al cual nos referiremos como Wearable System, consiste en detectar obstáculos a la altura de los hombros por medio de sensores ultrasónicos. El Wearable System notifica al usuario la proximidad de un objeto a través de la vibración del chaleco; la vibración tiene lugar en determinado(s) lugar(es). Es posible calibrar la vibración dependiendo de la sensibilidad de cada usuario. Las unidades de procesamiento de datos son: un microcontrolador PIC16F87, un módulo

Bluetooth (modelo no especificado) y un PDA (asistente digital personal). Este artículo describe la arquitectura, mostrada en la figura 1, además discute los beneficios potenciales del sistema.

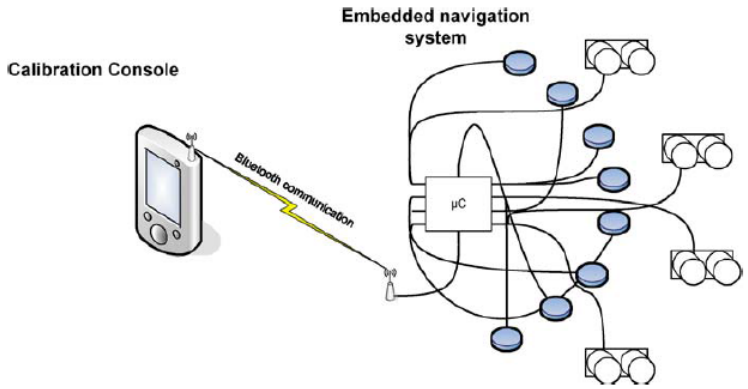

Figura 1. Esquema de la arquitectura del sistema *[1]*.

2.2. Cinturón y Manillas Vibradores Ultrasónicos para No Videntes *[2]*

El objetivo de este trabajo es ofrecer independencia y seguridad en la movilidad de la persona con discapacidad visual crítica, por ello se fabricó un cinturón y manillas vibradores ultrasónicos que detectan obstáculos a media altura en tres direcciones (izquierda, derecha y frontal), aumentando la movilidad de la persona. En este proyecto, al cual nos referiremos como Cinturón y manillas, vibra en determinado(s) lugar(es) para notificarle al usuario la proximidad de un objeto, mientras más fuerte la vibración, más cercano el objeto. Las unidades de procesamiento de datos son: Arduino Pro-micro, el módulo Xbee y tarjeta myRIO. También se usó el PIC18F2550 para controlar la carga de las baterías.

Figura 2. Cinturón y Manillas Vibradores Ultrasónicos para No Videntes *[2]*.

2.3. Similitudes de los sistemas mencionados con el proyecto de integración

- Fueron diseñados para mejorar la movilidad al caminar de las personas con discapacidad visual.
- Están construidos a base de sensores ultrasónicos.
- Ambos son wearables, ya que son accesorios vestibles portables.

2.4. Diferencias de los sistemas mencionados con el proyecto de integración

- En el presente proyecto las unidades de procesamiento son:
  - Dispositivo *Android* con versión 4.4KITKAT o superior
  - Microcontrolador PIC-16F886.
- El módulo *Bluetooth* envía la distancia capturada por los sensores ultrasónicos hacia el dispositivo *Android*, a diferencia del Wearable System, que utiliza al módulo bluetooth y PDA con el único fin de calibrar la vibración del wearable.
- En el proyecto de integración, el dispositivo *Android* procesa los datos y notifica al usuario por medio de diferentes sonidos y vibración cuando exista algún obstáculo en su camino, esto evita el costo por la adición de motores de vibración (integrados a la prenda de vestir) u otro tipo de dispositivos de procesamiento como se observa en los proyectos Wearable System y Cinturón y manillas. Además, se ahorra energía que le permite al usuario un tiempo mayor de uso.

3. Justificación

La ingeniería electrónica actualmente tiene una gama muy amplia de aplicaciones que permiten mejorar la calidad de vida de las personas. La importancia de este proyecto consiste en contribuir con un dispositivo electrónico (*wearable*) que ahorre energía y dinero (consultar Apéndice C. Comparación de precios) al trabajar en conjunto con una aplicación diseñada para OS *Android*. La función conjunta de estos dispositivos será prevenir a la persona con discapacidad visual de los objetos que estén a su paso y obstaculicen su avance de forma segura. El usuario podrá conocer anticipadamente si hay algún riesgo en su movilidad, así podrá cambiar de ruta en sus traslados a diferentes lugares, de esta manera se procurará su bienestar y se mejorará su calidad de vida.

4. Objetivos

4.1. Objetivo general

Diseñar y construir un sistema electrónico portable que determina y alerta acerca de la existencia de un objeto que obstaculiza la movilidad de una persona con discapacidad visual.

4.2. Objetivos particulares

1. Dimensionar arreglo de sensores ultrasónicos.

2. Procesar la distancia a partir de las señales del arreglo de sensores mediante el microcontrolador (PIC16F886).

3. Diseñar e implementar la aplicación para el dispositivo con OS *Android*.

4. Realizar la interfaz de comunicación para enviar la distancia de los sensores del dispositivo wearable hacia la APP mediante el protocolo RS-232.

5. Probar e integrar los módulos que conforman el sistema wearable para su implementación en un circuito impreso.

5. Marco Teórico

5.1. Wearable

Según la etimología de la lengua inglesa, wearable significa “vestible” o también “llevable”. Hoy en día es común la traducción de” tecnología ponible” o “computadoras corporales”. Los wearables implementan un dispositivo electrónico que incluye un microprocesador o microcontrolador, para brindar distintas funciones encapsuladas. La principal característica de esta tecnología es la de ejecutar una función específica. Por ejemplo, los wearables ligados a la salud otorgan al usuario algunas capacidades y nuevas habilidades; algunos son pequeños y se incorporan a la ropa fácilmente, e incluso dentro del cuerpo humano para registrar información sobre las funciones biológicas.*[3]*

5.2. App

Una aplicación móvil o App, es un software aplicativo que está diseñada para ser utilizados en tables, *smartphones* y otros dispositivos móviles. Existen Apps para diferentes ámbitos, por ejemplo, para el entretenimiento, para la educación, para las finanzas, para la salud, entre otras. En particular, las Apps para la salud cada vez se proponen más y mejores aplicaciones para *smartphones* que ayudan a sobrellevar determinadas patologías y enfermedades. La Organización Mundial de la Salud propone el concepto de *mHealth* o *salud móvil*, y lo define como el *“uso de móviles y tecnologías inalámbricas para apoyar el logro de objetivos de salud”[4]*. El *mHealth* tiene potencial para jugar un papel fundamental en el presente y futuro de la salud. Sin duda, será una herramienta clave para solventar los actuales problemas del sector. Problemas como la inequidad en el acceso a la salud o la prevalencia de enfermedades crónicas comprometen la sostenibilidad del sistema de salud.*[4]*

5.3. Comunicación *Bluetooth*

En palabras de J. T. Gironés *[5]* el proceso para crear la conexión con un dispositivo con tecnología *Android* es el siguiente:

“Un dispositivo debe abrir un socket de servidor y el otro debe inicializar la conexión (usando la dirección MAC del dispositivo del servidor para inicializar la conexión). El servidor y el cliente se consideran conectados entre sí cuando tienen un BluetoothSocket conectado en el mismo canal RFCOMM. En este punto, cada dispositivo puede obtener flujos de entrada y salida, y se puede iniciar la transferencia de datos. El dispositivo del servidor y el dispositivo del cliente obtienen el BluetoothSocket obligatorio de diferentes maneras. El servidor lo recibe cuando se acepta una conexión entrante. El cliente, cuando abra un canal RFCOMM hacia el servidor.” [5].

5.4. Microcontrolador PIC

Un microcontrolador es un circuito integrado programable, es el componente principal de una aplicación embebida. El microcontrolador contiene todos los componentes necesarios para controlar el funcionamiento de una tarea determinada; por ejemplo, el control un teclado de ordenador, un sistema de alarma, etcétera. Para esto, se utilizan los recursos mínimos siguientes: microprocesador, periféricos (unidades de entrada/salida) y memoria (flash y RAM) *[6]*. Un

microprocesador incluye al menos tres elementos, ALU, unidad de control y registros. La memoria se utiliza para almacenar el programa que gobierna el funcionamiento de un microcontrolador, que una vez configurado, solo sirve para realizar la función asignada *[6]*.

5.5. Sensores ultrasónicos

Los sensores son dispositivos de entrada que proporcionan una salida manipulable de la variable física medida. Por otra parte, el transductor es un dispositivo que tiene la misión de convertir una señal física o química en otra distinta entendible por el sistema *[7]*. Los sensores ultrasónicos son sensores de proximidad utilizados en sistemas de medición no invasivos para determinar la distancia del emisor a un objeto dado. El núcleo del sensor ultrasónico es un material piezoeléctrico. La piezoelectricidad es la propiedad que presenta un material de generar un voltaje debido a una fuerza aplicada. La onda ultrasónica que emite el material piezoeléctrico se genera por medio de una excitación eléctrica al material (el efecto piezoeléctrico también se presenta de manera inversa; es decir, al aplicar un voltaje, el material experimenta una deformación, y como resulta este emite una onda mecánica). Esta onda es emitida por todo el material, lo que significa que no es puntual. Por tanto, los sensores ultrasónicos cuyo elemento piezoeléctrico es de forma redonda se conocen como transductores fuente pistón, debido a la geometría que presenta el campo sonoro que emite. Se dice que una onda sonora es ultrasónica cuando está por encima de la frecuencia audible para el oído humano; esto es, por encima de los 20KHz aproximadamente *[7]*.

El efecto Doppler consiste en un cambio aparente de frecuencia de la onda sonora respecto al emisor cuando esta es reflejada en un objeto móvil (o partículas inmersas en un flujo). Este cambio resulta proporcional a la velocidad relativa del emisor reflector. Un sensor ultrasónico se auxilia del efecto Doppler, ya que un elemento ultrasónico (considerado como emisor) emite una onda ultrasónica, la cual es absorbida en parte y reflejada en parte por el objeto a medir; así, a través de la medición de la atenuación de la onda percibida por el receptor, el tiempo que le toma a esta ser percibida por el receptor, o por la presencia o ausencia de dicha onda en el emisor, es posible obtener características de la variable física que se desea determinar *[7]*. El principio de funcionamiento de los sensores ultrasónicos consiste en la emisión de una onda de manera cíclica, la cual es de alta frecuencia y corta duración, además de que se propaga en el medio. Al encontrar un objeto a su paso, esta es reflejada y vuelve en forma de eco al receptor. El circuito de acondicionamiento tiene la tarea de determinar el periodo transcurrido entre la emisión de la señal acústica y la recepción del eco *[7]*. Cuando este tipo de sensores se basa en la medición del tiempo del recorrido del sonido, por ejemplo, para determinar la distancia del objeto al sensor, es difícil que el ruido de fondo influya en la medición; por el contrario, si la medición se basa en a la intensidad de la onda que es reflejada, el sistema es sensitivo al ruido de fondo. Una de las principales ventajas de este tipo de sensado es que todo material que refleje el sonido puede ser detectado, independiente del color, la textura o el grosor del objeto. La desventaja es que, debido a la velocidad de propagación de la onda ultrasónica, esta depende de la temperatura del ambiente; ante esto, entonces se debe realizar una compensación *[7]*. Un sensor ultrasónico permite medir distancias de entre 20mm hasta 10m; no obstante, con un buen acondicionamiento de señal es posible obtener valores con hasta 1mm de precisión. Para lograr una buena medición es

importante tener en cuenta la forma en que se coloca tanto el emisor como el receptor, además de una compensación por la temperatura *[7]*. Los diferentes modos de operación de un sensor ultrasónico son los siguientes: configuración de reflexión, configuración de ventana, configuración de barrera ultrasónica bidireccional, configuración de supresión del primer plano, configuración de medidor de distancia, configuración de barrera para la detección de defectos *[7]*.

6. Desarrollo del proyecto

Se desarrolló un wearable que trabaja en sincronía con una App para apoyar a personas con discapacidad visual en su movilidad; el objetivo es detectar posibles obstáculos en su camino. Primeramente, se desarrolló el wearable, con el fin de obtener distancias entre el usuario y los obstáculos. Después se desarrolló la App, de modo que ésta recibe las distancias enviadas por el wearable, y aplica un algoritmo para revelar obstáculos en el camino. Finalmente, se realizaron pruebas del sistema con personas que simularon discapacidad visual.

6.1. Metodología

Observación y planteamiento del problema. La movilidad de las personas con discapacidad visual es un problema conocido, donde la detección de obstáculos es uno de los retos principales. En espacios como pasillos de universidades y oficinas, se realizarán pruebas utilizando dispositivos móviles inteligentes en personas que simulen discapacidad visual.

Hipótesis. En un sistema electrónico portable que determina y alerta sobre la existencia de un objeto que obstaculiza la movilidad de una persona con discapacidad visual, es posible disminuir el costo monetario y el consumo energético, si se delega el procesamiento de las señales al *smartphone Android* que posee la persona.

Experimentación. Será necesario la colaboración de usuarios para realizar mediciones del alcance de los sensores ultrasónicos, colocándolos en diferentes lugares del cuerpo de los usuarios, como el pecho, cintura y cabeza. Además de dimensionar y ajustar el tamaño del arreglo de sensores.

Resultados esperados. Se espera obtener un wearable con al menos cuatro sensores que midan la distancia de los objetos que se encuentren a 180° tanto al frente-horizontal como frente-vertical. Las señales serán procesadas en un *smartphone Android*, el cual emitirá sonidos y vibrará para prevenir a la persona con discapacidad visual sobre los obstáculos.

Análisis de resultados. El sistema será comparado con aquellos sistemas *wearables* que se han reportado en la literatura. La comparación se hará en cuanto al costo monetario y al consumo energético, con el propósito de comprobar la hipótesis.

6.2. Especificación del sistema

Esta sección describe los requerimientos funcionales y no funcionales del sistema.

Incluye también la lista de los usuarios y la descripción de las actividades que puede realizar.

6.2.1. Requerimientos funcionales del *wearable* y Aplicación OS *Android*

Requerimientos funcionales	Descripción
Crear un dispositivo *wearable*	Construir un chaleco hecho a base de un arreglo de cuatro sensores ultrasónicos.
Obtener la distancia de los sensores	Calcular distancia en centímetros de los sensores ultrasónicos a un posible obstáculo. Rango de los sensores [0 metros – 4 metros].
Enviar distancias a la *App*	Inicializar protocolo para establecer comunicación con *App*.
Crear una *App*	Desarrollar aplicación en SO *Android* capaz de comunicarse con el *wearable*
Módulo para interfaz grafica	Desplegar pantallas con reconocimiento y emisión de voz para la navegación entre la *App*. Alertar con sonidos y vibración de un obstáculo.
Módulo para comunicación *Bluetooth*	Recepción de la cadena de caracteres que puede contener las distancias de los sensores ultrasónicos.
Módulo de procesamiento de datos	Determinar si las distancias recibidas representan un posible obstáculo y alertar por medio de un sonido particular y una secuencia de vibración.

6.2.2. Requerimientos no funcionales

Requerimientos no funcionales aplicación OS *Android*	Descripción
Configuración dispositiva con OS *Android*	Antes de utilizar la *App*, una persona deberá ayudar al usuario con discapacidad para vincular el *wearable* con el dispositivo *Android*. Este paso solo se realizará una sola ocasión.
Activación *Bluetooth*	El usuario activará el *Bluetooth* y ejecutará la *App* por medio de comandos de voz. La *App* no continuará ejecutándose a menos que el *Bluetooth* se encuentre activado, de lo contrario la *App* lo indicará por medio de la voz artificial *Android*.
Rango Distancia	Si la distancia de algún(os) sensor(es) se encuentra en un rango mayor a 120 cm, la *App* no emitirá ninguna instrucción.

Requerimientos no funcionales *wearable*	Descripción
Obstáculo	Un objeto se considerará obstáculo, si tiene una superficie mayor o igual a 2.5 cm^2 y se encuentra a una distancia menor o igual a 120 cm de algún(os) sensor(es).

Distancia de los Sensores desfasados	Existirá un tiempo de aproximadamente $100\,ms$ de desfase entre la distancia actual y la enviada al dispositivo con OS *Android*. No obstante, se esperaría que, al ser un tiempo relativamente pequeño, no afecte la sincronización con el andar del usuario. Este tiempo podría ser mejorado aplicando un algoritmo optimizado.
Ambiente	El *wearable* estará diseñado para ambientes sin exceso de obstáculos, como los son las oficinas, despachos, bufetes, de igual manera, para escenarios educativos, como universidades, salas de juntas, entre otros; además, deberá operar únicamente cuando la persona se encuentre desplazándose.

6.2.3. Lista de usuarios

Usuario	Actividades
Persona con discapacidad visual	1. Acceder a la aplicación móvil 2. Escuchar y sentir. Alerta de un posible obstáculo

6.3. Diagramas de Casos de Uso

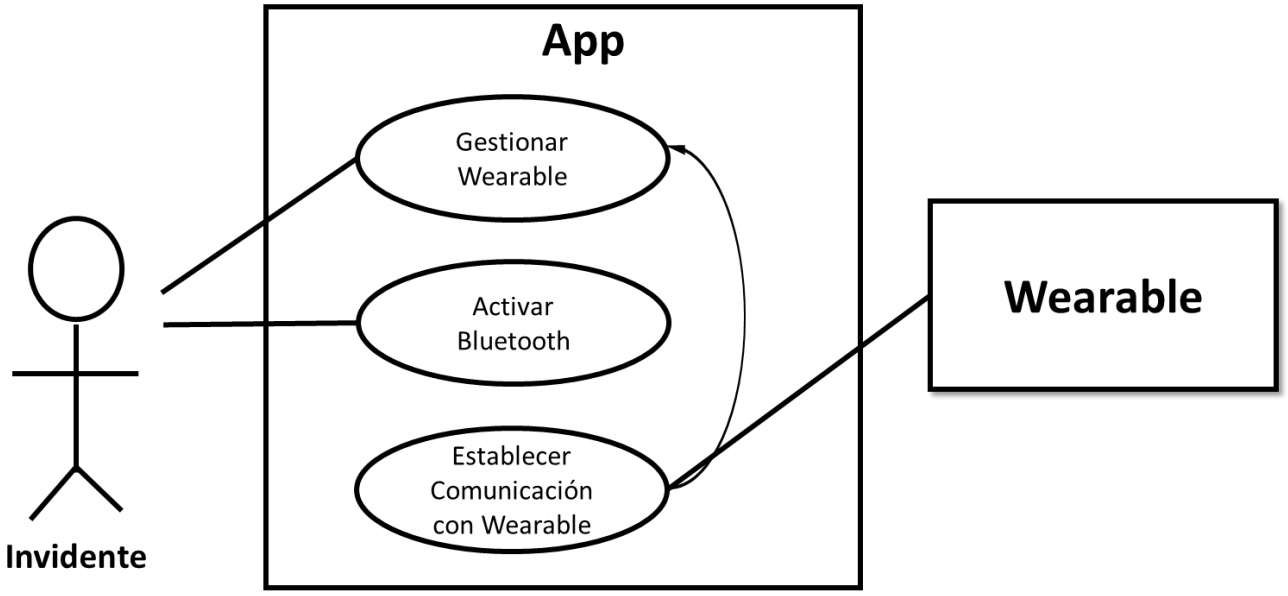

Figura 3. Diagrama De Caso De Uso *App*.

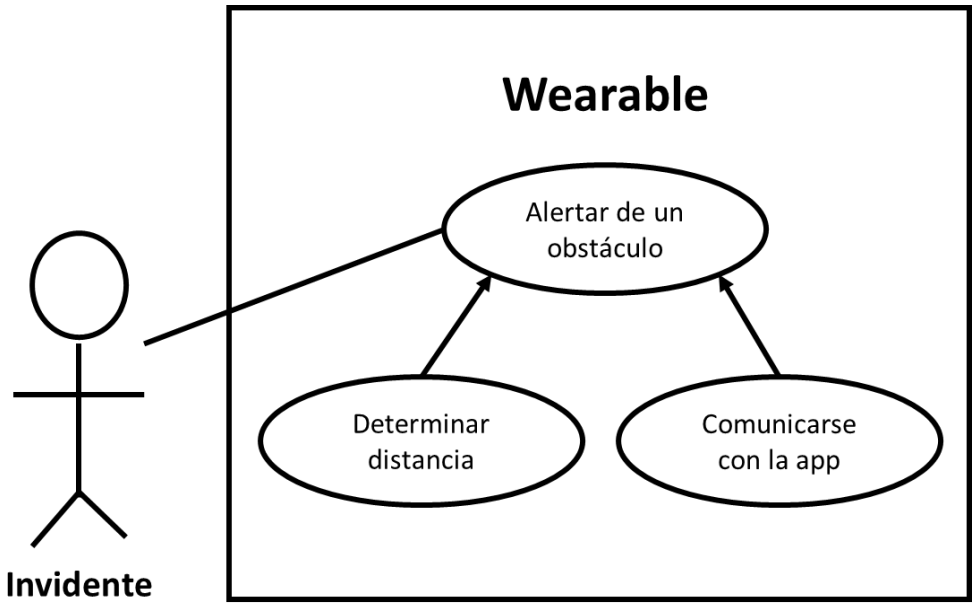

Figura 4. Diagrama de caso de uso *wearable*.

6.3.1. Casos de uso de *App*

Gestionar wearable

Resumen	Ejecuta la aplicación móvil en el dispositivo con OS *Android* versión igual o superior a la 4.4. *KITKAT*. Posteriormente, elimina el ruido de la trama recibida y se analiza cuando es una posible distancia. Se implementa algoritmo de alerta de obstáculo, para emitir diferentes sonidos y patrones de vibración.
Actores	Invidente.
Precondiciones	1. La conexión *Bluetooth* deberá estar activada en el Dispositivo móvil. 2. Los comandos por voz deberán estar activados. 3. El dispositivo con OS *Android* soporta conexiones *Bluetooth*. 4. El *wearable* debe estar encendido con las baterías con la energía suficiente.
Descripción	Flujo principal: 1. El usuario accede a la aplicación por medio del comando de voz “Abrir Caminador”. 2. El sistema despliega el logotipo de la UAM Azcapotzalco. 3. El sistema emite voz artificial *Android* saludando al Usuario. 4. Se cuenta que el número de tokens que contiene la trama sea igual al número de sensores. 5. Se asigna cada *token* a la distancia de cada sensor. 6. Implementa algoritmo de alerta de obstáculo (*pseudocódigo 3*). Flujo alternativo: a. El sistema detecta inhabilitada la conexión *Bluetooth*.

	b. El usuario activa el *Bluetooth* a través del comando de voz "activar *Bluetooth*". c. El sistema activa el *Bluetooth*, y el flujo continúa en el paso 3.
Excepción	• No es posible abrir la aplicación sin comandos de voz para el Usuario. • La petición de *Bluetooth* es rechaza, se cierra la *App*.
Postcondición	Alerta de un obstáculo en el camino.

