UNIVERSIDAD AUTÓNOMA DE GUERRERO

UNIDAD ACADÉMICA DE INGENIERÍA

TESIS

MONITOR DE SIGNOS VITALES PORTÁTIL

QUE PRESENTA

ING. ZAIC REBOLLEDO NANDI

PARA OBTENER EL GRADO DE

MAESTRÍA EN INGENIERÍA

DIRECTOR DE TESIS

DR. GUSTAVO ADOLFO ALONSO SILVERIO

CHILPANCINGO, GUERRERO, DICIEMBRE DE 2016

Contenido

INTRODUCCIÓN

Hoy en día es sabido que parámetros biomédicos del cuerpo humano como ECG (Electrocardiograma), SpO_2 (saturación de oxígeno), frecuencia cardiaca, presión arterial, niveles de glucosa y temperatura son esenciales en el tratamiento de enfermedades crónicas, el tratamiento de los pacientes está estrechamente ligado con la evolución de estos parámetros. Por esta razón se necesita un sistema para el cuidado de la salud móvil y portátil, para poder obtener dichos parámetros biomédicos en cualquier momento y en cualquier lugar, y así poder incrementar la longevidad de los seres humanos. Un equipo móvil con Android OS (Sistema Operativo) es una herramienta poderosa capaz de procesar, almacenar y visualizar datos biomédicos.

Este trabajo presenta un dispositivo portátil capaz de medir ECG, SpO2, frecuencia cardiaca, presión arterial, niveles de glucosa y temperatura utilizando métodos no invasivos; los 5 parámetros previamente mencionados son enviados inalámbricamente utilizando comunicación Bluetooth a un dispositivo con Android OS para poder ser visualizados, almacenados, procesados y finalmente compartidos vía internet con un especialista.

ESTRUCTURA DE LA TESIS

La estructura de la tesis es la siguiente: Consta de introducción, Justificación, alcances, objetivos, cuatro capítulos, referencias, índice de tablas y figuras y anexos. En la introducción se presenta de forma general el contenido de la investigación y desarrollo del dispositivo descrito en este trabajo. En la Justificación se mencionan los argumentos del por qué realizar este trabajo y su utilidad. En los alcances se especifica los límites y cobertura de la investigación. En el apartado de objetivos generales se enuncian los propósitos a los que se quieren llegar con este trabajo de investigación. En el capítulo I, se describen las variables biomédicas que se desean medir en el dispositivo elaborado, así como la teoría necesaria sobre las herramientas utilizadas para el desarrollo. En el capítulo II, se mencionan los antecedentes en el monitoreo de variables biomédicas, así como el estado actual utilizando dispositivos portátiles. En el capítulo III se describe los materiales y métodos empleados en la investigación, así como la elaboración del dispositivo presentado en este trabajo. En el capítulo IV se presentan los resultados logrados en contraste con el de otros trabajos. En la sección de Referencias se enlistan todas las referencias utilizadas en este trabajo. Hay una sección de Índice de tablas de figuras así como de anexos donde se especifican los componentes electrónicos utilizados, además del código del Arduino y las clases principales en lenguaje Java de la aplicación Android.

JUSTIFICACIÓN

En años recientes, pacientes con enfermedades crónico-degenerativas, como cardiovasculares (por ejemplo, los infartos de miocardio o accidentes cerebrovasculares), el cáncer, las enfermedades respiratorias crónicas (por ejemplo, la neumopatía obstructiva crónica o el asma), y la diabetes (WHO, 2015), necesitan un constante monitoreo de los signos vitales, niveles de glucosa y actividad eléctrica del corazón, que debido a estudios anteriores se ha encontrado que están estrechamente ligados con el padecimiento de enfermedades de este tipo. El monitoreo de signos vitales es una tarea complicada, ya que medir estos parámetros biomédicos requiere de equipos robustos y costosos que necesitan de personal especializado para su operación. Hasta ahora, en el mercado no existe un dispositivo de este tipo a un precio relativamente bajo. Es por eso que se quiere cubrir esta necesidad con un monitor de signos vitales que sea portátil y wearable para que pueda ser llevado a cualquier lugar a donde tenga que desplazarse el paciente. El monitor debe ser de fácil operación y de bajo costo, para que pueda ser implementado en pacientes que no tengan acceso a un especialista.

ALCANCES

Contar con un dispositivo portátil y wearable, eficaz para aquellas personas que requieran un monitoreo constante del cuerpo y así puedan llevar un control confiable en el tratamiento de enfermedades crónico-degenerativas.

OBJETIVOS

Objetivo general

Desarrollar un monitor de signos vitales portátil y wearable utilizando software y hardware abierto con el fin de optimizar costos, que sea capaz de medir la frecuencia respiratoria, frecuencia cardiaca, temperatura, presión arterial, saturación de oxígeno, niveles de glucosa y obtener el electrocardiograma, con el fin de reducir costos en el monitoreo de pacientes; las variables serán enviadas vía Bluetooth y serán visualizadas en un dispositivo móvil Android.

Objetivos específicos

- Adquirir la señal de ECG utilizando la tarjeta AD8232 de SparkFun.
- Medir el pulso cardiaco y saturación de oxígeno utilizar un sensor de hardware abierto.
- Medir la presión arterial utilizando un sensor de presión que será adaptado a un baumanómetro.

- Determinar de forma no invasiva el nivel de glucosa en la sangre utilizando análisis de imágenes.
- Realizar el procesamiento de señales utilizar una tarjeta Arduino Nano Generica.
- Diseñar las interfaces para el dispositivo móvil Android en Java utilizando Android Studio IDE.
- Diseñar una cubierta para el alojamiento de la circuitería necesaria, cubriendo los requisitos de ser portátil y wearable al cuerpo.

CAPÍTULO I. MARCO TEÓRICO E HIPÓTESIS

1.1. Marco Teórico

1.1.1. Variables Biomédicas

Actualmente el monitoreo de las variables vitales es una tarea ordinaria en hospitales, incluso se cuenta con monitores especializados para la obtención de estas variables; estos monitores suelen ser de tamaño robusto. Para poder tomar las lecturas es necesario de cierto adiestramiento en el uso de la instrumentación para la medición o, en su defecto, sería necesario conocer cómo operar un monitor de signos vitales. Estas variables indican cómo es la evolución de un paciente ante alguna enfermedad; en algunas enfermedades crónico-degenerativas es necesario un monitoreo de forma constante, tarea que puede resultar difícil si la persona está fuera de un centro médico, y no tiene acceso a la instrumentación necesaria para la obtención de parámetros biomédicos.

Para poder tomar mediciones de las variables vitales de forma constante, aún sin estar en algún centro médico o sin la ayuda de personal capacitado en la toma de éstas, es necesario un dispositivo portátil y que pueda ser llevado a todos lados, sin que se vuelva un estorbo para las actividades diarias del paciente. Tomar ventaja de un dispositivo celular como monitor resulta ser una opción sustentable, ya que en la actualidad la mayoría de las personas cuenta con uno. Por esta razón, sólo haría falta desarrollar la parte electrónica necesaria para obtener las variables vitales del cuerpo humano, enviarlas al dispositivo celular y visualizarlas en él. Si la parte electrónica, se desarrolla con software y hardware abierto, hace que el dispositivo sea de un costo más accesible en comparación con un sistema de monitoreo de un hospital o centro médico.

Si se considera un monitor de signos vitales, es indispensable conocer algunos términos. Estos signos son las señales fisiológicas que indican la presencia de vida de una persona y proporcionan datos para evaluar el estado de salud del paciente, así como los cambios en su evolución, ya sea positiva o negativamente (Garibay Rubio et al., 2006). A continuación se definen cada una de las variables que puede medir un monitor de signos vitales.

Frecuencia respiratoria: se toma usando la mnemotecnia VES (ver, oír, sentir) contando cuantas ventilaciones por minuto realiza la persona. Este es el único signo vital que uno mismo puede controlar, por lo que es importante no decirle al paciente que se va a valorar, para que no altere su patrón ventilatorio (Garibay Rubio et al., 2006).

Frecuencia cardiaca: se toma con un estetoscopio, el cual se coloca a la altura del quinto espacio intercostal en la línea media clavicular, es decir, a la altura del pezón izquierdo inclinando el estetoscopio un poco hacia la izquierda. Al igual que la frecuencia

respiratoria, se cuenta cuántas veces late el corazón en un minuto (Garibay Rubio et al., 2006).

Pulso*:* este signo indica que está llegando la sangre a todas las zonas del cuerpo. Se debe contabilizar cuántas pulsaciones hay en un minuto y detectar si es débil o fuerte. Existen diferentes zonas para tomar el pulso:

- Pulso carótido: se coloca el dedo índice y medio en el mentón, se sigue en línea recta hacia el cartílago cricoides (manzana de adán) y se recorre lateralmente 2 cm aproximadamente haciendo cierta presión.

- Pulso radial: se descubre la muñeca, con el dedo índice y medio se sigue la línea del dedo pulgar hasta la muñeca y se ejerce presión hacia el hueso.

- Pulso braquial: éste se utiliza sobre todo en niños, debido a que ellos tienen mucho más sensible el nervio del cuello. La manera de tomarlo es descubrir el brazo, el dedo índice y medio se colocan en el bíceps y se recorren hacia la cara interior del brazo separando los músculos y haciendo presión hacia el hueso (Garibay Rubio et al., 2006).

Presión arterial: es la presión que ejerce la sangre contra la pared de las arterias. Esta presión es imprescindible para que circule la sangre por los vasos sanguíneos y aporte el oxígeno y los nutrientes a todos los órganos del cuerpo, para que puedan funcionar. Para obtener su medición se coloca el baumanómetro en el brazo con la flecha o las mangueras en la zona de la arteria (el doblez del codo), se cierra pero no se aprieta al brazo, se busca el pulso de la arteria que pasa en esa zona y ahí se coloca la campana del estetoscopio; con la perilla se hace subir la aguja del baumanómetro, hasta los 160 mmHg o dependiendo de la presión que maneje normalmente el paciente, después se abre la perilla lentamente para poder escuchar en donde se empieza a oír el latido cardiaco y donde se deja de escuchar. El primer ruido y el último que se escuche indicará cuál es la tensión arterial (Garibay Rubio et al., 2006).

Temperatura corporal: se toma por medio de un termómetro, ya sea debajo del brazo o debajo de la lengua. También a grandes rasgos se puede saber la temperatura corporal palpando la piel de la persona, ya que ésta se puede sentir muy caliente o fría. Una temperatura normal es de 36 a 37 °C (Garibay Rubio et al., 2006).

Un monitor de signos vitales, por lo tanto, será un dispositivo capaz de medir todas estas variables mencionadas con anterioridad, pero también se podrán visualizar otros datos, tales como:

Saturación de oxigeno: la oximetría de pulso o pulsioximetría es la medición, no invasiva, del oxígeno transportado por la hemoglobina en el interior de los vasos sanguíneos. El color de la sangre varía dependiendo de lo saturada de oxígeno que se encuentre, debido a las propiedades ópticas del grupo hemo de la molécula de hemoglobina. Cuando la molécula de hemoglobina libera oxígeno pierde su color rosado, adquiriendo un tono más azulado y deja pasar menos la luz roja.

El pulsioxímetro determina la saturación de oxígeno midiendo espectrofotométricamente el "grado" de azules de la sangre arterial y expresa estos "azules" en términos de saturación. Dado que la absorción de luz de los tejidos y de la sangre venosa es constante, cualquier cambio en la absorción de la luz entre un tiempo dado y uno posterior se debe exclusivamente a la sangre arterial. Los pulsioxímetros miden pues la relación, en un intervalo de tiempo, entre las diferencias de absorción de las luces rojas e infrarrojas. Esta relación se vincula directamente con la saturación de oxihemoglobina.

Se precisa de un aparato de pulsioximetría, con un sensor en forma de pinza. En la pinza tiene un emisor de luz que se refleja en la piel del dedo, este sensor mide la cantidad de luz absorbida por la oxihemoglobina circulante en el paciente (Garibay Rubio et al., 2006).

ECG (Electrocardiograma): es la representación gráfica de la actividad eléctrica del corazón, lo que constituye un instrumento principal de la electrofisiología cardíaca y tiene una función relevante en la identificación y diagnóstico de las enfermedades cardiovasculares. Consiste en colocar electrodos en zonas específicas del cuerpo para así poder captar las señales eléctricas del corazón, estas señales son representadas en una gráfica a la que se nombra como electrocardiograma (Becerra-Luna et al., 2012).

La actividad eléctrica del corazón recogida en el ECG se observa en forma de un trazado que presenta diferentes deflexiones (ondas del ECG) que se corresponden con el recorrido de los impulsos eléctricos a través de las diferentes estructuras del corazón. Para intentar comprender los principios básicos que explican las oscilaciones en las líneas del ECG conviene conocer, si bien de forma somera, los fundamentos por los cuales se produce el movimiento del corazón, generado a través de microcorrientes eléctricas. De ello es responsable el sistema de conducción eléctrica del corazón (Kim et al., 2015).

El ECG presenta como línea guía la denominada línea isoeléctrica o línea basal, que puede identificarse fácilmente como la línea horizontal existente entre cada latido, ver fig. 1.1. Los latidos cardíacos quedan representados en el ECG normal por las diferentes oscilaciones de la línea basal en forma de ángulos, segmentos, ondas e intervalos, constituyendo una imagen característica que se repite con una frecuencia regular a lo largo de la tira de papel del ECG. Como se ha comentado, entre latido y latido va discurriendo la línea base (Castellano et al., 2004).

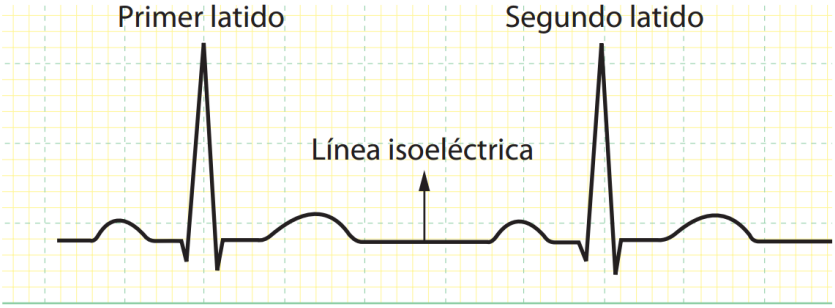


Fig 1.1 Representación de dos latidos cardíacos consecutivos en el electrocardiograma (Castellano et al., 2004).

El recorrido en sentido horizontal hace referencia al tiempo transcurrido, y la distancia en sentido vertical (altura o profundidad) al voltaje que se está produciendo. El papel por el que discurre el registro de la línea se encuentra milimetrado. Cada cuadrado pequeño del papel mide 1 mm y al observarlo con detenimiento puede comprobarse que cinco cuadrados pequeños forman un cuadrado grande, remarcado por un grosor mayor en la tira de papel del ECG. Para conocer cómo transcurren los tiempos durante la actividad del corazón, basta con recordar que cinco cuadrados grandes en sentido horizontal equivalen exactamente a un segundo (Pourafkari et al., 2016).

En un ECG normal, cada complejo consta de una serie de deflexiones (ondas del ECG) que alternan con la línea basal. Realizando la lectura de izquierda a derecha, se distinguen la onda P, el segmento P-R, el complejo QRS, el segmento ST y finalmente la onda T, ver fig. 1.2 (Jeroudi et al., 2015).

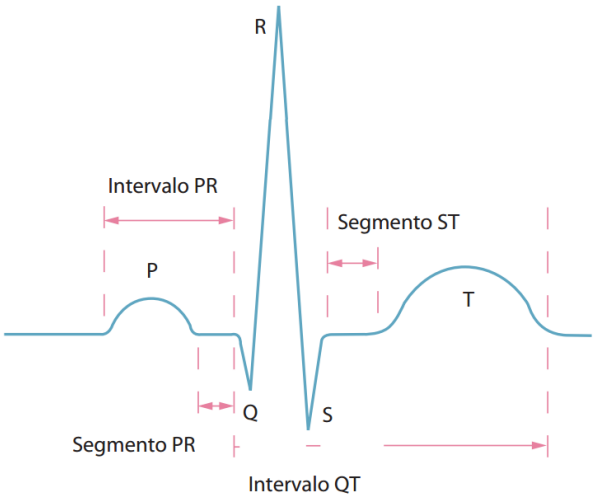


Fig. 1.2 Diferentes ondas del electrocardiograma (Jeroudi et al., 2015).

La onda P es la primera deflexión hacia arriba que aparece en el ECG. Su forma recuerda a una mezcla entre una U y una V invertidas. Suele durar unos dos cuadrados pequeños (con duración se hace referencia al tiempo, por lo que se debe mirar el número de cuadrados en sentido horizontal). Representa el momento en que las *aurículas* se están contrayendo y enviando sangre hacia los *ventrículos* (Karimipour and Homaeinezhad, 2014).

El segmento P-R Es el tramo de la línea basal (línea isoeléctrica) que se encuentra entre el final de la onda P y la siguiente deflexión, que puede ser hacia arriba (positiva) o hacia abajo (negativa), del ECG. Durante este período, las aurículas terminan de vaciarse y se produce una relativa desaceleración en la transmisión de la corriente eléctrica a través del corazón, justo antes del inicio de la contracción de los ventrículos (Karimipour and Homaeinezhad, 2014).

El complejo QRS Corresponde con el momento en que los ventrículos se contraen y expulsan su contenido sanguíneo. Como su nombre indica, consta de las ondas Q, R y S. La onda Q no siempre está presente. Se identifica por ser la primera deflexión negativa presente después del segmento P-R. Toda deflexión positiva que aparezca después del segmento P-R corresponde ya a la onda R propiamente dicha y, como se ha comentado anteriormente, el hecho de que no vaya precedida por una onda Q no es en absoluto patológico. De hecho, y siempre en relación con un ECG normal, las ondas Q deben ser de pequeño tamaño, no mayores que un cuadrado pequeño, tanto en longitud (duración) como en profundidad (voltaje), y encontrarse presentes sólo en ciertas derivaciones. La onda R es muy variable en altura (no debe olvidarse que las mediciones en el eje vertical tanto en altura como en profundidad expresan voltaje), ya que puede llegar a medir desde medio cuadrado hasta incluso cuatro o cinco cuadrados grandes en el caso de personas jóvenes deportistas. La onda S se observa como continuación directa de la onda R y comienza a partir del punto en que esta última, en su fase decreciente, se hace negativa. En conjunto, el complejo formado por las ondas Q, R y S no debe exceder en duración más de dos cuadrados pequeños (Wellens et al., 2000, Karimipour and Homaeinezhad, 2014).

Segmento ST es el trazado de la línea basal que se encuentra entre el final de la onda S y el comienzo de la onda T. Su elevación o descenso en relación con la línea basal puede significar insuficiencia en el riego del corazón, especialmente si dichas oscilaciones coinciden con sintomatología característica que pueda expresar afectación en el aporte de oxígeno al corazón. En este sentido, su valor como herramienta diagnóstica resulta insustituible (Karimipour and Homaeinezhad, 2014, Pourafkari et al., 2016).

Onda T se inscribe a continuación del segmento ST. Consiste en una deflexión normalmente positiva (es decir, por encima de la línea basal) que asemeja el relieve de una montaña más o menos simétrica. Su altura suele estar entre dos y cuatro cuadrados

pequeños y su duración no debe exceder los tres. La onda T representa el momento en que el corazón se encuentra en un período de relajación, una vez que ha expulsado la sangre que se hallaba en los ventrículos (Karimipour and Homaeinezhad, 2014, Pourafkari et al., 2016).

Niveles de Glucosa: los niveles óptimos de glucosa en la sangre son 80 a 100 mg/dl en ayunas y menor de 140 mg/dl después de comer. Unos niveles de 100 a 125 mg/dl en ayunas 140 a 199 mg/dl después de comer indicarían pre-diabetes. Niveles mayor a 126 mg/dl en ayunas y mayor a 200 mg/dl después de comer indican diabetes (Secretaria de Salud, 2009).

La medición de glucosa en la sangre generalmente se hace de forma invasiva al cuerpo extrayendo sangre de él y haciendo el análisis en laboratorio, otro tipo de análisis más, que no involucra ir a un laboratorio es a través de un glucómetro, que es un dispositivo portátil, que también toma una muestra de sangre de forma invasiva.

1.1.2. Arduino

Arduino es una plataforma de prototipos de código abierto basado en hardware y software fácil de usar. Las placas Arduino son capaces de leer entradas tales como la luz en un sensor, un dedo en un botón, o un mensaje de Twitter y convertirlas en una salida como la activación de un motor, encender un LED o publicar algo en línea. Para ello se utiliza el lenguaje de programación de Arduino (basado en *Wiring*) y el software de Arduino (IDE), basado en *Processing*.

A través de los años Arduino ha sido el cerebro de miles de proyectos, a partir de objetos cotidianos hasta instrumentos científicos complejos. Una comunidad mundial de fabricantes, estudiantes, aficionados, artistas, programadores y profesionales han optado por trabajar con esta plataforma de código abierto; sus contribuciones han añadido hasta ahora una increíble cantidad de conocimiento accesible que puede ser de gran ayuda, tanto para los principiantes como para expertos.

Arduino nació en Ivrea Interaction Design Institute como una herramienta que facilita la creación de prototipos, dirigido a estudiantes sin experiencia en electrónica y programación. Tan pronto como alcanzó una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, generando una gran variedad de productos desde simples tarjetas de 8 bits para aplicaciones de la IO (entrada salida), hasta aplicaciones especializadas como *wearables*, impresión en 3D entre otras aplicaciones de software embebido. Todas las placas Arduino son completamente de código abierto, permitiendo a los usuarios crear de forma independiente y, finalmente, adaptarlos a sus necesidades particulares. El software también es de código abierto, y ha mejorado a través de las aportaciones de los usuarios en todo el mundo.

Gracias a su sencilla y accesible experiencia de usuario, Arduino se ha utilizado en miles de diferentes proyectos y aplicaciones. El software de Arduino es fácil de usar para los principiantes, pero lo suficientemente flexible para los usuarios avanzados. Se ejecuta en Mac, Windows y Linux. Los profesores y los estudiantes lo utilizan para construir instrumentos científicos de bajo costo, para demostrar los principios de química y física, o para empezar con la programación y la robótica. Diseñadores y arquitectos construyen prototipos interactivos, músicos y artistas lo utilizan para instalaciones de luces y experimentar con nuevos instrumentos musicales. Los fabricantes, por supuesto, lo utilizan para construir muchos de los proyectos expuestos en la Maker Faire, por ejemplo. Arduino es una herramienta clave para aprender cosas nuevas. Cualquier persona ya sea niño, aficionado, artista o programador, puede comenzar a juguetear simplemente siguiendo paso a paso las instrucciones de un kit de desarrollo, o compartir ideas en línea con otros miembros de la comunidad de Arduino.

Hay muchos otros *microcontroladores* y plataformas de microcontroladores disponibles para computación física. Parallax Basic Stamp, de Netmedia BX-24, Phidgets, Handyboard del MIT, y muchos otros ofrecen una funcionalidad similar. Todas estas herramientas toman los detalles complicados de programación de microcontroladores y se envuelve en un paquete fácil de usar. Arduino también simplifica el proceso de trabajar con microcontroladores, pero ofrece algunas ventajas para los profesores, estudiantes y aficionados interesados sobre otros sistemas:

Bajo costo: las Placas Arduino son relativamente económicas, en comparación con otras plataformas de microcontroladores. La versión menos costosa del módulo Arduino puede ser montado a mano e incluso los módulos de Arduino premontados son de costos accesibles.

Multiplataforma: El software de Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los sistemas de microcontrolador se limitan a Windows.

Entorno de desarrollo y programación simple: El software de Arduino (IDE) es fácil de usar para los principiantes, pero lo suficientemente flexible para los usuarios avanzados. Se basa convenientemente en el entorno de programación Processing, por lo que los estudiantes aprenden a programar en ese entorno y estarán familiarizados con cómo funciona el Arduino IDE.

El código abierto: El software de Arduino está publicado como herramientas de código abierto, disponible para la extensión por programadores experimentados. El idioma se puede ampliar a través de bibliotecas C++, y las personas con ganas de entender los detalles técnicos pueden dar el salto de Arduino al lenguaje de programación AVR-C, en

el que se basa. Del mismo modo, se puede agregar código AVR-C directamente en los programas de Arduino.

Hardware abierto: Los planes de las placas Arduino se publican bajo una licencia de Creative Commons, por lo que los diseñadores de circuitos experimentados pueden hacer su propia versión del módulo, ampliándolo y mejorándolo. Incluso los usuarios con poca experiencia pueden construir la versión tablero del módulo con el fin de entender cómo funciona y tener un beneficio económico (Arduino, 2015).

1.1.3. Android OS

Android OS (operative system, sistema operativo) es un sistema operativo basado en el núcleo *Linux*. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.11. Los dispositivos de Android venden más que las ventas combinadas de Windows Phone e IOS.

1.1.4. Java

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente (Deitel and Deitel, 2004).

1.1.5. Android Studio

Es el entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014. Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux (google, 2015).

1.1.6. OpenCV

OpenCV es una biblioteca libre de *visión artificial*, originalmente desarrollada por Intel. Desde que apareció su primera versión alfa, en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación, con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica. El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi núcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel (Dawson-Howe, 2014).

1.1.7. FreeCAD

FreeCAD es una aplicación libre de diseño asistido por computadora en tres dimensiones, ingeniería asistida por computadora, para la asistencia en ingeniería mecánica y el diseño de elementos mecánicos. Está basado en Open CASCADE y programado en los lenguajes C++ y Python (FreeCAD, 2016).

FreeCAD presenta un entorno de trabajo similar CATIA, SolidWorks, SolidEdge, ArchiCAD o Autodesk Revit. Utiliza técnicas de modelado paramétrico y está provisto de una arquitectura de software modular, pudiendo añadir de forma sencilla funcionalidades sin tener que cambiar el núcleo del sistema.

