FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS CARDIOVASCULARES UTILIZANDO TECNOLOGÍA IoMT

AUTOR

Hamilton Renan Lugmaña Pillajo

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS CARDIOVASCULARES UTILIZANDO TECNOLOGÍA IoMT

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Redes y Telecomunicaciones.

Profesor Guía

MSc. Milton Neptalí Román Cañizares

Autor

Hamilton Renan Lugmaña Pillajo

Año

2020

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Implementación de un sistema de adquisición de datos cardiovasculares utilizando tecnología IoMT, a través de reuniones periódicas con el estudiante Hamilton Renan Lugmaña Pillajo, en el semestre 202010, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

Milton Neptalí Román Cañizares

Magister en Gerencia de Redes y Telecomunicaciones

CI: 0502163447

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Implementación de un sistema de adquisición de datos cardiovasculares utilizando tecnología IoMT, del estudiante Hamilton Renan Lugmaña Pillajo, en el semestre 202010, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

Iván Ricardo Sánchez Salazar

Magíster en Calidad Seguridad y Ambiente

CI: 1803456142

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

"Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes"

Hamilton Renan Lugmaña Pillajo

CI: 1724010515

AGRADECIMIENTOS

A Dios en primer lugar por permitirme culminar una nueva etapa de mi formación profesional.

A mi familia quienes con su paciencia y esfuerzo me han apoyado durante todo el trascurso de mis estudios.

DEDICATORIA

A mis padres por su sacrificio constante y permanente acompañamiento durante mis años de estudio, a mis hermanos por motivar mi crecimiento personal y profesional y haber sido mi apoyo en momentos de debilidad.

RESUMEN

En el presente trabajo de titulación se describe el desarrollo e implementación de un sistema de monitoreo para personas propensas a enfermedades cardiovasculares, utilizando el principio de *IoMT* que es considerado como el internet de las cosas médicas. Por otro lado, este sistema utiliza como dato principal la medición de frecuencia cardíaca, para lo cual, el primer paso en la realización de este trabajo es el levantamiento de todos los requerimientos en base a la problemática. También, es necesario llevar a cabo una investigación de los distintos tipos de elementos que se pueden considerar para la implementación del *hardware*. De la misma manera, es fundamental realizar otro estudio comparando las distintas tecnologías actuales para llevar acabo el desarrollo del *software*, así como también alojar dicho sistema en una plataforma *hosting*.

Una vez realizado los respectivos estudios, se procede a iniciar con el diseño de la red y posteriormente el diseño arquitectónico del prototipo electrónico. Es importante mencionar que dentro de la red se encuentra la base de datos, aplicaciones *web* y móvil, mismas que utilizan para su desarrollo varios lenguajes de programación.

Luego de haber terminado con todo el desarrollo del *software*, se procede a realizar la implementación del dispositivo electrónico, utilizando los elementos más apropiados definidos en el estudio previo. Finalmente, el sistema de monitoreo es el encargado de tomar los datos de frecuencia cardíaca del paciente y almacenarlos en una aplicación *web* alojada en un *hosting*, de tal manera, que esta información pueda ser accesible cuando el médico o el paciente lo requieran. Por otro lado, en caso de que el dato de frecuencia cardíaca sobrepase los límites establecidos como normales en el desarrollo del *software*, se realizara el registro de una agenda automáticamente incluyendo los datos de hora y fecha actuales, en los cuales se tomó el último dato del paciente, garantizando de esta manera que el usuario del tipo paciente pueda ser atendido por un médico lo más pronto posible.

Palabras clave: sistema de monitoreo, aplicación *web*, base de datos, frecuencia cardíaca, médico, paciente, lenguajes de programación.

ABSTRACT

In this degree work, describe the development and implementation of a monitoring system for people prone to cardiovascular diseases, using the principle of IoMT that is considered as the internet of medical things. On the other hand, this system uses the heart rate measurement as the main data, for which, the first step in carrying out this work is the lifting of all the requirements based on the problem. Also, it is necessary to carry out an investigation of the different types of elements that can be considered for the hardware implementation. In the same way, it is essential to carry out another study comparing the different current technologies to carry out the software development, as well as said system in a hosting platform.

Once the respective studies have been carried out, we proceed with the design of the network and then the architectural design of the electronic prototype. It is important to mention that within the network is the database, web and mobile applications, which use several programming languages for their development.