Establecer comunicación con wearable

Resumen	Genera el evento de conexión, además de inicializar un canal seguro de comunicación para envió y recepción de datos con *wearable*.
Actores	*Wearable.*
Precondiciones	1. Vincular el *wearable* al dispositivo con SO *Android*. 2. El *wearable* debe estar encendido con energía en las baterías suficiente para enviar tramas de caracteres y activar el arreglo de sensores ultrasónicos.
Descripción	Flujo principal: 1. El cliente inicializa la conexión (usando la dirección *MAC* del servidor) realizando una petición al servidor (*wearable*). 2. El servidor y el cliente se consideran conectados entre sí cuando el servidor acepta la solicitud y un `BluetoothSocket` está conectado en el mismo canal *RFCOMM*. 3. El cliente y el servidor pueden obtener flujos de entrada y salida, y se puede iniciar la transferencia de datos.
Excepción	• La comunicación falló. • Intentar volver a realizar la conexión, máximo 3 veces.
Postcondición	Recepción de tramas de caracteres

6.3.2. Casos de Uso *Wearable*

Determinar distancia

Resumen	Cuantificar la señal recibida del arreglo de sensores ultrasónicos en centímetros.
Actores	*Wearable.*

Precondiciones	El *wearable* debe estar encendido con las baterías con la energía suficiente.
Descripción	Flujo principal: 1. Enviar un pulso de 40ms por el pin `trigger` del sensor. 2. Inicializar un temporizador asignado al pin `echo` del sensor ultrasónico para obtener el tiempo de recepción del `trigger`. 3. Realizar conversión a centímetros y almacenar en una variable. 4. Replicar el paso 1 para 4 sensores.
Postcondición	Cadena de caracteres con distancias en centímetros.

Comunicarse con la App

Resumen	Enviar tramas de caracteres a la *App*.
Actores	*Wearable.*
Precondiciones	El *wearable* debe estar encendido con las baterías con la energía suficiente.
Descripción	Flujo principal: 1. Inicializar protocolo `RS-232` con velocidad de transmisión a 9600 baudios. 2. Enviar tramas de caracteres cada 20ms.
Postcondición	Tramas de caracteres enviadas a la *App*.

6.4. Diseño del *Wearable*

La base del dispositivo electrónico portable (*wearable*) consiste en un arreglo de sensores ultrasónicos, que capturan el tiempo aproximado a un obstáculo. Los 4 sensores forman una cruz al frente del cuerpo del usuario, y cada uno de ellos tiene una dirección diferente: izquierda, derecha, arriba y abajo. Cuando el arreglo de sensores determina el tiempo hacia posibles obstáculos, el microcontrolador PIC16F886 procesa esta información y establece 4 distancias en centímetros. Posteriormente, esas distancias se envían al dispositivo con OS *Android* en tramas de bits a través del módulo *Bluetooth* (modelo HC-06) utilizando el protocolo `RS-232`, como se muestra en la figura 5. El dispositivo *Android* es la entidad principal que alerta al usuario por medio de sonidos particulares y patrones de vibración.

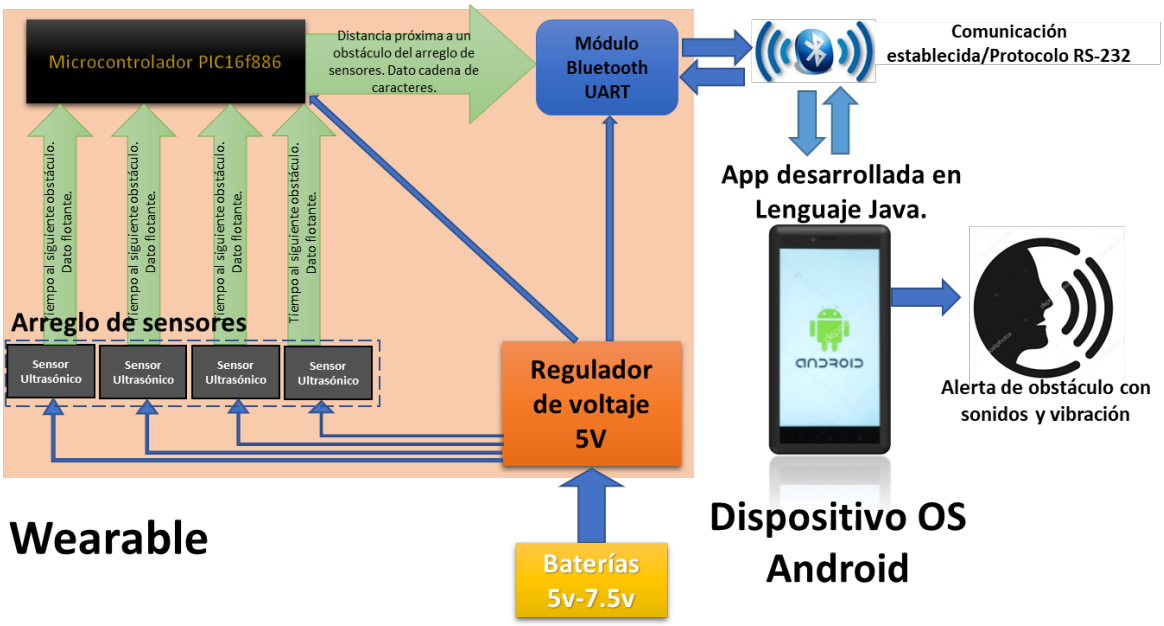

Figura 5. Diagrama a bloques del sistema.

Los transductores son suficientemente ligeros y pequeños como para fijarlos a una gama muy amplia de ropa. Para la experimentación se utilizó una prenda de vestir similar a un chaleco.

6.4.1. Arquitectura del Dispositivo

La arquitectura principal consiste en un microcontrolador, un arreglo de sensores, el módulo de comunicación *Bluetooth* y un regulador de voltaje. Los pines `TRIGGER` y `ECHO` de los cuatro sensores ultrasónicos se asignaron al `puerto B` del microcontrolador, como se aprecia en la figura 6. Por otro lado, los pines de recepción y trasmisión de datos del módulo *Bluetooth* se conectaron a los `puertos Rx` y `Tx` (consultar figura 6) donde se encuentra el módulo mejorado EUSART propio del microcontrolador, el cual soporta los siguientes protocolos *[8]*.

- RS-485, RS-232 y LIN 2.0
- Auto-Baud Detect
- Auto-Wake-Up on Start bit

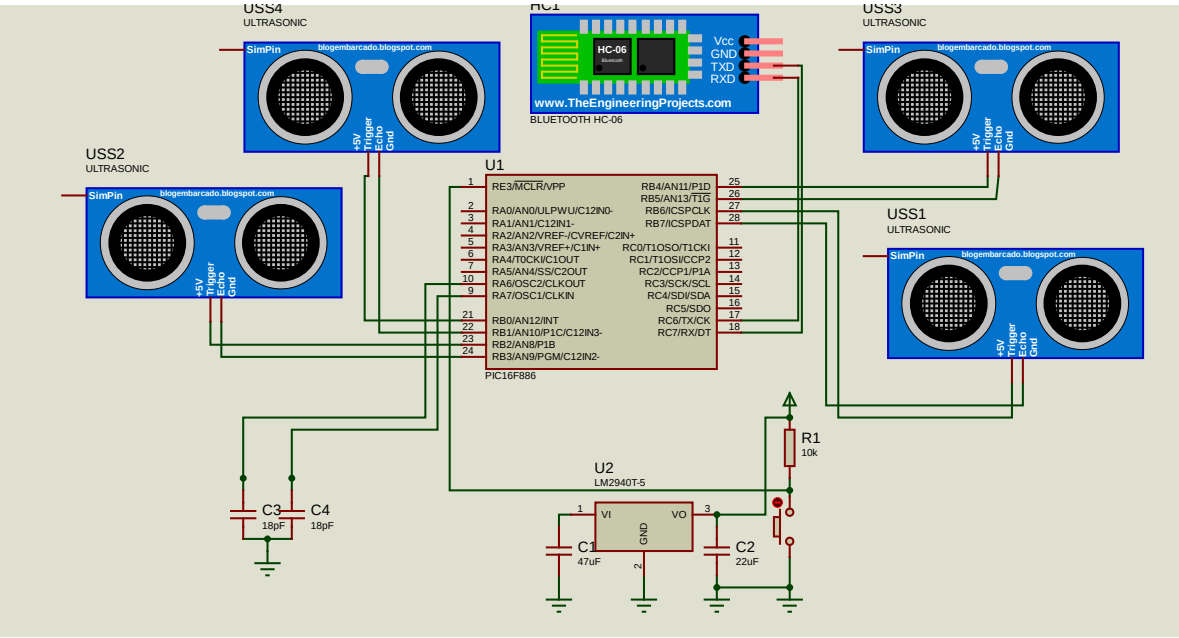

Figura 6. Diseño del circuito.

6.4.2. Diseño eléctrico PCB

Se diseñó una placa que tiene acceso a todos los pines del PIC16F886 con los *header's PINES MICRO 1* y *PINES MICRO 2*, como se muestra en la figura 7. Se integró un regulador de voltaje en la entrada *POWER PLACA* y posteriormente un *reset* en la entrada *POWER MICRO*. Consultar figura 5. Es posible alimentar directamente el circuito por las entradas *POWER MICRO* y *POWER DESDE PLACA* con $5v$ para evitar que la energía se disipe en el regulador, de lo contrario la fuente de energía debe ser de $6v - 7v$ desde la entrada *POWER PLACA*. También se agregó una hilera de *header´s* llamada *COMPONENTES*, que sirve para alimentar diferentes componentes externos. La energía de los componentes externos puede ser suministrada por el mismo circuito, o retirando el *JUMPER* pueden ser alimentados desde una fuente externa. La placa es versátil, diseñada para la experimentación. El diseño eléctrico en Proteus se muestra en la figura 7.

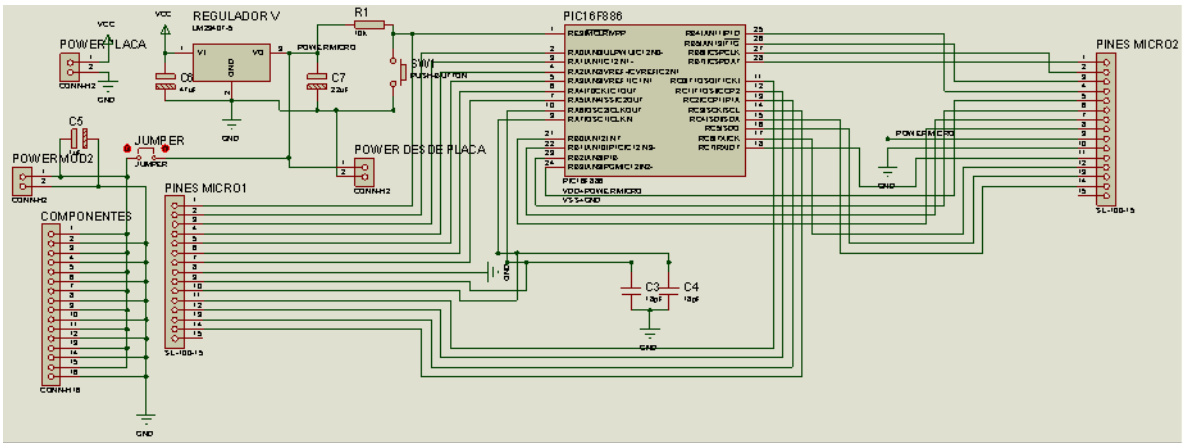

Figura 7. Diseño PCB.

El diseño de la placa fue realizado en el software *Altium*. Los *footprints* fueron hechos a mano con la guía de los *datasheet* de cada componente. El diseño de la placa se aprecia en la figura 8. La máscara de componentes se muestra en la figura 9.

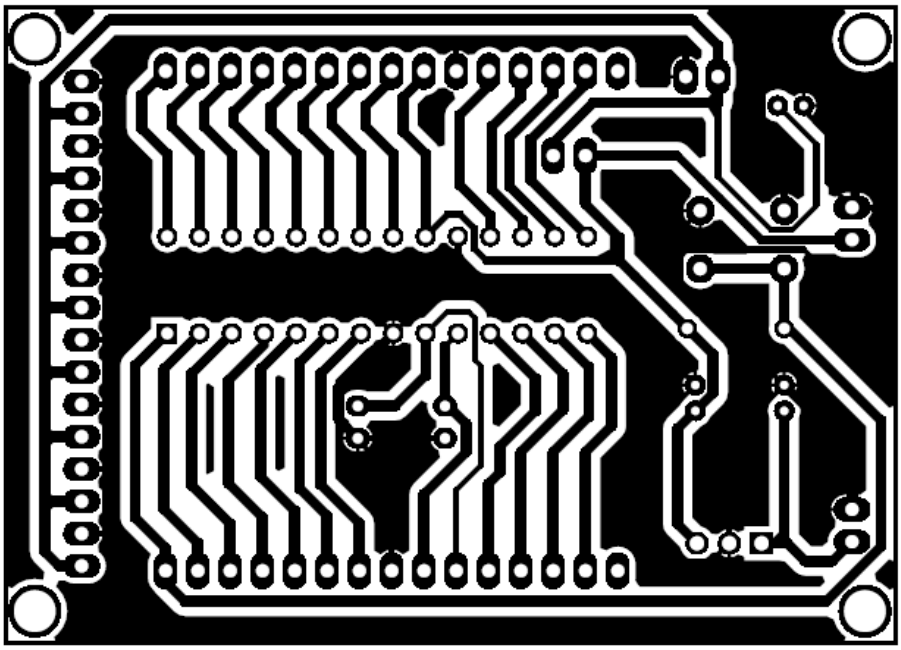

Figura 8. Diseño PCB.

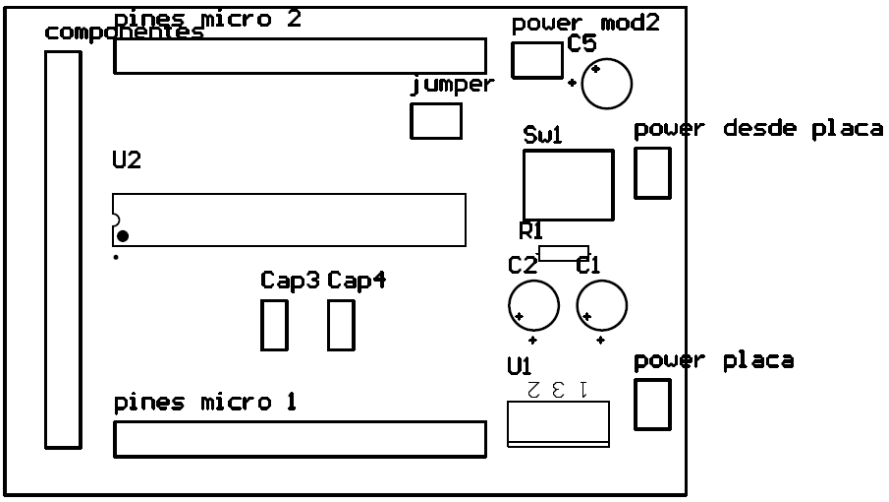

Figura 9. Mascara de Componentes.

6.5. Implementación del *Wearable*

6.5.1. Introducción

El *wearable* fue construido con base en los diseños de la sección 6.4. A continuación se describen características importantes de los componentes utilizados para que el *wearable* opere adecuadamente,

6.5.2. Arreglo de Sensores

El *wearable* está basado en sensores ultrasónicos funcionando sincronizadamente. Se analizó y determinó el lugar y la posición de los sensores en el cuerpo del usuario. El modelo del sensor ultrasónico que se empleó en este sistema es el US-100 con tecnología TTL que incluye transmisores ultrasónicos, receptor y el circuito de control, como se muestra en la figura 10. Mientras que en la figura 11, se observa el patrón de radiación aproximado para cada sensor. Las pruebas de calidad no fueron superadas por el modelo HC-SR04 inicialmente propuesto, por lo que fue reemplazado por el US-100.

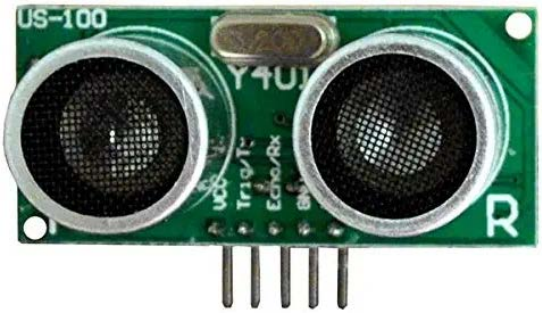

Figura 10. Sensor Ultrasónico US-100.*[9]*

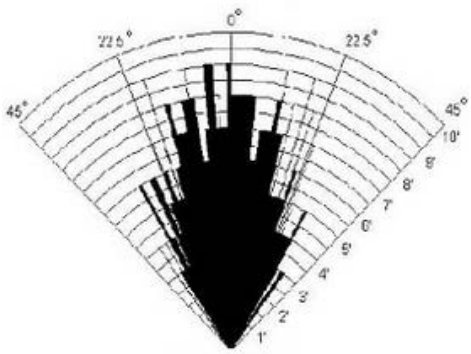

Figura 11. Patrón de radiación del modelo US-100.*[9]*

La forma de operar del sensor ultrasónico a utilizar consiste en enviar un pulso alto de al menos $10uS$ por el pin *Trigger* (Disparador). Una vez que el disparador haya mandado la señal, el sensor envía 8 pulsos de 40KHz hacia el medio, tal y como se muestra en la figura 12. Posteriormente se inicializa en alto el pin *Echo*, se detecta este evento para comenzar un conteo de tiempo. La salida del pin *Echo* se mantiene en alto hasta recibir el eco reflejado por el obstáculo, en ese momento cambia a un estado bajo, es decir, al terminar de contar el tiempo. Observar figura 12.

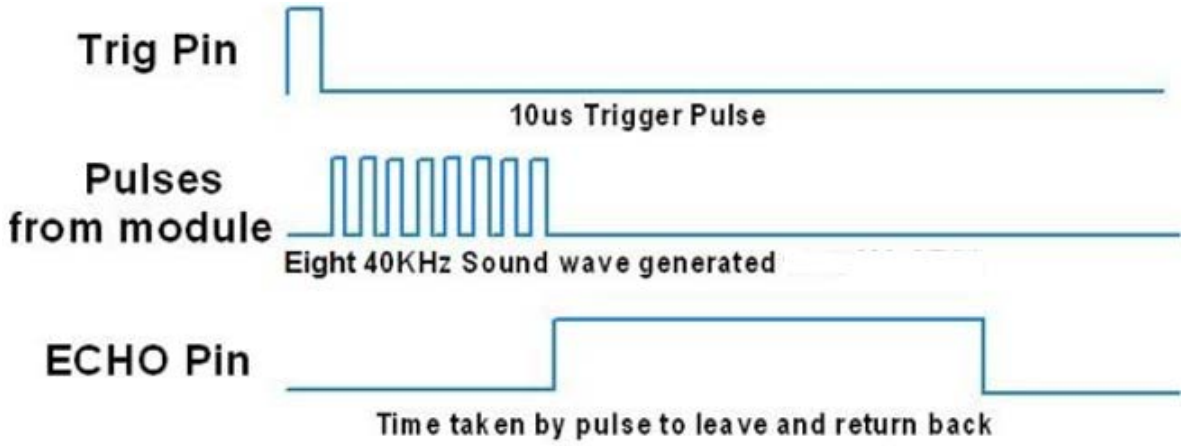

Figura 12. Funcionamiento del sensor ultrasónico.*[9]*

La distancia es proporcional a la duración del pulso y se calculará con las siguiente formula.

$$Velocidaddelsonido = 340\,m/s$$
$$Distancia(centímetros) = (Tiempomedidoenmicrosegundos)(0.017) \quad (1)$$

6.5.3. PIC16F886

Para determinar la distancia de posibles obstáculos, se escogió el microcontrolador PIC16F886 (figura 13) de Microchip por sus múltiples ventajas. El tamaño es relativamente pequeño comparado a otros modelos con las mismas funciones, es muy comercial, tiene un costo competitivo, además dispone de un módulo EUSART para implementar el protocolo RS-232 de una manera simple.

Figura 13. PIC16F886.*[8]*

El módulo (EUSART) es un periférico de comunicación serie de entrada y salida. El módulo contiene todos los generadores de reloj, registros de desplazamiento y buffer de datos necesarios para realizar una transferencia de datos en forma serie logrando ser independiente de la ejecución del programa del dispositivo. La EUSART también es conocida como *interface* de comunicación serie (SCI), y permite que se pueda configurar como *full-duplex*, (sistema asíncrono o síncrono) o *half-duplex*. El modo *full-duplex* es útil para comunicaciones con sistemas periféricos y computadoras. Por otro lado, el modo síncrono está destinado a las comunicaciones con dispositivos periféricos, tales como *A/D* o *D/A*, *EEPROM´s* u otros microcontroladores. Estos dispositivos generalmente no tienen relojes internos para la generación y la velocidad de transmisión, ya que se requiere el reloj externo y la señal de reloj es proporcionada por un dispositivo síncrono maestro.

Características del módulo EUSART *[10]*:

- Transmisión y recepción asíncrona *full-duplex*
- *Buffer* de entrada de dos caracteres
- *Buffer* de salida de un carácter
- Programable de 8 bits o 9 bits en la longitud de caracteres a transmitir
- Detección de direcciones en el modo de 9 bits
- Detección de error por desbordamiento en el *buffer*
- Detección de error de trama en carácter recibido
- Maestro síncrono *semi-duplex*
- Esclavo síncrono *semi-duplex*
- Reloj programable y polaridad de datos

Los registros para controlar las operaciones del módulo EUSART son las siguientes.