A diferencia de los CAD analíticos tradicionales, como pueden ser AutoCAD o Microstation, FreeCAD es un CAD paramétrico que utiliza parámetros para definir sus límites o acciones. En el diseño paramétrico cada elemento del dibujo (muros, puertas,

ventanas, etc.) es tratado como un objeto, el cual no es definido únicamente por sus coordenadas espaciales (x, y, z), sino también por sus parámetros, ya sean estos gráficos o funcionales. Las bases de datos relacionadas con el objeto hacen que este software, y especialmente su banco de trabajo de arquitectura, esté muy relacionado con el enfoque BIM, en el que un modelo BIM contiene el ciclo de vida completo de la construcción, desde el concepto hasta la edificación (FreeCAD, 2016).

Como muchos modernos modeladores CAD en 3D, tiene un componente para dos dimensiones para extraer un diseño detallado de un modelo 3D y con ello producir dibujos en 2D, pero el diseño directo en 2D (como el de AutoCAD LT) no es la meta, ni tampoco la animación ni formas orgánicas (como las creadas por Maya, 3ds Max o Cinema 4D) (FreeCAD, 2016).

1.2. Hipótesis

Utilizar un dispositivo celular Android, como monitor de signos vitales, permitirá –tanto a médicos como a pacientes– un mejor seguimiento y control de enfermedades crónico degenerativas a un bajo costo

CAPÍTULO II. ESTADO DEL ARTE

2.1. Antecedentes

Los principios de la monitorización se remontan a 1887, cuando Ludwig y Waller fueron los primeros en descubrir el diferencial de potencial eléctrico en el pecho humano, iniciando de esta manera la electrofisiología cardiaca. El electrocardiograma comenzó en 1893 con el trabajo del galvanómetro de cuerda de Einthoven. El cuerpo del trabajo siguió en este campo, pero el desarrollo de dispositivos que permitían una escritura directa en papel revolucionaron y promovieron el uso de una nueva herramienta de diagnóstico, esta herramienta sería el electrocardiógrafo (Einthoven, 1957).

En la realización de estas pruebas el paciente debía estar inmóvil o evitar hacer ejercicio, debido a que estaba enchufado a una máquina mediante electrodos. Este problema sirvió de inspiración al profesor Norma J. Holter, quien en los años 40 realizó una serie de estudios en los que consiguió estimular preparaciones de músculo por control remoto y registrar potenciales eléctricos de sistemas biológicos a distancia. Con la colaboración de Joseph A. Gengerelli aprendieron a emitir señales de radio al cerebro de ratas, posteriormente, amplificar la respuesta emitida por el cerebro del animal, transmitiéndola a distancia. En 1947 logran, trabajando ya en humanos, la primera transmisión de un electroencefalograma mientras el sujeto se encontraba realizando ejercicio físico (Einthoven, 1957).

Las señales emitidas por el músculo cardiaco eran 10 veces más potentes que las señales de cualquier otro músculo, por lo que decidió continuar sus investigaciones en el campo de la cardiología. Esta decisión fue tomada no sólo por la facilidad que conlleva tratar una señal más fuerte, sino también por el aumento de interés de la comunidad médica por lo problemas cardiovasculares (Einthoven, 1957).

El primer éxito del profesor Holter fue un transmisor de radio de 38 Kg de peso con un alcance de una manzana de casas, ver fig. 1. Sin embargo, con la colaboración de un físico llamado W.R. Glasscock, desarrolló un transmisor más pequeño con un receptor con grabadora que iba incorporado en un maletín que llevaba el propio sujeto de la investigación (Holter, 1957).

Fig. 2.1. Holter original de 38 kg (Holter, 1957).

A mediados de los 60, los avances técnicos favorecidos por el descubrimiento del transistor permitieron disminuir el tamaño del registrador, que ya pudo ser conectado directamente a los electrodos que se colocaban sobre el pecho del paciente. Así se eliminó la necesidad de transmitir la señal por radio y el sistema se hizo totalmente portátil.

Por este invento, el profesor Norman J. Holter consiguió algo infrecuente en la medicina, prestar su nombre a un método diagnóstico y que este método se convirtiera en pieza fundamental del diagnóstico en cardiología. En un principio los registradores permitían 10 horas de grabación en cintas magnéticas, ver fig 2, pero el desarrollo comercial del sistema impulsó el desarrollo de nuevos diseños que han conducido a grabadoras de tan sólo unos gramos de peso y con capacidad de grabaciones ininterrumpidas de hasta 48 horas en una memoria de estado sólido (Holter, 1957).

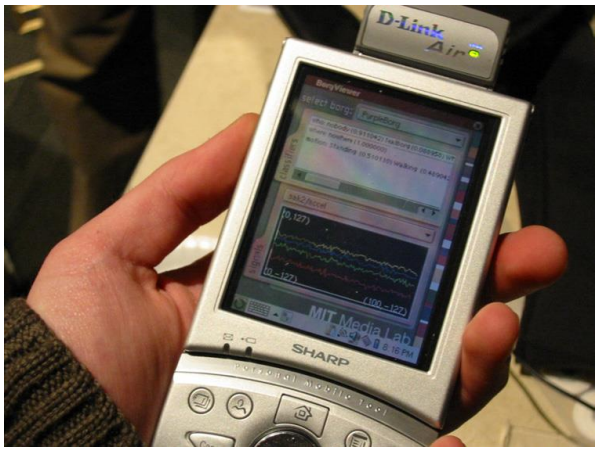
Fig. 2.2. Holter portátil capaz de grabar en cinta magnética (Holter, 1957).

El trabajo del Dr. Holter y sus colaboradores no se detuvo solamente en el grabador, también idearon un sistema para el análisis rápido de la señal de la ECG grabada en las cintas magnéticas. Desarrollaron en un osciloscopio de rayos catódicos un sistema que permitía la sobreimposición de los complejos cardiacos a una velocidad 60 veces superior a la grabada. La posterior adición de una señal de audio que aumentaba con los incrementos de la frecuencia cardiaca, completó el sistema que se denominó AVSEP, del inglés: audiovisual superimposed electrocardiographic presentation (Holter, 1957). La introducción de este nuevo método de diagnóstico en la medicina amplió enormemente el estudio de los problemas del ritmo cardiaco.

2.2. Estado actual

El monitoreo de los signos vitales del cuerpo humano, como se ha comentado, se hace con un dispositivo electrónico llamado monitor de signos vitales. Estos dispositivos son muy aparatosos y costosos, además de requerir de personal entrenado para su manipulación; sin embargo, existe cierta tendencia hacia los monitores portátiles.

En el 2004 el MIT (Instituto Tecnológico de Massachusetts) se desarrolla LiveNet (ver fig. 2.3), el cual es un sistema que tiene como monitor un PDA, es capaz de medir el ECG, presión arterial, temperatura y saturación de oxígeno. En este dispositivo, la comunicación entre los sensores microcontrolador y PDA (Asistente Personal Digital) es totalmente alámbrica (Sung et al., 2005).



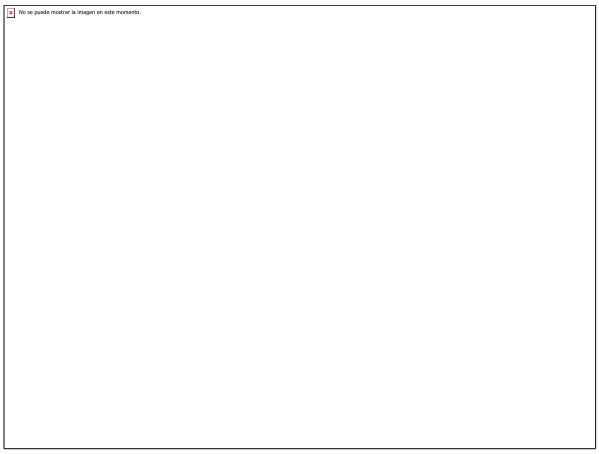


Fig. 2.3. LiveNet desarrollado por el MIT (Sung et al., 2005)..

La Universidad de Stanford y la NASA, en el 2004, desarrollaron el proyecto LifeGuard (ver fig. 2.4). Este dispositivo portátil toma la medición de las variables vitales y las envía de forma inalámbrica a una PC, la cual hace la función de monitor. El dispositivo puede tomar mediciones de ECG, presión arterial, saturación de oxígeno y temperatura (Mundt et al., 2005).

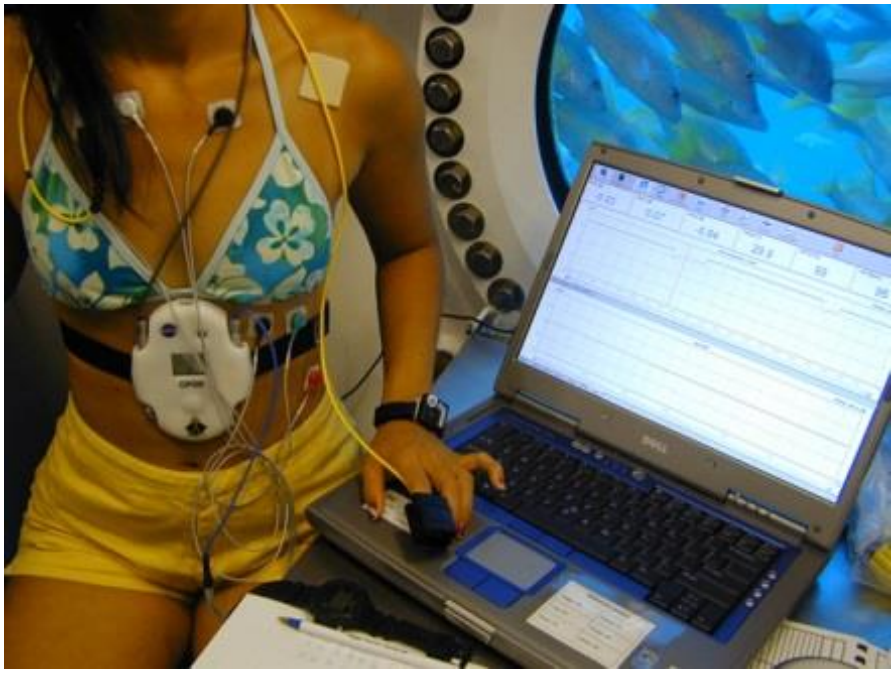


Fig. 2.4. Proyecto LifeGuard por la universidad de Stanford y NASA (Mundt et al., 2005).

La Universidad de Yonsei, en el 2005, desarrolló un sistema de monitoreo de glucosa para el cuidado de la salud. Este sistema está diseñado para medir la glucosa en la orina durante las actividades diarias del paciente. El sistema consiste en un sensor bioquímico conectado a un microcontrolador PIC, además de circuitos de control y análisis de señales (ver fig. 2.5). El dispositivo únicamente muestra el nivel de glucosa en una pequeña pantalla LCD (liquid cristal display) (Park et al., 2005).

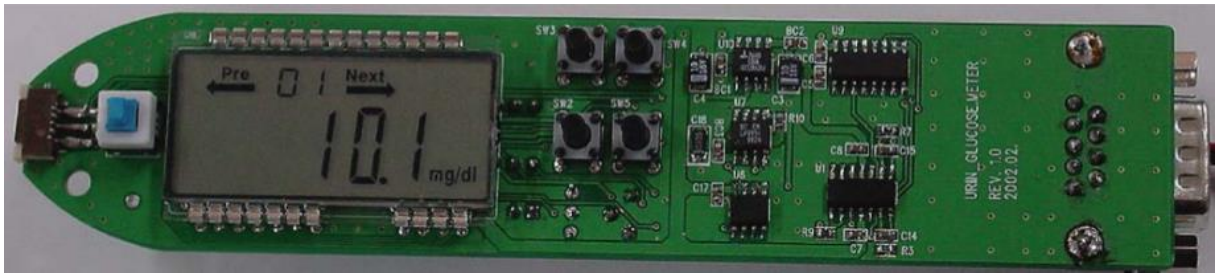


Fig. 2.5. Sistema para medir niveles de glucosa a través de la orina (Park et al., 2005).

Health Gear es el desarrollo por parte de Microsoft en el 2006, sólo mide el pulso cardiaco y la saturación de oxígeno (ver fig. 2.6). Este desarrollo fue pensado para operar con un teléfono celular, aunque también el dispositivo tiene que ir conectado de forma alámbrica al celular (Oliver and Flores-Mangas, 2006).

Fig. 2.6. Helth Gear desarrollado por Microsoft (Park et al., 2005).

En el 2009 la Universidad de Pittsburgh desarrolló HeartToGo (ver fig. 2.7). Dicho desarrollo también funciona con un teléfono celular como monitor, sólo mide el ECG pero es capaz de mandar los datos de la medición de forma inalámbrica al teléfono celular (Jin et al., 2009).

Fig. 2.7. HeartToGo systema para medir ECG por la Universidad de Pittsburg (Jin et al., 2009).

En el año 2010 investigadores en la Universidad de Virginia Commonwalth desarrollaron un sistema para obtener parametros biomedicos en tiempo real para astronautas durante las caminatas espaciales (ver fig. 2.8). Este sistema únicamente mide temperatura, saturacion de oxigeno, pulso cardiaco y la conductivdad de la piel (Fei et al., 2010).

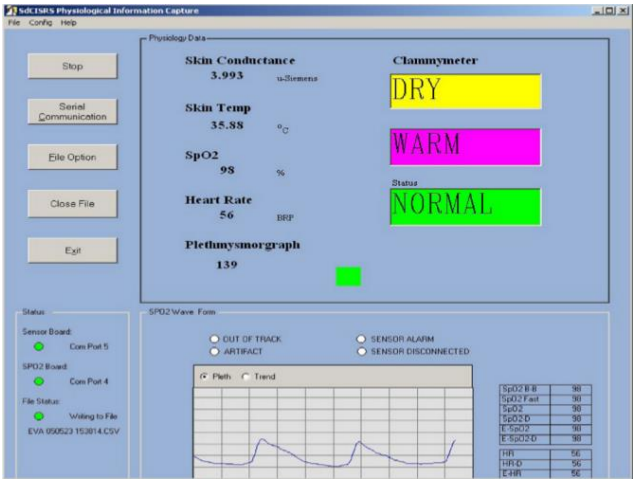


Fig. 2.8. Interfaz gráfica mostrada en un pequeño monitor que va aderido al casco del astronauta (Fei et al., 2010).

.

Investigadores de la universidad de Texas desarrollaron un sistema llamado Biowatch (ver fig. 2.9). Este dispositivo es un pequeño reloj de pulsera el cual es capaz de adquirir unicamente las señales de ECG y SpO_2. Los datos son enviados via comunicación Bluetooth a una PC para ser monitoreados y procesados en MATLAB (Thomas et al., 2014).

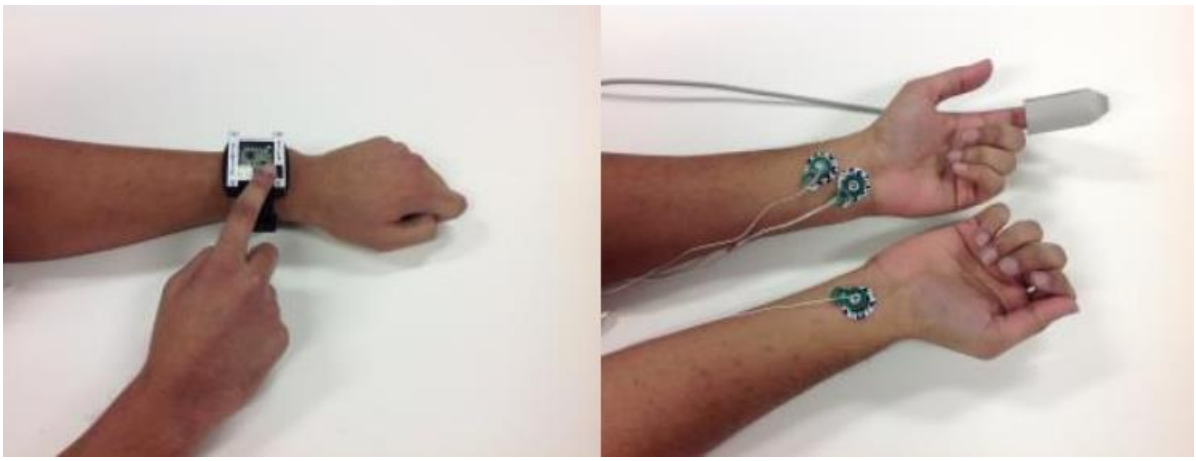


Fig. 2.9. Biowatch es un reloj de pulsera desarrollado por la Universidad de Texas (Thomas et al., 2014).

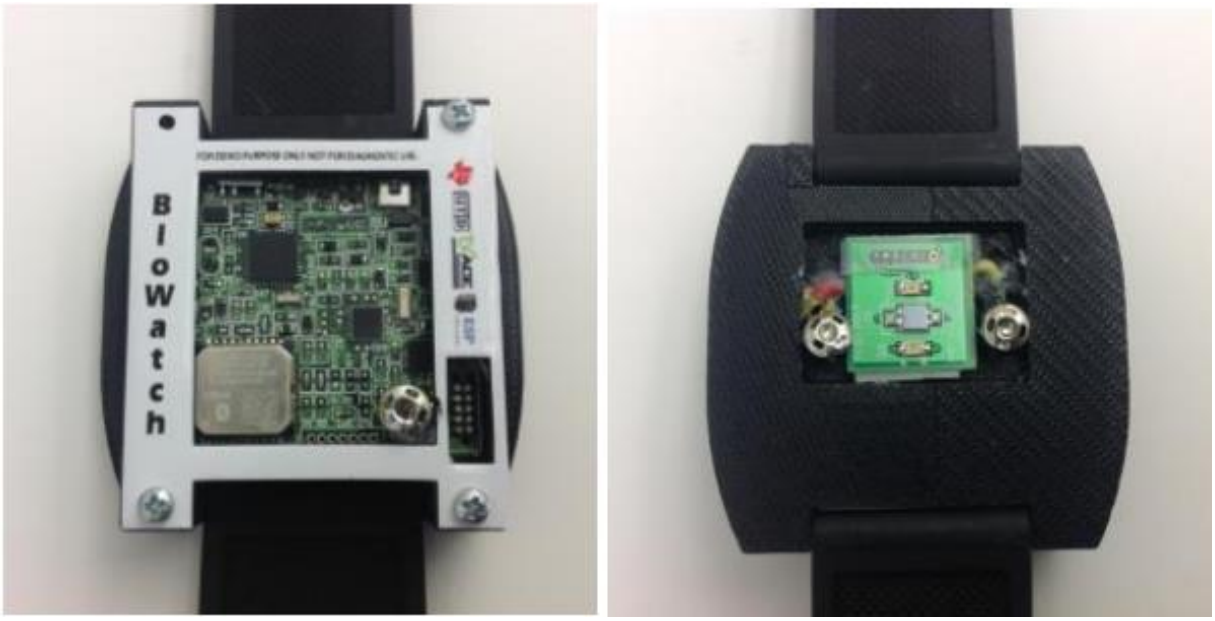


Fig. 2.9. Biowatch es un reloj de pulsera desarrollado por la Universidad de Texas. Continuación... (Thomas et al., 2014).

La empresa privada Vivometrics, en el 2004, desarrolló el proyecto LiveShirt (ver fig 2.10), que básicamente es un chaleco capaz de obtener la señal de ECG, frecuencia cardiaca y frecuencia respiratoria. El monitor obtiene las señales de forma alámbrica (Heilman and Porges, 2007).

Fig. 2.10. Liveshirt alámbrica (Heilman and Porges, 2007).

Actualmente algunos teléfonos móviles de alta gama tienen la capacidad de tomar algunas variables vitales como el pulso cardiaco y la presión arterial, con el uso de relojes de pulsera que tienen comunicación con el dispositivo móvil (ver fig. 2.11). Sin embargo, no son capaces de medir tantas variables biomédicas como un monitor de signos vitales comercial.

Fig. 2.11. Sistema para el cuidado de la salud desarrollado por Apple (Apple, 2015)

CAPÍTULO III. DISEÑO Y CONSTRUCCIÓN DEL MONITOR DE SIGNOS VITALES

3. Desarrollo del prototipo

El prototipo mostrado en este trabajo es capaz de medir diferentes parámetros biomédicos de forma no invasiva, los cuales son ECG, pulso cardiaco, SpO_2, presión arterial, temperatura y niveles de glucosa en la sangre. El dispositivo es alimentado por una batería recargable LiPo de 3.7 V. Todo el sistema está basado en software y hardware abierto, asegurando un sistema móvil para el cuidado de la salud de bajo costo. Todas las señales biomédicas son procesadas usando un microcontrolador basado en la plataforma Arduino, a excepción de los niveles de glucosa, los cuales son calculados utilizando un novedoso método no invasivo que consiste en el análisis de imágenes. Posteriormente, los datos son enviados a un Smartphone con Android OS, usando un módulo Bluetooth. Se desarrolló una aplicación para Android OS, con el fin de poder visualizar y almacenar los datos recibidos por el Smartphone; una vez que los datos son recibidos, es posible compartirlos instantáneamente vía internet. El diagrama de bloques del sistema implementado se muestra en la fig. 3.1.

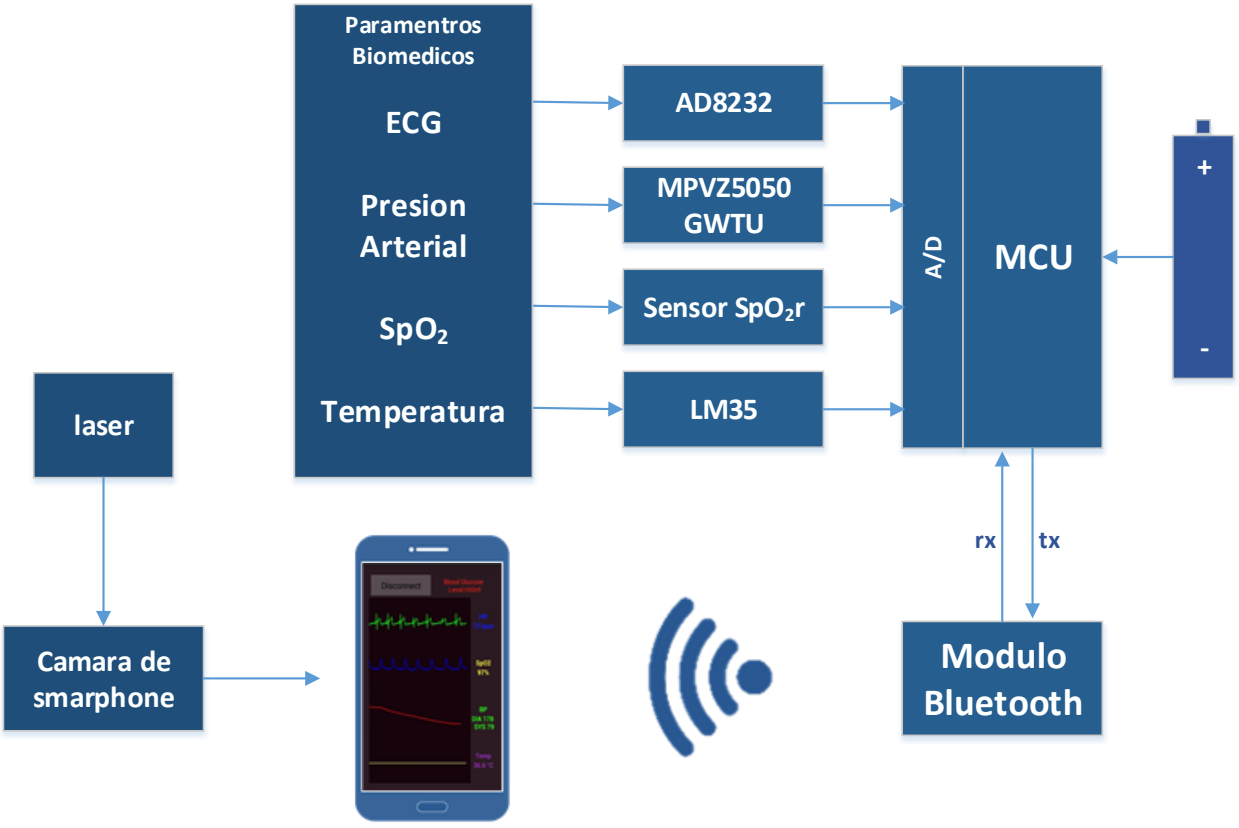


Fig. 3.1. Diagrama a bloques del sistema para el cuidado de la salud desarrollado.

3.1. Electrocardiograma ECG

La señal de ECG fue adquirida usando un chaleco deportivo con electrodos de Ag/AgCl embebidos en él, el chaleco puede ser ajustado para garantizar el contacto de los electrodos con la piel (ver fig. 3.2). Esta señal es recibida por la tarjeta AD8232 SparkFun Single Lead Heart Rate Monitor, la cual es de hardware abierto. La tarjeta tiene como componentes un circuito integrado AD8232, el cual fue calibrado y validado por la compañía ANALOG DEVICES (Analog Devices, 2012); este circuito integrado puede extraer una señal de ECG y después amplificarla y filtrarla antes de enviarla a un microcontrolador, en este caso a una entrada analógica del Arduino Nano. El circuito esquemático se muestra en la fig 3.3.