After having finished all the software development, the electronic device is implemented, using the most appropriate elements defined in the previous study. Finally, the monitoring system is in charge of taking the patient's heart rate data and storing it in a web application hosted in a hosting, so that this information can be accessible when the doctor or the patient requires it. On the other hand, in the event that the heart rate data exceeds the limits established as normal in the development of the software, an agenda will be registered automatically including the current time and date data, in which the last data was taken of the patient, thus ensuring that the patient-type user can be seen by a doctor as soon as possible.

Keywords: Monitoring system, web application, database, heart rate, doctor, patient, programming languages.

ÍNDICE

1. CAPÍTULO I. INTRODUCCIÓN

1.1 Antecedentes

El avance tecnológico presente en los últimos años ha tenido un fuerte impacto en el área de la medicina, teniendo en consideración lo anterior, un tema interesante se presenta en Cataluña en el año 2014 con una propuesta innovadora, la cual trata acerca de la visita médica no presencial, permitiendo que los datos del paciente se almacenen en una plataforma virtual haciendo posible que dicha información se encuentre siempre al alcance de quien lo necesite. A partir de esta iniciativa, se desarrolla el sistema informático *eBPlataform*, mismo que se encuentra bajo el concepto de *IoMT* siendo este último una rama de *IoT*, enfocado en la medicina o salud. Este sistema, utiliza un sensor *eBox* para monitorear la presión arterial de los pacientes con la ventaja de mantener los datos en un portal *web*, de tal forma que los médicos puedan proporcionar un tratamiento en línea (Lui, Niu, Yang, Shu, 2014).

Por otro lado, uno de los pioneros en el tema de *Smart Health* es la empresa de *Google*, quienes llevaron a cabo la propuesta de lanzar una banda inteligente que le sirva como ayuda a los médicos para poder llevar un monitoreo a distancia de los pacientes. El primer lanzamiento de este proyecto se da en el 2008, mismo que fracaso debido a la falta de usuarios. En el 2014 se vuelve a lanzar nuevamente el proyecto, pero con nuevo enfoque el cual se centra más en el control de ejercicio y nutrición (Llordachs, 2016).

Samsung, se convierte en la principal competencia de *Google* mediante *S-Health* una aplicación diseñada específicamente para llevar un registro por medio del monitoreo y control del estado físico de una persona (Bret, 2018).

Otro enfoque interesante se presenta en el 2017 por Oswaldo Arias estudiante egresado de la Universidad de las Américas, quien realiza la implementación de un sistema de monitoreo basado en la medición del pulso cardíaco y saturación de oxígeno en la sangre, utilizando sensores que recopilan los datos

del paciente cada cierto tiempo para posteriormente ser analizados, alojados y mostrados en una aplicación (Arias, 2017).

1.2 Planteamiento del problema

El ritmo de vida actual de las personas hace que estas no vean su salud como una prioridad. Así mismo, existen personas que tienen alguna enfermedad cardiovascular por su estilo de vida que necesita de una continuidad asistencial. Basados en este enfoque, las últimas estadísticas tomadas por la organización mundial de la salud en el año 2015 presentan a una cifra de 17,7 millones de personas que fallecieron, debido a problemas cardiovasculares que no fueron atendidos a tiempo. De la misma manera, este número representa un total del 31% de las muertes registradas en el mundo (OMG, 2015).

Los procedimientos actuales para controlar el estilo de vida de una persona son costosos y debido a esto, la accesibilidad a nuevas tecnologías que permitan monitorear en tiempo real los distintos factores responsables de un problema cardiovascular, como por ejemplo la frecuencia cardíaca y actividad física, son casi nulas.

1.3 Objetivos

1.3.1 General

Implementar un prototipo para la adquisición de parámetros de frecuencia cardíaca utilizando tecnología *IoMT*.

1.3.2 Específicos

- Analizar los requerimientos y componentes (*hardware* y *software*) para el diseño del sistema.
- Diseñar la arquitectura del *hardware* para el dispositivo propuesto.

- ❖ Implementar la red de datos y el sistema informático utilizando los requerimientos y componentes analizados.
- ❖ Efectuar las pruebas de funcionamiento del prototipo.

1.4 Justificación

Disminuir la cantidad de muertes y enfermedades por problemas cardiovasculares en la población ecuatoriana, mediante el desarrollo y la implementación de un sistema de red basada en la tecnología *IoMT* que le permita al usuario poder llevar un control continuo de su estado de salud, con la supervisión de un médico profesional y de confianza.