- Control y estado del transmisor (TXSTA)
- Control y estado del receptor (RCSTA)
- Control del BAUD Rate (BAUDCON)

Para todos los modos de operación de la EUSART. Los bits de control *TRIS* correspondientes a *RX/DT* y *TX/CK* se les asigna ‘1’. Automáticamente los pines de la EUSART son reconfigurados como entrada o salida, según sea el caso. Cuando no se activa la transmisión o recepción, los

pines *RX* y *TX* pueden ser usados como de propósito general. En las figuras 14 y 15 se aprecia el diagrama de Bloques del Transmisor y del Receptor EUSART respectivamente.

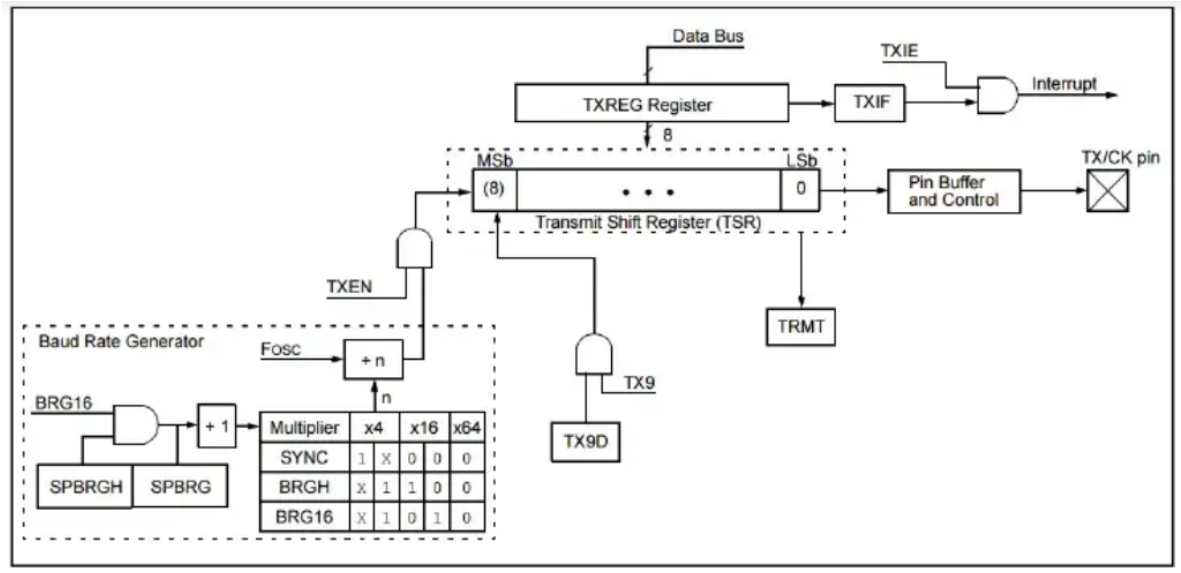

Figura 14. Diagrama a Bloques del Transmisor EUSART. *[8]*

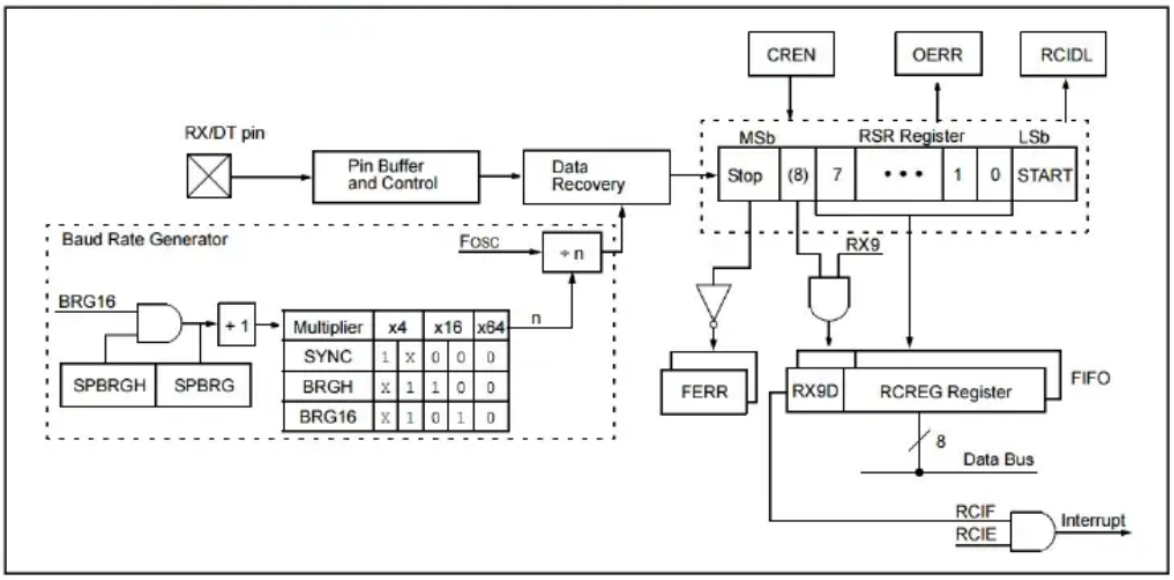

Figura 15. Diagrama a Bloques del Receptor EUSART. *[8]*

6.5.4. Módulo *Bluetooth*

El módulo serial *Bluetooth* HC-06 (esclavo), dispone de tecnología inalámbrica *Bluetooth* versión 2 (figura 16. Este módulo permite a cualquier microcontrolador con un puerto serial que implementa el protocolo RS-232 estándar, comunicarse con una CPU o un *smartphone* equipado con un módulo *Bluetooth* maestro. El módulo HC-06 ofrece el servicio de puerto serie (RFCOMM), creando un enlace de datos transparente entre un dispositivo que

implemente un cliente con *Bluetooth* como maestro y el microcontrolador. La salida del módulo es una señal serial asíncrona estándar de 9600 baudios.

Figura 16. Módulo *Bluetooth* HC-06.*[11]*

6.5.5. Regulador de voltaje

Con respecto a la fuente de energía utilizada, se implementó el circuito de la figura 17 con el fin de regular el voltaje de entrada de $7v$ a $5v$ y pobra diferentes fuentes de alimentación.

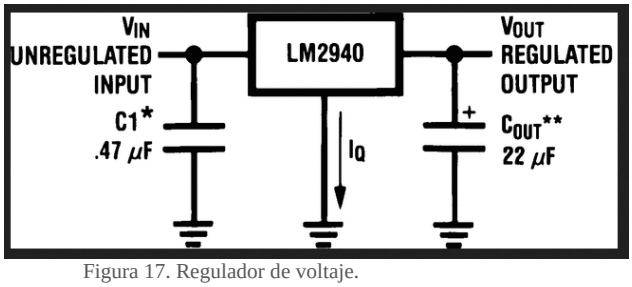

Figura 17. Regulador de voltaje.

6.5.6. Implementación de la placa PCB

Los módulos y componentes de las secciones 6.5.2. A la 6.5.5. Se soldaron a la placa, como se aprecia en la figura 18 y 19. Se probaron todas las conexiones y los posibles cortocircuitos que pudieran existir. Una vez comprobado los terminales, a la placa se le cargo un programa *Blink* que funcionó correctamente.

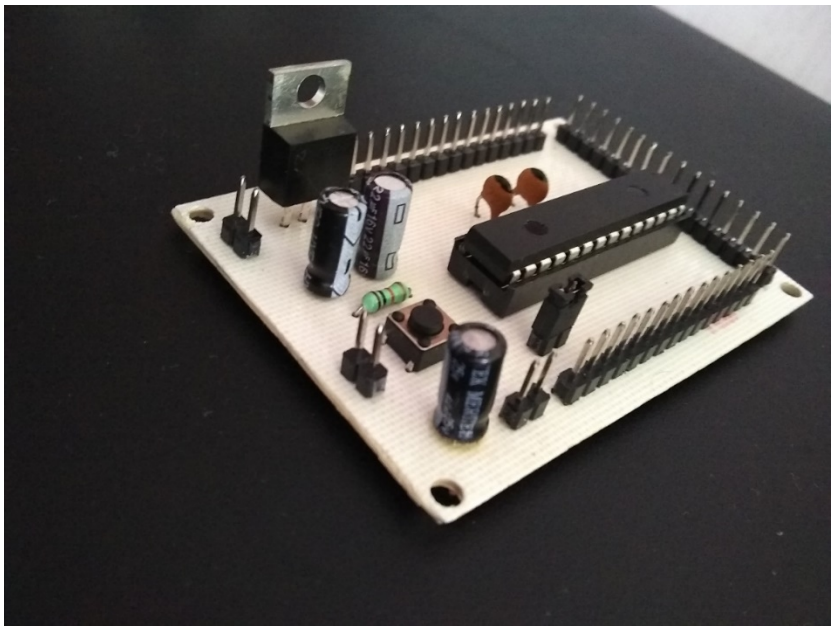

Figura 18. PCB; 1.

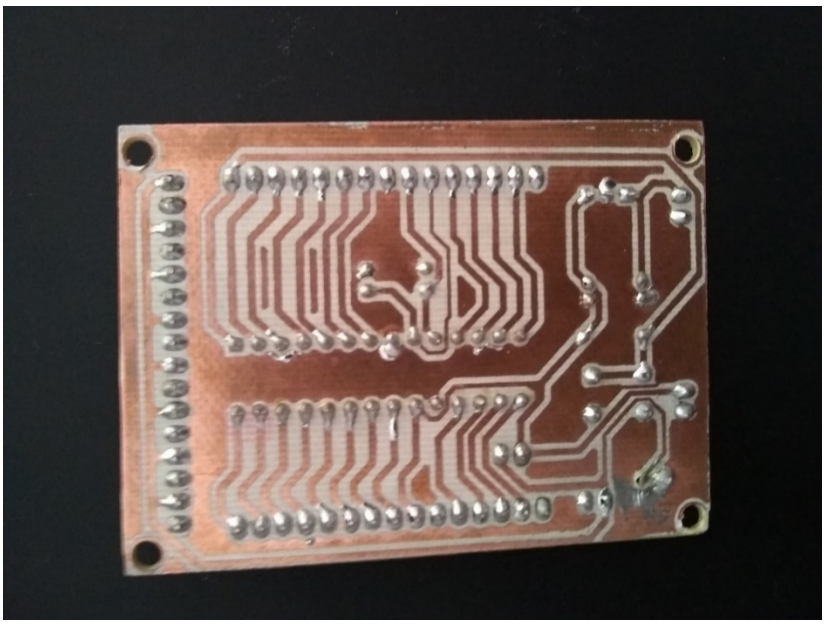

Figura 19. PCB;2

6.5.7. Algoritmo para Microcontrolador

El algoritmo para el microcontrolador PIC16F886 fue implementado a través del software *PIC C Compiler* con la finalidad de disminuir las instrucciones escritas en lenguaje ensamblador. El objetivo del algoritmo es enviar hacia la *App* las distancias a la que se encuentran posibles obstáculos. Inicialmente se captura el tiempo de un primer sensor en dirección al medio; enseguida, se calcula la distancia con la *ecuación 1*. Después se deja pasar un retardo de 20 ms y se obtiene el tiempo de un segundo sensor en dirección al medio;

posteriormente, se calcula la distancia con la *ecuación 1*. Seguidamente se deja pasar un retardo de 20 ms, y así sucesivamente con cada sensor. Finalmente, se envían las distancias en una trama de bits a través del módulo *Bluetooth*, tal y como se muestra en la figura 20.

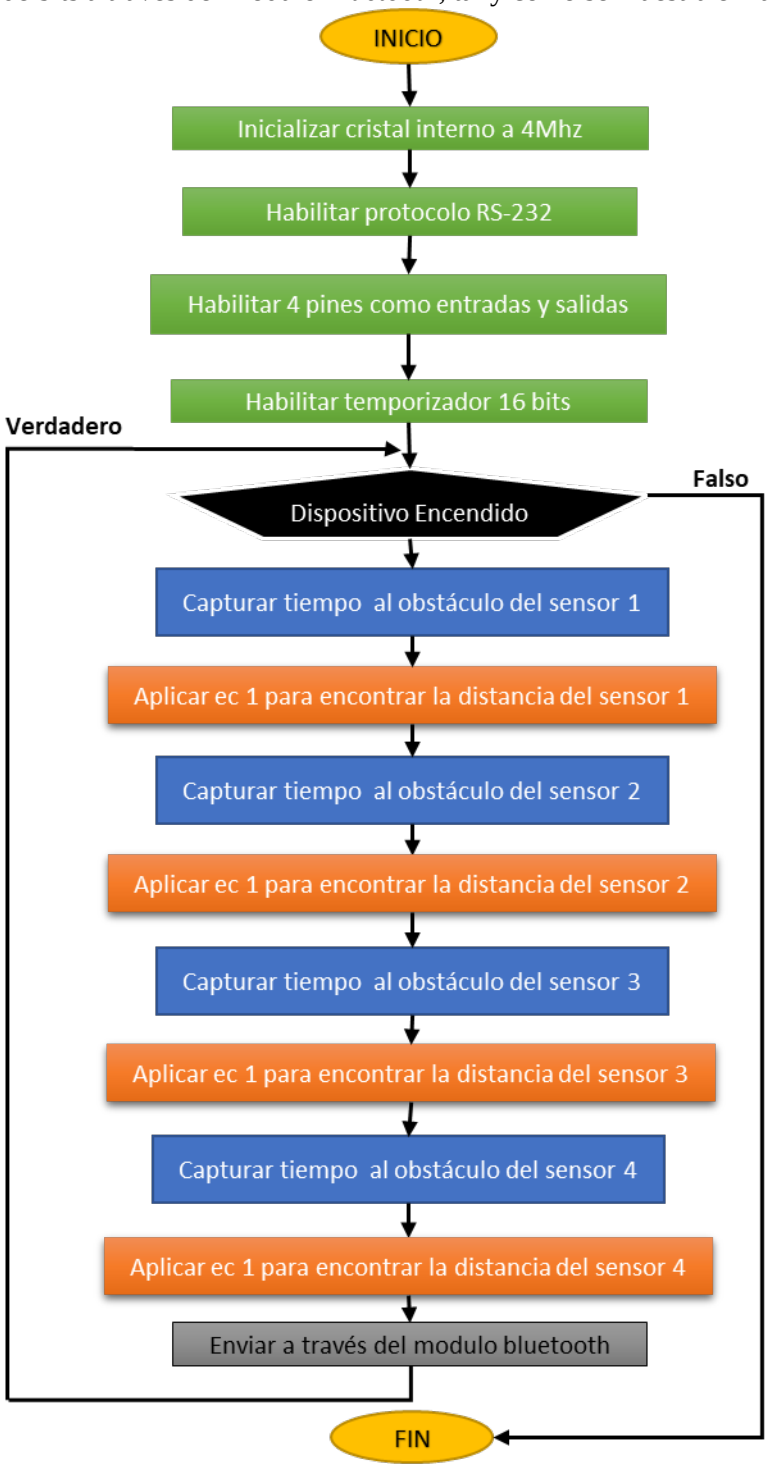

Figura 20. Diagrama de bloques de la programación del PIC16F886.

El algoritmo para el microcontrolador PIC16F886 se expresa en el pseudocódigo 1 y la implementación en lenguaje ensamblador se encuentra en el Apéndice A. *Wearable.*

1. Inicializar el cristal interno del pic16F886 a 4MHz.
2. Habilitar el protocolo de comunicación RS-232/*Bluetooth* a 9600 baudios.
3. Habilitar como salidas los pines del microcontrolador donde se conectarán los pines TRIGGER de los cuatro sensores.
4. Habilitar como entradas los pines del microcontrolador donde se conectarán los pines ECHO de los cuatro sensores.
5. Habilitar el temporizador de 16 bits a la frecuencia adecuada.
6. Enviar un estado alto al pin TRIGGER del sensor ultrasónico 1.
7. Inicializar un retardo de 10us.
8. Enviar un estado bajo al pin TRIGGER del sensor ultrasónico 1.
9. Mientras el pin ECHO del sensor ultrasónico 1 devuelva un estado bajo, el temporizador no inicializará la cuenta. Cuando este pin cambie a estado alto, el temporizador comenzará la cuenta.
10. Asignar el valor del temporizador de 16 bits a una variable flotante. Si el tiempo del temporizador es mayor a un tiempo aproximado a medio metro, asignar 8600 a la variable flotante.
11. Utilizar ec. 1. Para obtener la distancia y asignarla a una variable flotante correspondiente al sensor 1.
12. Inicializar un retardo de 20ms.
13. Enviar un estado alto al pin TRIGGER del sensor ultrasónico 2.
14. Inicializar un retardo de 10us.
15. Enviar un estado bajo al pin TRIGGER del sensor ultrasónico 2.
16. Mientras el pin ECHO del sensor ultrasónico 2 devuelva un estado bajo, el temporizador no inicializará la cuenta. Cuando este pin cambie a estado alto, el temporizador comenzará la cuenta.
17. Asignar el valor del temporizador de 16 bits a una variable flotante. Si el tiempo del temporizador es mayor a un tiempo aproximado a medio metro, asignar 8600 a la variable flotante.
18. Utilizar ec. 1. Para obtener la distancia y asignarla a una variable flotante correspondiente al sensor 2.
19. Inicializar un retardo de entre 20ms.
20. Enviar un estado alto al pin TRIGGER del sensor ultrasónico 3.
21. Inicializar un retardo de 10us.
22. Enviar un estado bajo al pin TRIGGER del sensor ultrasónico 3.
23. Mientras el pin ECHO del sensor ultrasónico 3 devuelva un estado bajo, el temporizador no inicializará la cuenta. Cuando este pin cambie a estado alto, el temporizador comenzará la cuenta.
24. Asignar el valor del temporizador de 16 bits a una variable flotante. Si el tiempo del temporizador es mayor a un tiempo aproximado a medio metro, asignar 8600 a la variable flotante.
25. Utilizar ec. 1. Para obtener la distancia y asignarla a una variable flotante correspondiente al sensor 3.
26. Inicializar un retardo de 20 ms.
27. Enviar un estado alto al pin TRIGGER del sensor ultrasónico 4.
28. Inicializar un retardo de 10us.
29. Enviar un estado bajo al pin TRIGGER del sensor ultrasónico 4.
30. Mientras el pin ECHO del sensor ultrasónico 4 devuelva un estado bajo, el temporizador no inicializará la cuenta. Cuando este pin cambie a estado alto, el temporizador comenzará la cuenta.
31. Asignar el valor del temporizador de 16 bits a una variable flotante. Si el tiempo del temporizador es mayor a un tiempo aproximado a medio metro, asignar 8600 a la variable flotante.

```
32. Utilizar ec. 1. Para obtener la distancia y asignarla a una variable flotante
    correspondiente al sensor 4.
33. Inicializar un retardo de 10ms.
34. Enviar las 4 distancias por el pin asignado a la transmisión de datos vía Bluetooth.
35. Volver al paso 6.
```

Pseudocódigo 1. Algoritmo para medir las distancias de los sensores.

6.6. Diseño de la *App*

La *App* que trabaja en conjunto con el *wearable* se desarrolló con la tecnología *Android*. Una vez establecida la comunicación con el *wearable*, la *App* procesa los datos y notifica al usuario si hay un posible obstáculo, emitiendo un sonido particular, además de un patrón de vibración específico, tal y como se aprecia en la figura 5.

A continuación, se describe la arquitectura de la *App,* a través de la vista lógica y la vista física del sistema. La sección 6.6.1. Describe el modelo de clases que da origen a la *App,* y la sección 6.6.2. presenta la distribución de las clases en dos capas.

6.6.1. Vista lógica: diagrama de clases del sistema para la App

El diagrama de clases está compuesto de 12 clases concretas, donde dos de ellas son *interfaces,* ver figura 21. A continuación se describe cada una. El código de cada clase se puede encontrar en el Apéndice B. *App*.

`LogoUam`: Primera Pantalla; muestra un *splashscreen* de 8 segundos de duración con el logo Uam Azcapotzalco. Informa al usuario invidente, por medio de la voz Artificial *Android*, si está activada la conexión *Bluetooth* en el dispositivo con OS *Android*, y si es que el dispositivo soporta la conexión, de lo contrario cierra la *App*. Si la conexión está apagada, por reconocimiento de voz se activa.

`MiLogo`: Segunda Pantalla; muestra un *splashscreen* de 6 segundos de duración con el Logo de la *App*. Informa al usuario invidente, por medio de la voz Artificial *Android* la inicialización de la comunicación con *wearable*.

`MainActivity`: Tercera Pantalla; función principal de la Aplicación. Inicializa el evento de conexión, recepción y trasmisión de datos. Recibe tramas de caracteres desde el *wearable*. Limpia la trama para obtener las distancias entre el *wearable* y los objetos. Implementa un algoritmo para determinar las distancias, y determinar si se trata de un obstáculo; en consecuencia, alertará por medio de vibraciones y sonidos específicos.

`TextoVoz`: Convertir una cadena de caracteres a voz Artificial de *Android*.

`HiloTemporizador`: Permite cambiar de una *activity* a otra, en una determinada cantidad de tiempo.

`HiloTempoComandVoice`: Lanza un servicio, lanza una *activity* y cierra la *activity* actual en una determinada cantidad de tiempo.

`ConnectedThread`: Crea un evento de conexión *Bluetooth* sobre un Hilo.

`MiHandler`: Desde el sistema operativo de *Android* obtiene el mensaje proveniente del *wearable,* y filtra los caracteres indeterminados del mensaje. La trama de caracteres recibida

en el mensaje puede contener ruido adicionado por distintos factores, lo que impediría recuperar el mensaje original. El mensaje filtrado es enviado al siguiente filtro `SubProcesos`.

`SubProcesos`: Separa en *tokens* el mensaje que contiene las distancias de los sensores. Con esta acción se implementa un algoritmo para manipular la vibración y el sonido. Además, la clase muestra en pantalla las distancias obtenidas; con el propósito de que sean analizadas por desarrolladores e investigadores, que deseen mejorar el sistema.

`SonidoVibracionRunnable`: Reproduce un sonido y hace vibrar el dispositivo al mismo tiempo. El patrón de vibración y el sonido para reproducir se asignarán dependiendo la distancia dada.