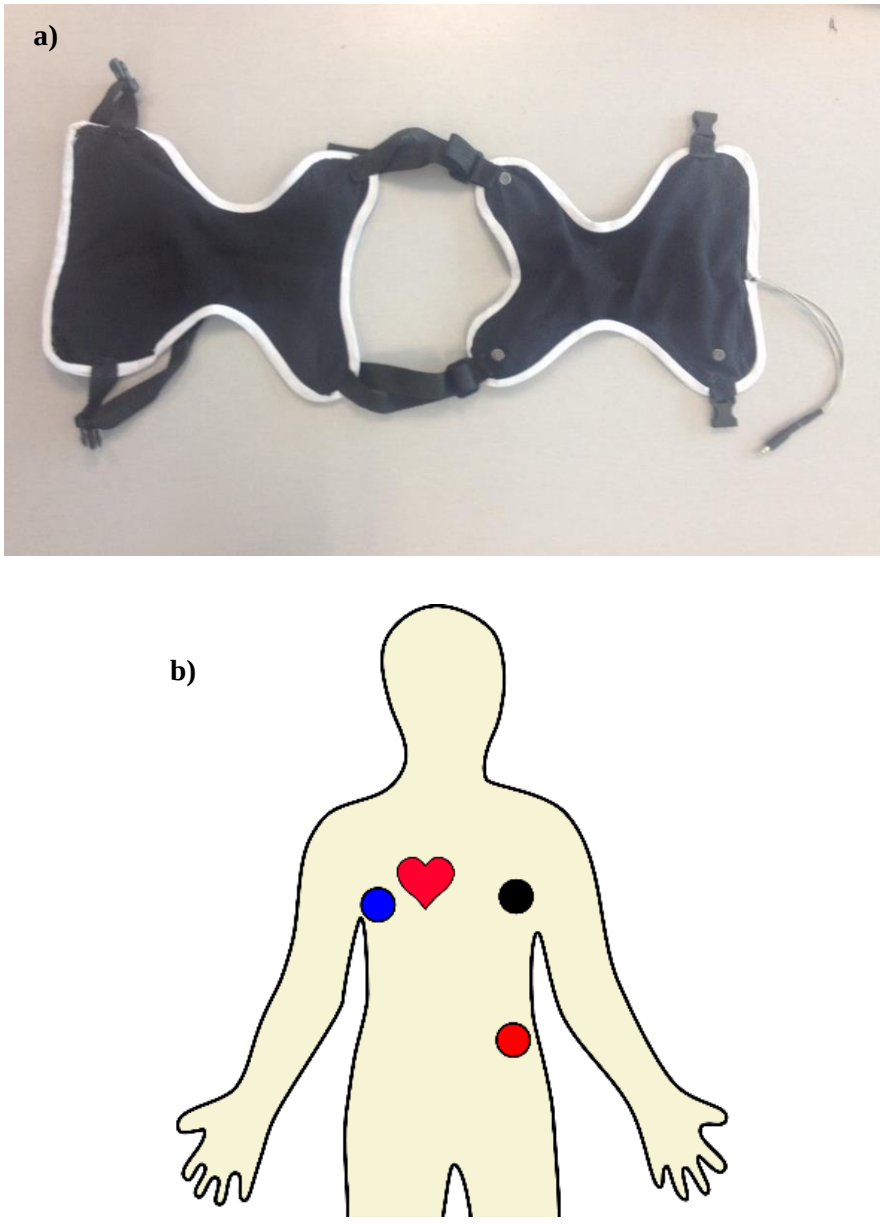


Fig. 3.2. a) Chaleco deportivo wearable, b) Posición de los electrodos

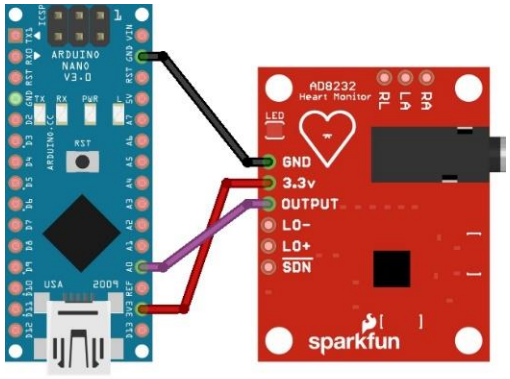


Fig. 3.3. Conexión de la tarjeta AD8232
SparkFun Single Lead Hear Rate Monitor con el Arduino Nano

Una de las aplicaciones del circuito integrado AD8232 es la de monitor ECG wearable debido a su pequeño tamaño y bajo consumo de energía, este circuito es capaz de eliminar los artefactos por el movimiento del paciente implementando un filtro pasa altas de dos polos con una frecuencia de corte de 0.5 Hz, seguido de un filtro pasa bajas de dos polos con una frecuencia de corte de 40 Hz. Esta etapa de filtrado también está configurada para una ganancia de 11, resultando un total de 1100 de ganancia del sistema.

La salida de la tarjeta AD8232, se recibe en la entrada analógica del Arduino A0, el código para recibir los datos en el arduino es el siguiente.

```
1. const int analogInPin0 = A0; //declaración del puerto a utilizar
2. char outStrECG[5]; //variable para enviar el dato en un formato valido
3.                    //para ser leído por el dispositivo android
4. int ECG=0; //variable para almacenar el valor del adc
5.
6. void readSensors() //función para leer el valor del ECG
7. {
8.   ECG = analogRead(analogInPin0);
9.   sprintf(outStrECG,"%04d",ECG);
10.  }
```

3.2. Saturación de oxigeno SpO_2

Para adquirir la señal de SpO_2 y el pulso cardiaco se utiliza un sensor de hardware abierto. El sensor obtiene la señal comparando la cantidad de luz absorbida por la sangre; dependiendo de la cantidad de oximoglobina y deoxihomoglobina presente, la cantidad de luz absorbida también cambia. Usando este cambio el sensor de pulso puede encontrar la saturación de oxígeno. El diagrama de conexión se muestra en la fig. 3.4.

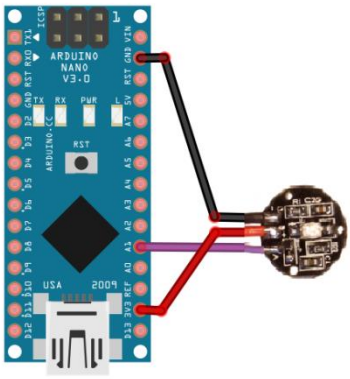


Fig. 3.4. Conexión Arduino Nano Generico y sensor de S_pO_2.

Para calcular la frecuencia cardiaca, la señal es grabada y analizada en el teléfono inteligente cada 60 segundos. La señal periódica incrementa rápidamente a un valor máximo para luego decrementar; la frecuencia cardiaca se calcula utilizando el pico más alto de la onda y el tiempo de muestreo.

La señal de salida del sensor de SpO_2 es capturada por la entrada analógica A1 del Arduino Nano, el código es el siguiente.

```
1. const int analogInPin1 = A1; //declaración del puerto a utilizar
2. char outStrSPO2[5]; //variable para enviar el dato en un formato valido
3.                     //para ser leído por el dispositivo android
4. int SPO2=0; //variable para almacenar el valor del adc
5.
6. void readSensors() //función para leer el valor del SPO2
7. {
8.   SPO2 = analogRead(analogInPin1);
9.   sprintf(outStrSPO2,"%04d", SPO2);
10.  }
```

3.3. Presión arterial

La señal de presión arterial se obtiene con un baumanómetro común, pero el manómetro fue remplazado con el sensor de presión MPVZ5050GWTU de Freescale Semiconductors (Arteta et al., 2012). Cuando se infla el brazalete, los valores de la señal empiezan a incrementar, después el aire es liberado y los valores de la señal empiezan a decrecer; cuando un ligero incremento sucede seguido de un decremento de la señal, el valor de la presión sistólica es obtenido. La señal continúa decreciendo y nuevamente un diminuto incremento es observado seguido de un decremento en la señal; cuando este cambio sucede, el valor de la presión diastólica es encontrado. Finalmente, para reducir posibles artefactos por movimientos, la señal es inicialmente pre filtrada usando el filtro de la mediana (Ball-Llovera et al., 2003). La fig. 3.5 muestra la conexión con el Arduino Nano Genérico.

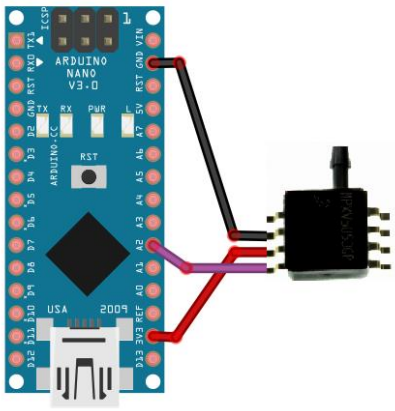


Fig. 3.5. Conexión del Arduino Nano Generico
con el sensor de presión MPVZ5050GWTU

La señal de salida del sensor de presión arterial es capturada por la entrada analógica A2 del Arduino Nano, el código es el siguiente.

```
1. const int analogInPin2 = A2; //declaración del puerto a utilizar
2. char outStrBP[5]; //variable para enviar el dato en un formato valido
3.                   //para ser leído por el dispositivo android
4. int BP =0 ; //variable para almacenar el valor del adc
5.
6. void readSensors() //función para leer el valor del SPO2
7. {
8.   BP = analogRead(analogInPin2);
9.   sprintf(outStrSPO2,"%04d", SPO2);
10.  }
```

3.4. Temperatura corporal

La temperatura corporal se adquiere por medio del sensor de temperatura LM35, el cual está unido al chaleco deportivo; el sensor puede dar una salida linealmente proporcional a la temperatura Celsius. El sensor no requiere calibración externa para poder proveer un rango de temperatura entre -55 y 150 °C. La conexión del LM35 con el Arduino Nano Genérico es mostrada en la fig. 3.6.

La señal de salida del sensor de presión arterial es capturada por la entrada analógica A3 del Arduino Nano, el código es el siguiente.

```
1. const int analogInPin2 = A3; //declaración del puerto a utilizar
2. char outStrTEMP[5]; //variable para enviar el dato en un formato valido
3.                   //para ser leído por el dispositivo android
4. int TEMP =0 ; //variable para almacenar el valor del adc
5.
6. void readSensors() //función para leer el valor del SPO2
```

```
{
  TEMP = analogRead(analogInPin3);
  sprintf(outStrTEMP,"%04d", TEMP);
}
```

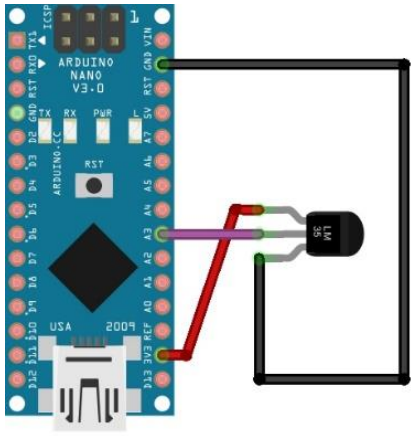


Fig. 3.6. Conexión del Arduino Nano Generico
con el sensor de temperatura LM35.

3.5. Arduino Nano Genérico

La tarjeta Arduino Nano está basada en el MCU ATMEGA328P-PU del fabricante ATMEL; esta tarjeta se usa para recibir todas las señales antes mencionadas usando su canal A/D (convertidor analógico-digital). La señal de ECG tiene una frecuencia en un rango de 10 Hz a 17 Hz, debido a que esta señal tiene la frecuencia más alta con respecto a las otras señales, esta señal restringe la frecuencia de muestreo del Arduino Nano genérico. De acuerdo con el teorema de Nyquist, la frecuencia de muestreo debe de ser más alta que la frecuencia de la señal (Pollock et al., 1999); por esta razón se requiere una frecuencia de al menos 34 Hz.

La resolución del Arduino Nano genérico es de 10 bits, y la frecuencia de muestreo fue configurada a 125 kHz, debido a que el Arduino Nano genérico sólo tiene un A/D el cual tiene que ser multiplexado para adquirir todos los parámetros biomédicos previamente descritos, es decir, se repartirá el tiempo de uso del conversor entre las entradas siempre cumpliendo el teorema de Nyquist. Para ello es necesario saber los tiempos críticos para cada medida. Para el caso de la frecuencia cardiaca, se desea detectar el pulso, la frecuencia que tiene la señal es la generada por el pulso humano, siendo este de 40 a 250 pulsos por minuto en sus casos más extremos, por lo tanto hay un latido cada 240 milisegundos (en el peor caso), es decir, una frecuencia de 4.17 Hz. Por lo tanto, se necesita tomar muestras cada 120 milisegundos. Por último, la medición de la temperatura corporal es otra medida que usa el conversor A/D. Esta medida no varía rápidamente en el tiempo, por lo que no es necesaria una frecuencia muy alta de

muestreo. Se ha considerado que una muestra por segundo es suficiente para esta medida.

3.6. Modulo Bluetooth HC-06.

Para comunicar el Arduino Nano Genérico con el Smatphone se utiliza el módulo Bluetooth HC-06, el cual es manufacturado por Wavesen Industries. Este módulo tiene una memoria FLASH de 8 Mbit y puede operar utilizando bajo voltaje en un rango de 3.1 hasta 4.2 V; también cuenta con un transmisor digital inalámbrico, así como con una potencia de emisión de 3 dBm. La fig. 3.7 muestra la conexión del módulo HC-06 con el Arduino Nano Genérico.

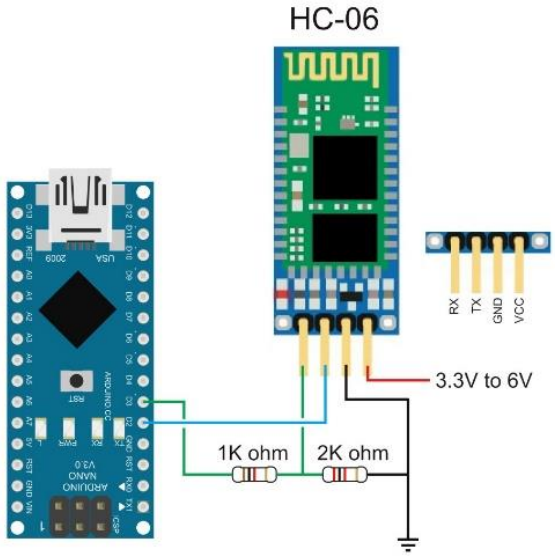


Fig. 3.7. Conexión del módulo bluetooth HC-06 con el Arduino Nano genérico.

Para poder enviar datos desde el Arduino Nano genérico se utiliza la siguiente función.

```
1. void sendAndroidValues(){
2.  BT.print("*"); //indentificador para marcar el inicio del String
3.  BT.write(outStrECG);  //valor de la variable de Electrocardiograma
4.  BT.write(outStrSPO2); //valor de la variable de saturación de oxigeno
5.  BT.write(outStrBP);   //valor de la variable de presión arterial
6.  BT.write(outStrTEMP); // valor de la temperatura
7. }
```

3.7. Desarrollo de una cubierta plástica para proteger el monitor de signos vitales de bajo costo.

Para mantener el dispositivo desarrollado protegido de caídas, golpes y polvo una cubierta en impresión 3D fue diseñada. La impresora implementada fue la da Vinci 1.0, la cual utiliza fabricación de filamentos fusionados; el material del filamento es acrilonitrilo

butadieno estireno (ABS) y tiene una resolución de hasta 0.1 mm. La fig 3.8 muestra la impresora 3D implementada.

Fig. 3.8 Impresora 3D utilizada para el desarrollo
de la cubierta del dispositivo.

El diseño fue realizado utilizando la herramienta de software libre FreeCAD. La cubierta plástica fue diseñada en dos partes, inferior y superior respectivamente. La fig 3.9, fig 3.10 y fig 3.11 muestra diferentes vistas del diseño.

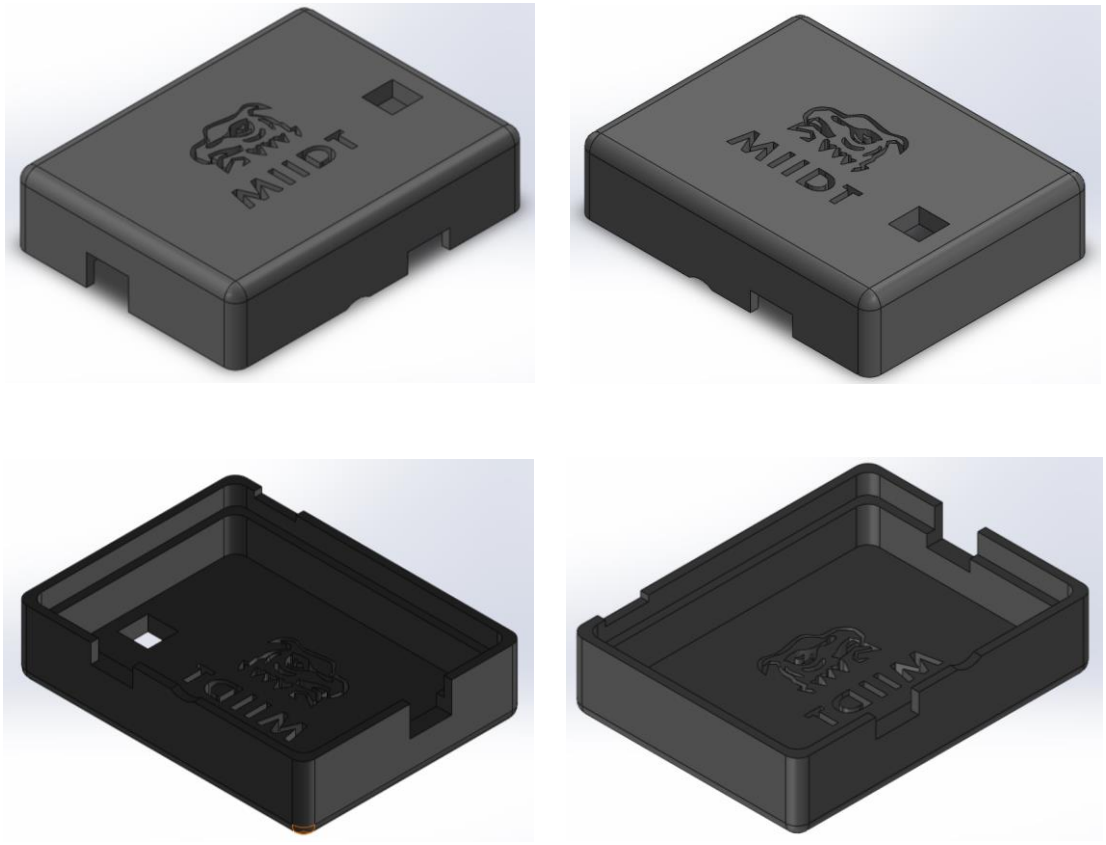


Fig. 3.9 Diferentes vistas de la parte superior de la cubierta.

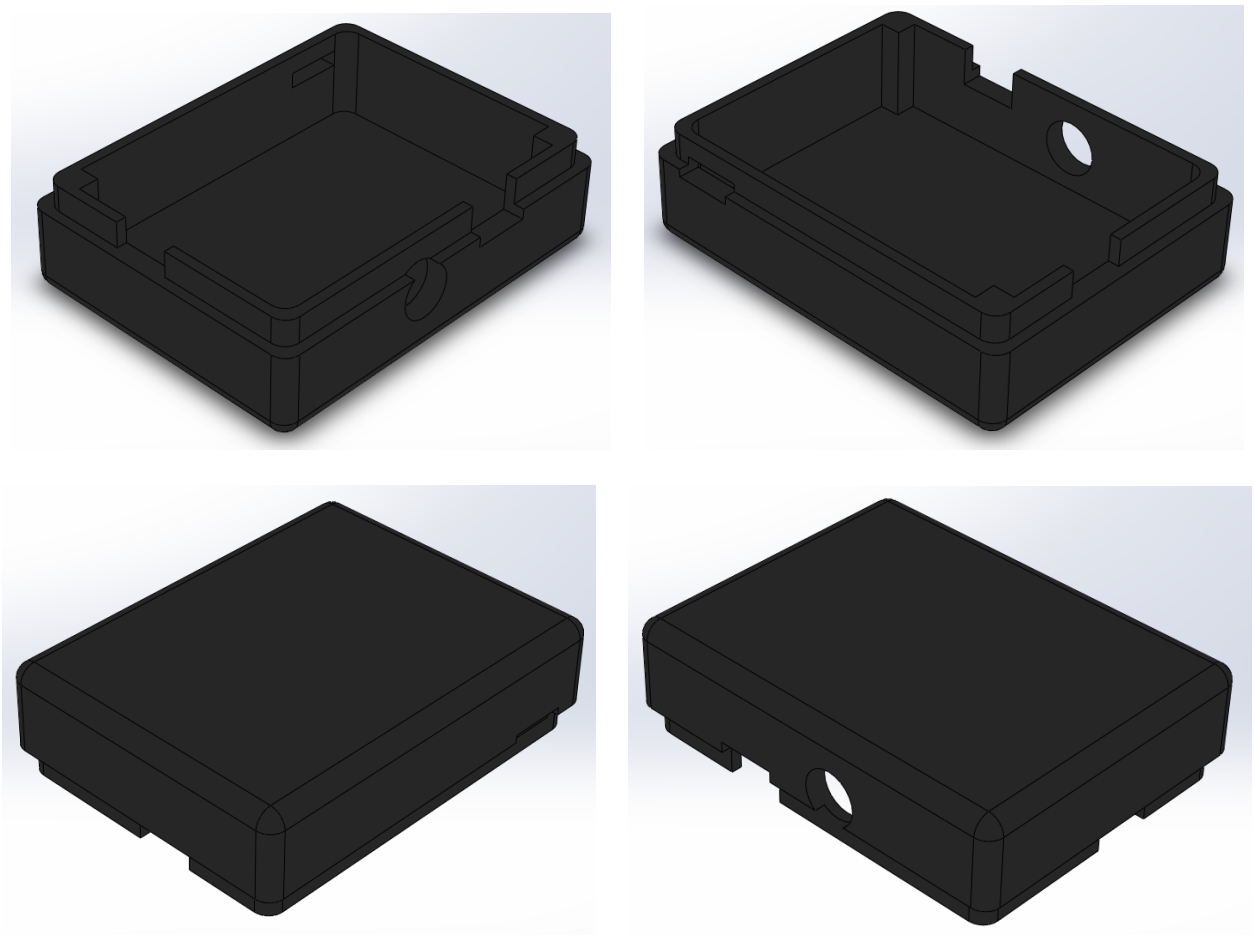


Fig. 3.10 Diferentes vistas de la parte inferior de la cubierta.

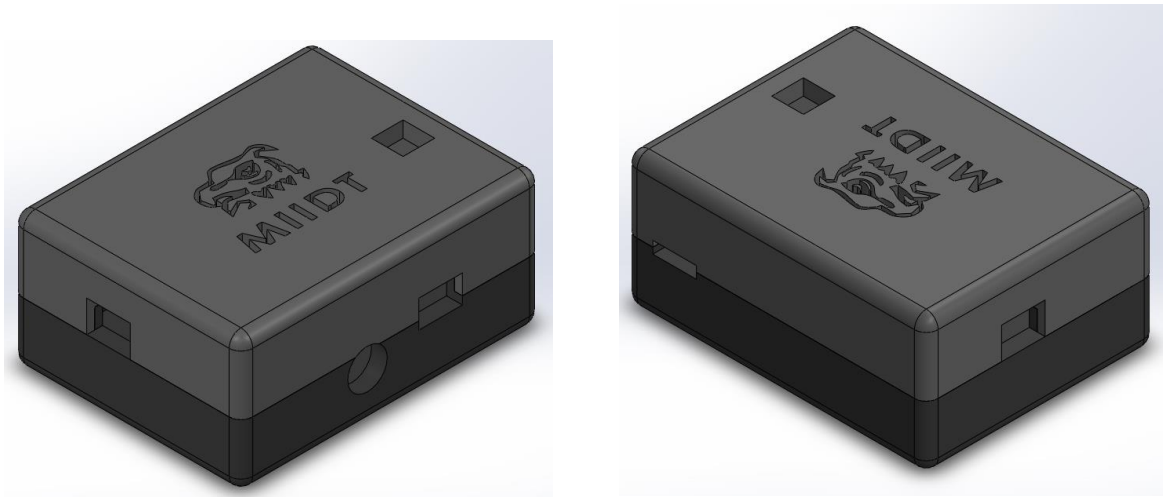


Fig. 3.11 Diferentes vistas del ensable de la parte superior e inferior.

3.8. Desarrollo de un accesorio para el Smartphone para poder tomar fotografías del dedo índice para el análisis de glucosa.

Otro accesorio que se diseñó para su posterior manufactura en impresión 3D fue el utilizado para poder obtener fotografías confiables sin variabilidad de luz. Dichas fotografías serán utilizadas para hacer un análisis de imágenes, en el apartado 3.10 se hablara a detalle del proceso y de la utilización del accesorio. La fig 3.12 muestra diferentes vistas del accesorio diseñado.

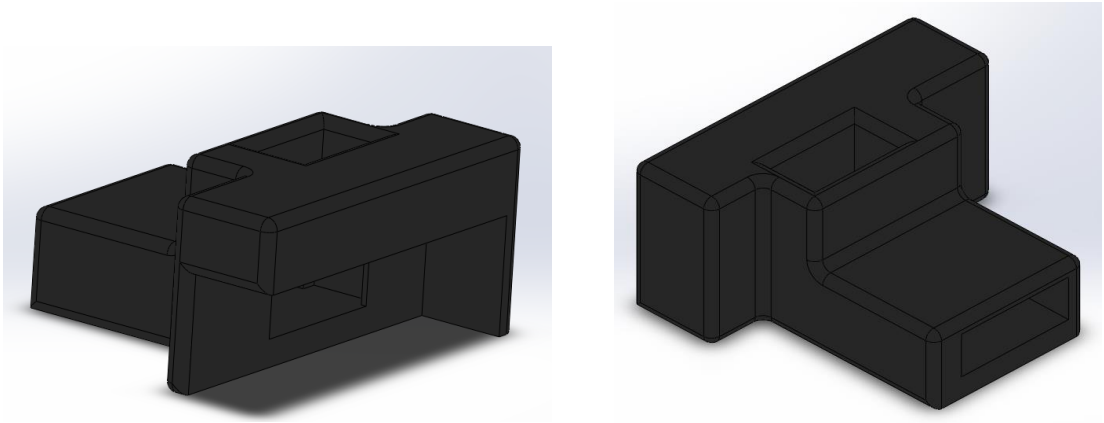


Fig. 3.12 Accesorio diseñado para la toma de fotografías confiables para el análisis de glucosa.