Así mismo, se pretende optimizar el tiempo necesitado por los médicos para evaluar el estado de salud de los pacientes, de tal manera que ambos tipos de usuarios pueda aprovechar al máximo su tiempo.

Con el desarrollo de este proyecto a más de prevenir un sin número de enfermedades a causa de problemas cardiovasculares, se busca también aportar información relacionada con *IoT* en la rama de la medicina, proporcionando resultados confiables que puedan ser utilizados como una base en la Universidad de las Américas para futuros proyectos de titulación orientados a *IoMT*.

2. CAPÍTULO II. MARCO TEÓRICO

Este capítulo aborda una descripción en términos generales de las diferentes tecnologías a utilizarse, así como también los distintos puntos de referencia necesarios a tomar en consideración, además la descripción de cada uno de estos para llevar acabo el desarrollo e implementación del tema general que utiliza los principios de la tecnología *IoMT.*

2.1 *IoT*

Internet of Things con sus siglas *IoT*, es considerado como la próxima evolución de Internet, debido a que permite conectar a una red inalámbrica varios elementos sin la necesidad de la intervención humana, estos distintos dispositivos pueden ir desde sensores, objetos mecánicos, instrumentos médicos, hasta objetos de la vida cotidiana, tales como ropa, calzado, entre otros.

Entre los distintos modelos de conectividad que se pueden hallar en esta tecnología son:

- *Device-to-Device* (dispositivo a dispositivo). – Esta arquitectura establece una conexión directa entre 2 o más dispositivos sin la necesidad de algún elemento intermediario que se encargue de procesar la información. Además, para llevar a cabo dicha comunicación esta arquitectura utiliza protocolos de comunicación entre los cuales se encuentran: *Bluetooth*, *Z-Wave* o *ZigBee*, como se puede observar en la figura 1, (Thaler, 2015).

Figura 1. Arquitectura *Device to Device*.
Tomado de (Thaler, 2015).

- *Device-to-Cloud* (dispositivo a la nube). – A diferencia de la anterior arquitectura, esta enlaza un dispositivo a una red con la capacidad de conectarse directamente a un proveedor de servicios en la nube,

mediante la utilización de *wifi* o cableado tradicional ethernet. Como se puede apreciar en la figura 2, (Thaler, 2015).

Figura 2. Arquitectura *Device to Cloud*.
Tomado de (Thaler, 2015).

- *Device-to-Gateway* (dispositivo a puerta de enlace). – Permite la conexión de un elemento utilizando el servicio *ALG* denominado también como modelo de dispositivo a puerta de enlace de capa de aplicación. Mismo que posee un intermediario o puerta de enlace que permite llevar acabo la conexión de un dispositivo cualquiera con un proveedor de servicios en la nube. Como se muestra a continuación en la figura 3. Este modelo permite un mejor manejo de los datos, debido a la traducción de protocolos, así como también una mayor seguridad (Thaler, 2015).

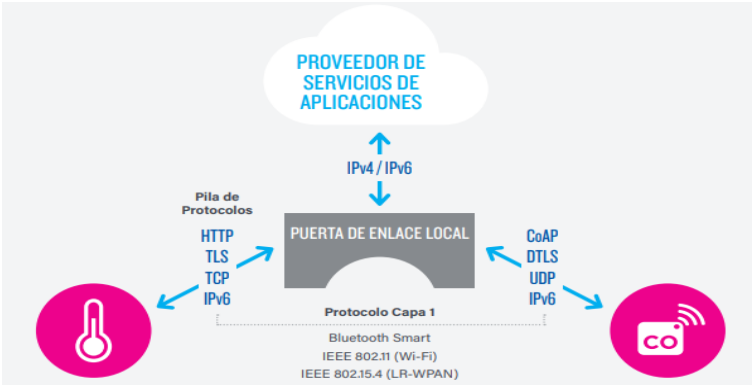


Figura 3. Arquitectura *Device to Gateway*.
Tomado de (Thaler, 2015).

- *Back-End Data-Sharing* (intercambio de datos a través del back-end). – La característica fundamental de este tipo de arquitectura es el manejo y la recopilación de información de varias fuentes que puede ser compartida con usuarios terceros, mediante la utilización de estándares basados en texto plano como *JSON* de *JavaScript*. Además, utiliza como base el modelo *device to cloud*. Como se puede observar en la figura 4, (Thaler, 2015).