`TextViewRunnable`: Refresca en la pantalla del dispositivo móvil *Android* la distancia de algún sensor. La clase se ejecutará sobre un Hilo y se iniciará el componente de texto.

`ServicioActivo`: Servicio que monitorea en segundo plano cuando el *Bluetooth* se haya habilitado. Una vez ocurrida esta acción, lanza una *activity*.

`TextToSpeech`: Sintetiza la voz del texto para su reproducción inmediata o para crear un archivo de sonido.

`Runnable`: La interfaz debe ser implementada por cualquier clase cuyos ejemplares estén destinados a ser ejecutados por un hilo.

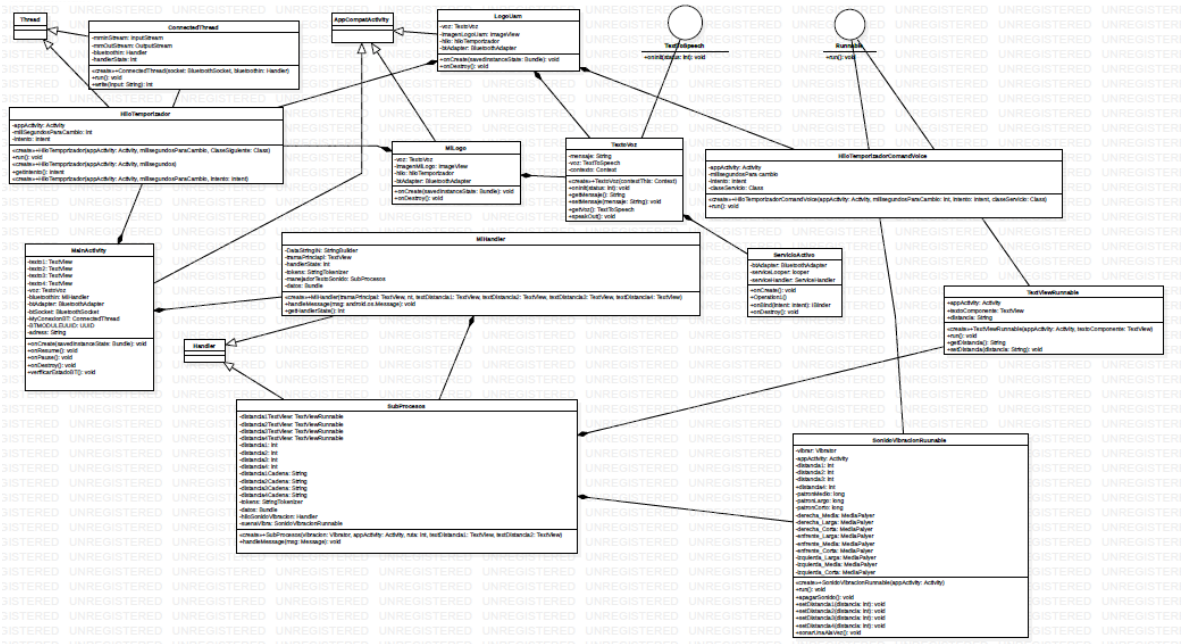

Figura 21. Diagrama UML.

6.6.2. Vista lógica: Distribución de las clases

La *App* fue diseñada siguiendo una arquitectura de dos capas: presentación y lógica de negocio. La capa de presentación está formada por tres medios de comunicación: sonido,

vibración e interfaz gráfica. La capa de lógica de negocio lleva a cabo el procesamiento de los datos. Es importante mencionar que la *App* no cuenta con capa de datos, porque no requiere almacenar la información que procesa, a saber las tramas de caracteres que recibe del *wearable*.

El modelo de negocio se encarga de recuperar la trama de caracteres enviada por *wearable* con la clase `MiHandler`. Seguidamente, la clase `SubProcess` filtra la información para manipularla y asignarla a una determinada distancia. La capa de presentación (o vista), muestra las distancias recuperadas, correspondientes a cada sensor, a través de la clase `TextViewRunnable`. Para que los desarrolladores e investigadores puedan realizar determinados experimentos. La capa de presentación también cuenta con sonido y vibración, la clase que implementa el algoritmo de alerta con sonido y vibración es `SonidoVibracionRunnable`. La figura 22 muestra como están distribuidas las clases del sistema.

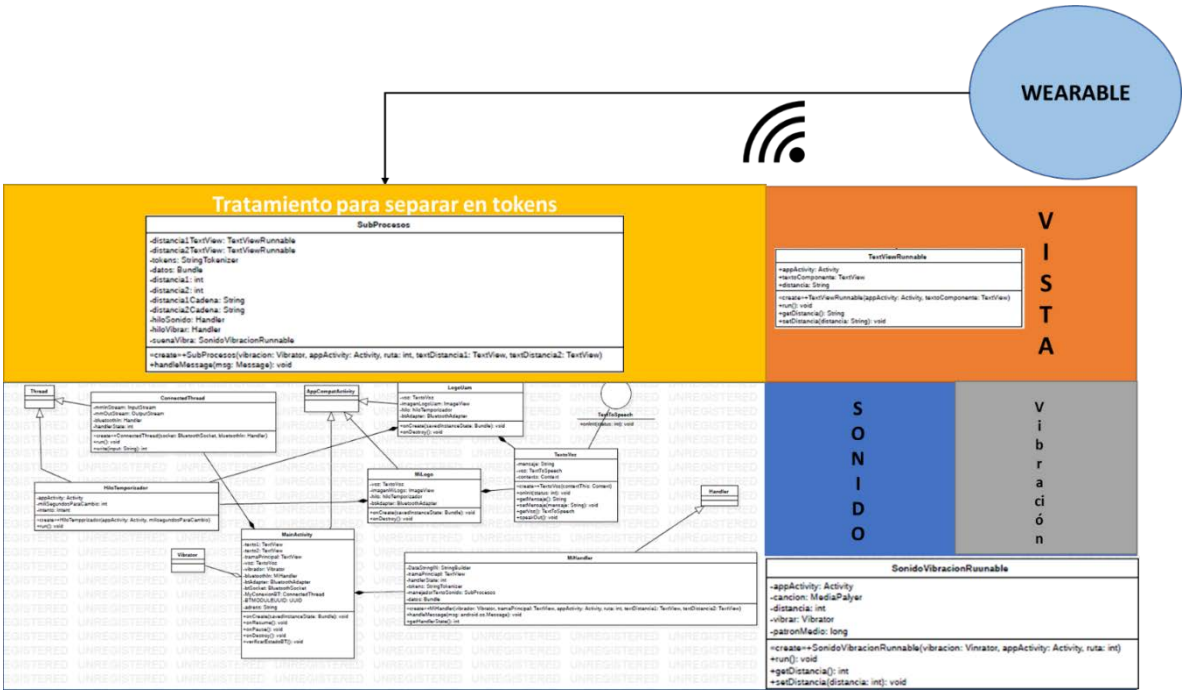

Figura 22, Modelo 2 Capas.

6.6.3. Vista física: modelo cliente-servidor

Se implementó el modelo cliente-servidor para crear una conexión por medio de la *API* de *Bluetooth* (ver figura 23), y así transferir las distancias del arreglo de sensores. El *wearable* funciona como servidor y la *App* en el dispositivo *Android* como cliente, ver figura 24.

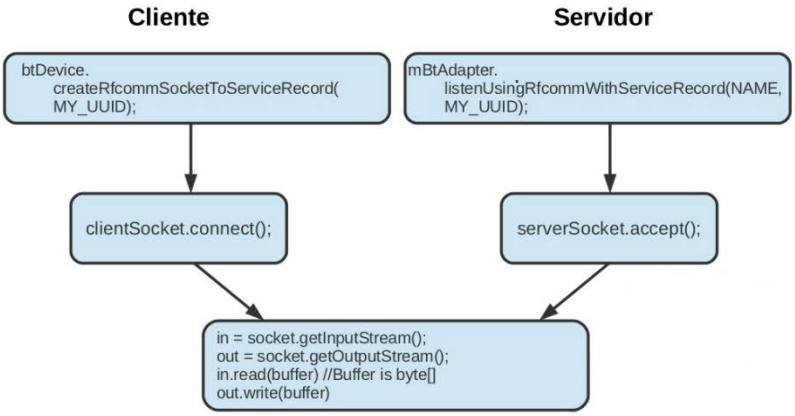

Figura 23. Arquitectura cliente servidor utilizando la *API* de *Bluetooth*.

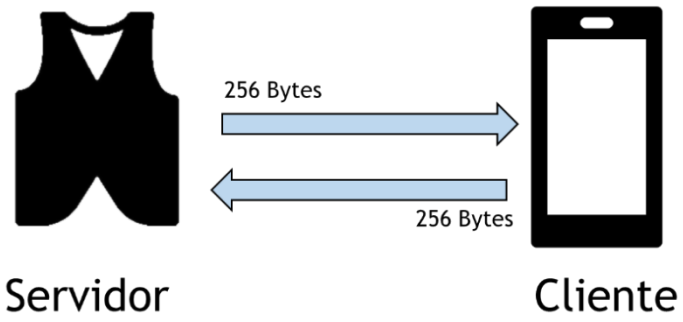

Figura 24. Arquitectura cliente servidor.

6.7. Implementación de la *App*

6.7.1. Algoritmos implementados

A continuación, se describen los algoritmos implementados para los módulos de comunicación entre el *wearable* y la *App*, y el módulo que realiza el procesamiento de los datos para alertar al usuario. El código del pseudocódigo 2 y 3 se puede encontrar en el Apéndice B. *App*.

6.7.2. Módulo de comunicación entre el wearable y la *App*

Con base en el modelo cliente-servidor descrito en la sección 6.6.3, el algoritmo para establecer la comunicación entre el *wearable* y la *App* está expresado en el pseudocódigo 2.

1. El *wearable* debe mantener un socket de servidor abierto para inicializar la conexión con el dispositivo *Android*.
2. Utilizar la dirección MAC del *wearable* y obtener un objeto BluetoothDevice que represente al *wearable*.

3. Usar el BluetoothDevice del paso anterior y obtener un objeto BluetoothSocket e inicializar la conexión.
4. Llamar al método connect() del ejemplar BluetoothSocket para la conexión.
5. El sistema realizara una búsqueda del SDP en el dispositivo remoto a fin de que coincida el UUID.
6. Si la búsqueda tiene éxito y el dispositivo remoto acepta la conexión, este último comparte el canal RFCOMM que se usa durante la conexión y se muestra connect().
7. De lo contrario, si la conexión falla o el método connect() caduca (después de unos 12 segundos), se genera una excepción (este procedimiento de conexión siempre debe realizarse en un proceso por separado del subproceso de la actividad principal).
8. Obtener los ejemplares InputStream y OutputStream que administran transmisiones a través del socket mediante los métodos getInputStream() y getOutputStream(), respectivamente.
9. Leer y escribir datos para los flujos se realiza con read(byte[]) y write(byte[]), respectivamente.
10. Obtener un objeto DataStringIN y utilizar el método append() , de tal manera se consigue la cadena de tipo String que contiene las distancias de los 4 sensores .Se debe implementar en un subproceso aparte de la activity principal.
11. Cuando el usuario cierre la *App* se debe finalizar la conexión utilizando el objeto BluetoothSocket llamando al método close().

Pseudocódigo 2. Comunicación *Bluetooth*.

6.7.2.1. Módulo de procesamiento de datos para alertar al usuario

La *App* determina si existe algún obstáculo potencialmente cercano al usuario. Posteriormente, utilizando la clase `MediaPayer` y `Vibrator` genera un sonido acompañado de vibración para alertar al usuario.

El algoritmo para el procesamiento de las tramas, recibidas del *wearable*, está especificado con base en una serie de condiciones, las cuales se presentan en las tablas 1, 2 y 3.

Para aplicar el algoritmo adecuadamente, el sensor 1 se debe encontrar del lado derecho, mientras que el sensor 2 al lado izquierdo y finalmente el sensor 3 y 4 se ubican en el centro. En la tabla 1 se presentan las condiciones para que el sonido y la vibración se desactiven.

Tabla 1. Condiciones para Anular Sonido y Vibración.

Entradas (distancias de los sensores a un obstáculo) d = distancia (cm) sensor 1 && Sensor 2 && (sensor 3 && sensor 4)			**Salida (Sonido y Vibración)**	
Sensor 1 (derecha)	**Sensor 2 (izquierda)**	**Sensor 3 y 4 (enfrente)**	**Sonido**	**Vibración**
d >= 120	d >= 120	d >= 120	----------	----------

En la tabla 2 se presentan todas las condiciones para activar el sonido y la vibración dependiendo primordialmente de la distancia de un solo sensor o de varios, para el caso del sensor 3 y 4.

Tabla 2. Condiciones para activar Sonido y Vibración dependiendo principalmente de una distancia.

Entradas (distancias de los sensores a un obstáculo) d = distancia (cm) sensor 1 & Sensor 2 & (sensor 3 || sensor 4)			**Salida (Sonido y Vibración)**	
Sensor 1 (derecha)	**Sensor 2 (izquierda)**	**Sensor 3 y 4 (enfrente)**	**Sonido**	**Vibración**
120 > d >= 60	d >= 120	d >= 120	Derecha_larga	Patrón_largo
60 > d >= 30	d >= 60	d >= 60	Derecha_media	Patrón_medio
d < 30	d >= 30	d >= 30	Derecha_corta	Patrón_corto
d >= 120	**120 > d >= 60**	d >= 120	Izquierda_larga	Patrón_largo
d >= 60	**60 > d >= 30**	d >= 60	Izquierda_media	Patrón_medio
d >= 30	**d < 30**	d >= 30	Izquierda_corta	Patrón_corto
d >= 120	d >= 120	**120 > d >= 60**	Enfrente_larga	Patrón_largo
d >= 60	d >= 60	**60 > d >= 30**	Enfrente _media	Patrón_medio
d >= 30	d >= 30	**d < 30**	Enfrente _corta	Patrón_corto

En el caso de que varios sensores detecten distancias similares, se activa el sonido y la vibración, como lo muestra la tabla 3.

Tabla 3. Condiciones para activar Sonido y Vibración dependiendo principalmente de varias distancias.

Entradas (distancias de los sensores a un obstáculo) d = distancia (cm) sensor 1 & Sensor 2 & (sensor 3 || sensor 4)			**Salida (Sonido y Vibración)**	
Sensor 1 (derecha)	**Sensor 2 (izquierda)**	**Sensor 3 y 4 (enfrente)**	**Sonido**	**Vibración**
120 > d >= 60	d >= 120	**120 > d >= 60**	Enfrente_larga	Patrón_largo
60 > d >= 30	d >= 60	**60 > d >= 30**	Enfrente_media	Patrón_medio
d < 30	d >= 30	**d < 30**	Enfrente_corta	Patrón_corto
d >= 120	**120 > d >= 60**	**120 > d >= 60**	Enfrente_larga	Patrón_largo
d >= 60	**60 > d >= 30**	**60 > d >= 30**	Enfrente_media	Patrón_medio
d >= 30	**d < 30**	**d < 30**	Enfrente_corta	Patrón_corto
120 > d >= 60	**120 > d >= 60**	**120 > d >= 60**	Enfrente_larga	Patrón_largo
60 > d >= 30	**60 > d >= 30**	**60 > d >= 30**	Enfrente _media	Patrón_medio
d < 30	**d < 30**	**d < 30**	Enfrente _corta	Patrón_corto

Con base en las condiciones de las tablas 1, 2 y 3 se describe el pseudocódigo 3. El código del pseudocódigo 3 se encuentra en el Apéndice B. *App*.

1. Si las distancias de los sensores 1, 2, 3 y 4 son mayores o igual a 120 cm, no se emite ningún sonido ni vibración.
2. De lo contrario, si la distancia del sensor 1 es menor que 120 cm y mayor o igual a 60 cm, y las distancias de los sensores 2, 3 y 4 son mayores o guales a 120 cm, emitir sonido Derecha_Larga y Patrón_largo de vibración.
3. De lo contrario, si la distancia del sensor 1 es menor que 60 cm y mayor o igual a 30 cm, y las distancias de los sensores 2, 3 y 4 son mayores o guales a 60 cm, emitir sonido Derecha_media y Patrón_medio de vibración.
4. De lo contrario, si la distancia del sensor 1 es menor que 30 cm, y las distancias de los sensores 2, 3 y 4 son mayores o guales a 30 cm, emitir sonido Derecha_corta y Patrón_corto de vibración.

```
5.  De lo contrario, si la distancia del sensor 2 es menor que 120 cm y mayor o igual a 60
    cm, y las distancias de los sensores 1, 3 y 4 son mayores o guales a 120 cm, emitir
    sonido Izquierda_Larga y Patrón_largo de vibración.
.
.
.
19. De lo contrario, si las distancias de los sensores 1, 2, 3 y 4 son menores a 30 cm,
    emitir sonido Enfrente_corta y Patrón_corto de vibración.
```

Pseudocódigo 3. Algoritmo de Alerta de Obstáculo.

Fue seleccionado un sonido de alerta y a partir de este se crearon los demás en un tono más grave para el lado derecho y agudos para el lado izquierdo. También se aceleró la pista y se ralentizó para las determinadas distancias. Consultar tabla 4.

Tabla 4. Descripción de Sonidos.

SONIDO	DESCRIPCIÓN
IZQUIERDA_CORTA	Son una serie de pitidos agudos que suenan en un periodo corto de tiempo entre uno y otro.
IZQUIERDA_MEDIA	Son una serie de pitidos agudos que suenan en un periodo medio de tiempo entre uno y otro.
IZQUIERDA_LARGA	Son una serie de pitidos agudos que suenan en un periodo largo de tiempo entre uno y otro.
ENFRENTE_CORTA	Son una serie de pitidos medios que suenan en un periodo corto de tiempo entre uno y otro.
ENFRENTE_MEDIA	Son una serie de pitidos medios que suenan en un periodo medio de tiempo entre uno y otro.
ENFRENTE_LARGA	Son una serie de pitidos medios que suenan en un periodo largo de tiempo entre uno y otro.
DERECHA_CORTA	Son una serie de pitidos graves que suenan en un periodo corto de tiempo entre uno y otro.
DERECHA_MEDIA	Son una serie de pitidos graves que suenan en un periodo medio de tiempo entre uno y otro.
DERECHA_LARGA	Son una serie de pitidos largos que suenan en un periodo largo de tiempo entre uno y otro.

Tabla 5.Descripción de Vibración.

VIBRACIÓN	DESCRIPCIÓN
PATRÓN_CORTO	Vibrará por intervalos, 900ms encendido y 900ms apagado
PATRÓN_MEDIO	Vibrará por intervalos, 600ms encendido y 600ms apagado
PATRÓN_LARGO	Vibrará por intervalos, 400ms encendido y 400ms apagado

6.7.3. Tiempo de Operación del Wearable y elección de la Fuente de Energía

De acuerdo con la arquitectura del *wearable* y necesidades del usuario no vidente, se determinó los requerimientos de corriente para la fuente adecuada.

Consumo de Corriente del *wearable*

- Sensor ultrasónico US-100: $2mA$
- Módulo HC-06: 30 mA a 40 mA
- Microcontrolador PIC16F886: Para un oscilador de 4 MHz el consumo es de aproximadamente 2 mA, más el puerto B en uso es de 200 mA; el total de $202mA$

$$I_{Total\ del\ Circuito} = 4(2\ mA) + 40\ mA + 202\ mA = 250\ mA \quad (2)$$

Entendiendo que el usuario debe permanecer con el *wearable* en uso durante un día completo, aproximadamente entre 18-20 hrs.

Si C es la capacidad de la batería,

$$C = I\,T \quad (3)$$

→

$$C = (250\ mA)\,(20\ h) = 5\ Ah$$

No es recomendable descargar la batería hasta llegar a un valor muy cercano a cero en cada ciclo de carga. Omitir esta información podría llevar a que la batería se degrade más rápido y su capacidad de carga disminuya con el tiempo. Por lo tanto, se propone una batería que no trabaje a menos de 50 % de su carga.

$$C = \frac{5\ Ah}{50\%} = \frac{5\ Ah}{0.5} = 10\ Ah$$

La batería que cumple con las especificaciones es un arreglo en serie con 4 elementos de baterías Energizer recargables NH-15-2300 (HR6). La capacidad de cade elemento es de $2300\ mAh$ y 1.2 v, con un total de 9.2 Ah y 4.8 v. La velocidad de descarga es lenta ya que son baterías hechas de NiMH, conforme la hoja de especificaciones del fabricante *[13]*.

Además, las baterías son comerciales y se pueden conseguir en una gran cantidad de establecimientos.

6.8. Experimentación

La experimentación del sistema para apoyar a una persona con discapacidad visual en su movilidad fue realizada en las siguientes etapas: Experimentación del *wearable* y Experimentación del sistema. A continuación, se describirá cada etapa de la experimentación.

6.8.1. Experimentación del *Wearable*

El *wearable* transmite datos hacia la aplicación móvil. Sin las tramas proporcionadas por el *wearable* es imposible analizar y corroborar el funcionamiento empírico de la *App*. Durante esta etapa se descargó una aplicación gratuita estándar para observar únicamente la recepción de las tramas de caracteres.

El archivo hexadecimal generado para el *pseudocódigo 1*, se cargó al microcontrolador por medio del programador Pick-kit V2. Para ello, se armó el circuito de la figura 25, en una tablilla de pruebas. Es preciso señalar la realización de pruebas efectuadas con anterioridad para comprobar el funcionamiento adecuado de los sensores individualmente.