3.9. Smartphone con Sistema Operativo Android (HUAWEI Y330).

El Smartphone utilizado en el presente trabajo para procesar y visualizar los parámetros biomédicos mencionados fue el HUAWEI Ascend Y330, este modelo fue elegido debido a su bajo costo y tener recursos limitados en comparación con dispositivos de gama alta, de esta forma se asegura que la aplicación puede correr en la mayoría de dispositivos Android. La Tabla 3.1 muestra las características del Smartphone utilizado.

Tabla 3.1.
Características del Smartphone
(Telcel, 2014)

Modelo:	HUAWEI Ascend Y330	
Sistema operativo:	Android 4.2 + EMUI 2.0 Lite	

Tabla 3.1.
Características del Smartphone
Continuación… (Telcel, 2014)

Banda:	HSPA+: 21Mbps // UMTS: 900/2100MHz // GSM: 850/900/1800/1900 MHz
Dimensiones:	122.1 x 63.5 x 11.3 mm
Pantalla:	4" WVGA LCD, Resolución: 800 x 480 pixels (10,16 cm)
Batería:	1500mAh Li-Ion // Tiempo en reposo: 350 horas // Tiempo conversación: 400 min
Memoria interna:	512MB RAM + 4 GB ROM (2GB disponible)
Memoria expansión:	MicroSD ampliable hasta 32GB
Conectividad:	WiFi b/g/n // Bluetooth 4.0 HS // GPS/A-GPS // USB 2.0
Cámara:	3.0MP FF // Vídeo: 640x480
Peso:	126 gr.
Procesador:	ARM Cortex A7, Dual Core 1.3 GHz
Otros:	Mini SIM

3.10. Niveles de Glucosa utilizando análisis de imágenes.

Para determinar los niveles de glucosa, se modificó un experimento basado en espectroscopia de infrarrojo cercano (Dantu et al., 2014, Scully et al., 2012, Alonso et al., 2010). La ley de Beer-Lambert mostrada en la ec. 3.1 determina la cantidad de un material utilizando espectroscopia de infrarrojo cercano. La absorción de luz por un material es proporcional a la cantidad de material (Acosta et al., 2006).

$$Absorption = \left(\frac{intensity\ of\ incident\ light}{intensity\ of\ transmitted\ light}\right) \qquad \text{(Ec. 3.1)}$$

Un histograma de una imagen cuantifica los valores de brillo/intensidad. La información global con respecto a la imagen está contenida en el histograma. En algunos casos, el histograma o información derivada de él como su desviación estándar o el promedio de

la intensidad, puede ser utilizado para realizar análisis de imágenes (Dawson-Howe, 2014).

La cámara del Smartphone tiene una resolución de 5 megapíxeles y se usa como foto detector. Para medir los niveles de glucosa, un haz de luz, pasa a través de la punta de un dedo, el cual es colocado en la lente de la cámara del Smartphone. La aplicación es capaz de controlar la cámara para tomar fotografías de una dimensión de 480x854 pixeles; las fotografías son analizadas con ayuda de las librerías de OpenCV para Android, para asegurar fotografías confiables, un pequeño accesorio para el Smartphone fue manufacturado en impresion 3D. La fig. 3.13 muestra el accesorio manufacturado.

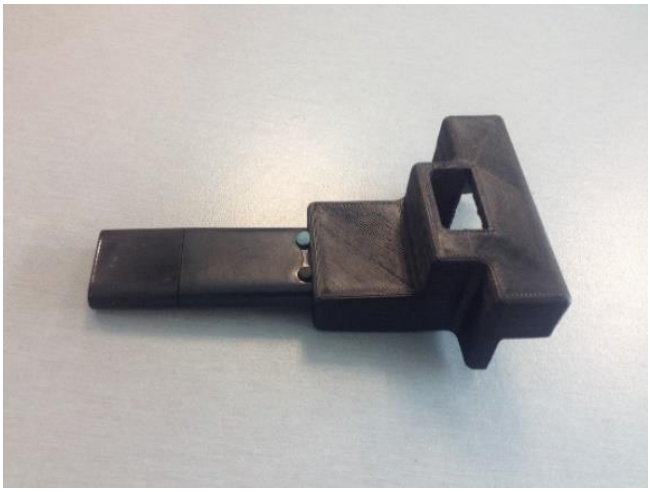


Fig. 3.13 Accesorio impreso en 3D para tomar fotografías y calcular niveles de glucosa.

El análisis digital de imágenes consiste en determinar un cambio en la intensidad de color en el canal azul cuando hay un cambio en la concentración de glucosa en la sangre, este canal fue utilizado debido a que las características de la imagen en este rango espectral hay contenida información relevante.

El dispositivo Android toma 5 fotografías, después el algoritmo utiliza el canal azul para obtener los histogramas de cada imagen y después estimar un promedio de los histogramas para así obtener un solo histograma. Una vez que el promedio de los histogramas es obtenido, estos datos son filtrados usando un filtro pasa bajas de segundo orden con una frecuencia de corte de 0.03 H, con el fin de poder analizar la señal como una señal eléctrica y así poder obtener los valores de energía de la señal (Pollock et al., 1999). La fig. 3.14 muestra el diagrama del proceso de adquisición de niveles de glucosa.

La aplicación Android fue desarollada en lenguaje Java con el fin de tener los beneficios de una aplicación nativa para android, lo cual permite que la aplicación tenga un mejor desempeño, ademas de ser mas ligera. El entorno de desarrollo utilizado fue Android Studio, el cual es la herramienta oficial de Google para el desarrollo de aplicaciones

Android. Esta herramiente permite que el desarrollo de aplicaciones nativas android sea mas sencillo ademas de ofrecer herramientes de control de versiones y. La fig. 3.15 muestra el diagrama de flujo de la Aplicación Android.

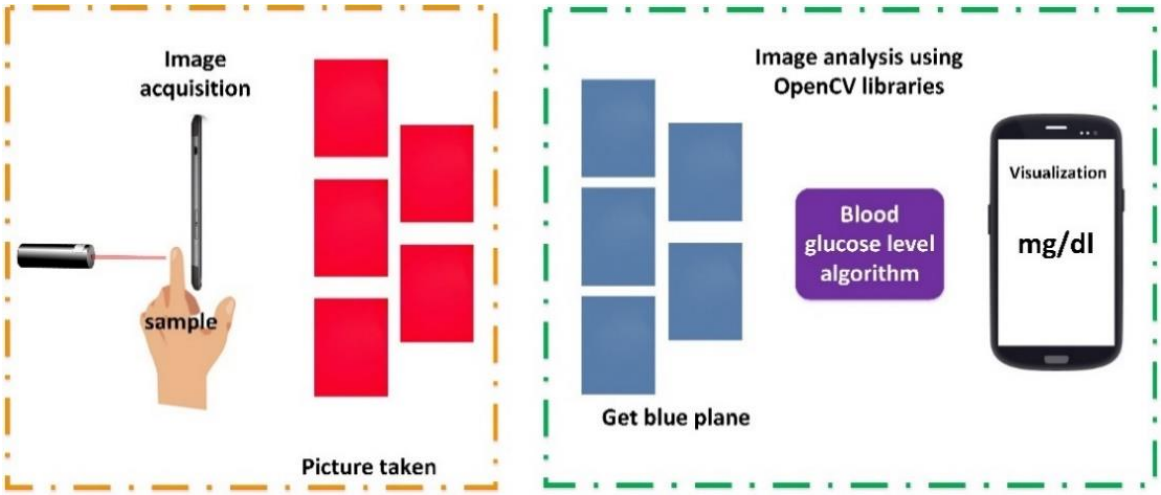


Fig. 3.14. Diagrama general del sistema para adquirir los niveles de glucosa.

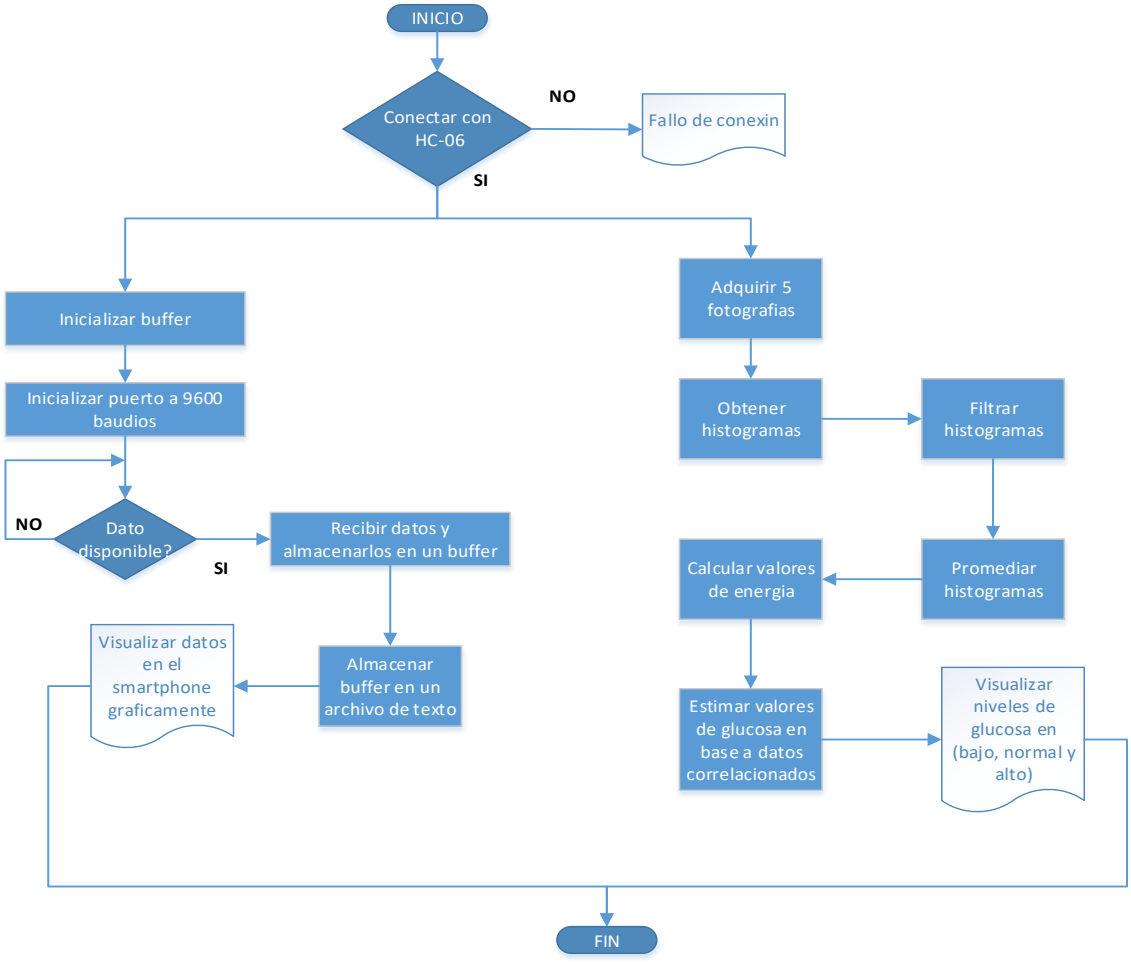


Fig. 3.15 Diagrama de flujo de la aplicación Android.

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1. Resultados

En este trabajo se integró satisfactoriamente la adquisicion, visualizacion y almancenamiento de las señales biomédicas mencionadas en capítulos anteriores, en un sistema wearable, el cual permite al paciente desarrollar sus actividades diarias. Los sensores pueden ser fácilmente conectados al dispositivo desarrollado, y el paciente puede escoger qué sensores conectar, proviendo al sistema versatilidad.

El sistema desarrollado consume 138.6 mW y es alimentado usando una pila recargable LiPo de 3.7 V y 3800 mAh capaz de operar durante 8 horas, el programa del microcontrolador es de apenas 5.5 kB el cual representa el 18% de la memoria total de programa, la fig. 4.1 muestra el pequeño dispositivo portatil que fue desarrollado, el cual tiene una tamaño de 58 mm x 76 mm x 34 mm, y un peso de 96 g y cabe en la palma de una mano.

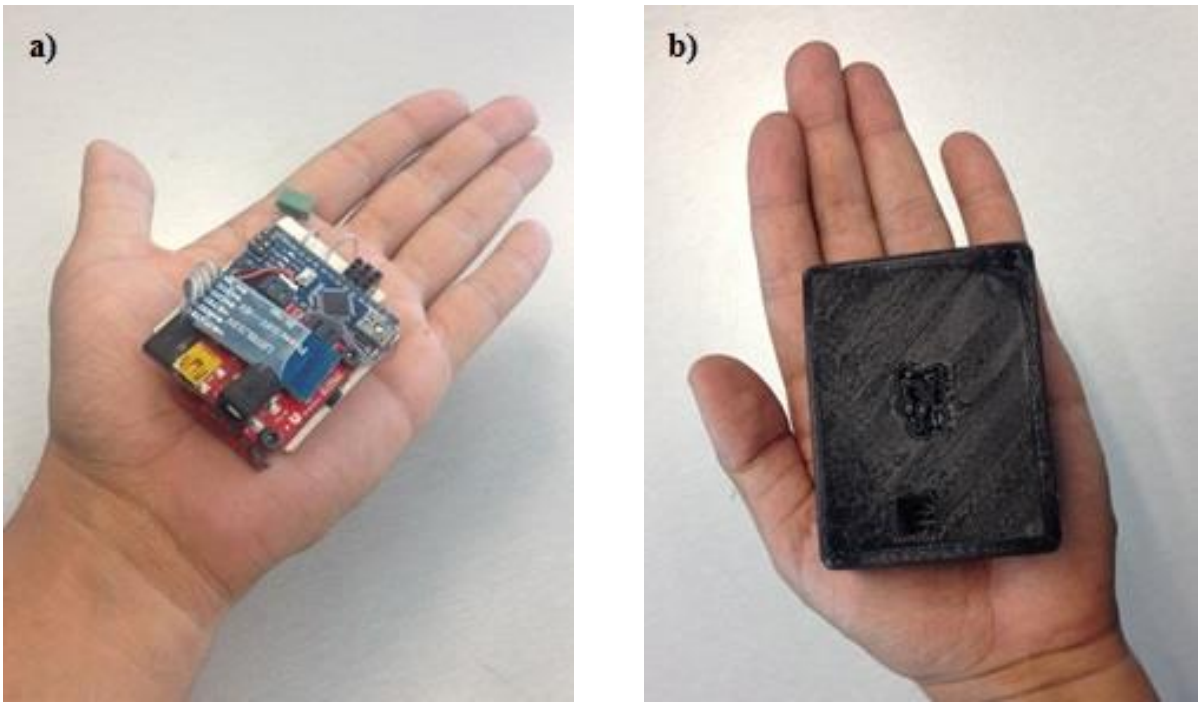


Fig. 4.1 Sistema de adquisicion de datos integrado a) tarjeta de adquisicion de datos con bluetooth y alimentacion con pilas recargables b) Cubierta del dispositivo impreso manufacturado en Impresora 3D.

Los datos del ECG fueron comparados con el monitor de signos vitales comercial Smartwave, el cual es manufacturado por Senolige; ambos exibieron resultados similares, los cuales fueron revisados y validados por un cardiólogo. La fig. 4.2 muestra los datos del ECG almacenados en el smartphone durante un periodo de 8 horas.

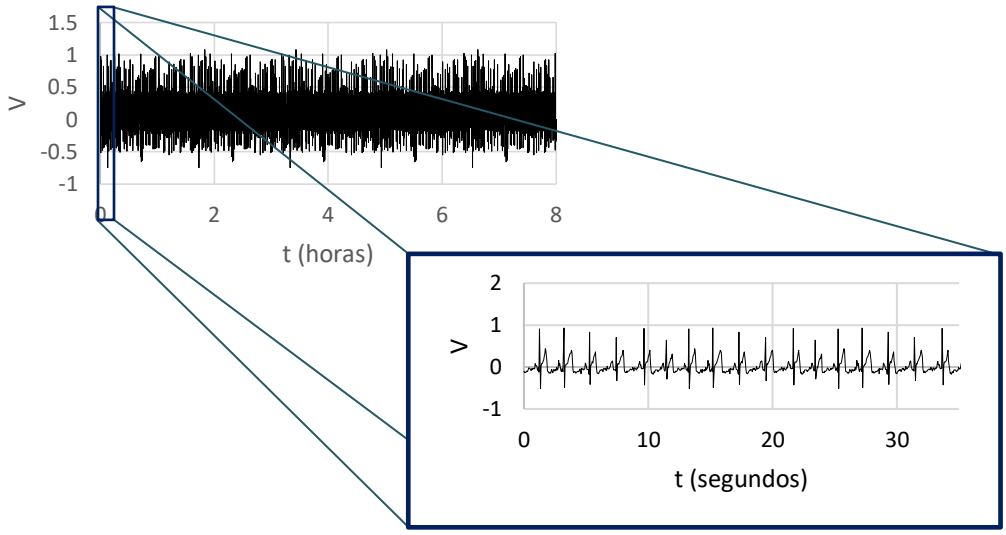


Fig. 4.2 Señal de ECG grabada durante 8 horas.

Los valores de saturación de oxígeno (SpO2), presión arterial y temperatura corporal pueden ser comparados directamente con los obtenidos usando instrumentación biomédica. Para validar estos datos, fueron recolectadas varias muestras usando instrumentación medica estándar, como termómetros, oxímetros, y baumanómetros, respectivamente, y después los datos fueron comparados con los recolectados usando el monitor de signos vitales portátil de bajo costo propuesto en este trabajo. Los resultados mostrados en el Smartphone fueron similares a los resultados provenientes de la instrumentación biomédica, con un error menor al 5%.

La señal de SpO2 fue comparada con los valores obtenidos del oxímetro incluido en el monitor de signos vitales Smartwave 12 mencionado anteriormente, presentando valores similares a los medidos con el monitor portátil de bajo costo. La fig. 4.3 muestra la señal de SpO2 adquirida durante 8 horas.

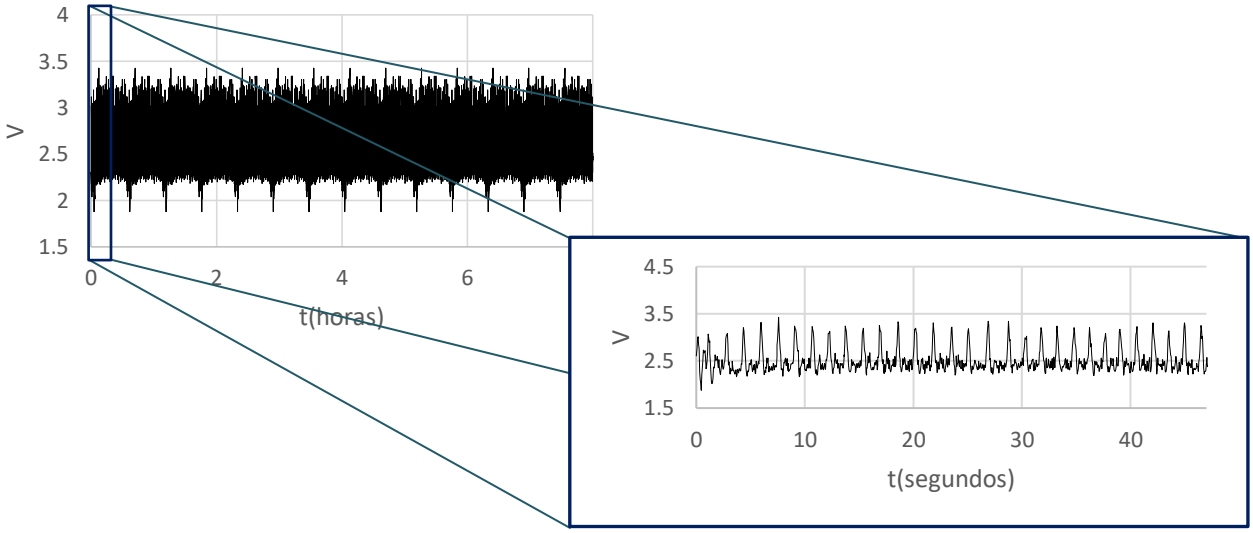


Fig. 4.3 Señal de S_pO_2 grabada durante 8 horas.

Los valores de presion arterial fueron comparados con los que fueron obtenidos usando un baumanometro de mercurio, mostrando valores similares. Para las mediciones de temperatura corporal se utilizó un termómetro de mercurio; los valores mostrados en el smartphone fueron muy cercanos a los del termómetro. En ambos parámetros, en 10 muestras, se registró un error menor al 10%.

Para validar los experimentos realizados para determinar los niveles de glucosa utilizando análisis de imágenes, ver la fig 4.4, los datos obtenidos usando análisis de imágenes, en 10 sujetos de prueba saludables, fueron correlacionados con los resultados de pruebas invasivas estandard de laboratorio para obtener niveles de glucosa. La primera prueba fue realizada en ayunas; en la segunda prueba los sujetos ingirieron “torrejas” (el cual es un postre tradicional en el estado de Guerrero); la tercera prueba fue realizada despues de comer. Una prueba de reproducibilidad demostró que hay un cambio significativo en los histogramas del canal azul extraidos de las fotografías tomadas con el smartphone; los histogramas el canal rojo y verde tambien fueron analizados, sin embargo, no hubo cambios significativos observados. La fig. 4.5 muestra cómo las 5 fotografías tamadas en un sujeto de pruebas, despues de la ingesta de torrejas, presenta histogramas similares; por este motivo, se calculó el promedio de los histogramas para posteriormente tratarlo como señal eléctrica.

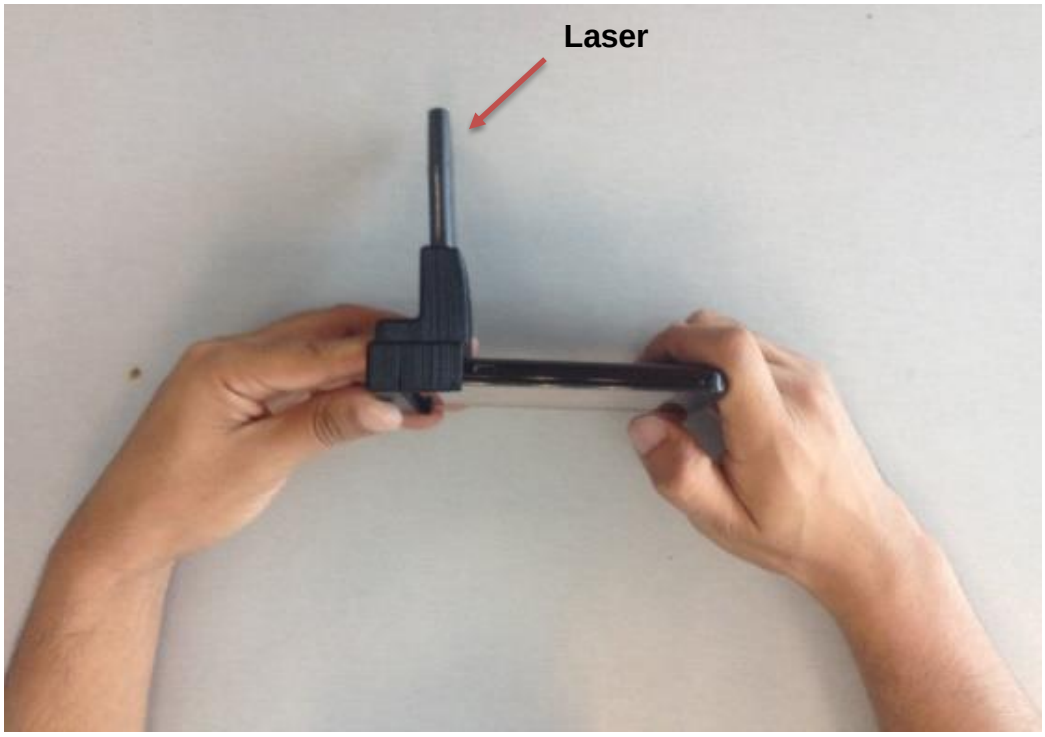


Fig. 4.4. Adquisición de fotografías para el análisis de imágenes.

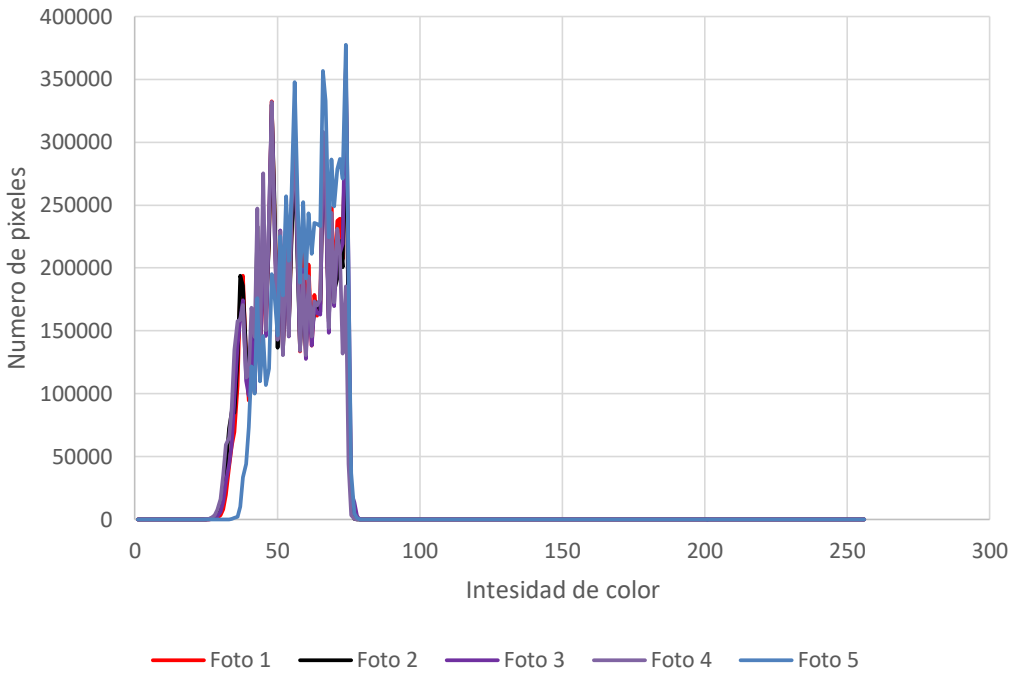


Fig. 4.5 Medición de niveles de glucosa después de la ingesta de “torrejas”.