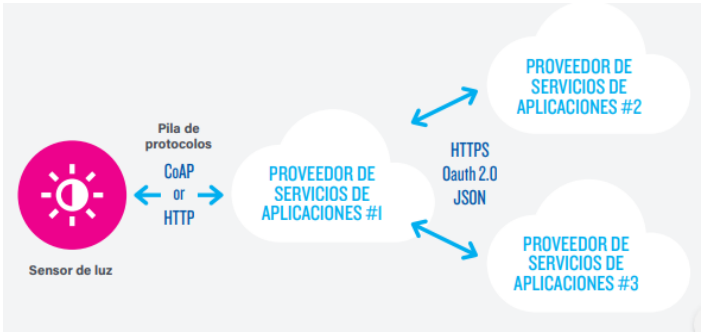


Figura 4. Arquitectura *BacK-End Data-Sharing*.
Tomado de (Thaler, 2015).

2.1.1 IoMT

IoMT denominado también como *Internet of Medical things* es la tecnología que se encarga de conectar varios instrumentos, dispositivos o sensores médicos a una red o sistema de TI, los mismos que se encargan de recolectar información de los signos vitales del paciente, de tal manera que se pueda monitorear en tiempo real a los usuarios para obtener un diagnóstico médico más preciso (Editorial Equipo, 2018).

La finalidad de *IoMT* es utilizar sensores para monitorear a los pacientes de manera constante, ahorrando tiempo y recursos. De la misma manera, entre las plataformas más utilizadas para llevar a cabo proyectos de este tipo se encuentran:

- *Arduino*. – *Arduino* es una plataforma de desarrollo basada en una placa electrónica de *hardware* libre que incorpora un microcontrolador reprogramable y una serie de pines hembra o macho, los que permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla.

  Cuando se habla de *Arduino* es fundamental especificar el modelo concreto, ya que se han fabricado diferentes modelos de placas *Arduino* oficiales, cada una pensada con un propósito diferente y características variadas (como el tamaño físico, número de pines E/S, modelo del microcontrolador, etc.). A pesar de las varias placas que existen todas pertenecen a la misma familia (microcontroladores AVR marca Atmel), esto significa que comparten la mayoría de sus características de *software*, como arquitectura, librerías y documentación, como se puede apreciar en la figura 5, (Arduino, s.f).

  Figura 5. Modelos de placas *Arduino*.
  Tomado de (Arduino, s.f).

- *CheapDuino*. – Dispositivo de tamaño reducido, como se puede ver en la figura 6, con gran capacidad de procesamiento, compatible con el entorno de desarrollo *Arduino*, el objetivo de este procesador es competir el en mercado tecnológico ofreciendo soluciones para distintos tipos de proyectos a bajos costos.

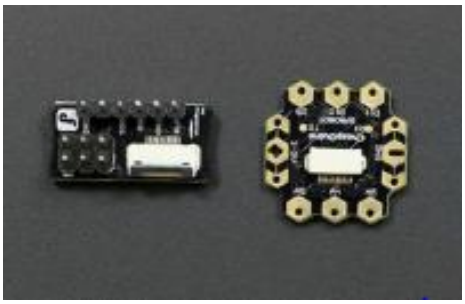


Figura 6. Microcontrolador *CheapDuino*.
Tomado de (DFROBOT, 2019).

- *Vocore2*. – Microprocesador de tamaño pequeño similar al de una moneda como se puede observar en la figura 7, que utiliza como sistema operativo una versión reducida de *Linux*, manteniendo las características de código abierto, además este dispositivo puede ser utilizado para distintos tipos de proyectos tales como por ejemplo llevar acabo conexiones VPN para controlar una red a distancia, así como también manipular estaciones con contenido multimedia, entre otros, etc.

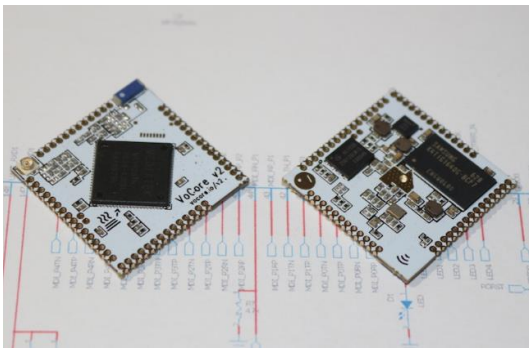


Figura 7. Microprocesador *vocore2*.
Tomado de (*Vocore2*, s.f).