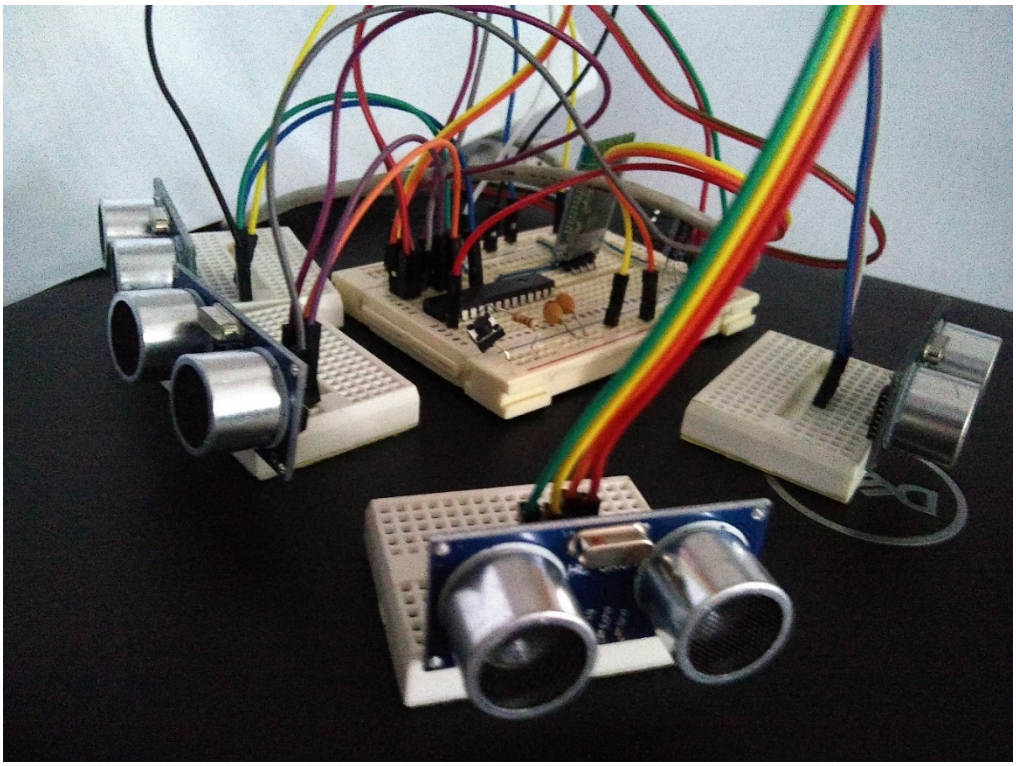

Figura 25. Circuito Principal en Tablilla de Pruebas.

Una vez armado el circuito, el programa cargado en el PIC16F886 y la aplicación genérica instalada en un dispositivo *Android*, se procedió a corroborar las distancias determinadas por el circuito. Las 4 variables flotantes (`distancia1, distancia2, distancia3 y distancia4`) son enviadas al dispositivo *Android* en una trama de caracteres por medio del módulo *Bluetooth*. Los obstáculos eran objetos de superficie variada y diferente textura. Estaban colocados intencionalmente delante de los sensores por periodos de tiempo, para después aproximarlos o alejarlos. Principalmente se buscaba monitorear el tiempo de respuesta del sistema y la precisión de la medida. Las distancias enviadas y cuantificadas por el circuito fueron mostradas por una aplicación gratuita.

6.8.2. Experimentación del sistema

El sistema para apoyar a una persona con discapacidad visual en su movilidad finalmente fue puesto a prueba en circunstancias semejantes a las que enfrentan usuarios con discapacidad visual. Los lugares elegidos para los ensayos fueron universidades, plazas comerciales, casas habitación, parques públicos, entre otros. La edad de los participantes oscila entre los 20 hasta los 55 años. Los participantes usaban el producto de 20 a 40 minutos aproximadamente bajo supervisión constante por parte del responsable del proyecto. Al terminar la prueba, cada usuario proporcionó su experiencia empleando el wearable y externó su opinión sobre ventajas, fallas y mejoras sobre el sistema.

Figura 26. *Wearable*; vista frontal.

Figura 27. *Wearable*; vista posterior.

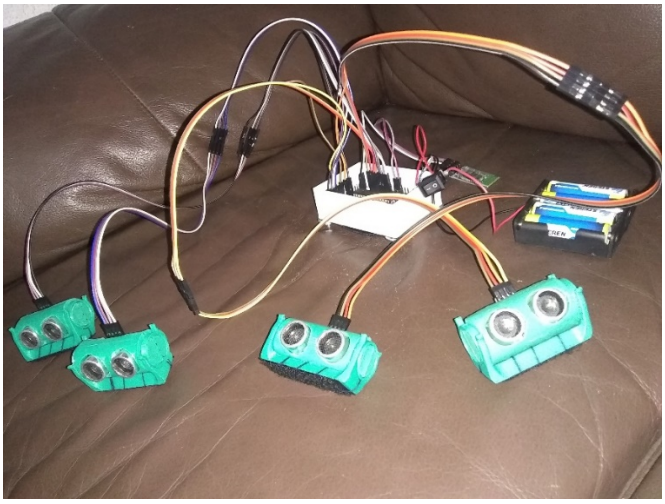

Figura 28. *Wearable*.

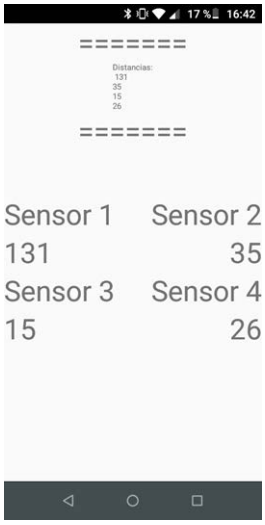

Figura 29. *App.*

Para proteger al circuito principal, como a los sensores ultrasónicos se mandaron imprimir los modelos de las figuras 30 y 31.

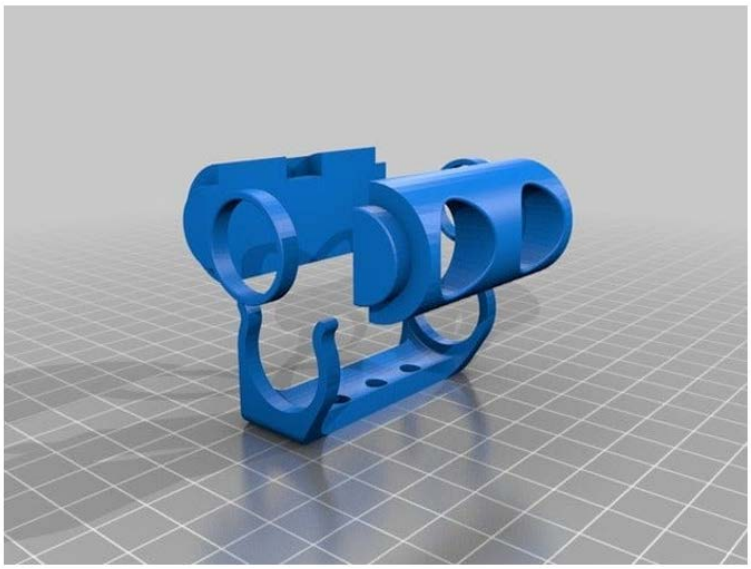

Figura 30. Diseño impresión 3D carcasa sensor.

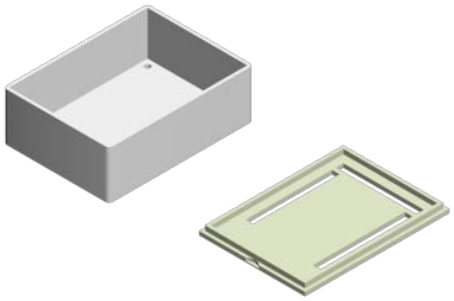

Figura 31. Diseño impresión 3d carcasa circuito.

Las carcasas se hicieron con una pasta que al enfriarse es bastante dura y resiste, aunque los diseños físicos no quedaron tan detallados como en el diseño digital, resultaron muy útiles para el *wearable*. Además, a la carcasa de los sensores ultrasónicos se les unió con hilo una cinta *contactel* para que se pudieran adherir al chaleco, como se muestra en la figura 32. Por otro lado, la carcasa del circuito se unió a la cinta *contactel* utilizando tornillos, como lo muestra la figura 33. Todas las partes de cinta *contactel* pueden ser retiradas, el sistema está hecho para ser modificado.

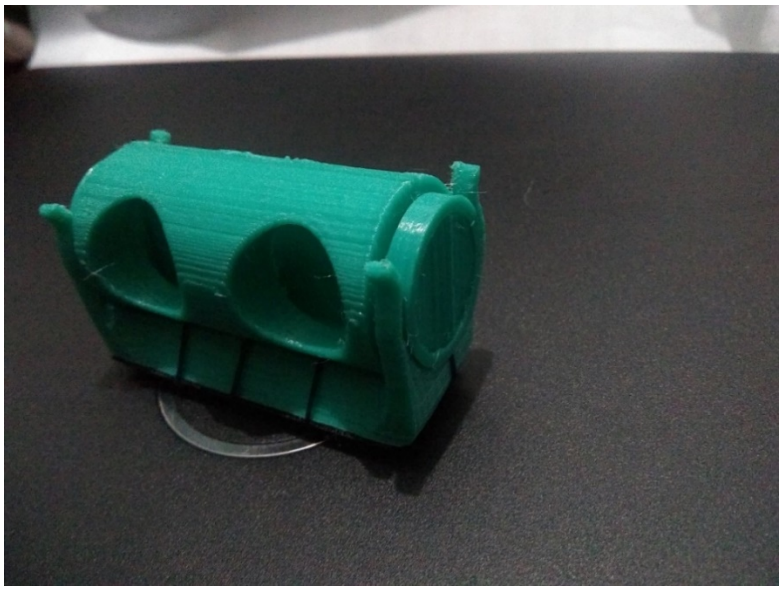

Figura 32. Carcasa Sensor armada.

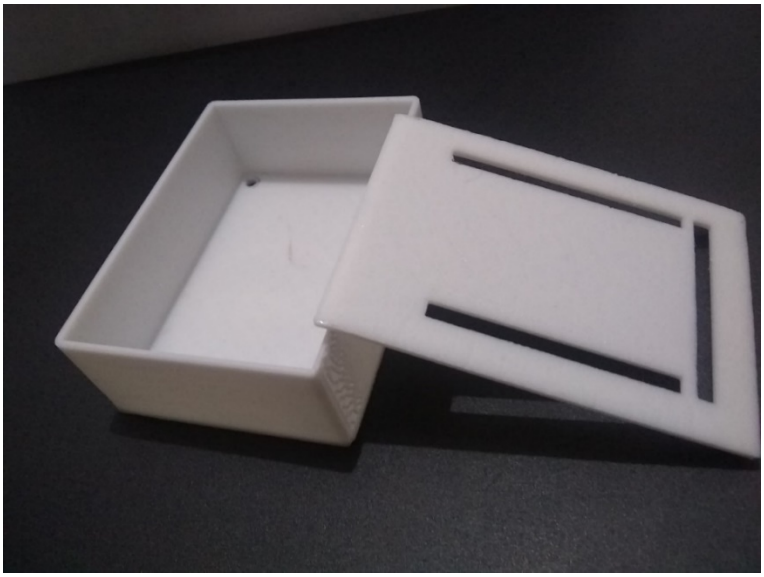

Figura 33. Carcasa Circuito.

Las pruebas experimentales consistieron en cubrir los ojos de algunas personas, evitando que pudieran hacer uso del sentido de la vista, desplazándose de un punto A hacia un punto B. Los lugares seleccionados fueron universidades, plazas comerciales, y casas habitación, como se muestra en las figuras 34 - 39. En algunos casos los participantes desconocían la zona en la que se aplicó el ensayo. Se destaca que las personas emplean *Apps* en *smartphones* bajo la plataforma *Android*, pero no tuvieron conocimiento previo del *wearable* y la *App* que alerta sobre los obstáculos.

Figura 34. Prueba 1.

Figura 35. Prueba 2.

Figura 36. Prueba 3.

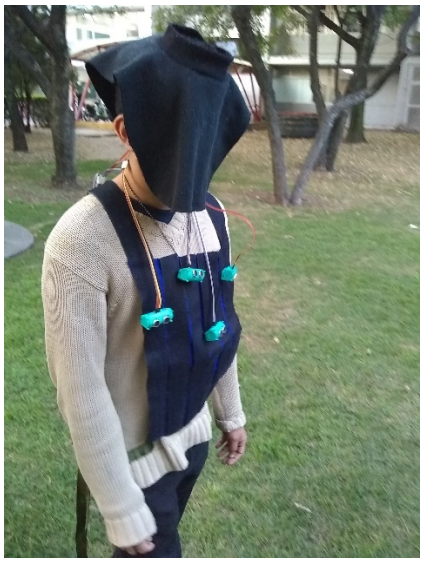

Figura 37. Prueba 4.

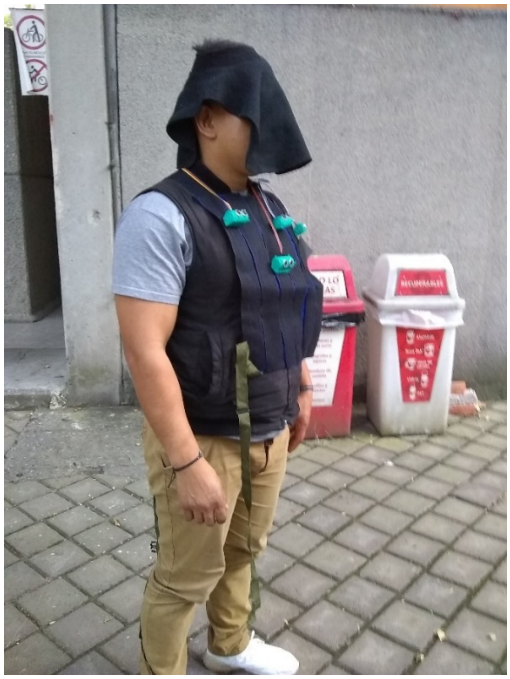

Figura 38. Prueba 5.

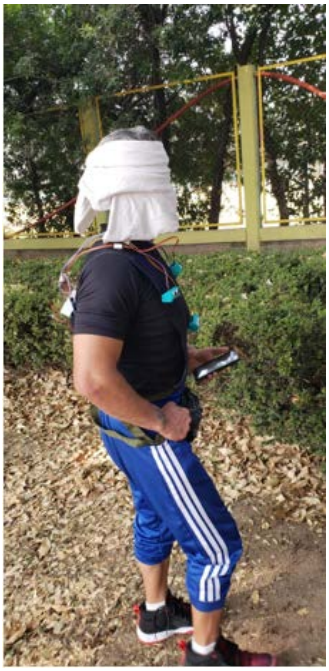

Figura 39. Prueba 6.

7. Resultados

En desplazamientos con obstáculos arriba de la cadera o superiores, la movilidad de los participantes utilizando el sistema era un poco lenta, aunque segura. Los sensores laterales principalmente demostraron tener un gran rango en el patrón de radiación por lo que detectaban obstáculos con facilidad y se lograban diferenciar los tonos agudos, graves y medios. Igualmente, la proximidad de los objetos se diferenciaba con la vibración y la frecuencia en el sonido. El tiempo de respuesta del sistema era casi instantáneo, o con algunos milisegundos de retraso, sin embargo, cuando el usuario tenía un andar de moderado a lento, el sistema detectaba obstáculos sin problema alguno.

La mayoría de los usuarios reportó inconvenientes en obstáculos menores a la altura de la rodilla, aun cuando se modificó la dirección en la que apuntaban los sensores, con el fin de abarcar mayor cobertura en la zona baja del cuerpo. Los obstáculos en donde se presentaron estas circunstancias fueron principalmente en pavimento deteriorado con grietas gruesas, raíces de troncos talados, banquetas, entre otros. Además, si hay un hoyo o un escalón el usuario está en un riesgo potencial de sufrir un accidente.

También se detectaron dificultades en el momento en que los usuarios se colocaban el *wearable* con los ojos cubiertos. La cinta *contactel* se pegaba con otras partes de la prenda, esto ocasionaba algunas veces retraso e incomodidad en los usuarios.

Para navegar y abrir la *App* en el dispositivo *Android*, era necesario, en algunas ocasiones, configurar el dispositivo, en otras ocasiones bastó activar al asistente de voz, el cual es lanzado dejando el dedo presionado en el botón principal de menú.

El tiempo utilizando el sistema fue de dos días seguidos, cada uno con aproximadamente 8 horas de trabajo, dando como resultado un estimado de 16 horas sin presentar alguna falla. En las pruebas posteriores se recargo la fuente de energía para evitar posibles fallos.

8. Análisis y discusión de Resultados

Las pruebas del sistema para apoyar a personas con discapacidad visual efectuadas con diferentes usuarios, demostraron que estos pueden desplazarse con "fluidez" sobre superficies planas. Aunque inicialmente los usuarios mostraron un bajo nivel de confiabilidad en el sistema; conforme aumentó el tiempo de uso del sistema, el nivel de confiabilidad creció positivamente; el usuario fue moviéndose naturalmente y fue adaptándose al medio. Se resalta que los usuarios no tuvieron un conocimiento previo del funcionamiento con respecto al *wearable*, únicamente se dieron a conocer los 3 diferentes sonidos emitidos por la *App* para indicar, respectivamente, las direcciones al frente, izquierda y derecha.

Una vez instalada la *App* en el dispositivo *Android*, fue lanzada fácilmente con el asistente de voz. Las notificaciones por comandos de voz fueron útiles e informaron al usuario de los diferentes mensajes de navegación, los cuales proporcionaron seguridad en la comunicación con el sistema. Cuando hubo problemas con el asistente voz, simplemente se configuró el dispositivo *Android* con las herramientas de accesibilidad, como *TalkBack*.

El tiempo de respuesta fue rápido, casi instantáneo o con algunos milisegundos de retraso. Sin embargo, para los obstáculos más amorfos y de superficie reducida, por ejemplo, menor a $2\ cm^2$ la precisión y el tiempo de respuesta fueron inexactos o completamente erróneos. No obstante, algunos sensores mostraron diferente patrón de radiación que otros, por lo que ciertos transductores detectaron objetos con superficies más pequeñas. Los objetos que provocaban esta situación eran escasos, como ramas demasiado delgadas, hilos, cordones, cintas de anuncios en el techo, entre otros. Al ser tan delgados y pequeños estos elementos no ocasionaban dificultad en el andar del usuario.

En las detecciones correspondientes a obstáculos menores a la altura de la rodilla, existieron inconvenientes. El problema se debió a la precisión de los sensores con respecto a la talla del usuario, porque si se ajustaban muy cerca del suelo, se detectaba al mismo como obstáculo, igualmente cuando el usuario se inclinaba ligeramente hacia adelante se perdía la precisión, y si se ajustaban muy lejos del piso, no había respuesta del sistema. Para tratar de resolver este problema, los sensores se colocaron apuntando al suelo (con mayor cercanía al cuerpo de la persona) pero al caminar, algunas veces las rodillas de los participantes eran detectadas como obstáculos.

La elección del arreglo en serie de ENERGIZER NH15-2300, tiene muchas ventajas a otras baterías. Primordialmente cumple con la demanda de corriente durante el tiempo estipulado y es fácil su adquisición.

Para ensamblar cada parte electrónica en el chaleco, existieron algunas dificultades. Dada la naturaleza del wearable, se diseñó un chaleco para la experimentación; el chaleco está cubierto de cinta *contactel* para mover con facilidad los sensores durante las pruebas; sin embargo, al empezar a vestir el *wearable*, en ocasiones la cinta *contactel* se adhería entre sí. Esto no es grave, pues se trató de un prototipo; la prenda final podrá mejorarse a partir de los resultados experimentales.

El costo monetario y de energía del del sistema se logró reducir, en comparación a los antecedentes de este proyecto (sección 2), porque el sistema no utiliza motores vibradores de cd y tampoco necesita de tarjetas como myRio o Xbee, de precios elevados para la programación de sistemas embebidos. Además, el costo medio de un *Smartphone* con SO *Android* no excede los USD $328.00. Las estadísticas indican que un alto porcentaje de personas en México cuenta con *Smartphone*, por lo que no habría necesidad de adquirir otro dispositivo, sino únicamente la pieza *wearable*, reduciendo así significativamente el precio del producto.

9. Conclusiones

El sistema permite al usuario detectar frontalmente obstáculos en superficies planas, con alcance de aproximadamente $1.50\ metros$ y un rango de altitud desde la cintura, hasta la barbilla del usuario con discapacidad visual. Es capaz de revelar obstáculos potencialmente peligrosos como anuncios, arboles, muros, muebles, objetos metálicos, entre otros. El sistema puede tener un tiempo de trabajo de 18 horas continuas, aunque en la experimentación se probó durante 16 horas repartidas en dos días, sin presentar ningún problema.

El sistema es útil e intuitivo, la *App* emite diferentes patrones de sonidos y diferentes patrones de vibración, el *wearable* mide la distancia entre el usuario y los obstáculos. No obstante, con base en las pruebas efectuadas, se apreciaron algunas deficiencias. Existen objetos amorfos con superficies menores a $2\ cm^2$que el sistema difícilmente detectará; el riesgo es que el usuario tropiece o caiga. También hay obstáculos que se encuentran en una altitud inferior a la rodilla del usuario que el sistema no detectará. Finalmente, el mayor riesgo se encuentra al bajar las escaleras o en pisos con hoyos.

El costo del sistema se redujo, ya que no se utilizó ningún *PDA* o tarjeta de datos especializados, como en los antecedentes. El consumo de energía es menor, debido a que el *wearable* no contiene motores de vibración, los cuales consumen desde 70 mA a hasta los 110 mA por motor.

El sistema resuelve la movilidad en algunos casos, disminuye el costo económico y ahorra energía. En el corto plazo este sistema puede ser empleado junto con un bastón u otro instrumento mecánico principalmente cuando el usuario está bajando escaleras o cuando camina sobre un piso inestable o con hoyos. En el largo plazo, este sistema se puede extender integrando el API de GPS para ampliar los espacios de uso. También se pueden integrar instrumentos mecánicos o electrónicos para detectar obstáculos en las zonas inferiores a las rodillas. De esta manera, el sistema formado por el *wearable*

y la *App* podrán solventar las limitantes observadas durante la experimentación, y potenciarían la movilidad de las personas con discapacidad visual.

El alcance de los objetivos particulares se describe en la tabla 6, así como el grado de cumplimiento de cada uno.

Tabla 6. Alcance de los Objetivos Particulares.

1.	Dimensionar arreglo de sensores ultrasónicos.	90 %	El sistema funcionaría de mejor manera si se agregarán un par de sensores más, para aumentar rango de detección.
2.	Procesar la distancia a partir de las señales del arreglo de sensores mediante el microcontrolador (PIC16F886).	99 %	Probar el uso de interrupciones para lograr un mayor rendimiento y velocidad.
3.	Diseñar e implementar la aplicación para el dispositivo con OS *Android*.	87 %	Mantener en segundo plano la App con el fin de minimizar el consumo de la energía en el dispositivo *Android*. Una vez en conexión el *wearable* con *App*, si el *wearable* se desconecta por alguna razón, cerrar App.
4.	Realizar la interfaz de comunicación para enviar la distancia de los sensores del dispositivo *wearable* hacia la *App* mediante el protocolo RS-232.	100 %	La prueba fue exitosa en su totalidad.
5.	Probar e integrar los módulos que conforman el sistema wearable para su implementación en un circuito impreso.	97 %	Probar con un mejor diseño de chaleco.