La fig. 4.6 muestra los histogramas de las 3 mediciones realizadas (ayunas, después de la ingesta de torrejas y después de comer). Puede observarse que en cada medición los histogramas tienen características bien definidas.

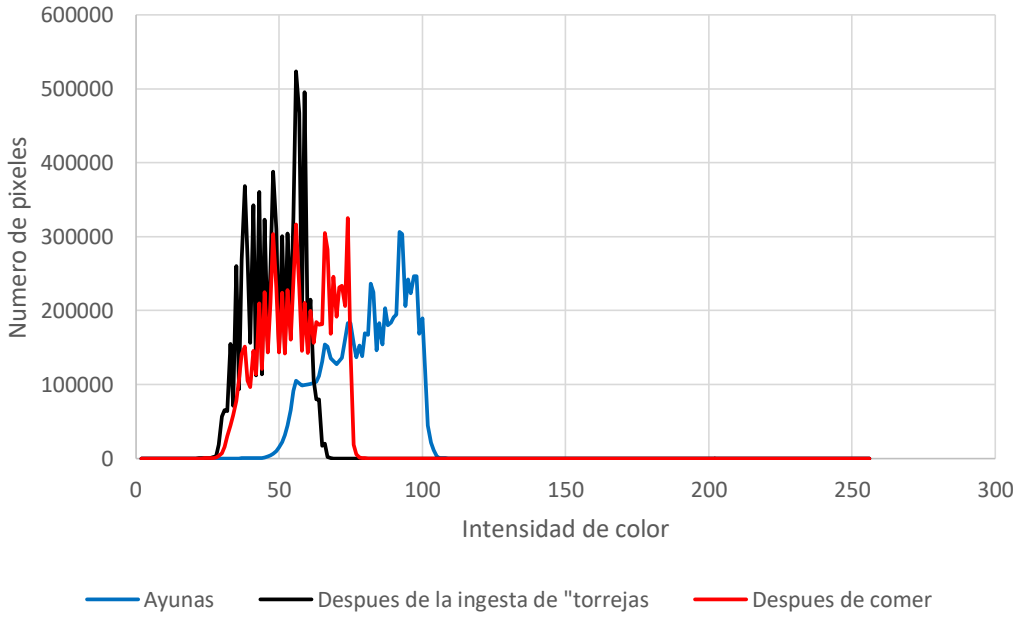


Fig. 4.6. Histogramas de las 3 mediciones (Ayunas, después de la ingesta de torrejas, después de comer).

Una vez que se pudo observar que los histogramas tenían características diferentes y bien definidas, se procedió con el filtrado de los histogramas para poder manejarlos como señales eléctricas y así poder obtener los niveles de energia con el fin de posteriormente compararlos con los niveles de glucosa obtenidos con la prueba invasiva de laboratorio, ver fig. 4.7.

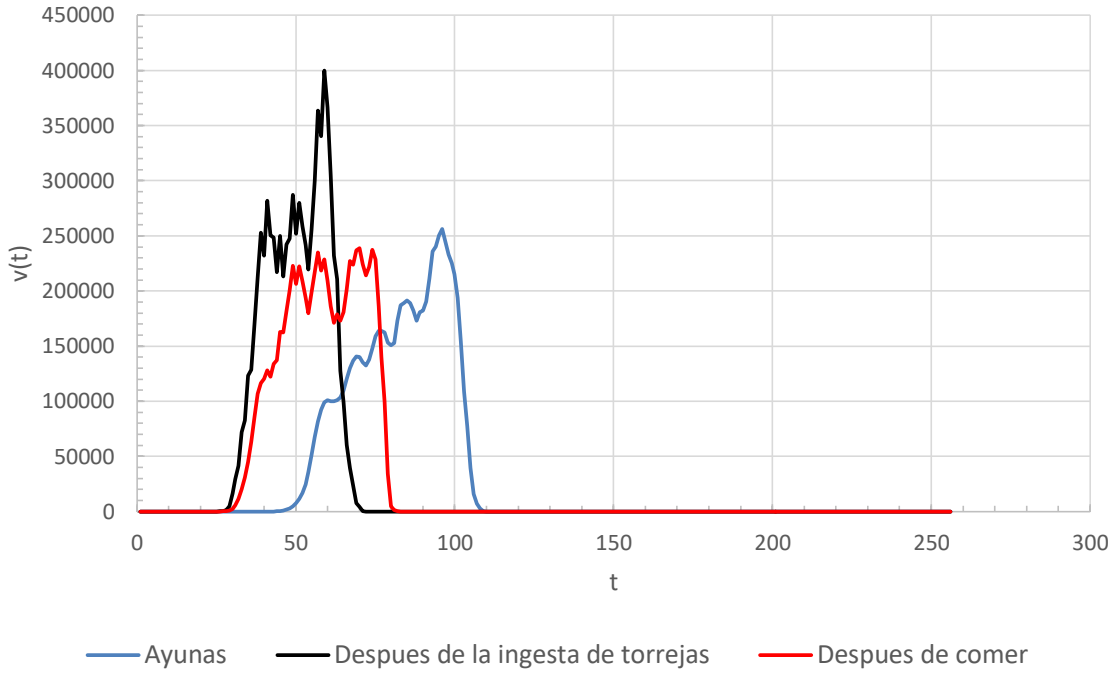


Fig. 4.7. Histogramas analisados como señales electricas.

Los valores de eneriga calculados usando análisis de imágenes muestran cierta tendencia con respecto a los niveles de glucosa. La fig 4.8 muestra la correlación (Ec. 2) entre los valores de energía obtenidos usando análisis de imágenes y los niveles de glucosa obtenidos usando el método invasivo de laboratorio. Usando los valores recollectados de todas las mediciones, el algoritmo puede estimar los valores de glucosa en tres categoria: bajo, normal y alto.

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \qquad (Ec. 4.1)$$

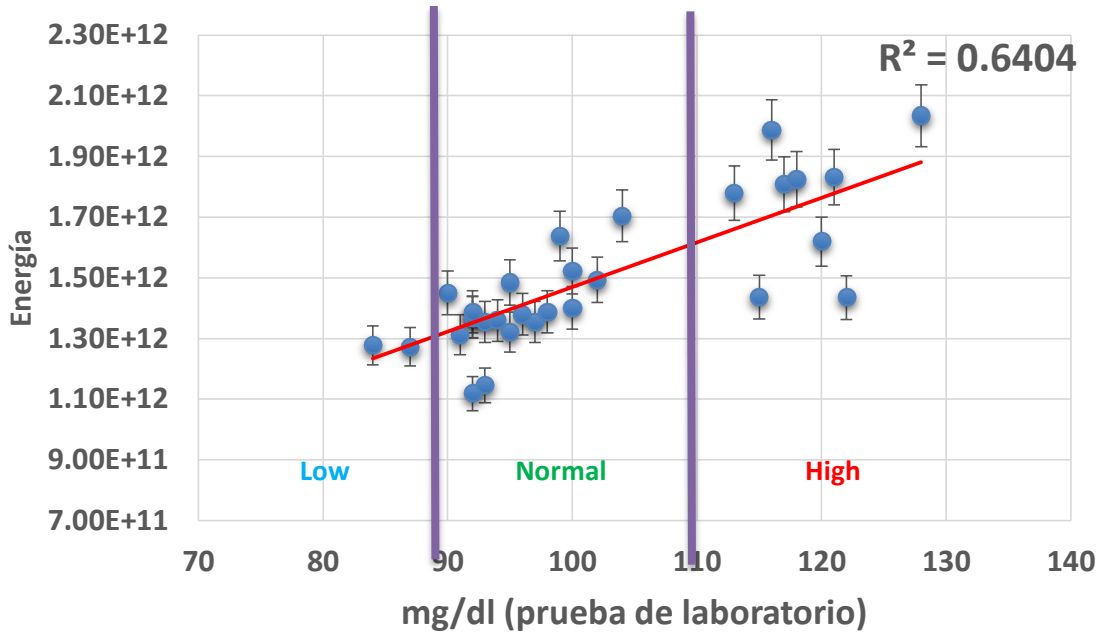


Fig. 4.8. Correlacion entre energia y niveles de glucosa en la sangre.

La aplicacion Android fue desarrollada para operar en conjunto con el dispositivo para el cuidado de la salud wearable por medio de comunicacion Bluetooth. La aplicacion es capaz de mostrar graficamente todas los parametros biomédicos antes mencionados, y tan solo consume 4 MB de la memoria RAM. La fig. 4.9 muestra todo el sistema integrado en operación.

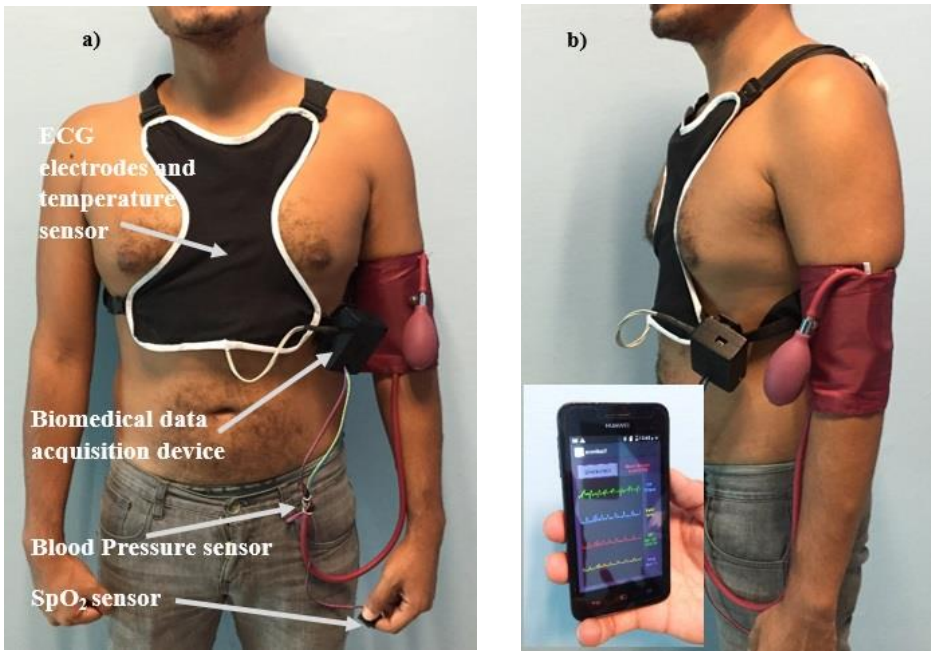


Fig. 4.9 Monitor de Signos Vitales Portatil en operacion a) Vista Frontal y b) vista lateral del sujeto de pruebas usando el sistema.

La aplicacion es facil de usar. Cuando la aplicacion es ejecutada, el paciente sólo necesita escoger el módulo Bluetooth, el cual debió ser emparejado en las configuración del dispositivo Android a utilizar. Una vez que la conexión se realizó, el dispositivo Android empieza la captura de datos provenientes del monitor portatil para posteriormente almacenarlos en un documento de texto. Para estimar los niveles de glucosa, el paciente debe colocar su dedo índice sobre la lente de la cámara del smartphone; al presionar sobre la pantalla del smartphone, se empezarán a calcular los niveles de glucosa y se mostrarán en pantalla. La aplicación puede ser descargada desde los repositorios Git bajo la licencia GNU General Public License v2.0 (http://github.com/ZaicRN/LowCost HealthCareMonitoringSystem). La figura 4.10 muestra las capturas de pantalla de la aplicación Android.

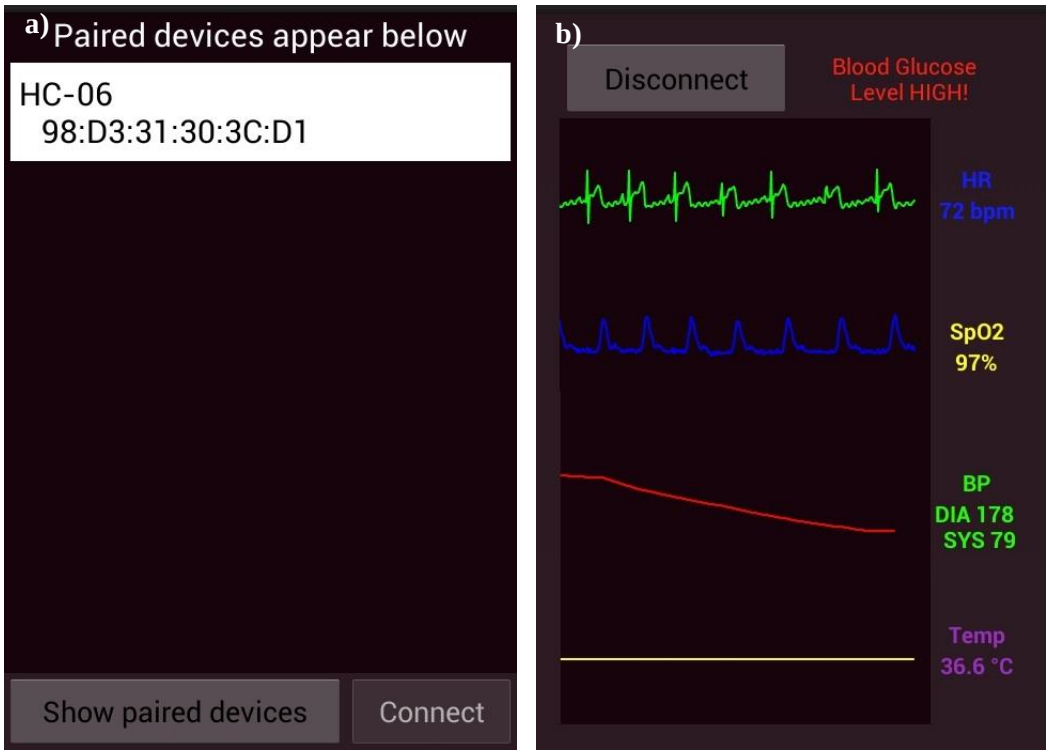


Fig. 4.10 a), b) Capturas de pantalla Android.

4.2. Discusión

El sistema propuesto en este trabajo es capaz de medir 6 parámetros biomédicos por un periodo de más de 8 horas, alimentado con una bateria LiPo de 3800 mAh; este rendimiento representa una mejora en comparación con otros trabajos, los cuales pueden medir sólo hasta 3 parámetros (Mundt et al., 2005, Park et al., 2005, Oliver and Flores-Mangas, 2006, Jin et al., 2009, Fei et al., 2010, Heilman and Porges, 2007). El dispositivo es de un tamaño pequeño, mide 58 mm x 76 mm x 35 mm, y tiene un peso de sólo 96 g, lo cual asegura su portabilidad. El archivo txt generado durante las 8 horas tiene un peso

apenas de 5 MB y puede ser facilmente compartido por Internet; adicionalmente, el sistema puede medir los niveles de glucosa en forma cualitativa, y por primera vez un método no invasivo para obtener los niveles de glucosa en la sangre es integrado a un monitor de signos vitales wearable.

Llevar el brazalete para la medicion de presion arterial puede llegar a ser ligeramente aparatoso; sin embargo, la modularidad del sistema permite conectar o desconectar cualquier sensor, de esta manera no es necesario llevar el baumanómetro todo el tiempo y sólo conectarlo cuando se desea tomar la lactura. Además, si algún sensor deja de funcionar, no afecta el funcionamiento de otros sensores.

Resumiendo, el sistema propuesto en este trabajo representa un buen arquetipo para desarrollar un sistema para el cuidado de la salud de bajo costo y wearable.

CONCLUSIONES

El sistema descrito en esta tesis ofrece una versátil alternativa para monitorear la salud de un paciente, el prototipo portátil es capaz de adquirir el ECG, pulso cardíaco, presión arterial, SpO_2, temperatura y niveles de glucosa en la sangre, y mostrarlos gráficamente en un Smartphone con sistema operativo Android en una aplicación muy intuitiva con las ventajas de un monitor comercial, además de poder ser portátil y poder ser llevado a cualquier sitio.

El dispositivo puede operar cerca de 8 horas continúas adquiriendo los datos biomédicos del usuario, mientras éste realiza sus actividades diarias, y compartir estos datos con un especialista a cualquier hora del día. Esta es una nueva y poderosa forma de vigilar la condición de un paciente, permitiendo a los médicos mejorar el tratamiento y calidad de vida de los pacientes. Además de usar un método no invasivo es más atractivo que los actuales métodos de pinchar el dedo para tomar la lectura con un glucómetro, o un análisis de laboratorio donde tengan que extraer una muestra de sangre. Estos métodos no son placenteros y son dolorosos para ser frecuentemente usados.

Los sistemas para el monitoreo de la salud de bajo costo son importantes. Cuando se trata de enfermedades crónicas, los medicamentos y equipo pueden llegar a ser extremadamente caros. El sistema de monitoreo para la salud propuesto en este trabajo se ajusta de manera perfecta para aquellas personas que no tienen los suficientes recursos económicos para adquirir un monitor de signos vitales comercial.

REFERENCIAS

ACOSTA, G. M., HENDERSON, J. R., HAJ, N. A. A., RUCHTI, T. L., MONFRE, S. L., BLANK, T. B. & HAZEN, K. H. 2006. Compact apparatus for noninvasive measurement of glucose through near-infrared spectroscopy. Google Patents.

ALONSO, G., RAMIREZ, A. & MUNOZ, R. Hardware implementation of the wavelet transform coupled with Artificial Neural Network for quantification purposes. Health Care Exchange (PAHCE), 2010 Pan American, 2010. IEEE, 32-37.

ANALOG DEVICES 2012. Single-Lead, Heart Rate Monitor Front End. *http://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf: AD8232 datasheet*

3-27.

APPLE. 2015. Apple. Available: http://www.apple.com/ios/health/ [Accessed].

ARDUINO. 2015. Arduino. Available: https://www.arduino.cc/en/Guide/Introduction [Accessed].

ARTETA, C., DOMINGOS, J. S., PIMENTEL, M. A., SANTOS, M. D., CHIFFOT, C., SPRINGER, D., RAGHU, A. & CLIFFORD, G. D. 2012. Low-cost blood pressure monitor device for developing countries. *Wireless Mobile Communication and Healthcare.* Springer.

BALL-LLOVERA, A., DEL REY, R., RUSO, R., RAMOS, J., BATISTA, O. & NIUBO, I. An experience in implementing the oscillometric algorithm for the noninvasive determination of human blood pressure. Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE, 2003. IEEE, 3173-3175.

BECERRA-LUNA, B., DÁVILA-GARCÍA, R., SALGADO-RODRÍGUEZ, P., MARTÍNEZ-MEMIJE, R. & INFANTE-VÁZQUEZ, Ó. 2012. Monitor de señales de electrocardiografía y frecuencia cardiaca mediante un teléfono móvil con el protocolo de comunicación Bluetooth. *Archivos de cardiología de México,* 82**,** 197-203.

CASTELLANO, C., DE JUAN, F. P., CASTELLANO, M. A., DE JUAN, M. P. & ATTIE, F. 2004. *Electrocardiografía clínica*, Elsevier.

DANTU, V., VEMPATI, J. & SRIVILLIPUTHUR, S. Non-invasive blood glucose monitor based on spectroscopy using a smartphone. Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE, 2014. IEEE, 3695-3698.

DAWSON-HOWE, K. 2014. *A Practical Introduction to Computer Vision with OpenCV*, John Wiley & Sons.

DEITEL, H. M. & DEITEL, P. J. 2004. *Cómo programar en Java*, Pearson Educación.

EINTHOVEN, W. 1957. The telecardiogram. *American Heart Journal,* 53**,** 602-615.

FEI, D.-Y., ZHAO, X., BOANCA, C., HUGHES, E., BAI, O., MERRELL, R. & RAFIQ, A. 2010. A biomedical sensor system for real-time monitoring of astronauts' physiological parameters during extra-vehicular activities. *Computers in biology and medicine,* 40**,** 635-642.

FREECAD 2016.

GARIBAY RUBIO, C. R., PELÁEZ CORDEIRO, I. J. & CANO RODRIGUEZ, Á. I. 2006. *Primeros Auxilios Basicos*.

GOOGLE. 2015. *Android Studio* [Online]. google. Available: http://developer.android.com/intl/es/tools/studio/index.html [Accessed].

HEILMAN, K. J. & PORGES, S. W. 2007. Accuracy of the LifeShirt®(Vivometrics) in the detection of cardiac rhythms. *Biological psychology,* 75**,** 300-305.

HOLTER, N. J. 1957. Radioelectrocardiography: a new technique for cardiovascular studies. *Annals of the New York Academy of Sciences,* 65**,** 913-923.

JEROUDI, O. M., CHRISTAKOPOULOS, G., CHRISTOPOULOS, G., KOTSIA, A., KYPREOS, M. A., RANGAN, B. V., BANERJEE, S. & BRILAKIS, E. S. 2015. Accuracy of remote electrocardiogram interpretation with the use of Google Glass technology. *The American journal of cardiology,* 115**,** 374-377.

JIN, Z., ORESKO, J., HUANG, S. & CHENG, A. C. HeartToGo: a personalized medicine technology for cardiovascular disease prevention and detection. Life Science Systems and Applications Workshop, 2009. LiSSA 2009. IEEE/NIH, 2009. IEEE, 80-83.

KARIMIPOUR, A. & HOMAEINEZHAD, M. R. 2014. Real-time electrocardiogram P-QRS-T detection–delineation algorithm based on quality-supported analysis of characteristic templates. *Computers in biology and medicine,* 52**,** 153-165.

KIM, B.-H., NOH, Y.-H. & JEONG, D.-U. A Wearable ECG Monitoring System Using Adaptive EMD Filter Based on Activity Status. Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on, 2015. IEEE, 11-16.

MUNDT, C. W., MONTGOMERY, K. N., UDOH, U. E., BARKER, V. N., THONIER, G. C., TELLIER, A. M., RICKS, R. D., DARLING, B., CAGLE, Y. D. & CABROL, N. A. 2005. A multiparameter wearable physiologic monitoring system for space and terrestrial applications. *Information Technology in Biomedicine, IEEE Transactions on,* 9**,** 382-391.

OLIVER, N. & FLORES-MANGAS, F. HealthGear: a real-time wearable system for monitoring and analyzing physiological signals. Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on, 2006. IEEE, 4 pp.-64.

PARK, H. D., LEE, K. J., YOON, H. R. & NAM, H. H. 2005. Design of a portable urine glucose monitoring system for health care. *Computers in biology and medicine,* 35**,** 275-286.

POLLOCK, D. S. G., GREEN, R. C. & NGUYEN, T. 1999. *Handbook of time series analysis, signal processing, and dynamics*, Academic Press.

POURAFKARI, L., GHAFFARI, S. & NADER, N. D. 2016. Incorporating Fragmented QRS on Surface Electrocardiogram to Exercise Stress Test. *Annals of noninvasive electrocardiology: the official journal of the International Society for Holter and Noninvasive Electrocardiology, Inc*.

SCULLY, C. G., LEE, J., MEYER, J., GORBACH, A. M., GRANQUIST-FRASER, D., MENDELSON, Y. & CHON, K. H. 2012. Physiological parameter monitoring from optical recordings with a mobile phone. *Biomedical Engineering, IEEE Transactions on,* 59**,** 303-306.

SECRETARIA DE SALUD 2009. Norma Oficial Mexicana NOM-015-SSA"-2010, para la prevención, tratamiento y control de la diabetes mellitus. Distrito Federal.

SUNG, M., MARCI, C. & PENTLAND, A. 2005. Journal of NeuroEngineering and Rehabilitation. *Journal of neuroengineering and rehabilitation,* 2**,** 0003-2.

TELCEL. 2014. Telcel. Available: http://www.telcel.com/content/telcel/personas/equipos/telefonos-y-smartphones.html [Accessed].

THOMAS, S. S., NATHAN, V., ZONG, C., AKINBOLA, E., AROUL, A. L. P., PHILIPOSE, L., SOUNDARAPANDIAN, K., SHI, X. & JAFARI, R. BioWatch—A wrist watch based signal acquisition system for physiological signals including blood pressure. Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE, 2014. IEEE, 2286-2289.

WELLENS, H. J., BÄR, F. W. & LIE, K. 2000. The value of the electrocardiogram in the differential diagnosis of a tachycardia with a widened QRS complex. *Professor Hein JJ Wellens.* Springer.

WHO. 2015. World Health Organization. Available: http://www.who.int/topics/chronic_diseases/es/ [Accessed].