- *Pic*. – Es considerado un microcontrolador debido a que es un dispositivo de procesamiento muy eficaz, ademas posee una amplia gama que se pueden ajustar a las necesidades del usuario, las cual van desde dispositivos con 6 pines hasta un máximo de 100. Así mismo, Microchip es la organización mas grande encargada de fabricar estos

microcontroladores. En la figura 8, se puede observar el microcontrolador *Pic* de la familia *PIC*16 de 8 bits.

Figura 8. Microcontrolador PIC16F84A.
Tomado de (Rossano, 2009).

- *Raspberry*. – Es un ordenador que se encuentra alojado en una placa de tamaño reducido, la cual se asemeja a una tarjeta de crédito. Esta tecnología nace o fue desarrollada en el Reino Unido por la fundación *Raspberry* Pi su logo se puede observar en la figura 9, con el objetivo de causar un impacto importante en las personas que desean estudiar y realizar proyectos que puedan satisfacer las necesidades del mundo actual.

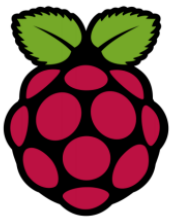


Figura 9. Logo *Raspberry*.
Tomado de (*Rossano, 2009*).

2.1.2 Telemedicina

El término de telemedicina hace referencia específicamente a la atención medica no presencial, por medio de la utilización de nuevas tecnologías que permitan prestar servicios de forma remota. Así mismo, se puede asociar este término con el *Internet of Medical Things*, debido a que facilita la conexión

inalámbrica de distintos dispositivos médicos a un sistema configurado de tal manera que permita controlar y monitorear de forma remota a los pacientes. Además, con la nueva presencia y el lanzamiento de 5G, se espera llevar esta tecnología a un paso más grande, en el cual se pueda manejar datos médicos de poblaciones enteras (Nuñez, 2016).

2.2 Tecnologías de comunicación Inalámbricas

Entre las dos tecnologías de comunicación inalámbricas más utilizadas por *IoT* se encuentran:

- ***Wifi*.** – Tecnología de conexión inalámbrica para redes de área local, se basan en el estándar IEEE 802.11 que permite la movilidad de los usuarios en un área de terminada, con bajos costos de implementación (Castro, 2019). Además, es importante tener en consideración las distintas versiones que existen a partir de dicho estándar, esto se puede apreciar de mejor manera en la tabla 1.

  Tabla 1.
  Estándares Wifi.

	802.11	**802.11a**	**802.11b**	**802.11g**	**802.11n**
Año de creación	1997	1999	1999	2003	2009
Frecuencia		5 Ghz	2.4 Ghz	2.4 Ghz	2.4 Ghz – 5 Ghz
Velocidad	2 Mbps	54 Mbps	11 Mbps	50 Mbps	600 Mbps
Compatibilidad		802.11a - 802.11n	802.11b - 802.11g - 802.11n	802.11b - 802.11g - 802.11n	802.11a - 802.11b - 802.11g - 802.11n

Comentarios	Obsoleto	Limitado	Mejor alcance

Adaptado de (Cemeblog, 2019).

- *Bluetooth Low Energy*. – Denominado también como *Bluetooth Smart*, debido a la reducción de consumo y bajos niveles de transmisión erróneos con velocidades de 1 Mbps que utiliza mecanismos de cifrado *AES* de 128 bits, además, fue presentado alrededor del año 2010 y fue integrado a la versión 4.0.

2.3 Medicina y Salud

El estudio y desarrollo de nuevas tecnologías están revolucionando el área de la medicina, proporcionando mejoras que permitan llevar un monitoreo constante que asegure un mejor control de los pacientes, de tal manera que tanto los médicos como dichos pacientes puedan aprovechar al máximo su tiempo.

2.3.1 Constantes Vitales

Según (EP, 2019), los signos vitales también denominados como constantes son aquellos indicadores que nos permiten obtener información sobre el estado de salud o fisiológico de una persona. Entre los principales indicadores utilizados en la medicina o en la práctica clínica son:

- Frecuencia Cardíaca (FC). – Parámetro que se encarga de medir el número de latidos del corazón trascurridos en un minuto, su unidad de medida es (lpm).
- Frecuencia Respiratoria (FR). – Indicador que se encarga de contar el número de ciclos respiratorios completos que efectúa una persona en un minuto.