9.1. Perspectivas

Tabla 7. Perspectivas

Problema	Solución
Detección de obstáculos para alta y baja altura.	Agregar 2 sensores más. El primero apuntando a la parte superior del cuerpo y el otro a la inferior, aunado al uso del bastón para obstáculos muy pequeños o escales de bajada.
Mejor diseño en el chaleco	Utilizar un chaleco similar al del personal que labora en transporte/policía para mejorar el tiempo de colocación.
Implementación de GPS y mejorar interacción con el usuario.	1. Utilizar el api GPS de Google para ampliar los lugares de uso. 2. Utilizar el BLE para poder usar la *App* en segundo plano o con el dispositivo *Android* en repaso, de esta manera se aumenta el rendimiento del dispositivo *Android*. 3. Añadir opción para calibrar las distancias del Algoritmo de Alerta de Obstáculo. 4. Integrar un *buzzer* al *wearable*, el cual pitaría para avisar al usuario sobre el encendido del dispositivo.

10. Referencias bibliográficas

[1] S. Cardin, D. Thalmann, y F. Vexo, " A wearable System for Mobility Improvement of Visually Impaired ",The Visual Computer, vol. 23, pp. 109-118, febrero 2007.
Disponible:https://link.springer.com/article/10.1007/s00371-006-0032-4
Último acceso: noviembre del 2019

[2] H. M. Grijalva, y C. D. Mejía, " Cinturón y Manillas Vibradores Ultrasónicos para No Videntes ", tesis licenciatura, FICA, Universidad Técnica del Norte, Ibarra, Ecuador,2016.
Disponible: http://repositorio.utn.edu.ec/bitstream/123456789/5612/3/ARTICULO.pdf
Último acceso: noviembre del 2019

[3] S. González y M. M. Sidrón, " Estudio prospectivo en España: la tecnología wearable en el ámbito empresarial. Posibilidades como herramienta de comunicación ", Icono 14, volumen 15 , pp. 220-243, Cádiz, España , 2017.

[4] C. Rodríguez, " Salud Móvil: Nuevos horizontes para la promoción de la salud ", Artículo de Divulgacion científica, Funsalud, México, 2018.

[5] J. T. Gironés, " El gran libro de Android ", Segunda edición, Alfaomega Grupo Editor, México.

[6] E. Palacios, F. Remiro y L. López, " Microcontrolador PIC 16F84. Desarrollo de proyectos ", Primera edición, Alfaomega Grupo Editor, México, agosto, 2004.

[7] L. G. Corona, G. E. Abarca y J. Mares, " Sensores y actuadores ", Primera edición, Grupo Editorial Patria, Colonia San Juan Tlihuaca, Azcapotzalco, México, CD México, 2014.

[8] Microchip Technology Inc., " PIC16F87/88 ", Microchip Technology Inc., Datasheet Microcontrolador, 2013.
Disponible: http://ww1.microchip.com/downloads/en/devicedoc/30487d.pdf
Último acceso: noviembre del 2019

[9] Okystar Tech Co, " Sensor de distancia US-100Ultrasonis ", Okystar Tech Co., Descripción técnica US-100 Ultrasonic Sensor, Shenzhen, China, 2018.
Disponible: https: https://hardtofind.com.mx/pdfs/textos/O/OKY3262.PDF
Último acceso: noviembre del 2019

[10] G. Stettler, " EUSART PIC ", Manual técnica EUSART PIC.
Disponible: https://www.academia.edu/25237495/EUSART_PIC
Último acceso: noviembre del 2019

[11] Okystar Tech Co, " Wireless Cheap *Bluetooth* Module *Bluetooth* Transeiver RF Module HC-06 *Bluetooth* ", Okystar Tech Co., Descripción técnica Module *Bluetooth* , Shenzhen, China, 2018.
Disponible: https://okystar.com/product-item/rf-module-module-*Bluetooth*-hc-06-oky3373/
Último acceso: noviembre del 2019

[12] Android Inc., " Configuración *Bluetooth* ", Manual de configuración, Android Inc., abril 25, 2018.
Disponible: https://developer.android.com/
Último acceso: noviembre del 2019

[13] Energizer Inc., " ENERGIZER NH15-2300 (HR6) ", Datasheet Battery, Energizer Inc., abril 25, 2018.
Disponible: https://data.energizer.com/PDFs/nh15-2300.pdf
Último acceso: noviembre del 2019

Apéndice A. Wearable (entregables)

Código en ensamblador ocupado para la programación del microcontrolador

```
03-dic.-19 11:54

               ROM used: 1242 words (15%)
                         Largest free fragment is 2048
               RAM used: 41 (11%) at main() level
                         63 (17%) worst case
               Stack:    2 locations


0000:  MOVLW  02
0001:  MOVWF  0A
0002:  GOTO   2A0
0003:  NOP
00EB:  MOVLW  40
00EC:  MOVWF  04
00ED:  BCF    03.7
00EE:  MOVF   00,W
00EF:  BTFSC  03.2
00F0:  GOTO   0FF
00F1:  MOVLW  01
00F2:  MOVWF  78
00F3:  CLRF   77
00F4:  DECFSZ 77,F
00F5:  GOTO   0F4
00F6:  DECFSZ 78,F
00F7:  GOTO   0F3
00F8:  MOVLW  4A
00F9:  MOVWF  77
00FA:  DECFSZ 77,F
00FB:  GOTO   0FA
00FC:  GOTO   0FD
00FD:  DECFSZ 00,F
00FE:  GOTO   0F1
00FF:  RETURN
02A0:  CLRF   04
02A1:  BCF    03.7
02A2:  MOVLW  1F
02A3:  ANDWF  03,F
02A4:  BSF    03.5
02A5:  BSF    03.6
02A6:  BCF    07.3
02A7:  MOVLW  19
02A8:  BCF    03.6
02A9:  MOVWF  19
02AA:  MOVLW  A6
02AB:  MOVWF  18
02AC:  MOVLW  90
02AD:  BCF    03.5
02AE:  MOVWF  18
02AF:  BSF    03.5
02B0:  BSF    03.6
02B1:  MOVF   09,W
02B2:  ANDLW  C0
02B3:  MOVWF  09
02B4:  BCF    03.6
02B5:  BCF    1F.4
02B6:  BCF    1F.5
02B7:  MOVLW  00
02B8:  BSF    03.6
02B9:  MOVWF  08
```

```
02BA:  BCF    03.5
02BB:  CLRF   07
02BC:  CLRF   08
02BD:  CLRF   09
02BE:  MOVLW  05
02BF:  BCF    03.6
02C0:  MOVWF  10
02C1:  BSF    03.5
02C2:  BCF    06.0
02C3:  BCF    03.5
02C4:  BSF    06.0
02C5:  MOVLW  03
02C6:  MOVWF  77
02C7:  DECFSZ 77,F
02C8:  GOTO   2C7
02C9:  BSF    03.5
02CA:  BCF    06.0
02CB:  BCF    03.5
02CC:  BCF    06.0
02CD:  BSF    03.5
02CE:  BSF    06.1
02CF:  BCF    03.5
02D0:  BTFSS  06.1
02D1:  GOTO   2CD
02D2:  CLRF   0F
02D3:  CLRF   0E
02D4:  BSF    03.5
02D5:  BSF    06.1
02D6:  BCF    03.5
02D7:  BTFSS  06.1
02D8:  GOTO   2F1
02D9:  MOVF   0F,W
02DA:  MOVWF  7A
02DB:  MOVF   0E,W
02DC:  MOVWF  77
02DD:  MOVF   0F,W
02DE:  SUBWF  7A,W
02DF:  BTFSS  03.2
02E0:  GOTO   2D9
02E1:  MOVF   77,W
02E2:  MOVWF  40
02E3:  MOVF   7A,W
02E4:  MOVWF  41
02E5:  MOVF   41,W
02E6:  SUBLW  20
02E7:  BTFSC  03.0
02E8:  GOTO   2F0
02E9:  XORLW  FF
02EA:  BTFSS  03.2
02EB:  GOTO   2EF
02EC:  MOVF   40,W
02ED:  SUBLW  97
02EE:  BTFSS  03.0
02EF:  GOTO   2F1
02F0:  GOTO   2D4
02F1:  MOVF   0F,W
02F2:  MOVWF  7A
02F3:  MOVF   0E,W
02F4:  MOVWF  77
02F5:  MOVF   0F,W
02F6:  SUBWF  7A,W
02F7:  BTFSS  03.2
02F8:  GOTO   2F1
02F9:  MOVF   77,W
02FA:  MOVWF  40
```

```
02FB:  MOVF   7A,W
02FC:  MOVWF  41
02FD:  CALL   004
02FE:  MOVF   7A,W
02FF:  MOVWF  27
0300:  MOVF   79,W
0301:  MOVWF  26
0302:  MOVF   78,W
0303:  MOVWF  25
0304:  MOVF   77,W
0305:  MOVWF  24
0306:  MOVF   27,W
0307:  MOVWF  47
0308:  MOVF   26,W
0309:  MOVWF  46
030A:  MOVF   25,W
030B:  MOVWF  45
030C:  MOVF   24,W
030D:  MOVWF  44
030E:  CLRF   4B
030F:  CLRF   4A
0310:  CLRF   49
0311:  MOVLW  80
0312:  MOVWF  48
0313:  CALL   021
0314:  MOVF   77,W
0315:  MOVWF  40
0316:  MOVF   78,W
0317:  MOVWF  41
0318:  MOVF   79,W
0319:  MOVWF  42
031A:  MOVF   7A,W
031B:  MOVWF  43
031C:  MOVWF  47
031D:  MOVF   79,W
031E:  MOVWF  46
031F:  MOVF   78,W
0320:  MOVWF  45
0321:  MOVF   77,W
0322:  MOVWF  44
0323:  MOVLW  33
0324:  MOVWF  4B
0325:  MOVWF  4A
0326:  MOVLW  69
0327:  MOVWF  49
0328:  MOVLW  83
0329:  MOVWF  48
032A:  CALL   021
032B:  MOVF   7A,W
032C:  MOVWF  23
032D:  MOVF   79,W
032E:  MOVWF  22
032F:  MOVF   78,W
0330:  MOVWF  21
0331:  MOVF   77,W
0332:  MOVWF  20
0333:  MOVLW  14
0334:  MOVWF  40
0335:  CALL   0EB
0336:  BSF    03.5
0337:  BCF    06.2
0338:  BCF    03.5
0339:  BSF    06.2
033A:  MOVLW  03
033B:  MOVWF  77
```

```
033C:  DECFSZ 77,F
033D:  GOTO   33C
033E:  BSF    03.5
033F:  BCF    06.2
0340:  BCF    03.5
0341:  BCF    06.2
0342:  BSF    03.5
0343:  BSF    06.3
0344:  BCF    03.5
0345:  BTFSS  06.3
0346:  GOTO   342
0347:  CLRF   0F
0348:  CLRF   0E
0349:  BSF    03.5
034A:  BSF    06.3
034B:  BCF    03.5
034C:  BTFSS  06.3
034D:  GOTO   366
034E:  MOVF   0F,W
034F:  MOVWF  7A
0350:  MOVF   0E,W
0351:  MOVWF  77
0352:  MOVF   0F,W
0353:  SUBWF  7A,W
0354:  BTFSS  03.2
0355:  GOTO   34E
0356:  MOVF   77,W
0357:  MOVWF  40
0358:  MOVF   7A,W
0359:  MOVWF  41
035A:  MOVF   41,W
035B:  SUBLW  20
035C:  BTFSC  03.0
035D:  GOTO   365
035E:  XORLW  FF
035F:  BTFSS  03.2
0360:  GOTO   364
0361:  MOVF   40,W
0362:  SUBLW  97
0363:  BTFSS  03.0
0364:  GOTO   366
0365:  GOTO   349
0366:  MOVF   0F,W
0367:  MOVWF  7A
0368:  MOVF   0E,W
0369:  MOVWF  77
036A:  MOVF   0F,W
036B:  SUBWF  7A,W
036C:  BTFSS  03.2
036D:  GOTO   366
036E:  MOVF   77,W
036F:  MOVWF  40
0370:  MOVF   7A,W
0371:  MOVWF  41
0372:  CALL   004
0373:  MOVF   7A,W
0374:  MOVWF  2F
0375:  MOVF   79,W
0376:  MOVWF  2E
0377:  MOVF   78,W
0378:  MOVWF  2D
0379:  MOVF   77,W
037A:  MOVWF  2C
037B:  MOVF   2F,W
037C:  MOVWF  47
```

```
037D:  MOVF   2E,W
037E:  MOVWF  46
037F:  MOVF   2D,W
0380:  MOVWF  45
0381:  MOVF   2C,W
0382:  MOVWF  44
0383:  CLRF   4B
0384:  CLRF   4A
0385:  CLRF   49
0386:  MOVLW  80
0387:  MOVWF  48
0388:  CALL   021
0389:  MOVF   77,W
038A:  MOVWF  40
038B:  MOVF   78,W
038C:  MOVWF  41
038D:  MOVF   79,W
038E:  MOVWF  42
038F:  MOVF   7A,W
0390:  MOVWF  43
0391:  MOVWF  47
0392:  MOVF   79,W
0393:  MOVWF  46
0394:  MOVF   78,W
0395:  MOVWF  45
0396:  MOVF   77,W
0397:  MOVWF  44
0398:  MOVLW  33
0399:  MOVWF  4B
039A:  MOVWF  4A
039B:  MOVLW  69
039C:  MOVWF  49
039D:  MOVLW  83
039E:  MOVWF  48
039F:  CALL   021
03A0:  MOVF   7A,W
03A1:  MOVWF  2B
03A2:  MOVF   79,W
03A3:  MOVWF  2A
03A4:  MOVF   78,W
03A5:  MOVWF  29
03A6:  MOVF   77,W
03A7:  MOVWF  28
03A8:  MOVLW  14
03A9:  MOVWF  40
03AA:  CALL   0EB
03AB:  BSF    03.5
03AC:  BCF    06.4
03AD:  BCF    03.5
03AE:  BSF    06.4
03AF:  MOVLW  03
03B0:  MOVWF  77
03B1:  DECFSZ 77,F
03B2:  GOTO   3B1
03B3:  BSF    03.5
03B4:  BCF    06.4
03B5:  BCF    03.5
03B6:  BCF    06.4
03B7:  BSF    03.5
03B8:  BSF    06.5
03B9:  BCF    03.5
03BA:  BTFSS  06.5
03BB:  GOTO   3B7
03BC:  CLRF   0F
03BD:  CLRF   0E
```

```
03BE:  BSF    03.5
03BF:  BSF    06.5
03C0:  BCF    03.5
03C1:  BTFSS  06.5
03C2:  GOTO   3DB
03C3:  MOVF   0F,W
03C4:  MOVWF  7A
03C5:  MOVF   0E,W
03C6:  MOVWF  77
03C7:  MOVF   0F,W
03C8:  SUBWF  7A,W
03C9:  BTFSS  03.2
03CA:  GOTO   3C3
03CB:  MOVF   77,W
03CC:  MOVWF  40
03CD:  MOVF   7A,W
03CE:  MOVWF  41
03CF:  MOVF   41,W
03D0:  SUBLW  20
03D1:  BTFSC  03.0
03D2:  GOTO   3DA
03D3:  XORLW  FF
03D4:  BTFSS  03.2
03D5:  GOTO   3D9
03D6:  MOVF   40,W
03D7:  SUBLW  97
03D8:  BTFSS  03.0
03D9:  GOTO   3DB
03DA:  GOTO   3BE
03DB:  MOVF   0F,W
03DC:  MOVWF  7A
03DD:  MOVF   0E,W
03DE:  MOVWF  77
03DF:  MOVF   0F,W
03E0:  SUBWF  7A,W
03E1:  BTFSS  03.2
03E2:  GOTO   3DB
03E3:  MOVF   77,W
03E4:  MOVWF  40
03E5:  MOVF   7A,W
03E6:  MOVWF  41
03E7:  CALL   004
03E8:  MOVF   7A,W
03E9:  MOVWF  37
03EA:  MOVF   79,W
03EB:  MOVWF  36
03EC:  MOVF   78,W
03ED:  MOVWF  35
03EE:  MOVF   77,W
03EF:  MOVWF  34
03F0:  MOVF   37,W
03F1:  MOVWF  47
03F2:  MOVF   36,W
03F3:  MOVWF  46
03F4:  MOVF   35,W
03F5:  MOVWF  45
03F6:  MOVF   34,W
03F7:  MOVWF  44
03F8:  CLRF   4B
03F9:  CLRF   4A
03FA:  CLRF   49
03FB:  MOVLW  80
03FC:  MOVWF  48
03FD:  CALL   021
03FE:  MOVF   77,W
```

```
03FF:  MOVWF  40
0400:  MOVF   78,W
0401:  MOVWF  41
0402:  MOVF   79,W
0403:  MOVWF  42
0404:  MOVF   7A,W
0405:  MOVWF  43
0406:  MOVWF  47
0407:  MOVF   79,W
0408:  MOVWF  46
0409:  MOVF   78,W
040A:  MOVWF  45
040B:  MOVF   77,W
040C:  MOVWF  44
040D:  MOVLW  33
040E:  MOVWF  4B
040F:  MOVWF  4A
0410:  MOVLW  69
0411:  MOVWF  49
0412:  MOVLW  83
0413:  MOVWF  48
0414:  CALL   021
0415:  MOVF   7A,W
0416:  MOVWF  33
0417:  MOVF   79,W
0418:  MOVWF  32
0419:  MOVF   78,W
041A:  MOVWF  31
041B:  MOVF   77,W
041C:  MOVWF  30
041D:  MOVLW  14
041E:  MOVWF  40
041F:  CALL   0EB
0420:  BSF    03.5
0421:  BCF    06.6
0422:  BCF    03.5
0423:  BSF    06.6
0424:  MOVLW  03
0425:  MOVWF  77
0426:  DECFSZ 77,F
0427:  GOTO   426
0428:  BSF    03.5
0429:  BCF    06.6
042A:  BCF    03.5
042B:  BCF    06.6
042C:  BSF    03.5
042D:  BSF    06.7
042E:  BCF    03.5
042F:  BTFSS  06.7
0430:  GOTO   42C
0431:  CLRF   0F
0432:  CLRF   0E
0433:  BSF    03.5
0434:  BSF    06.7
0435:  BCF    03.5
0436:  BTFSS  06.7
0437:  GOTO   450
0438:  MOVF   0F,W
0439:  MOVWF  7A
043A:  MOVF   0E,W
043B:  MOVWF  77
043C:  MOVF   0F,W
043D:  SUBWF  7A,W
043E:  BTFSS  03.2
043F:  GOTO   438
```

```
0440:  MOVF   77,W
0441:  MOVWF  40
0442:  MOVF   7A,W
0443:  MOVWF  41
0444:  MOVF   41,W
0445:  SUBLW  20
0446:  BTFSC  03.0
0447:  GOTO   44F
0448:  XORLW  FF
0449:  BTFSS  03.2
044A:  GOTO   44E
044B:  MOVF   40,W
044C:  SUBLW  97
044D:  BTFSS  03.0
044E:  GOTO   450
044F:  GOTO   433
0450:  MOVF   0F,W
0451:  MOVWF  7A
0452:  MOVF   0E,W
0453:  MOVWF  77
0454:  MOVF   0F,W
0455:  SUBWF  7A,W
0456:  BTFSS  03.2
0457:  GOTO   450
0458:  MOVF   77,W
0459:  MOVWF  40
045A:  MOVF   7A,W
045B:  MOVWF  41
045C:  CALL   004
045D:  MOVF   7A,W
045E:  MOVWF  3F
045F:  MOVF   79,W
0460:  MOVWF  3E
0461:  MOVF   78,W
0462:  MOVWF  3D
0463:  MOVF   77,W
0464:  MOVWF  3C
0465:  MOVF   3F,W
0466:  MOVWF  47
0467:  MOVF   3E,W
0468:  MOVWF  46
0469:  MOVF   3D,W
046A:  MOVWF  45
046B:  MOVF   3C,W
046C:  MOVWF  44
046D:  CLRF   4B
046E:  CLRF   4A
046F:  CLRF   49
0470:  MOVLW  80
0471:  MOVWF  48
0472:  CALL   021
0473:  MOVF   77,W
0474:  MOVWF  40
0475:  MOVF   78,W
0476:  MOVWF  41
0477:  MOVF   79,W
0478:  MOVWF  42
0479:  MOVF   7A,W
047A:  MOVWF  43
047B:  MOVWF  47
047C:  MOVF   79,W
047D:  MOVWF  46
047E:  MOVF   78,W
047F:  MOVWF  45
0480:  MOVF   77,W
```

```
0481:  MOVWF  44
0482:  MOVLW  33
0483:  MOVWF  4B
0484:  MOVWF  4A
0485:  MOVLW  69
0486:  MOVWF  49
0487:  MOVLW  83
0488:  MOVWF  48
0489:  CALL   021
048A:  MOVF   7A,W
048B:  MOVWF  3B
048C:  MOVF   79,W
048D:  MOVWF  3A
048E:  MOVF   78,W
048F:  MOVWF  39
0490:  MOVF   77,W
0491:  MOVWF  38
0492:  MOVLW  14
0493:  MOVWF  40
0494:  CALL   0EB
0495:  MOVLW  89
0496:  MOVWF  04
0497:  MOVF   23,W
0498:  MOVWF  43
0499:  MOVF   22,W
049A:  MOVWF  42
049B:  MOVF   21,W
049C:  MOVWF  41
049D:  MOVF   20,W
049E:  MOVWF  40
049F:  CLRF   44
04A0:  CALL   142
04A1:  MOVLW  0A
04A2:  BTFSS  0C.4
04A3:  GOTO   4A2
04A4:  MOVWF  19
04A5:  MOVLW  89
04A6:  MOVWF  04
04A7:  MOVF   2B,W
04A8:  MOVWF  43
04A9:  MOVF   2A,W
04AA:  MOVWF  42
04AB:  MOVF   29,W
04AC:  MOVWF  41
04AD:  MOVF   28,W
04AE:  MOVWF  40
04AF:  CLRF   44
04B0:  CALL   142
04B1:  MOVLW  0A
04B2:  BTFSS  0C.4
04B3:  GOTO   4B2
04B4:  MOVWF  19
04B5:  MOVLW  89
04B6:  MOVWF  04
04B7:  MOVF   33,W
04B8:  MOVWF  43
04B9:  MOVF   32,W
04BA:  MOVWF  42
04BB:  MOVF   31,W
04BC:  MOVWF  41
04BD:  MOVF   30,W
04BE:  MOVWF  40
04BF:  CLRF   44
04C0:  CALL   142
04C1:  MOVLW  0A
```

```
04C2:  BTFSS  0C.4
04C3:  GOTO   4C2
04C4:  MOVWF  19
04C5:  MOVLW  89
04C6:  MOVWF  04
04C7:  MOVF   3B,W
04C8:  MOVWF  43
04C9:  MOVF   3A,W
04CA:  MOVWF  42
04CB:  MOVF   39,W
04CC:  MOVWF  41
04CD:  MOVF   38,W
04CE:  MOVWF  40
04CF:  CLRF   44
04D0:  CALL   142
04D1:  MOVLW  0D
04D2:  BTFSS  0C.4
04D3:  GOTO   4D2
04D4:  MOVWF  19
04D5:  MOVLW  02
04D6:  MOVWF  40
04D7:  CALL   0EB
04D8:  GOTO   2C1
04D9:  SLEEP

Configuration Fuses:
   Word  1: 2FF9   XT WDT NOPUT MCLR NOPROTECT NOCPD BROWNOUT IESO FCMEN NOLVP NODEBUG

   Word  2: 3FFF   NOWRT BORV40
```