INDICE DE TABLAS, FIGURAS Y ECUACIONES

Tablas

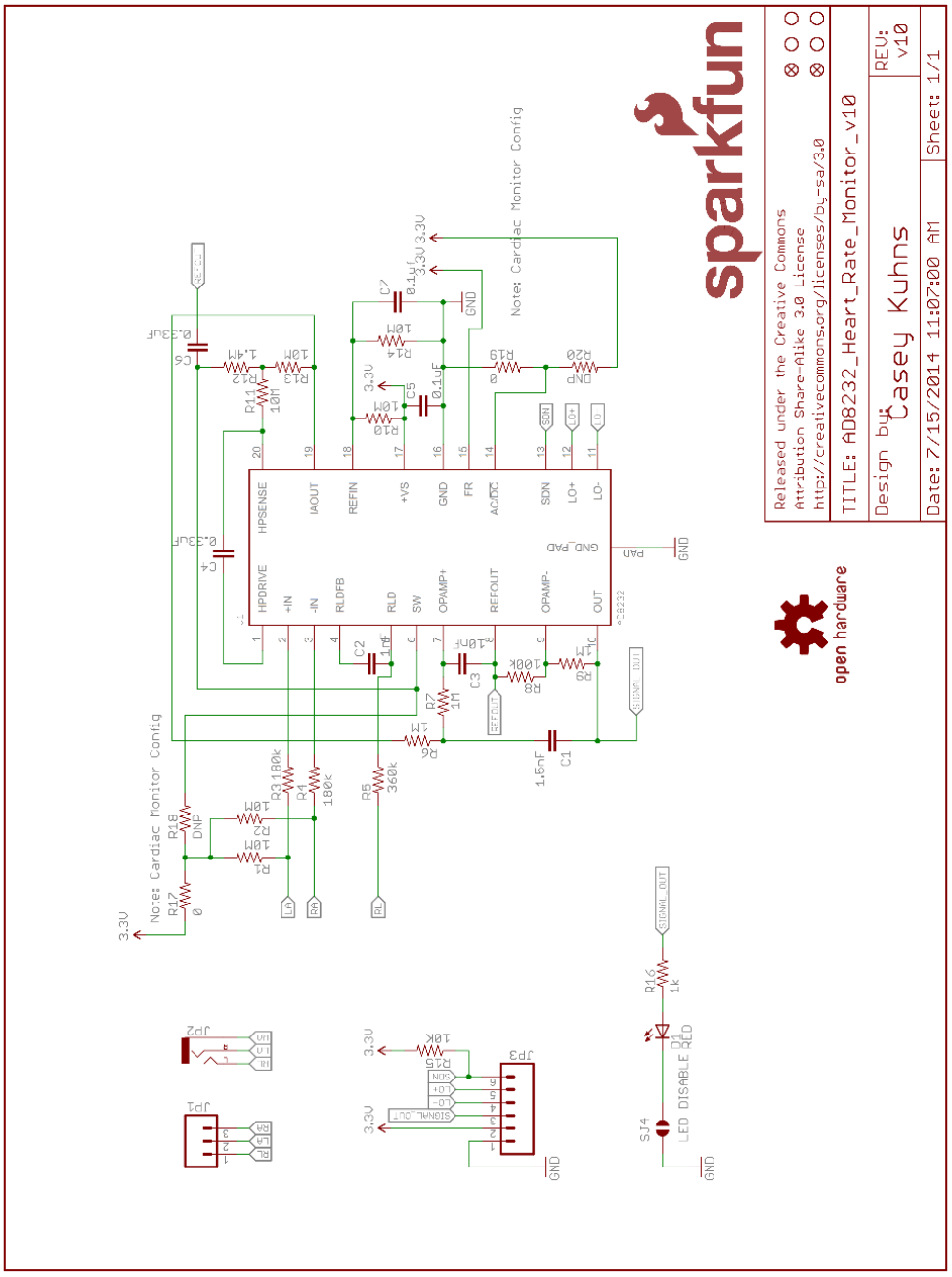
Figuras

ECUACIONES

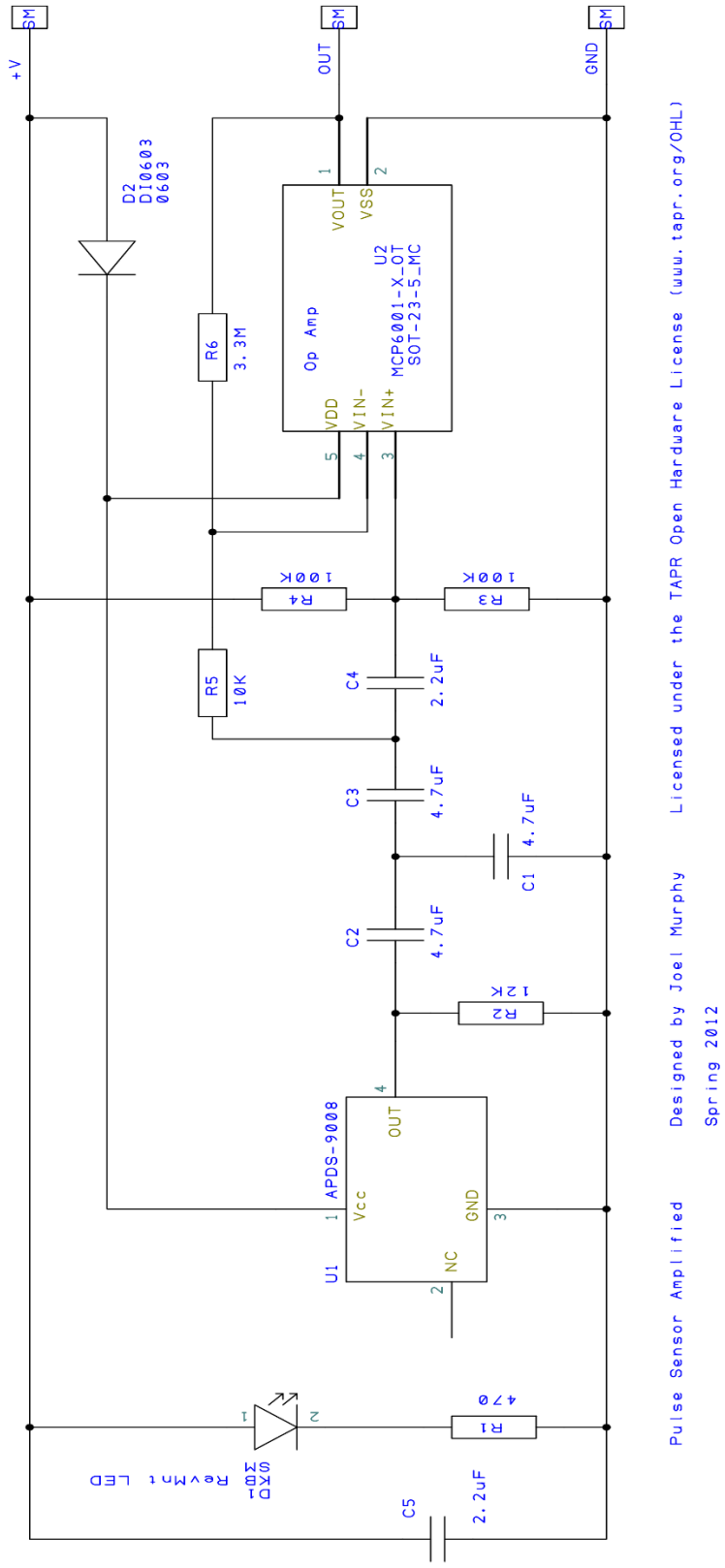
ANEXOS

A. 1. Planos

Esquema eléctrico de la Tarjeta AD823 Heart Rate Monitor V10

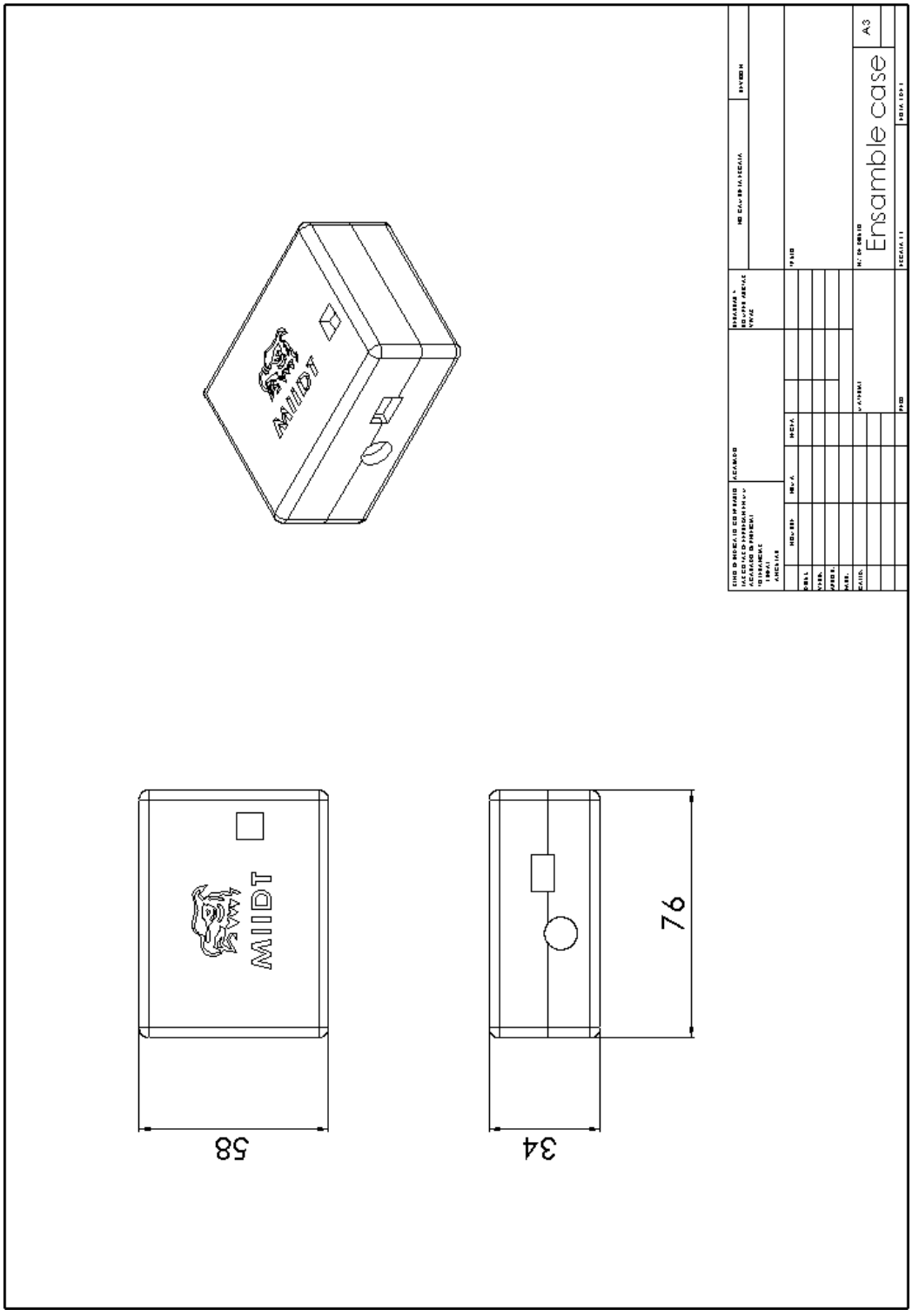


Esquema eléctrico del Sensor de S_pO_2.

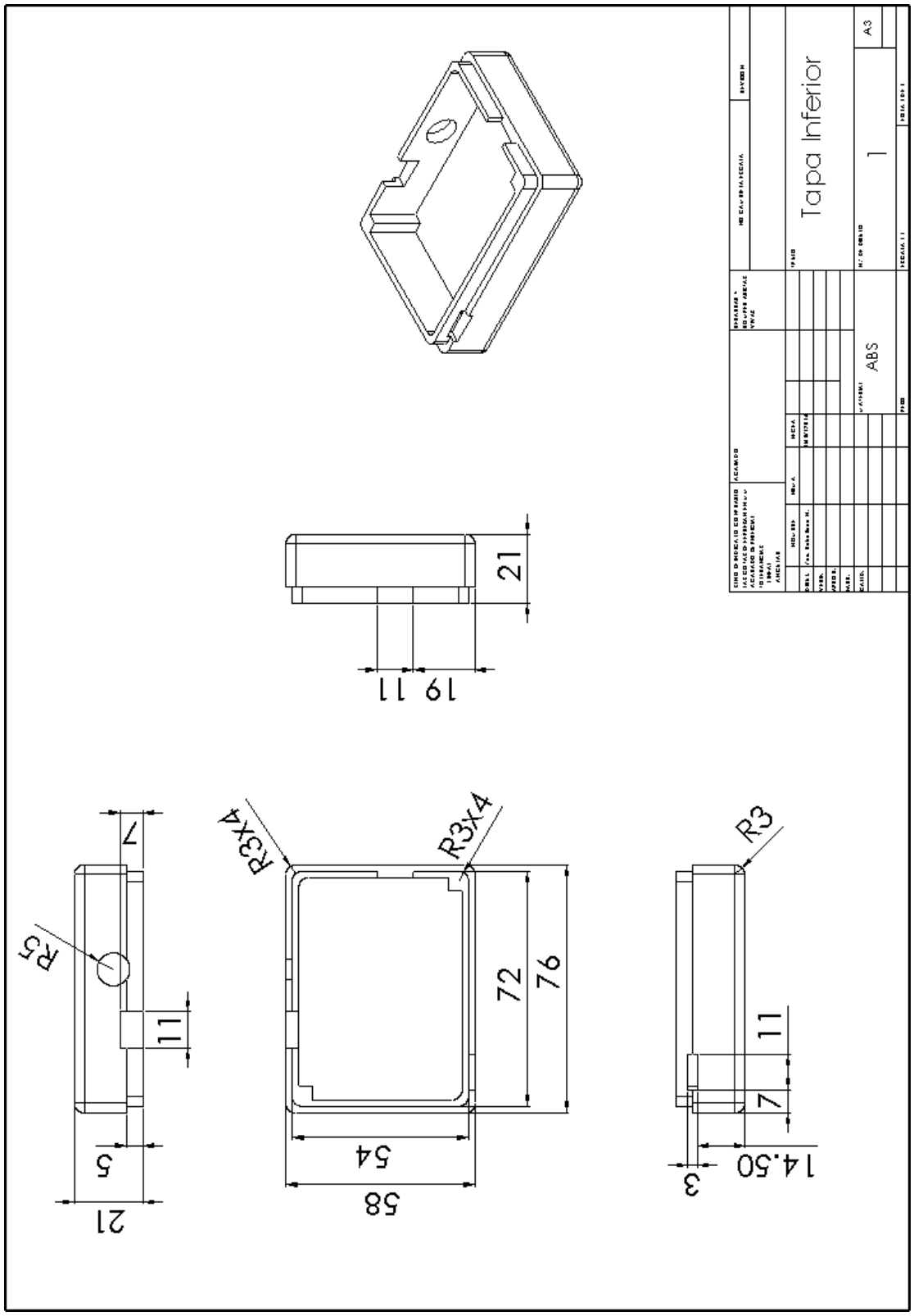


Cubierta plástica Ensamble del case

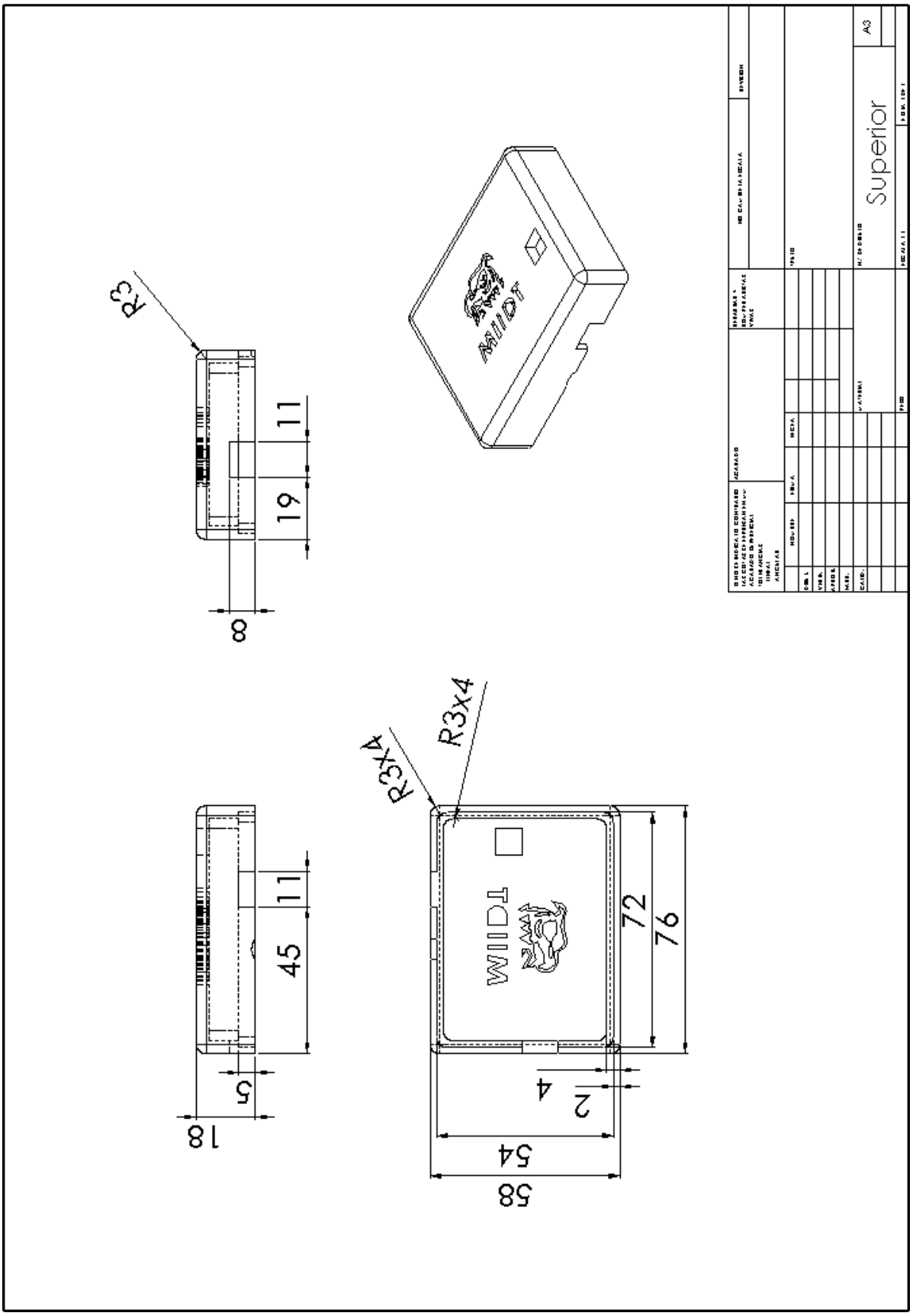
A.



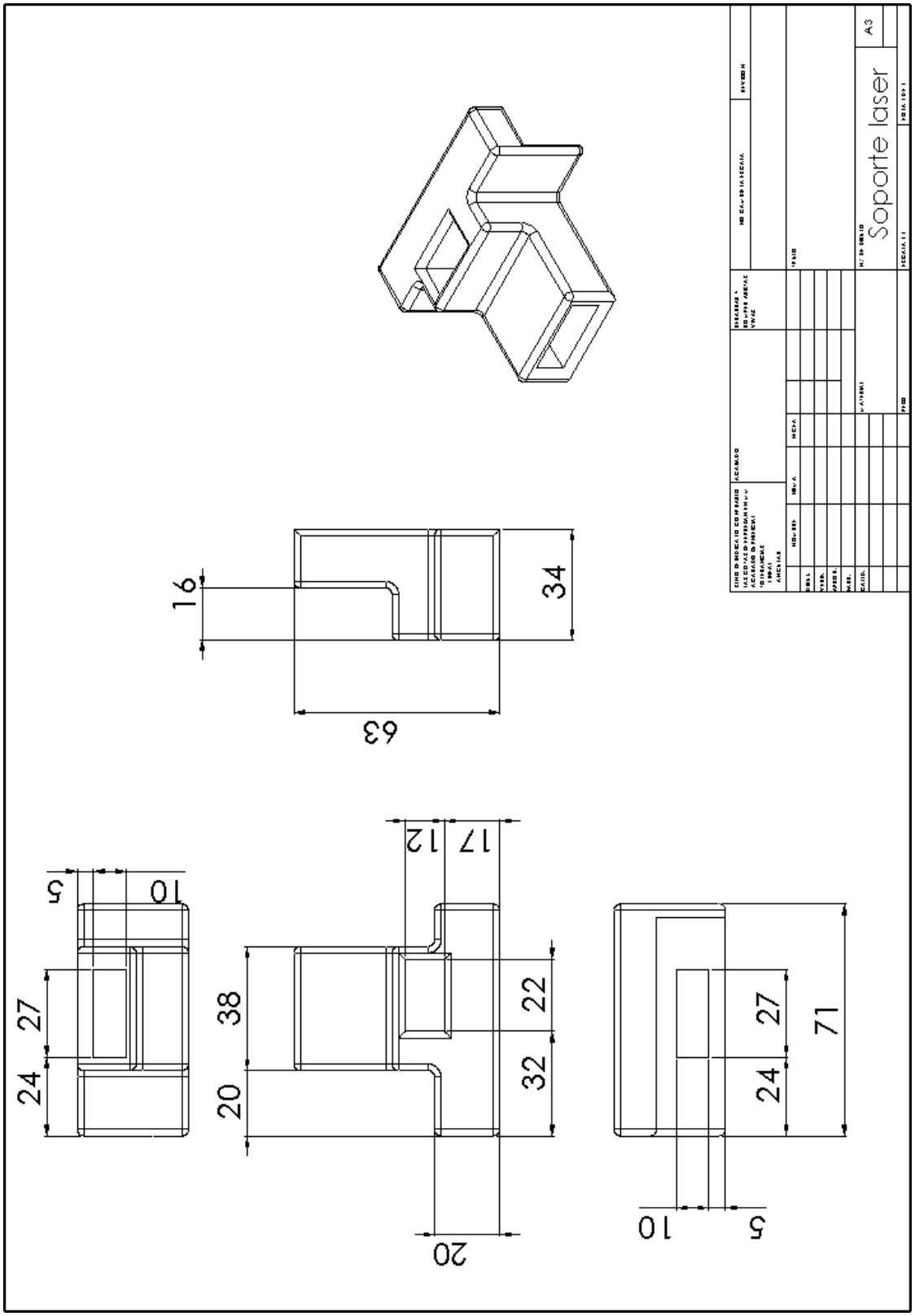
Tapa Inferior



Tapa Superior



Soporte para apuntador laser



A. 2. Hojas de datos

Especificaciones del sensor de presión

Pressure

Freescale Semiconductor

MPX5050
Rev 11, 03/2010

Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPX5050
MPXV5050
MPVZ5050
Series
0 to 50 kPa (0 to 7.25 psi)
0.2 to 4.7 V Output

The MPXx5050 series piezoresistive transducer is a state-of-the-art monolithic silicon pressure sensor designed for a wide range of applications, but particularly those employing a microcontroller or microprocessor with A/D inputs. This patented, single element transducer combines advanced micromachining techniques, thin-film metallization, and bipolar processing to provide an accurate, high level analog output signal that is proportional to the applied pressure.

Features

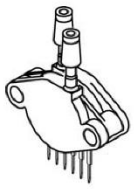
- 2.5% Maximum Error over 0° to 85°C
- Ideally suited for Microprocessor or Microcontroller-Based Systems
- Temperature Compensated Over -40° to +125°C
- Patented Silicon Shear Stress Strain Gauge
- Durable Epoxy Unibody Element
- Easy-to-Use Chip Carrier Option

Pressure

UNIBODY PACKAGES

MPX5050D
CASE 867-08

MPX5050GP
CASE 867B-04

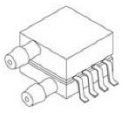


MPX5050DP
CASE 857C-05

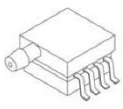
SMALL OUTLINE PACKAGES

MPVZ5050GW7U
CASE 1560-03

MPXV5100GC6U
CASE 482A-01



MPXV5050DP
CASE 1351-01



MPXV5050GP
CASE 1369-01

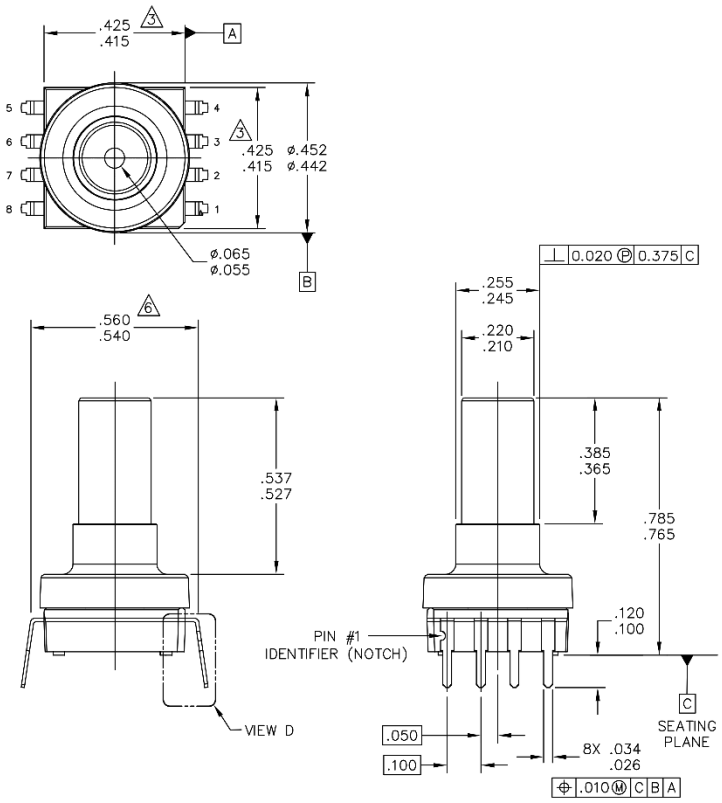
Operating Characteristics

Table 1. Operating Characteristics (V_S = 5.0 Vdc, T_A = 25°C unless otherwise noted, P1 > P2. Decoupling circuit shown in Figure 4 required to meet electrical specifications.)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range[1]		P_{OP}	0	—	50	kPa
Supply Voltage[2]		V_S	4.75	5.0	5.25	Vdc
Supply Current		I_o	—	7.0	10	mAdc
Minimum Pressure Offset[3] @ V_S = 5.0 Volts	(0 to 85°C)	V_{off}	0.088	0.2	0.313	Vdc
Full Scale Output[4] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSO}	4.587	4.7	4.813	Vdc
Full Scale Span[5] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSS}	—	4.5	—	Vdc
Accuracy[6]	(0 to 85°C)	—	—	—	±2.5	$\%V_{FSS}$
Sensitivity		V/P	—	90	—	mV/kPa
Response Time[7]		t_R	—	1.0	—	ms
Output Source Current at Full Scale Output		I_{o+}	—	0.1	—	mAdc
Warm-Up Time[8]		—	—	20	—	ms
Offset Stability[9]		—	—	±0.5	—	$\%V_{FSS}$

1. 1.0 kPa (kiloPascal) equals 0.145 psi.
2. Device is ratiometric within this specified excitation range.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure at 25°C.
   TcSpan: Output deviation over the temperature range of 0° to 85°C, relative to 25°C.
   TcOffset: Output deviation with minimum pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.
   Variation from Nominal: The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS} at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm-up Time is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
9. Offset Stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.