Plano del Circuito Impreso

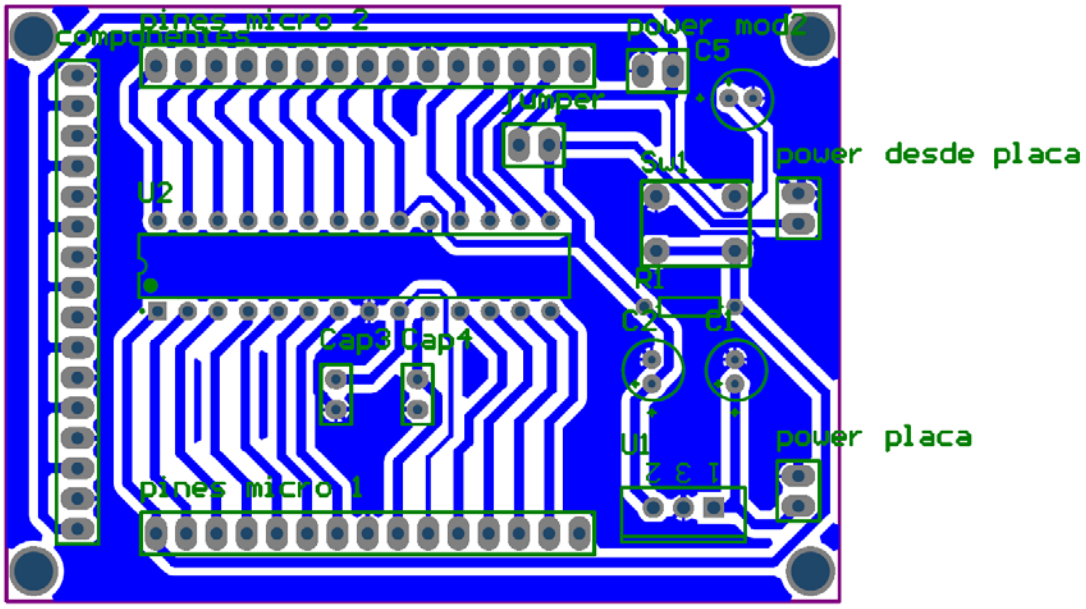

Figura 40. Plano del Circuito Impreso.

Wearable

Figura 41. *Wearable.*

Apéndice B. App (entregables)

Código Fuente Java

LogoUam

```
import android.app.SearchManager;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.Toast;

import com.example.adrian.ptadrianbuendia.Hilos.HiloTempoComandVoice;
import com.example.adrian.ptadrianbuendia.Hilos.HiloTemporizador;
import com.example.adrian.ptadrianbuendia.ParaVoz.TextoVoz;
import com.example.adrian.ptadrianbuendia.R;

import com.example.adrian.ptadrianbuendia.Servicio.ServicioActivo;


/**
 Primera Pantalla; muestra un splashScreen de 8 segundos de duración con el Logo Uam Azcapotzalco.
 Informa al usuario invidente, por medio de la voz Artificial Android, si está activada la conexión
 Bluetooth en el dispositivo con OS Android,y si es que el dispositivo soporta la conexión, de lo
 contrario cierra la App. Si la conexión está apagada, por reconocimiento de voz se activa.
 *
 */
public class LogoUam extends AppCompatActivity {

    private TextoVoz voz;
    private ImageView imagenLogoUam;
    private HiloTemporizador hilo;
    private BluetoothAdapter btAdapter = null ;

    //private static final int RECOGNIZE_SPEECH_ACTIVITY = 1;
    private Context contextThis =this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_logo_uam);

        btAdapter=BluetoothAdapter.getDefaultAdapter();
        imagenLogoUam = (ImageView) findViewById(R.id.logoUam);
        voz = new TextoVoz(this);

        /**
         *  Comprueba que el dispositivo Bluetooth está disponible
         *  y esté activado, de lo contrario se cerrará la APP
         *
         */
        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "El dispositivo no soporta bluetooth",
Toast.LENGTH_LONG).show();
            voz.setMensaje("El dispositivo no soporta bluetooth");
            voz.speakOut();

            /**
             * Hilo Temporizdor para terminar La Activity
             */
            hilo=new HiloTemporizador(this,8000);
            hilo.start();

        } else {

            if (btAdapter.isEnabled()) {
                voz.setMensaje("Proyecto Terminal del alumno Adrian Buendia");
                voz.speakOut();

                hilo = new HiloTemporizador(this,5000,MiLogo.class);
                hilo.start();
            } else{
```

```
                voz.setMensaje("El Bluetooth no está encendido. Para habilitarlo
                      repita el comando siguiente. Activar Bluetooth ");
                voz.speakOut();

                String query="";
                Intent intent = new Intent(Intent.ACTION_VOICE_COMMAND);
                intent.putExtra(SearchManager.QUERY, query);

                HiloTempoComandVoice hilo1=new
           HiloTempoComandVoice(this,5000,intent,ServicioActivo.class);

                if (intent.resolveActivity(getPackageManager()) != null) {

           Toast.makeText(getApplicationContext(),"aqui",Toast.LENGTH_LONG).show();
                    //Lanza Servicio
                    hilo1.start();

                }
             }
        }
    }
}
```

HiloTemporizador

```
import android.app.Activity;
import android.content.Intent;

/**
 * Permite cambiar de una activity a otra, en una determinada cantidad de tiempo.
 */
public class HiloTemporizador extends Thread {

    private Activity appActivity;
    private int milisegundosParaCambio;
    private Intent intento;

    public HiloTemporizador(Activity appActivity, int milisegundosParaCambio, Class
classSiguienteActivity){

        this.appActivity=appActivity;
        this.milisegundosParaCambio=milisegundosParaCambio;
        this.intento = new Intent(appActivity, classSiguienteActivity);

    }

    public HiloTemporizador(Activity appActivity, int milisegundosParaCambio){

        this.appActivity=appActivity;
        this.milisegundosParaCambio=milisegundosParaCambio;

    }

    public HiloTemporizador(Activity appActivity, int milisegundosParaCambio, Intent intento){

        this.appActivity=appActivity;
        this.milisegundosParaCambio=milisegundosParaCambio;
        this.intento = intento;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(milisegundosParaCambio);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        appActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(intento!=null){

                    appActivity.getApplicationContext().startActivity(intento);
                    appActivity.finish();
                }
                else
```

```
                appActivity.finish();
            }
        });
    }

    public Intent getIntento() {
        return intento;
    }
}
```

TextoVoz

```
import android.content.Context;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.widget.Toast;
import java.util.Locale;

/**
 * Convertir una cadena de caracteres a voz Artificial de Android.
 */

public class TextoVoz  implements TextToSpeech.OnInitListener {
    private String mensaje;
    private TextToSpeech voz;
    private Context contexto;
                    //   this            ,            this                    ,

    /**
     * Incializar la voz Artificial de Android
     * @param contextThis
     */
    public TextoVoz (Context contextThis  ){ //getBasecontext() --->Afuera de OnCreat
        this.contexto = contextThis;
        voz=new TextToSpeech(contexto,this); //new TextoVoz(Context contextThis,
TextToSpeech.OnInitListener listener)
        mensaje="";
    }

    /**
     * Metodo para comprobar si el dipositivo soporta voz Artifcial Android.
     * Ademas de obtener el idioma actual del dispositivo para poder repodrucir con el acento adecuado
     * @param status
     */
    @Override
    public void onInit(int status) {
        if(status==TextToSpeech.SUCCESS){
            int result= getVoz().setLanguage(Locale.getDefault());
            if(result==TextToSpeech.LANG_NOT_SUPPORTED || result==TextToSpeech.LANG_MISSING_DATA){
                Log.e("voz","Este lenguaje no es soportado");
                                //getBaseContext()
                Toast.makeText(contexto, "Este lenguaje no es soportado", Toast.LENGTH_LONG).show();
            }else{
                speakOut();
            }
        }else{
            Log.e("voz","Inicalizacion del lenguaje fallida");
            Toast.makeText(contexto, "Inicalizacion del lenguaje fallida", Toast.LENGTH_LONG).show();
        }
    }

    public void speakOut(){
        getVoz().speak(getMensaje(), TextToSpeech.QUEUE_FLUSH,null);
    }

    public String getMensaje() {
        return mensaje;
    }

    public void setMensaje(String mensaje) {
        this.mensaje = mensaje;
    }

    public TextToSpeech getVoz() {
        return voz;
    }
}
```

```
}
```

HiloTempoComandVoicejava

```
import android.app.Activity;
import android.content.Intent;

import com.example.adrian.ptadrianbuendia.Servicio.ServicioActivo;

/**
 * Lanza un servicio, lanza una activity y cierra la activity actual en una determinada cantidad de
tiempo.
 */

public class HiloTempoComandVoice extends Thread {
    private Activity appActivity;
    private int milisegundosParaCambio;
    private Intent intento;
    private Class claseServicio;

    /**
     *
     * @param appActivity
     * @param milisegundosParaCambio
     * @param intento
     * @param claseServicio
     */
    public HiloTempoComandVoice(Activity appActivity, int milisegundosParaCambio, Intent intento,Class
claseServicio){

        this.appActivity=appActivity;
        this.milisegundosParaCambio=milisegundosParaCambio;
        this.intento = intento;
        this.claseServicio = claseServicio;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(milisegundosParaCambio);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        appActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(intento!=null){
                    appActivity.startService(new Intent(appActivity, ServicioActivo.class));
                    appActivity.getApplicationContext().startActivity(intento);
                    appActivity.finish();
                }
                else appActivity.finish();
            }
        });
    }

}
```

ServicioActivo

```
import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.os.Process;
import android.widget.Toast;

import com.example.adrian.ptadrianbuendia.Logos.LogoUam;
```

```java
/**
 * Servicio que monitorea en segundo plano cuando el bluetooth se haya habilitado. Una vez ocurrida esta
acción, lanza una activity.
 */
public class ServicioActivo extends Service {

    private BluetoothAdapter btAdapter;

    private Looper serviceLooper;
    private ServiceHandler serviceHandler;

    private final class ServiceHandler extends Handler {

        @Override
        public void handleMessage(Message msg) {

            if(btAdapter.isEnabled()){
                stopSelf(msg.arg1);
            }

        }
    }

    @Override
    public void onCreate() {
        btAdapter= BluetoothAdapter.getDefaultAdapter();
        HandlerThread thread = new HandlerThread("ServiceStartArguments",
                Process.THREAD_PRIORITY_BACKGROUND);
        thread.start();

        // Get the HandlerThread's Looper and use it for our Handler
        serviceLooper = thread.getLooper();
        serviceHandler = new ServiceHandler(serviceLooper);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();

        // For each start request, send a message to start a job and deliver the
        // start ID so we know which request we're stopping when we finish the job

        while( !btAdapter.isEnabled() ){

            Message msg = serviceHandler.obtainMessage();
            msg.arg1 = startId;
            serviceHandler.sendMessage(msg);

        }
        // If we get killed, after returning from here, restart
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        // We don't provide binding, so return null
        return null;
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, "BluetoothActivado OnDestroy", Toast.LENGTH_SHORT).show();

        startActivity(new Intent(this, LogoUam.class));
    }

}
```

MiLogo

```java
import android.bluetooth.BluetoothAdapter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;

import com.example.adrian.ptadrianbuendia.Hilos.HiloTemporizador;
import com.example.adrian.ptadrianbuendia.MainActivity;
import com.example.adrian.ptadrianbuendia.ParaVoz.TextoVoz;
```

```java
import com.example.adrian.ptadrianbuendia.R;


/**
 * Esta clase es un splashScreen la cual durara 6 segundos activa y mostrara el Logo de la APP.
 * Mientras alertara al usuario para que encienda el dispositivo electronico si esq no lo ha hecho,
 * ,de lo contrario se cerrara la app.
 */
public class MiLogo extends AppCompatActivity {

    private TextoVoz voz;
    private ImageView imagenMiLogo;
    private BluetoothAdapter btAdapter = null ;
    private HiloTemporizador hilo = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mi_logo);

        imagenMiLogo = (ImageView) findViewById(R.id.miLogo);
        voz = new TextoVoz(this);

        int contador;
        Bundle paqueteContador= this.getIntent().getExtras();
        hilo = new HiloTemporizador(this,6000,MainActivity.class);

        if(paqueteContador==null){
            voz.setMensaje("Verifique que el dispositivo este encendido y las pilas bien cargadas ");
            voz.speakOut();
            contador=0;
            hilo.getIntento().putExtra("KEYReceptor",contador);
            hilo.start();
        }else{

            contador = paqueteContador.getInt("KEYTransmisor");
            voz.setMensaje("Verifique que el dispositivo este encendido y las pilas bien cargadas.
Intento "+contador);
            voz.speakOut();
            hilo.getIntento().putExtra("KEYReceptor",contador);

            if(contador==3){
                voz.setMensaje("Los 3 intentos han sido agotados. Aplicación finalizada");
                voz.speakOut();
                hilo = new HiloTemporizador(this,5000);
                hilo.start();
            }else hilo.start();
        }


    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        if(voz.getVoz().isSpeaking()){
            voz.getVoz().stop();
        }
        voz.getVoz().shutdown();
    }

}
```

TextViewRunnable

```java
import android.app.Activity;
import android.widget.TextView;

/**
 * La funcion principal de esta clase es la de cambiar en pantalla
 * la distancia de algun sensor.
 *
 * Se ejecutara sobre un Hilo y seteara el componente de texto, en este caso el obejto TextView
 */

public class TextViewRunnable implements Runnable{
    private  Activity appActivity;
```

```
    private TextView textoComponente;
    private String distancia;

    /**
     * Reibira la appActivity como parametro y el componente de texto
     * @param appActivity
     * @param textoComponente
     */
    public TextViewRunnable(Activity appActivity,TextView textoComponente){
        this.textoComponente = textoComponente;
        this.appActivity = appActivity;
    }

    /**
     * Seteara el componente de texto con un nuevo valor
     */
    @Override
    public void run() {

            appActivity.runOnUiThread(new Runnable() {
                @Override
                public void run() {

                    textoComponente.setText(getDistancia());
                }
            });
    }

    /**
     * Regresa la distancia en formato String
     * @return
     */

    public String getDistancia() {
        return distancia;
    }

    /**
     * Setea la distancia
     * @param distancia
     */
    public void setDistancia(String distancia) {
        this.distancia = distancia;
    }
}
```

MainActivity

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
import com.example.adrian.ptadrianbuendia.Hilos.MiHandler;
import com.example.adrian.ptadrianbuendia.Logos.MiLogo;
import com.example.adrian.ptadrianbuendia.ParaBluetooth.ConnectedThread;
import com.example.adrian.ptadrianbuendia.ParaVoz.TextoVoz;

import java.io.IOException;
import java.util.UUID;


/**
 * Esta clase será la pantalla principal de la Aplicacion.
 * En el metodo onCreate se incializaran los objetos
 * En el metodo onResume se Configurar la conexion Bluetooth,
 * con esta accion se empezaran a recibir mensajes en el objeto bluetoothIn.
 *
 */
public class MainActivity extends AppCompatActivity {

    private TextoVoz voz;
    private TextView tramaPrincipal,texto1,texto2,texto3,texto4;

    //----------------------------------------
    private MiHandler bluetoothIn;
```

```
    private BluetoothAdapter btAdapter = null ;
    private BluetoothSocket btSocket = null;
    private ConnectedThread MyConexionBT;
    // Identificador unico de servicio - SPP UUID
    private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    // String para la direccion MAC
    private static String address = null;
    //----------------------------------------



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        /**
         * Incializacion de varios objetos
         */

        voz=new TextoVoz(this);

        tramaPrincipal=(TextView) findViewById(R.id.trama_principal);
        texto1=(TextView) findViewById(R.id.texto1);
        texto2=(TextView) findViewById(R.id.texto2);
        texto3=(TextView) findViewById(R.id.texto3);
        texto4=(TextView) findViewById(R.id.texto4);

        btAdapter = BluetoothAdapter.getDefaultAdapter();
        VerificarEstadoBT();

        bluetoothIn = new MiHandler(tramaPrincipal,this,texto1,texto2,texto3,texto4);

    }


    @Override
    public void onResume()
    {
        super.onResume();
        VerificarEstadoBT();

        address="00:21:13:02:80:AA";
        //Setea la direccion MAC
        BluetoothDevice device = btAdapter.getRemoteDevice(address);

        try
        {
            //crea un conexion de salida segura para el dispositivo
            // usando el servicio UUID
            btSocket = device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        } catch (IOException e) {
            Toast.makeText(getBaseContext(), "La creacción del Socket fallo", Toast.LENGTH_LONG).show();
            Intent intento = new Intent (MainActivity.this, MiLogo.class);
            startActivity( intento );
        }
        // Establece la conexión con el socket Bluetooth.
        try
        {
            btSocket.connect();
        } catch (IOException e) {
            try {
                btSocket.close();
                Bundle paqueteContador= this.getIntent().getExtras();
                int contador = paqueteContador.getInt("KEYReceptor");
                ++contador;
                Toast.makeText(getBaseContext(), "Encienda el dispositivo. Contador-->"+contador,
Toast.LENGTH_LONG).show();
                Intent intentoRetornoLogo = new Intent (MainActivity.this, MiLogo.class);
                intentoRetornoLogo.putExtra("KEYTransmisor",contador);
                startActivity( intentoRetornoLogo );
                finish();

            } catch (IOException e2) { finish(); }
        }
        MyConexionBT = new ConnectedThread(btSocket,bluetoothIn);


        MyConexionBT.start();

    }
```

```
    @Override
    public void onPause()
    {
        super.onPause();
        try
        {
            /**
             * Cuando la aplicacion entra en Pausa el socket se cierra
             */
            btSocket.close();
        } catch (IOException e2) {}

    }

    /**
     * VerificarEstadoBT() comprueba que el dispositivo  Bluetooth está disponible y solicita que se
active si está desactivado
     */
    private void VerificarEstadoBT() {

        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "El dispositivo no soporta bluetooth",
Toast.LENGTH_LONG).show();
        } else {
            if (btAdapter.isEnabled()) {
            } else {
                Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
            }
        }
    }

}
```

ConnectedThread

```
import android.bluetooth.BluetoothSocket;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import com.example.adrian.ptadrianbuendia.Hilos.MiHandler;

/**
 * La clase ConnectedThread es la encargada de crear el evento de conexion sobre un hilo.
 */
public class ConnectedThread extends Thread {

    private final InputStream mmInStream;
    private final OutputStream mmOutStream;
    private MiHandler bluetoothIn;

    public ConnectedThread(BluetoothSocket socket, MiHandler bluetoothIn) {
        InputStream tmpIn = null;
        OutputStream tmpOut = null;
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
        }
        mmInStream = tmpIn;
        mmOutStream = tmpOut;

        this.bluetoothIn = bluetoothIn;

    }

    /**
     * En el metodo run(), el Cliente se mantiene en estado "listener"
     */
    public void run() {
        byte[] buffer = new byte[256];
        int bytes;

        /**
         * Se mantiene siempre en modo "listener" para determinar el ingreso de datos
         */
        while (true) {
```

```
            try {
                bytes = mmInStream.read(buffer);

                /**
                 * Envia los datos obtenidos hacia el manejador de mensajes. En este caso se enviar al
objeto bluetoothIn sobre este Hilo.
                 *
                 * https://developer.android.com/reference/android/os/Message.html#sendToTarget--
                 *
                 * Mientras el constructor Message sea publico, la mejor manera de obtener el mensaje es
llamando a Message.obtain() o Handler.obtainMessage(),
                 */
                bluetoothIn.obtainMessage(bluetoothIn.getHandlerState(), bytes, -1,
readMessage).sendToTarget();

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

}
```

MiHandler

```
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.TextView;

import java.util.StringTokenizer;

/**
 * La clase MiHandler obtiene el mensaje desde el sistema operativo de android,
 * y determina si el mensaje puede contener las repectivas distancias de los sensores.
 * La trama de caracteres recibida en el mensaje puede contener ruido adicionado por distintos factores,
 * lo que impediría recuperar el mensaje orginal.
 */

public class MiHandler extends Handler {
    private StringBuilder DataStringIN;
    private TextView tramaPrincipal;
    private final int handlerState;

    private StringTokenizer tokens;

    private SubProcesos manejadorTexoSonido;
    Bundle datos;

    public MiHandler( TextView tramaPrincipal, Activity appActivity, TextView textDistancia1, TextView
textDistancia2, TextView textDistancia3, TextView textDistancia4){
        this.tramaPrincipal=tramaPrincipal;
        this.DataStringIN = new StringBuilder();
        this.handlerState = 0;


        this.datos = new Bundle();
        this.manejadorTexoSonido = new
SubProcesos(appActivity,textDistancia1,textDistancia2,textDistancia3,textDistancia4);
    }

    /**
     * Al enviar desde el dispositivo electronico el mensaje separa las distancias por saltos de linea,
     * si no hay por lo menos dos sepraciones, este metodo no enviara un nuevo mensaje al objeto
     * manejadorTextoSonido el cual podra procesar las distancias y evitara errores de procesamiento
     * o un procesamiento mas profundo para captar los posibles errores.
     * @param msg
     */
    @Override
    public void handleMessage(android.os.Message msg) {

        if (msg.what == getHandlerState()) {

            String readMessage = (String) msg.obj;
            DataStringIN.append(readMessage);

            int endOfLineIndex = DataStringIN.indexOf("\r"); // <----ultimo caracter e la trama
```

```
            if (endOfLineIndex > 0) {
                String dataInPrint = DataStringIN.substring(0, endOfLineIndex);
                tramaPrincipal.setText("Distancias:\n " + dataInPrint);//<-<- PARTE A MODIFICAR >->->

                tokens=new StringTokenizer(dataInPrint);

                DataStringIN.delete(0, DataStringIN.length());

                /**
                 * TOKENS
                 */
                /**
                 * Son 4 sensores ultrasonicos asi que las distancias son 2 y los tokens deberian ser 2.
                 */
                if ( tokens.countTokens() == 4 ) {
                    Message mensaje = new Message();
                    datos.putString("llave",dataInPrint );
                    //Es necesario crear un nuevo mensaje cada vez y envirlo por el este hilo al otro
hnadler manejadorTexoSonido
                    mensaje.setData(datos);
                    manejadorTexoSonido.sendMessage(mensaje);
                }

            }

        }
    }

    public int getHandlerState() {
        return handlerState;
    }

}
```

SubProcess

```
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.widget.TextView;

import com.example.adrian.ptadrianbuendia.ClasesRunnable.SonidoVibracionRunnable;
import com.example.adrian.ptadrianbuendia.ClasesRunnable.TextViewRunnable;

import java.util.StringTokenizer;

/**
 * Esta clase fue creada para separar el mensaje en tokens y obetener las distancias de los sensores .
 * Con esta accion se podra manipular la vibracion  y el sonido para alertar a la persona ocn
discapacidad.
 *
 * Ademas para realizar la pruebas pertienetes esta clase tambien muestra en en pantalla las distancias
obtenidas.
 */
public class SubProcesos extends Handler {

    private TextViewRunnable distancia1TextView, distancia2TextView, distancia3TextView,
distancia4TextView;
    private int distancia1,distancia2,distancia3,distancia4;
    private String distancia1Cadena,distancia2Cadena,distancia3Cadena,distancia4Cadena;

    private StringTokenizer tokens;
    private Bundle datos;

    private Handler hiloSonidoVibracion;
    private SonidoVibracionRunnable suenaVibra;


    public  SubProcesos ( Activity appActivity, TextView textDistancia1, TextView textDistancia2,
TextView textDistancia3, TextView textDistancia4){

        this.distancia1TextView = new TextViewRunnable(appActivity,textDistancia1);
        this.distancia2TextView = new TextViewRunnable(appActivity,textDistancia2);
        this.distancia3TextView = new TextViewRunnable(appActivity,textDistancia3);
        this.distancia4TextView = new TextViewRunnable(appActivity,textDistancia4);
```

```
        this.hiloSonidoVibracion = new Handler(Looper.getMainLooper());

        this.suenaVibra = new SonidoVibracionRunnable(appActivity);
    }

    /**
     * En este metodo se recuperar el mensaje con las distancias para poder separarlas en tokens y
convertirlas a Enteros.
     * Tambien se mandaran llamar a los metodos de los objetos que muestren la distancia en patalla.
     * Se implementa la alerta con sonido y vibración con el objeto hiloSonidoVibracion y suenaVibra
     * @param msg
     */
    @Override
    public void handleMessage(Message msg) {
        datos = msg.getData();
        tokens = new StringTokenizer(datos.getString("llave"));

            for (int i=1;tokens.hasMoreTokens();i++) {

                switch (i) {

                    case 1:
                                suenaVibra.setDistancia1(asignacionDistancia( distancia1Cadena, distancia1,
distancia1TextView) );
                                hiloSonidoVibracion.post(suenaVibra);
                        break;
                    case 2:
                                suenaVibra.setDistancia2(asignacionDistancia( distancia2Cadena, distancia2,
distancia2TextView) );
                                hiloSonidoVibracion.post(suenaVibra);
                        break;
                    case 3:
                                suenaVibra.setDistancia3(asignacionDistancia( distancia3Cadena, distancia3,
distancia3TextView) );
                                hiloSonidoVibracion.post(suenaVibra);
                        break;
                    case 4:
                                suenaVibra.setDistancia4(asignacionDistancia( distancia4Cadena, distancia4,
distancia4TextView) );
                                hiloSonidoVibracion.post(suenaVibra);;
                        break;
                    default: tokens.nextToken();
                        break;
                }


            }

        }

        public int asignacionDistancia(String distanciaCadena, int distancia, TextViewRunnable
distanciaTextView){
            try {

                distanciaCadena = tokens.nextToken().trim();
                distancia = Integer.parseInt(distanciaCadena);
                distanciaTextView.setDistancia(distanciaCadena);
                post(distanciaTextView);
            } catch (NumberFormatException e) {
                distanciaCadena="Err";
                distancia = 2000;
                distanciaTextView.setDistancia(distanciaCadena);
                post(distanciaTextView);
            }

            return distancia;
        }


}
```

SonidoVibracionRunnable

```
import android.app.Activity;
import android.content.Context;
import android.media.MediaPlayer;
import android.os.Vibrator;

import com.example.adrian.ptadrianbuendia.R;
```

```
/**
 * Esta clase tiene como funcion principal reproducir un sonido y hacer vibrar el dispositivo a la vez.
 * El patron de vibracion y el sonido a reporducir se asignaran dependiendo la distancia dada.
 *
 */

public class SonidoVibracionRunnable implements Runnable{

    private Activity appActivity;
    private MediaPlayer
derechaCorta,derechaMedia,derechaLarga,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,izquierdaMedia,izquierdaLarga;

    private int distancia1;
    private int distancia2;
    private int distancia3;
    private int distancia4;

    private Vibrator vibrar;

    private long[] patronCorto={200,100,200,100,200,100,200};
    private long[] patronMedio={200,300,200,300,200,300,200};
    private long[] patronLargo={300,600,300,600,300,600,300};

    public SonidoVibracionRunnable(Activity appActivity){
        this.appActivity=appActivity;

        this.vibrar=(Vibrator)appActivity.getSystemService(Context.VIBRATOR_SERVICE);

        this.derechaCorta = MediaPlayer.create(appActivity,R.raw.derecha_corta);
        this.derechaCorta.setLooping(true);

        this.derechaMedia = MediaPlayer.create(appActivity,R.raw.derecha_media);
        this.derechaMedia.setLooping(true);

        this.derechaLarga = MediaPlayer.create(appActivity,R.raw.derecha_larga);
        this.derechaLarga.setLooping(true);

        this.enfrenteCorta = MediaPlayer.create(appActivity,R.raw.enfrente_corta);
        this.enfrenteCorta.setLooping(true);

        this.enfrenteMedia = MediaPlayer.create(appActivity,R.raw.enfrente_media);
        this.enfrenteMedia.setLooping(true);

        this.enfrenteLarga = MediaPlayer.create(appActivity,R.raw.enfrente_larga);
        this.enfrenteLarga.setLooping(true);

        this.izquierdaCorta = MediaPlayer.create(appActivity,R.raw.izquierda_corta);
        this.izquierdaCorta.setLooping(true);

        this.izquierdaMedia = MediaPlayer.create(appActivity,R.raw.izquierda_media);
        this.izquierdaMedia.setLooping(true);

        this.izquierdaLarga = MediaPlayer.create(appActivity,R.raw.izquierda_larga);
        this.izquierdaLarga.setLooping(true);

        this.setDistancia1(2000);
        this.setDistancia2(2000);
        this.setDistancia3(2000);
        this.setDistancia4(2000);


    }

    /**
     * Algoritmo para seleccionar una cancion y vibrar el smartphone dependiendo la distancia.
     */
    @Override
    public void run() {

        if(distancia1 >= 120 && distancia2 >= 120  && ( distancia3 >= 120 && distancia4 >= 120 )){ // Primera Tabla
            apagarSonido();
            vibrar.cancel();
        }else{
            if( (distancia1 >= 60 && distancia1 < 120)  && distancia2 >= 120  && ( distancia3 >= 120 || distancia4 >= 120 )){ // Segunda Tabla

sonarUnaAlaVez(derechaLarga,derechaCorta,derechaMedia,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,izquierdaLarga,izquierdaMedia);
                vibrar.vibrate(patronLargo,0);
            }else {
                if( (distancia1 >= 30 && distancia1 < 60)  && distancia2 >= 60  && ( distancia3 >= 60 || distancia4 >= 60 )){

sonarUnaAlaVez(derechaMedia,derechaLarga,derechaCorta,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,izquierdaLarga,izquierdaMedia);
                    vibrar.vibrate(patronMedio,0);
                }else{
                    if(  distancia1 < 30  && distancia2 >= 30  && ( distancia3 >= 30 || distancia4 >= 30 )){

sonarUnaAlaVez(derechaCorta,derechaLarga,derechaMedia,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,izquierdaLarga,izquierdaMedia);
                        vibrar.vibrate(patronCorto,0);
                    }else{
                        if( distancia1 >= 120   && (distancia2 >= 60 && distancia2 < 120) && ( distancia3 >= 120 || distancia4 >= 120 )){

sonarUnaAlaVez(izquierdaLarga,derechaCorta,derechaMedia,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,derechaLarga,izquierdaMedia);
                            vibrar.vibrate(patronLargo,0);
                        }else{
                            if( distancia1 >= 60   && (distancia2 >= 30 && distancia2 < 60) && ( distancia3 >= 60 || distancia4 >= 60 )){

sonarUnaAlaVez(izquierdaMedia,derechaCorta,derechaMedia,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaCorta,derechaLarga,izquierdaLarga);
                                vibrar.vibrate(patronMedio,0);
                            }else{
                                if( distancia1 >= 30   &&  distancia2 < 30 && ( distancia3 >= 30 || distancia4 >= 30 )){

sonarUnaAlaVez(izquierdaCorta,derechaCorta,derechaMedia,enfrenteCorta,enfrenteMedia,enfrenteLarga,izquierdaMedia,derechaLarga,izquierdaLarga);
                                    vibrar.vibrate(patronCorto,0);
                                }else {
                                    if( distancia1 >= 120   &&  distancia2 >= 120 && ( (distancia3 >= 60 && distancia3 < 120) || (distancia4 >= 60
&& distancia4 < 120) )){

sonarUnaAlaVez(enfrenteLarga,derechaCorta,derechaMedia,enfrenteCorta,enfrenteMedia,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                        vibrar.vibrate(patronLargo,0);
                                    }else{
                                        if( distancia1 >= 60   &&  distancia2 >= 60 && ( (distancia3 >= 30 && distancia3 < 60) || (distancia4 >= 30
&& distancia4 < 60) )){

sonarUnaAlaVez(enfrenteMedia,derechaCorta,derechaMedia,enfrenteCorta,enfrenteLarga,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
```

```
                                        vibrar.vibrate(patronMedio,0);
                                    }else{
                                        if( distancia1 >= 30   &&  distancia2 >= 30 && ( distancia3 < 30 || distancia4 < 30  ) ){

sonarUnaAlaVez(enfrenteCorta,derechaCorta,derechaMedia,enfrenteMedia,enfrenteLarga,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                            vibrar.vibrate(patronCorto,0);
                                        }else{
                                            if( (distancia1 >= 60 && distancia1 < 120)   &&  distancia2 >= 120 && ( (distancia3 >= 60 &&
distancia3 < 120) || (distancia4 >= 60 && distancia4 < 120)  ) ){ // Tercera Tabla

sonarUnaAlaVez(enfrenteLarga,derechaCorta,derechaMedia,enfrenteMedia,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                vibrar.vibrate(patronLargo,0);
                                            }else{
                                                if( (distancia1 >= 30 && distancia1 < 60)   &&  distancia2 >= 60 && ( (distancia3 >= 30 &&
distancia3 < 60) || (distancia4 >= 30 && distancia4 < 60)  ) ){

sonarUnaAlaVez(enfrenteMedia,derechaCorta,derechaMedia,enfrenteLarga,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                    vibrar.vibrate(patronMedio,0);
                                                }else{
                                                    if( distancia1 < 30  &&  distancia2 >= 30 && (  distancia3 < 30 || distancia4 < 30  ) ){

sonarUnaAlaVez(enfrenteCorta,derechaCorta,derechaMedia,enfrenteLarga,enfrenteMedia,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                        vibrar.vibrate(patronCorto,0);
                                                    }else{
                                                        if( distancia1 >=120    &&  (distancia2 >= 60 && distancia2 < 120) && ( (distancia3 >=
60 && distancia3 < 120) || (distancia4 >= 60 && distancia4 < 120)  ) ){

sonarUnaAlaVez(enfrenteLarga,derechaCorta,derechaMedia,enfrenteMedia,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                            vibrar.vibrate(patronLargo,0);
                                                        }else{
                                                            if( distancia1 >=60    &&  (distancia2 >= 30 && distancia2 < 60) && ( (distancia3
>= 30 && distancia3 < 60) || (distancia4 >= 30 && distancia4 < 60)  ) ){

sonarUnaAlaVez(enfrenteMedia,derechaCorta,derechaMedia,enfrenteLarga,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                                vibrar.vibrate(patronMedio,0);
                                                            }else{
                                                                if( distancia1 >=30    &&  distancia2 < 30  && ( ( distancia3 < 30) || (
distancia4 < 30)  ) ){

sonarUnaAlaVez(enfrenteCorta,derechaCorta,derechaMedia,enfrenteLarga,enfrenteMedia,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                                    vibrar.vibrate(patronCorto,0);
                                                                }else{
                                                                    if( (distancia1 >=60 && distancia1 < 120)    &&  (distancia2 >= 60 &&
distancia2 < 120) && ( (distancia3 >= 60 && distancia3 < 120) || (distancia4 >= 60 && distancia4 < 120)  ) ){

sonarUnaAlaVez(enfrenteLarga,derechaCorta,derechaMedia,enfrenteMedia,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                                        vibrar.vibrate(patronLargo,0);
                                                                    }else{
                                                                        if( (distancia1 >= 30 && distancia1 < 60)    &&  (distancia2 >= 30 &&
distancia2 < 60) && ( (distancia3 >= 30 && distancia3 < 60) || (distancia4 >= 30 && distancia4 < 60)  ) ){

sonarUnaAlaVez(enfrenteMedia,derechaCorta,derechaMedia,enfrenteLarga,enfrenteCorta,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                                            vibrar.vibrate(patronMedio,0);
                                                                        }else{
                                                                            if( distancia1 < 30    &&  distancia2 < 30 && ( distancia3 < 30 ||
distancia4 < 30  ) ){

sonarUnaAlaVez(enfrenteCorta,derechaCorta,derechaMedia,enfrenteLarga,enfrenteMedia,izquierdaCorta,izquierdaMedia,derechaLarga,izquierdaLarga);
                                                                                vibrar.vibrate(patronCorto,0);
                                                                            }
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

    }

    public void sonarUnaAlaVez(MediaPlayer sonar,MediaPlayer apagar1,MediaPlayer apagar2,MediaPlayer apagar3,MediaPlayer apagar4,MediaPlayer
apagar5,MediaPlayer apagar6,MediaPlayer apagar7,MediaPlayer apagar8){
        if( !sonar.isPlaying() ){sonar.start();}

        if( apagar1.isPlaying() ){apagar1.pause();}

        if( apagar2.isPlaying() ){apagar2.pause();}

        if( apagar3.isPlaying() ){apagar3.pause();}

        if( apagar4.isPlaying() ){apagar4.pause();}

        if( apagar5.isPlaying() ){apagar5.pause();}

        if( apagar6.isPlaying() ){apagar6.pause();}

        if( apagar7.isPlaying() ){apagar7.pause();}

        if( apagar8.isPlaying() ){apagar8.pause();}

    }

    public void apagarSonido(){
        if( derechaCorta.isPlaying() ){derechaCorta.pause();}

        if( derechaMedia.isPlaying() ){derechaMedia.pause();}

        if( derechaLarga.isPlaying() ){derechaLarga.pause();}

        if( enfrenteCorta.isPlaying() ){enfrenteCorta.pause();}

        if( enfrenteMedia.isPlaying() ){enfrenteMedia.pause();}
```

```
        if( enfrenteLarga.isPlaying() ){enfrenteLarga.pause();}
        if( izquierdaCorta.isPlaying() ){izquierdaCorta.pause();}
        if( izquierdaMedia.isPlaying() ){izquierdaMedia.pause();}
        if( izquierdaLarga.isPlaying() ){izquierdaLarga.pause();}
    }
    public void setDistancia1(int distancia1) {this.distancia1 = distancia1;}
    public void setDistancia2(int distancia2) {this.distancia2 = distancia2;}
    public void setDistancia3(int distancia3) {this.distancia3 = distancia3;}
    public void setDistancia4(int distancia4) {this.distancia4 = distancia4;}
}
```

App

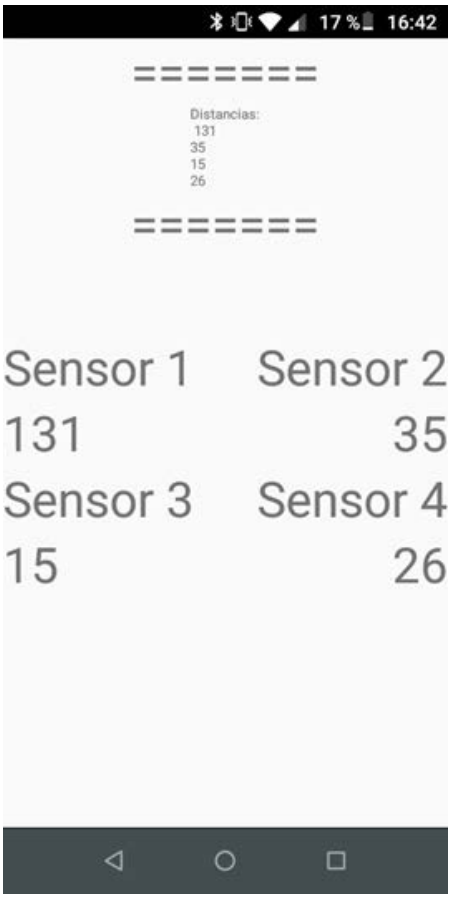

Figura 42. *App*.

Apéndice C. Comparación de Precios

Comparación estimada de precios de los componentes entre el proyecto de integración [PI] y los proyectos antecedentes
(Proyecto Wearable System *[1]* y Cinturón y Manillas *[2]*)

Tabla 8. Comparación de precios con el proyecto de integración.

	PI	Antecedente *[1]*	Antecedente*[2]*
Arquitectura similares	Si	Si	Si
¿Es configurable?	Si	Si	No
Cobertura/obstáculos	Media (inicialmente)	Alta	Baja
Precio del S.O.Android (*)	USD $328.00	---	---
PDA (**)	---	USD $500.00	---
Tarjeta myRio (***)		---	USD $ 768.25
Módulo Xbee	---	---	USD $ 30.00
Motores vibradores Precio c/motor USD $5.00.	---	Usa 8 motores. Total, USD $40.00	Usa 3 motores. Total, USD $15.00
Costo total	USD $328.00 + Arquitectura similar	USD $540.00 + Arquitectura similar	USD $813.25 + Arquitectura similar

(*) Sistema Operativo Android

Según el portal *Statista*, basado en datos recopilados en más de 75 mercados diferentes, excluyendo acuerdos especiales, el precio medio de venta de Smartphones en el mundo es cercano a los USD $ 328.00. Y si refináramos las estadísticas, llegaríamos a la conclusión de que el precio de los dispositivos con sistema operativo Android, es menor en relación con los demás.

Por otro lado, fue presentada la Encuesta Nacional sobre Disponibilidad y Uso de las Tecnologías de la Información en los Hogares (ENDUTHI) 2015. La cual informa que el 71.5 % usa telefonía móvil y 3 de cada 4 mexicanos cuenta con un Smartphone.

(**) PDA (Asistente Digital Personal)

Los PDA han perdido el auge que tenían en sus inicios, ya que están siendo sustituidos por Smartphones, los cuales tienen más funcionalidades. Algunos portales web, como *LogisCenter*, muestran que el promedio de venta de un PDA oscila alrededor de USD $ 500.00.

(***) Tarjeta myRio

En la tabla 9. Se observan los precios de las diferentes tarjetas. Se obtuvo un promedio y se comparó con los demás dispositivos.

Tabla 9. Precios entre los diferentes dispositivos myRIO para estudiantes.

Model	Price	Bus Connector Select
myRIO-1900	**$ 548.00**	USB 2.0, Wi-Fi
myRIO-1900	**$ 1,098.00**	USB 2.0, Wi-Fi
myRIO-1950	**$ 439.00**	USB 2.0
myRIO-1950	**$ 988.00**	USB 2.0

Proyecto de integración [PI] vs Proyecto Wearable System *[1]*

Con base en la tabla 1. Se puede observar el costo total estimado del PI contra el antecedente *[1]*, hay una diferencia de USD $212.00 favorable para el PI, dado que el costo medio de un Smartphone con SO Android no excede los USD $328.00; además, el PI no utiliza motores vibradores de cd.

Proyecto de integración [PI] vs Cinturón y Manillas *[2]*

Comparado el PI contra el antecedente *[2]*, la diferencia de costos (USD $ 485.25,) es notablemente favorable para el PI, debido al precio elevado de componentes como la tarjeta myRIO para la programación de sistemas embebidos.

En las comparaciones anteriores, las estadísticas indican que un alto porcentaje de personas en México cuenta con Smartphone, por lo que no habría necesidad de adquirir otro dispositivo, sino únicamente la pieza wearable, reduciendo significativamente el precio del producto.