PACKAGE DIMENSIONS



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: SO, 8 I/O, .420 X .420 PKG, .100 IN PITCH	DOCUMENT NO: 98ASA10611D	REV: D
	CASE NUMBER: 1560-03	25 FEB 2009
	STANDARD: NON-JEDEC	

CASE 1560-03
ISSUE D
SMALL OUTLINE PACKAGE

Especificaciones del sensor de temperatura LM35

LM35 Precision Centigrade Temperature Sensors

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

3 Description

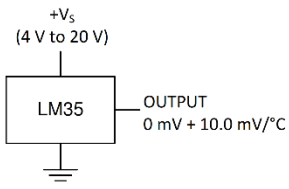
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

Device Information(1)

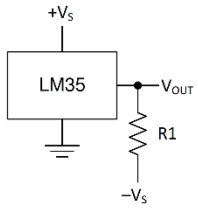
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM35	TO-CAN (3)	4.699 mm × 4.699 mm
	TO-92 (3)	4.30 mm × 4.30 mm
	SOIC (8)	4.90 mm × 3.91 mm
	TO-220 (3)	14.986 mm × 10.16 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Basic Centigrade Temperature Sensor (2°C to 150°C)



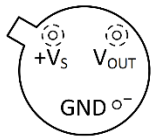
Full-Range Centigrade Temperature Sensor



Choose $R_1 = -V_S / 50\ \mu A$
$V_{OUT} = 1500$ mV at 150°C
$V_{OUT} = 250$ mV at 25°C
$V_{OUT} = -550$ mV at −55°C

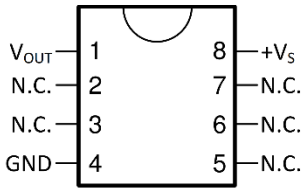
5 Pin Configuration and Functions

NDV Package
3-Pin TO-CAN
(Bottom View)



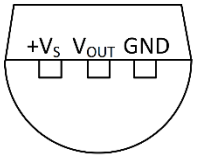
Case is connected to negative pin (GND)

D Package
8-PIN SOIC
(Top View)

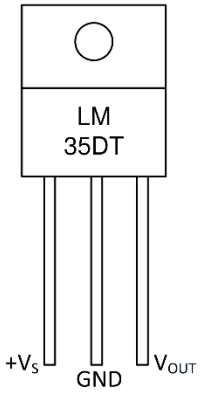


N.C. = No connection

LP Package
3-Pin TO-92
(Bottom View)



NEB Package
3-Pin TO-220
(Top View)



Tab is connected to the negative pin (GND).

NOTE: The LM35DT pinout is different than the discontinued LM35DP

Pin Functions

PIN					TYPE	DESCRIPTION
NAME	TO46	TO92	TO220	SO8		
V_{OUT}	—	—	—	1	O	Temperature Sensor Analog Output
N.C.	—	—	—	2	—	No Connection
	—	—	—	3		
GND	—	—	—	4	GROUND	Device ground pin, connect to power supply negative terminal
N.C.	—	—	—	5	—	No Connection
	—	—	—	6		
	—	—	—	7		
$+V_S$	—	—	—	8	POWER	Positive power supply pin

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)(2)

		MIN	MAX	UNIT
Supply voltage		–0.2	35	V
Output voltage		–1	6	V
Output current			10	mA
Maximum Junction Temperature, $T_J$max			150	°C
Storage Temperature, T_{stg}	TO-CAN, TO-92 Package	–60	150	°C
	TO-220, SOIC Package	–65	150	

(1) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2500	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
Specified operating temperature: T_{MIN} to T_{MAX}	LM35, LM35A	–55	150	°C
	LM35C, LM35CA	–40	110	
	LM35D	0	100	
Supply Voltage (+V_S)		4	30	V

6.4 Thermal Information

THERMAL METRIC(1)(2)		LM35				UNIT
		NDV	LP	D	NEB	
		3 PINS		8 PINS	3 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	400	180	220	90	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	24	—	—	—	

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) For additional thermal resistance information, see *Typical Application*.

8.2 Typical Application

8.2.1 Basic Centigrade Temperature Sensor

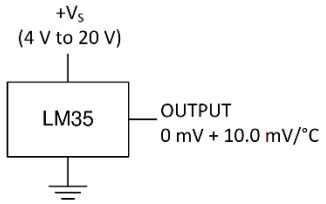


Figure 14. Basic Centigrade Temperature Sensor (2 °C to 150 °C)

8.2.1.1 Design Requirements

Table 1. Design Parameters

PARAMETER	VALUE
Accuracy at 25°C	±0.5°C
Accuracy from –55 °C to 150°C	±1°C
Temperature Slope	10 mV/°C

8.2.1.2 Detailed Design Procedure

Because the LM35 device is a simple temperature sensor that provides an analog output, design requirements related to layout are more important than electrical requirements. For a detailed description, refer to the *Layout*.

8.2.1.3 Application Curve

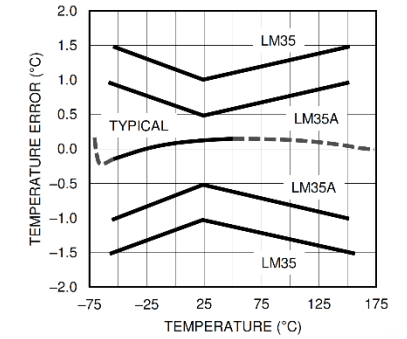


Figure 15. Accuracy vs Temperature (Ensured)

Especificaciones del módulo Bluetooth HC-06

Guangzhou HC Information Technology Co., Ltd.

Product Data Sheet

Module Data Sheet

Rev 1

1．0	2.0	2.1	2.2				
2006/6/18	2006/9/6	2010/4/22	2011/4/6				

DRAWN BY :	Ling Xin		**MODEL :** HC-06	
CHECKED BY :	Eric Huang		**Description::** BC04 has external 8M Flash and EDR module HC-06 is industrial, and compatible with civil HC-04	
APPD. BY:	Simon Mok		**REV: 2.0**	**Page :**
Former version introduction	HC-06 is the higher version of LV_BC_2.0. Linvor is the former of wavesen.			

www.wavesen.com Phone: 020-84083341 Fax: 020-84332079 QQ:1043073574
Address: Room 527, No.13, Jiangong Road, Tianhe software park, Tianhe district, Guangzhou Post: 510660
Technology consultant: support@wavesen.com Business consultant:sales@wavesen.com
Complaint and suggestion: sunbirdit@hotmail.com

1. Product's picture

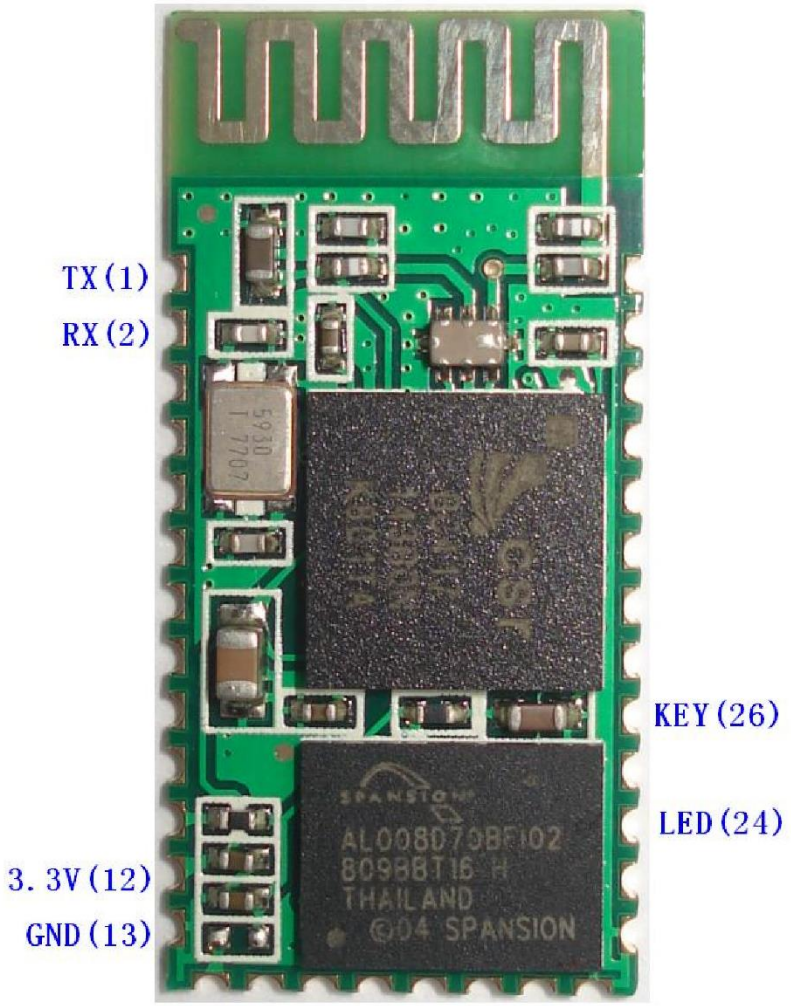


Figure 1 A Bluetooth module

www.wavesen.com Phone: 020-84083341 Fax: 020-84332079 QQ:1043073574
Address: Room 527, No.13, Jiangong Road, Tianhe software park, Tianhe district, Guangzhou Post: 510660
Technology consultant: support@wavesen.com Business consultant:sales@wavesen.com
Complaint and suggestion: sunbirdit@hotmail.com

2. Feature

- Wireless transceiver
  - Sensitivity (Bit error rate) can reach -80dBm.
  - The change range of output's power: -4 - +6dBm.
- Function description (perfect Bluetooth solution)
  - Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
  - Has a build-in 2.4GHz antenna; user needn't test antenna.
  - Has the external 8Mbit FLASH
  - Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA. The current in communication is 8mA.
  - Standard HCI Port (UART or USB)
  - USB Protocol: Full Speed USB1.1, Compliant With 2.0
  - This module can be used in the SMD.
  - It's made through RoHS process.
  - The board PIN is half hole size.
  - Has a 2.4GHz digital wireless transceiver.
  - Bases at CSR BC04 Bluetooth technology.
  - Has the function of adaptive frequency hopping.
  - Small (27mm×13mm×2mm)
  - Peripherals circuit is simple.
  - It's at the Bluetooth class 2 power level.
  - Storage temperature range: -40 ℃ - 85℃， work temperature range: -25 ℃ - +75℃
  - Any wave inter Interference: 2.4MHz， the power of emitting: 3 dBm.
  - Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.

- Low power consumption
- Has high-performance wireless transceiver system
- Low Cost

PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	CLK_REQ
PIO7	30	Bi-Directional	Programmable input/output line	CLK_OUT
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	
RESETB	11	CMOS Input with weak intemal pull-down		
UART_RTS	4	CMOS output, tri-stable with weak internal pull-up	UART request to send, active low	
UART_CTS	3	CMOS input with weak internal pull-down	UART clear to send, active low	
UART_RX	2	CMOS input with weak internal pull-down	UART Data input	
UART_TX	1	CMOS output, Tri-stable with weak internal pull-up	UART Data output	
SPI_MOSI	17	CMOS input with weak internal pull-down	Serial peripheral interface data input	
SPI_CSB	16	CMOS input with weak internal	Chip select for serial peripheral interface, active low	

A. 3. Código

Programa en Arduino para la adquisición y envío de variables biomédicas.

```
1. #include <SoftwareSerial.h> //Libreria que permite establecer
   comunicacion serie en otros pins

2. const int analogInPin0 = A0;
3. const int analogInPin1 = A1;
4. const int analogInPin2 = A2;
5. const int analogInPin3 = A3;
6.
7. char outStrX[5];
8. char outStrY[5];
9. char outStrZ[5];
10.  char outStrW[5];
11.
12.  unsigned long LoopTimer = 0;
13.
14.  int x=0,y=0,z=0,w=0;
15.
16.  const int LoopTime5000 = 1000;
17.
18.  SoftwareSerial BT(10,11);//10 RX, 11 TX.
19.  void setup(){
20.    BT.begin(9600);
21.    Serial.begin(9600);
22.  }
23.
24.  void loop(){
25.  if (micros() >= LoopTimer+LoopTime5000){
26.    LoopTimer += LoopTime5000;
27.
28.    readSensors();
29.    sendAndroidValues();
30.    sendSerialValues();
31.    }
32.  }
33.
34.  void readSensors(){
35.  // leer las señales biomedicas
36.  x = analogRead(analogInPin0);
37.  y = analogRead(analogInPin1);
38.  z = analogRead(analogInPin2);
39.  w = analogRead(analogInPin3);
40.
41.  sprintf(outStrX,"%04d",x);
42.  sprintf(outStrY,"%04d",y);
43.  sprintf(outStrZ,"%04d",z);
44.  sprintf(outStrW,"%04d",w);
45.  }
46.
47.  void sendSerialValues(){
48.    //Enviar datos
49.    Serial.print("*");
50.    Serial.write(outStrX);
51.    Serial.write(outStrY);
52.    Serial.write(outStrZ);
```

```
53.     Serial.write(outStrW);
54.   }

55.   void sendAndroidValues(){
56.     BT.print("*");
57.     BT.write(outStrX);
58.     BT.write(outStrY);
59.     BT.write(outStrZ);
60.     BT.write(outStrW);
61.   }
```

Archivos XML para el maquetado de la aplicación.

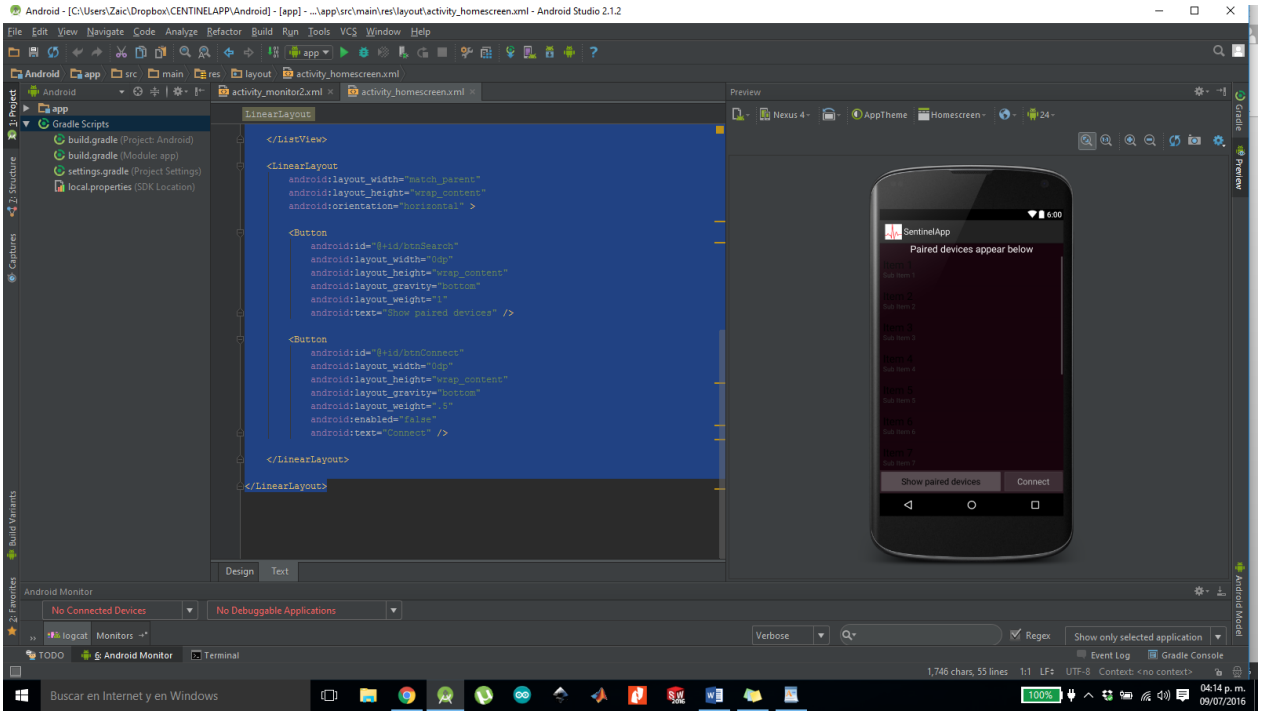


Fig. 5.1 Maquetado de la pantalla para la conexión Bluetooth.

```
1.  <LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
2.      xmlns:tools=http://schemas.android.com/tools
3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent"
5.      android:orientation="vertical"
6.      tools:context=".Homescreen"
7.      android:background="#e4140009">
8.
9.      <TextView
10.         android:id="@+id/txtListHeading"
11.         android:layout_width="fill_parent"
12.         android:layout_height="wrap_content"
```

```
        android:text="Paired devices appear below"
        android:textSize="20sp"
        android:gravity="center"
        android:allowUndo="false"
        android:background="#e4140009"
        android:textColor="#ffffff" />

    <ListView
        android:id="@+id/lstDevices"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingBottom="10dp"
        android:background="#e4140009"
        android:clickable="false">

    </ListView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnSearch"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_weight="1"
            android:text="Show paired devices" />

        <Button
            android:id="@+id/btnConnect"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_weight=".5"
            android:enabled="false"
            android:text="Connect" />

    </LinearLayout>

</LinearLayout>
```

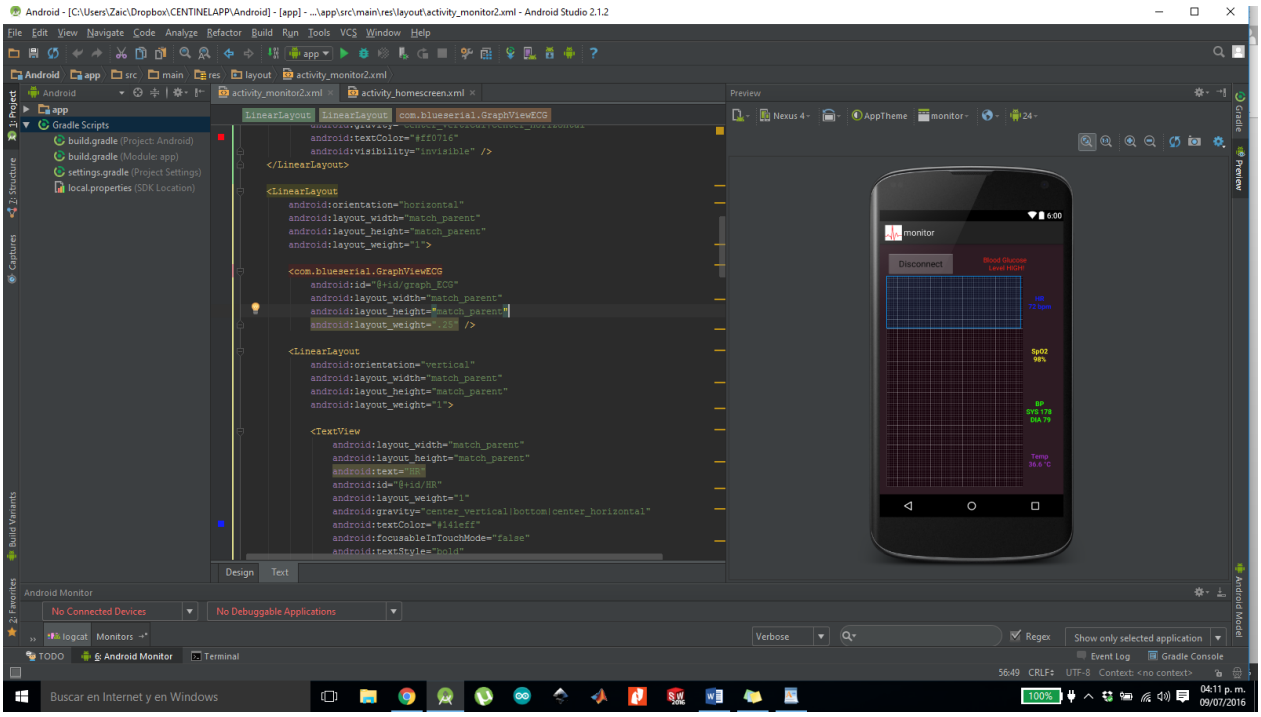


Fig. Maquetado de la pantalla principal del
donde se muestran todas la variables biomédicas.

```
1.  <LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
2.      xmlns:tools=http://schemas.android.com/tools
3.      android:orientation="vertical"
4.      android:layout_width="match_parent"
5.      android:layout_height="match_parent"
6.  android:paddingLeft="@dimen/activity_horizontal_margin"
7.      android:paddingRight="@dimen/activity_horizontal_margin"
8.      android:paddingTop="@dimen/activity_vertical_margin"
9.      android:paddingBottom="@dimen/activity_vertical_margin"
10. tools:context="com.blueserial.monitor"
11.     android:background="#e4140009">
12.
13.     <LinearLayout
14.         android:orientation="horizontal"
15.         android:layout_width="match_parent"
16.         android:layout_height="50dp">
17.
18.         <Button
19.             android:layout_width="match_parent"
20.             android:layout_height="match_parent"
21.             android:id="@+id/btnDisconnect"
22.             android:layout_weight="1.5"
23.             android:text="@string/disconnect_btn" />
24.
25.         <TextView
26.             android:layout_width="match_parent"
27.             android:layout_height="match_parent"
28.             android:text="Blood Glucose \n Level HIGH!"
29.             android:id="@+id/textView2"
30.             android:gravity="center_vertical|center_horizontal"
```

```
            android:layout_weight="1"
            android:textColor="#ff2215" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="."
            android:id="@+id/txtReceive"
            android:singleLine="false"
            android:layout_weight="2.5"

            android:typeface="monospace"
            android:hint="@string/txtReceive_str"
            android:gravity="center_vertical|center_horizontal"
            android:textColor="#ff0716"
            android:visibility="invisible" />
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1">

        <com.blueserial.GraphViewECG
            android:id="@+id/graph_ECG"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight=".25" />

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="HR"
                android:id="@+id/HR"
                android:layout_weight="1"
                android:gravity="center_vertical|bottom|center_horizontal"
                android:textColor="#141eff"
                android:focusableInTouchMode="false"
                android:textStyle="bold"
                android:typeface="normal" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/HRvalue"
                android:layout_weight="1"
                android:gravity="top|center_horizontal"
                android:textColor="#141eff"
                android:textStyle="bold"
                android:typeface="normal"
```

```
                android:text="72 bpm" />
        </LinearLayout>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1">


            <com.blueserial.GraphViewSp02
                android:id="@+id/graph_SpO2"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight=".25" />
      <LinearLayout
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1" >

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="SpO2"
                    android:id="@+id/Sp02"
                    android:layout_weight="1"
                    android:gravity="bottom|center_horizontal"
                    android:textColor="#fffc23"
                    android:textStyle="bold"
                    android:typeface="normal" />

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:id="@+id/SpO2value"
                    android:layout_weight="1"
                    android:gravity="top|center_horizontal"
                    android:textColor="#fffc23"
                    android:textStyle="bold"
                    android:typeface="normal"
                    android:text="98%" />
            </LinearLayout>
        </LinearLayout>

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1">

            <com.blueserial.GraphViewBP
                android:id="@+id/graph_BP"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
```

```
                android:layout_weight=".25" />

            <LinearLayout
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1" >

               <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="BP"
                    android:id="@+id/BP"
                    android:layout_weight="1"

    android:gravity="center_vertical|bottom|center_horizontal"
                    android:textColor="#21ff07"
                    android:textStyle="bold"
                    android:typeface="normal" />

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:id="@+id/BPvalue"
                    android:layout_weight="1"
                    android:gravity="top|center_horizontal"
                    android:textColor="#21ff07"
                    android:textStyle="bold"
                    android:typeface="normal"
                    android:text="SYS 178 \n DIA 79" />
            </LinearLayout>
        </LinearLayout>

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1">

            <com.blueserial.GraphViewTemp
                android:id="@+id/graph_Temp"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight=".25" />

            <LinearLayout
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1" >

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:text="Temp"
                    android:id="@+id/Temp"
                    android:layout_weight="1"
```

```
                    android:gravity="bottom|center_horizontal"
                    android:textColor="#cfb134e0"
                    android:textStyle="bold"
                    android:typeface="normal" />

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:id="@+id/TempValue"
                    android:layout_weight="1"
                    android:gravity="top|center_horizontal"
                    android:textColor="#cfb134e0"
                    android:textStyle="bold"
                    android:typeface="normal"
                    android:text="36.6 °C" />
            </LinearLayout>
        </LinearLayout>

    </LinearLayout>
```

Clases Java de la Aplicación Android.

Clase GraphView

```
public class GraphViewECG extends View
{
   private Paint paint,paint2,paint3;
   private Path path;
   private int availableWidth;
   private int availableHeight;
   int counter=0;
   private int data;
   long inicio,fin;
   float spacing = (float)1.00;

    public GraphViewECG(Context context)
    {
        this(context, null);
    }
    // This constructor is the player but all are necessary!!
    public GraphViewECG(Context context, AttributeSet attrs)
    {
      super(context, attrs);
      setBackgroundColor(Color.argb(228, 20, 0, 9));
      paint = new Paint(Paint.ANTI_ALIAS_FLAG);
      paint.setStyle(Paint.Style.STROKE);
      paint.setColor(Color.GREEN);
      paint.setStrokeWidth(2);
      paint.setStrokeCap(Paint.Cap.ROUND);
      paint.setStrokeJoin(Paint.Join.ROUND);
      path = new Path();

      paint2 = new Paint(Paint.ANTI_ALIAS_FLAG);
      paint2.setStyle(Paint.Style.STROKE);
      paint2.setColor(Color.GRAY);
      paint2.setStrokeWidth(0.5f);
      paint2.setStrokeCap(Paint.Cap.ROUND);
      paint2.setStrokeJoin(Paint.Join.ROUND);
```

```
35.
36.       paint3 = new Paint(Paint.ANTI_ALIAS_FLAG);
37.       paint3.setStyle(Paint.Style.STROKE);
38.       paint3.setColor(Color.WHITE);
39.       paint3.setStrokeWidth(0.5f);
40.       paint3.setStrokeCap(Paint.Cap.ROUND);
41.       paint3.setStrokeJoin(Paint.Join.ROUND);
42.     }
43.
44.    public GraphViewECG(Context context, AttributeSet attrs, int defStyle) {
45.        super(context, attrs, defStyle);
46.    }
47.
48.     @Override
49.    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec){
50.       super.onMeasure(widthMeasureSpec, heightMeasureSpec);
51.       availableWidth = getMeasuredWidth();
52.       availableHeight = getMeasuredHeight();
53.    }
54.
55.    public void plotProcedure(int data) {
56.       this.data = (data*availableHeight)/1023;
57.
58.       if (counter > availableWidth/spacing) {
59.          fin = System.currentTimeMillis();
60.          System.out.println("the task has taken " + (float)(fin - inicio)/1000
61. + " milliseconds "+availableWidth);
62.          counter = 0; path.reset();
63.       }
64.       if (counter == 0) {
65.          inicio = System.currentTimeMillis();
66.          path.moveTo(0, availableHeight - this.data);
67.       }
68.       else {
69.          path.lineTo( counter*spacing, availableHeight - this.data);
70.       }
71.       counter++;
72.       // Force redraw
73.       invalidate();
74.    }
75.
76.     @Override
77.     public void onDraw(Canvas canvas) {
78.       canvas.drawPath(path, paint);
79.
80.       for (int i=0; i<=availableHeight;i=i+((availableWidth)/60)) {
81.          canvas.drawLine(0, i, availableWidth, i, paint2);
82.       }
83.
84.       for (int i=0; i<=availableHeight;i=i+((availableWidth)/60)*5) {
85.          canvas.drawLine(0, i, availableWidth, i, paint3);
86.       }
87.
88.       for (int i=0; i<=availableWidth;i=i+((availableWidth)/60)) {
89.          canvas.drawLine(i, 0, i, availableHeight, paint2);
90.       }
91.
92.       for (int i=0; i<=availableWidth;i=i+((availableWidth)/60)*5) {
93.          canvas.drawLine(i, 0, i, availableHeight, paint3);
94.       }
95.
96.     }
97. }
```

Clase Homescreen

```
1. public class Homescreen extends Activity {
2.
3.    private Button mBtnSearch;
4.    private Button mBtnConnect;
5.    private ListView mLstDevices;
6.
7.    private BluetoothAdapter mBTAdapter;
8.
9.    private static final int BT_ENABLE_REQUEST = 10; // This is the code we use
10.for BT Enable
11.   private static final int SETTINGS = 20;
12.
13.   private UUID mDeviceUUID = UUID.fromString("00001101-0000-1000-8000
14.00805F9B34FB"); // Standard SPP UUID
15.   //
16.   private int mBufferSize = 50000; //Default
17.   public static final String DEVICE_EXTRA = "com.blueserial.SOCKET";
18.   public static final String DEVICE_UUID = "com.blueserial.uuid";
19.   private static final String DEVICE_LIST = "com.blueserial.devicelist";
20.   private static final String DEVICE_LIST_SELECTED =
21."com.blueserial.devicelistselected";
22.   public static final String BUFFER_SIZE = "com.blueserial.buffersize";
23.   private static final String TAG = "BlueTest5-Homescreen";
24.   public String path =
25.Environment.getExternalStorageDirectory().getAbsolutePath() + "/SentinelApp";
26.   @Override
27.   protected void onCreate(Bundle savedInstanceState) {
28.
29.      super.onCreate(savedInstanceState);
30.
31.      setContentView(R.layout.activity_homescreen);
32.      ActivityHelper.initialize(this);
33.      Log.d(TAG, "Created");
34.
35.      mBtnSearch = (Button) findViewById(R.id.btnSearch);
36.      mBtnConnect = (Button) findViewById(R.id.btnConnect);
37.
38.      mLstDevices = (ListView) findViewById(R.id.lstDevices);
39.
40.      File dir = new File(path);
41.      dir.mkdirs();
42.      File file = new File(dir,"biomedicalData.txt");
43.
44.      if (file.exists()){file.delete();}
45.
46.      /*********************************************************************/
47.      if (savedInstanceState != null) {
48.         ArrayList<BluetoothDevice> list =
49.savedInstanceState.getParcelableArrayList(DEVICE_LIST);
50.         if(list!=null){
51.            initList(list);
52.            MyAdapter adapter = (MyAdapter)mLstDevices.getAdapter();
53.            int selectedIndex =
54.savedInstanceState.getInt(DEVICE_LIST_SELECTED);
55.            if(selectedIndex != -1){
56.               adapter.setSelectedIndex(selectedIndex);
57.               mBtnConnect.setEnabled(true);
58.            }
59.         } else {
60.            initList(new ArrayList<BluetoothDevice>());
61.         }
62.
```

```
63.       } else {
64.          initList(new ArrayList<BluetoothDevice>());
65.       }
66.
67.       mBtnSearch.setOnClickListener(new OnClickListener() {
68.
69.          @Override
70.          public void onClick(View arg0) {
71.             mBTAdapter = BluetoothAdapter.getDefaultAdapter();
72.
73.             if (mBTAdapter == null) {
74.                Toast.makeText(getApplicationContext(), "Bluetooth not found",
75.                Toast.LENGTH_SHORT).show();
76.             } else if (!mBTAdapter.isEnabled()) {
77.                Intent enableBT = new
78. Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
79.                startActivityForResult(enableBT, BT_ENABLE_REQUEST);
80.             } else {
81.                new SearchDevices().execute();
82.             }
83.          }
84.       });
85.
86.       mBtnConnect.setOnClickListener(new OnClickListener() {
87.
88.          @Override
89.          public void onClick(View arg0) {
90.             BluetoothDevice device = ((MyAdapter)
91.             (mLstDevices.getAdapter())).getSelectedItem();
92.             Intent intent = new Intent(getApplicationContext(),
93.             monitor2.class);
94.             intent.putExtra(DEVICE_EXTRA, device);
95.             intent.putExtra(DEVICE_UUID, mDeviceUUID.toString());
96.             intent.putExtra(BUFFER_SIZE, mBufferSize);
97.             startActivity(intent);
98.          }
99.       });
100.    }
101.
102.  /************************************************************************/
103.    @Override
104.     protected void onSaveInstanceState(Bundle outState) {
105.     super.onSaveInstanceState(outState);
106.     MyAdapter adapter = (MyAdapter) (mLstDevices.getAdapter());
107.     ArrayList<BluetoothDevice> list = (ArrayList<BluetoothDevice>)
108.     adapter.getEntireList();
109.
110.     if (list != null) {
111.        outState.putParcelableArrayList(DEVICE_LIST, list);
112.        int selectedIndex = adapter.selectedIndex;
113.        outState.putInt(DEVICE_LIST_SELECTED, selectedIndex);
114.     }
115.  }
116.    @Override
117.   protected void onPause()
118.      // TODO Auto-generated method stub
119.      super.onPause();
120.   }
121.
122.  @Override
123.  protected void onStop() {
124.      // TODO Auto-generated method stub
125.      super.onStop();
126.  }
127.
128.  @Override
```

```
129.  protected void onActivityResult(int requestCode, int resultCode, Intent
130.   data) {
131.   switch (requestCode) {
132.     case BT_ENABLE_REQUEST:
133.        if (resultCode == RESULT_OK)
134.           msg("Bluetooth Enabled successfully");
135.           new SearchDevices().execute();
136.        } else {
137.           msg("Bluetooth couldn't be enabled");
138.        }
139.        break;
140.
141.        case SETTINGS:
142.         SharedPreferences prefs =
143.         PreferenceManager.getDefaultSharedPreferences(this);
144.         String uuid = prefs.getString("prefUuid", "Null");
145.        mDeviceUUID = UUID.fromString(uuid);
146.        Log.d(TAG, "UUID: " + uuid);
147.        String bufSize = prefs.getString("prefTextBuffer", "Null");
148.        mBufferSize = Integer.parseInt(bufSize);
149.
150.        String orientation = prefs.getString("prefOrientation", "Null");
151.        Log.d(TAG, "Orientation: " + orientation);
152.        if (orientation.equals("Landscape")) {
153.
   setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
154.        } else if (orientation.equals("Portrait")) {
155.           setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
156.        } else if (orientation.equals("Auto")) {
157.
158. setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
159.        }
160.        break;
161.        default:
162.        break;
163.     }
164.     super.onActivityResult(requestCode, resultCode, data);
165.  }
166.
167.  /*********************************************************************/
168.   private void msg(String str) {
169.     Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT).show();
170.   }
171.
172.   /*********************************************************************/
173.   private void initList(List<BluetoothDevice> objects) {
174.     final MyAdapter adapter = new MyAdapter(getApplicationContext(),
175. R.layout.list_item, R.id.lstContent, objects);
176.     mLstDevices.setAdapter(adapter);
177.     mLstDevices.setOnItemClickListener(new OnItemClickListener() {
178.
179.        @Override
180.        public void onItemClick(AdapterView<?> parent, View view, int
181.        position, long id) {
182.             adapter.setSelectedIndex(position);
183.             mBtnConnect.setEnabled(true);
184.       }
185.     });
186.  }
187.
188.  /*********************************************************************/
189.   private class SearchDevices extends AsyncTask<Void, Void,
190.   List<BluetoothDevice>> {
191.
192.     @Override
193.     protected List<BluetoothDevice> doInBackground(Void... params) {
```

```
194.         Set<BluetoothDevice> pairedDevices = mBTAdapter.getBondedDevices();
195.         List<BluetoothDevice> listDevices = new ArrayList<BluetoothDevice>();
196.         for (BluetoothDevice device : pairedDevices) {
197.            listDevices.add(device);
198.         }
199.         return listDevices;
200.
201.      }
202.
203.      @Override
204.      protected void onPostExecute(List<BluetoothDevice> listDevices) {
205.         super.onPostExecute(listDevices);
206.         if (listDevices.size() > 0) {
207.            MyAdapter adapter = (MyAdapter) mLstDevices.getAdapter();
208.            adapter.replaceItems(listDevices);
209.         } else {
210.            msg("No paired devices found, please pair your serial BT device
    and
211.            try again");
212.          }
213.      }
214.
215.   }
216.
217.   /*************************************************************/
218.
219.    private class MyAdapter extends ArrayAdapter<BluetoothDevice> {
220.      private int selectedIndex;
221.      private Context context;
222.      private int selectedColor = Color.parseColor("#abcdef");
223.      private List<BluetoothDevice> myList;
224.
225.      public MyAdapter(Context ctx, int resource, int textViewResourceId,
226.      List<BluetoothDevice> objects) {
227.         super(ctx, resource, textViewResourceId, objects);
228.         context = ctx;
229.         myList = objects;
230.         selectedIndex = -1;
231.      }
232.
233.      public void setSelectedIndex(int position) {
234.         selectedIndex = position;
235.         notifyDataSetChanged();
236.      }
237.
238.      public BluetoothDevice getSelectedItem() {
239.         return myList.get(selectedIndex);
240.      }
241.
242.      @Override
243.      public int getCount() {
244.         return myList.size();
245.      }
246.
247.      @Override
248.      public BluetoothDevice getItem(int position) {
249.         return myList.get(position);
250.      }
251.
252.      @Override
253.      public long getItemId(int position) {
254.         return position;
255.      }
256.
257.      private class ViewHolder {
258.         TextView tv;
```

```
259.      }
260.
261.      public void replaceItems(List<BluetoothDevice> list) {
262.         myList = list;
263.         notifyDataSetChanged();
264.      }
265.
266.      public List<BluetoothDevice> getEntireList() {
267.         return myList;
268.      }
269.
270.      @Override
271.      public View getView(int position, View convertView, ViewGroup parent) {
272.         View vi = convertView;
273.         ViewHolder holder;
274.         if (convertView == null) {
275.            vi = LayoutInflater.from(context).inflate(R.layout.list_item,
276.            null);
277.            holder = new ViewHolder();
278.
279.            holder.tv = (TextView) vi.findViewById(R.id.lstContent);
280.
281.            vi.setTag(holder);
282.         } else {
283.            holder = (ViewHolder) vi.getTag();
284.         }
285.
286.         if (selectedIndex != -1 && position == selectedIndex) {
287.            holder.tv.setBackgroundColor(selectedColor);
288.         } else {
289.            holder.tv.setBackgroundColor(Color.WHITE);
290.         }
291.         BluetoothDevice device = myList.get(position);
292.         holder.tv.setText(device.getName() + "\n   " + device.getAddress());
293.
294.         return vi;
295.      }
296.
297.   }
298.
299.   @Override
300.   public boolean onCreateOptionsMenu(Menu menu) {
301.      // Inflate the menu; this adds items to the action bar if it is present
302.      getMenuInflater().inflate(R.menu.homescreen, menu);
303.      return true;
304.   }
305.
306. }
```

Clase monitor

```
1. public class monitor2 extends Activity {
2.      private static final String TAG = "BlueSerial-monitor2";
3.      private int mMaxChars = 50000;//Default
4.      private UUID mDeviceUUID;
5.      private BluetoothSocket mBTSocket;
6.      private ReadInput mReadThread = null;
7.
8.      private boolean mIsUserInitiatedDisconnect = false;
9.
10.     // All controls here
11.     private TextView mTxtReceive;
12.     private Button mBtnDisconnect;
```

```
13.     private GraphViewECG graphECG;
14.     private GraphViewSp02 graphSpO2;
15.     private GraphViewBP graphBP;
16.     private GraphViewTemp graphTemp;
17.
18.     private TextView textECG, textSp02, textBP, textTemp;
19.
20.     private boolean mIsBluetoothConnected = false;
21.     private BluetoothDevice mDevice;
22.     private ProgressDialog progressDialog;
23.
24.     private StringBuilder recDataString = new StringBuilder(1024);
25.
26.     private String flag[];
27.     private String x="",y="",z="",w="";
28.     private String x2="",y2="",z2="",w2="";
29.     private int intX=0,intY=0,intZ=0,intW=0;
30.
31.     public String path =
32. Environment.getExternalStorageDirectory().getAbsolutePath() + "/SentinelApp";
33.
34.     @Override
35.     protected void onCreate(Bundle savedInstanceState) {
36.         super.onCreate(savedInstanceState);
37.
38.
39.         setContentView(R.layout.activity_monitor2);
40.
41.         textECG = (TextView) findViewById(R.id.HRvalue);
42.         textSp02 = (TextView) findViewById(R.id.SpO2value);
43.         textBP = (TextView) findViewById(R.id.BPvalue);
44.         textTemp = (TextView) findViewById(R.id.TempValue);
45.
46.         graphECG =(GraphViewECG) findViewById(R.id.graph_ECG);
47.         graphSpO2 = (GraphViewSp02) findViewById(R.id.graph_SpO2);
48.         graphBP = (GraphViewBP) findViewById(R.id.graph_BP);
49.         graphTemp = (GraphViewTemp)findViewById(R.id.graph_Temp);
50.
51.         File dir = new File(path);
52.         dir.mkdirs();
53.
54.         Intent intent = getIntent();
55.         Bundle b = intent.getExtras();
56.         mDevice = b.getParcelable(Homescreen.DEVICE_EXTRA);
57.         mDeviceUUID = UUID.fromString(b.getString(Homescreen.DEVICE_UUID));
58.         mMaxChars = b.getInt(Homescreen.BUFFER_SIZE);
59.
60.         Log.d(TAG, "Ready");
61.
62.         mBtnDisconnect = (Button) findViewById(R.id.btnDisconnect);
63.         mTxtReceive = (TextView) findViewById(R.id.txtReceive);
64.
65.         mBtnDisconnect.setOnClickListener(new OnClickListener() {
66.
67.             @Override
68.             public void onClick(View v) {
69.                 mIsUserInitiatedDisconnect = true;
70.                 new DisConnectBT().execute();
71.             }
72.         });
73.     }
74.
75.
76.     private class ReadInput implements Runnable {
77.
78.         private boolean bStop = false;
```

```
79.         private Thread t;
80.
81.         public ReadInput() {
82.             t = new Thread(this, "Input Thread");
83.             t.start();
84.         }
85.
86.         public boolean isRunning() {
87.             return t.isAlive();
88.         }
89.
90.         @Override
91.         public void run() {
92.             InputStream inputStream;
93.
94.             try {
95.                 inputStream = mBTSocket.getInputStream();
96.                 while (!bStop) {
97.                     // byte[] buffer = new byte[256];
98.                     // buffer size increased for ECG
99.                     byte[] buffer = new byte[512];
100.                      if (inputStream.available() > 0) {
101.                        inputStream.read(buffer);
102.                        int i = 0;
103.
104.                         for (i = 0; i < buffer.length && buffer[i] != 0;
    i++){
                            }
105.
106.                        final String strInput = new String(buffer, 0, i);
107.
108.                        mTxtReceive.post(new Runnable() {
109.                            @Override
110.                            public void run() {
111.                                mTxtReceive.append(strInput);
112.                                mTxtReceive.setText("Blood Glucose \n Level
113.                                HIGH!");
114.                                  String[] items = strInput.split("\\*");
115.                                for (String item : items )
116.                                {
117.                                    if(item.length() == 16){
118.                                          try{
119.                                            x = item.substring(0,4);
120.                                            y = item.substring(4,8);
121.                                            z = item.substring(8,12);
122.                                            w = item.substring(12, 16);
123.
124.                                            intX = Integer.parseInt(x);
125.                                            intY = Integer.parseInt(y);
126.                                            intZ = Integer.parseInt(z);
127.                                            intW = Integer.parseInt(w);
128.
129.                                            graphECG.plotProcedure(intX);
130.                                            graphSpO2.plotProcedure(intY);
131.                                            graphBP.plotProcedure(intZ);
132.                                            graphTemp.plotProcedure(intW);
133.
134.                                            textECG.setText("72 bpm");
135.                                            textSp02.setText("97%");
136.                                            textBP.setText("DIA 178 \n
    SYS79");
137.                                            textTemp.setText("36.6 °C");
138.                                            writeFile();
139.
140.                                        }catch(NumberFormatException ex){
141.  //handle your exception
```

```
                                        System.out.println("Not an
                                        integer");
                                    }
                                }

                            }
                        }
                    });
                }
                Thread.sleep(90);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
             e.printStackTrace();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
             e.printStackTrace();
        }
    }
    public void stop() {
        bStop = true;
    }
}

//---------------------------------------------------------------------------
-

 public void writeFile(){
    File dir = new File(path);
    dir.mkdirs();
    File file = new File(dir,"biomedicalData.txt");

    String message = intX+ "," + intY+ "," +intZ + ","+ intW + "\n" ;

    try {
        FileOutputStream fileOutputStream = new
        FileOutputStream(file,true);
          fileOutputStream.write(message.getBytes());
        fileOutputStream.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

//---------------------------------------------------------------------------


 private class DisConnectBT extends AsyncTask<Void, Void, Void> {
    @Override
    protected void onPreExecute() {
    }

    @Override
    protected Void doInBackground(Void... params) {

        if (mReadThread != null)
        try {
            mBTSocket.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
             e.printStackTrace();
        }
        return null;
    }
```

```
207.        @Override
208.        protected void onPostExecute(Void result) {
209.            super.onPostExecute(result);
210.            mIsBluetoothConnected = false;
211.            if (mIsUserInitiatedDisconnect) {
212.                finish();
213.            }
214.        }
215.    }
216.      private void msg(String s) {
217.        Toast.makeText(getApplicationContext(), s, Toast.LENGTH_SHORT).show();
218.    }
219.
220.    @Override
221.    protected void onPause() {
222.        if (mBTSocket != null && mIsBluetoothConnected) {
223.            new DisConnectBT().execute();
224.        }
225.        Log.d(TAG, "Paused");
226.        super.onPause();
227.    }
228.
229.    @Override
230.    protected void onResume() {
231.        if (mBTSocket == null || !mIsBluetoothConnected) {
232.            new ConnectBT().execute();
233.        }
234.        Log.d(TAG, "Resumed");
235.        super.onResume();
236.    }
237.
238.    @Override
239.    protected void onStop() {
240.        Log.d(TAG, "Stopped");
241.        super.onStop();
242.    }
243.
244.    @Override
245.    protected void onSaveInstanceState(Bundle outState) {
246.        // TODO Auto-generated method stub
247.         super.onSaveInstanceState(outState);
248.    }
249.
250.    private class ConnectBT extends AsyncTask<Void, Void, Void> {
251.
252.        private boolean mConnectSuccessful = true;
253.
254.        @Override
255.        protected void onPreExecute() {
256.            progressDialog = ProgressDialog.show(monitor2.this, "Hold on",
257.            "Connecting");// http://stackoverflow.com/a/11130220/1287554
258.          }
259.
260.        @Override
261.        protected Void doInBackground(Void... devices) {
262.              try {
263.                if (mBTSocket == null || !mIsBluetoothConnected) {
264.                    mBTSocket =
265.
    mDevice.createInsecureRfcommSocketToServiceRecord(mDeviceUUID);
266.                    BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
267.                    mBTSocket.connect();
268.                }
269.            } catch (IOException e) {
270.                // Unable to connect to device
271.                 e.printStackTrace();
```

```
272.                 mConnectSuccessful = false;
273.             }
274.             return null;
275.         }
276.         @Override
277.         protected void onPostExecute(Void result) {
278.             super.onPostExecute(result);

279.               if (!mConnectSuccessful) {
280.                 Toast.makeText(getApplicationContext(), "Could not connect to
281.                 device. Is it a Serial device? Also check if the UUID is
282.                 correct in the settings", Toast.LENGTH_LONG).show();
283.                   finish();
284.             } else {
285.                 msg("Connected to device");
286.                 mIsBluetoothConnected = true;
287.                 mReadThread = new ReadInput(); // Kick off input reader
288.              }
289.
290.               progressDialog.dismiss();
291.         }
292.     }
293.
294. }
```

A. 4. Glosario

Aurículas: El interior del corazón se divide en cuatro cavidades: las dos superiores, por donde entra sangre al corazón, se denominan aurículas. Las dos inferiores se denominan ventrículos.

Baumanómetro: es un instrumento médico empleado para la medición indirecta de la presión arterial proporcionando, por lo general, la medición en milímetros de mercurio (mmHg).

Enfermedades crónico-degenerativas: son enfermedades de larga duración y por lo general de progresión lenta. Las enfermedades cardíacas, los infartos, el cáncer, las enfermedades respiratorias y la diabetes, son las principales causas de mortalidad en el mundo.

Espectrofotometría: es la medición de la cantidad de energía radiante que absorbe o transmite un sistema químico en función de la longitud de onda; es el método de análisis óptico más usado en las investigaciones químicas y bioquímicas.

Estetoscopio: Instrumento médico en forma de trompetilla que sirve para explorar los sonidos producidos por los órganos de las cavidades del pecho y del abdomen.

Galvanometro: Instrumento que sirve para determinar la intensidad y el sentido de una corriente eléctrica mediante la desviación que esta produce en una aguja magnética.

Glucosa: Azúcar que se encuentra en la miel, la fruta y la sangre de los animales.

Glucometro: Un glucómetro es un instrumento de medida que se utiliza para obtener la concentración de glucosa en sangre, de forma instantánea

Hemoglobina: proteína que contiene hierro y que le otorga el color rojo a la sangre. Se encuentra en los glóbulos rojos y es la encargada del transporte de oxígeno por la sangre desde los pulmones a los tejidos.

Histograma: es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados, ya sea en forma diferencial o acumulada.

Microcontrolador: Es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Mnemotecnia: es una oración corta y fácil de recordar que ayuda de manera artificiosa a relacionar palabras, con el objetivo de memorizar conceptos con más facilidad.

Osiloscopio: Aparato que sirve para registrar oscilaciones de ondas y las presenta en una pantalla.

Processing: Lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

Ventrículos: Cavidades de la parte inferior del corazón de mamíferos, aves y reptiles que recibe la sangre procedente de las aurículas.

Visión artificial: también conocida como visión por computador o visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen

Wiring: Framework de programación de software abierto para microcontroladores.

Wearable: es aquel dispositivo que se lleva sobre, debajo o incluido en la ropa y que está siempre encendido, no necesita encenderse y apagarse. Otras de sus características es que permite la multitarea por lo que no requiere dejar de hacer otra cosa para ser usado y puede actuar como extensión del cuerpo o mente del usuario.