- Presión Arterial (PA). – Permite determina la fuerza que ejerce la sangre en el torrente circulatorio durante el ciclo cardíaco, además la Tensión Arterial máxima o sistólica corresponde a la contracción del ventrículo izquierdo para bombear la sangre y la Tensión Arterial mínima o diastólica hace referencia a la dilatación de éste, su unidad de medida es (mmhg).
- Temperatura (Tª). – Es el parámetro que se encarga de determinar el resultado calórico de los procesos metabólicos realizados por el organismo, su unidad de medida es (°C o °K).

2.3.1.1 Frecuencia Cardíaca

La frecuencia cardíaca es un dato que se obtiene en un determinado intervalo de tiempo y es el número de veces que late el corazón en dicho tiempo. Así mismo, este valor puede apreciarse de mejor manera en ciertos puntos del cuerpo humano, tales como: cuello, parte trasera de las rodillas, muñeca, entre otros (NCI, 2019).

Por otro lado, el valor de la frecuencia cardíaca en las personas depende de su estilo y hábitos de vida diarios, por lo general cuando una persona adulta se encuentra en estado de reposo oscila entre 50 y 100 latidos por minuto. En las Tablas 2 y 3 se puede apreciar de mejor manera como la edad y otros factores afectan directamente al rango de latidos por minuto establecido como normal (SEC, 2019).

Tabla 2.

Frecuencia Cardíaca según la edad y género, en estado de reposo (Hombres).

HOMBRES				
Edad	Mal	Normal	Bien	Excelente
20 – 29	86 +	70 – 84	62 – 68	60 o menos
30 - 39	86 +	72 – 84	64 – 70	62 o menos
40 - 49	90 +	74 – 88	66 – 72	64 o menos

50 +	90 +	76 – 88	68 – 74	66 o menos

Adaptado de (SEC, 2019).

Tabla 3.

Frecuencia Cardíaca según la edad y género, en estado de reposo (Mujeres).

MUJERES				
Edad	Mal	Normal	Bien	Excelente
20 - 29	96 +	78 – 94	72 – 76	70 o menos
30 - 39	98 +	80 – 96	72 – 78	70 o menos
40 - 49	100 +	80 – 98	74 – 78	72 o menos
50 +	104 +	84 – 102	76 – 86	74 o menos

Adaptado de (SEC, 2019).

2.3.1.2 Métodos de Medición Convencional

Método Manual

Según la (SEC, s.f), este método consiste en tomar datos de la frecuencia cardíaca con la ayuda de un especialista, teniendo en consideración el número de pulsos que se presentan en la muñeca de una persona, para esto es necesario llevar acabo los siguientes pasos:

1. Presionar de forma suave la parte interna de la muñeca del lado del pulgar bajo.
2. Colocar los dedos índice y corazón entre 1 y 2 cm por debajo del pliegue de la muñeca.
3. Calcular el número de latidos o pulsaciones durante un período de 30 segundos y multiplicarlo por 2.

Método con Dispositivo electrónico

Actualmente, se utilizan diferentes tipos de dispositivos preparados y acondicionados para tomar mediciones de las constantes vitales,

específicamente de la frecuencia cardíaca de los pacientes, entre los más destacados podemos encontrar:

- Correas de pecho. – Este dispositivo se encarga de medir la frecuencia cardíaca mediante la toma de muestras y el análisis de una señal eléctrica que ocasiona el cuerpo humano cuando el corazón se contrae.
- Monitor De Ritmo Cardíaco. – Este terminal permite detectar, procesar y desplegar los datos de la frecuencia cardíaca de forma continua.
- *SmartBand*. – El principio de funcionamiento de estos dispositivos electrónicos inteligentes es la toma de muestras del pulso cardíaco mediante la utilización de sensores, los cual esta diseñados específicamente para tomar datos de este tipo.

2.4 Desarrollo de *Software*

Es necesario tener en consideración distintos parámetros o puntos de referencia para el correcto desarrollo de un *software*, de tal manera que se pueda satisfacer al máximo las necesidades del usuario teniendo como punto de partida y base los requerimientos proporcionados por dicho usuario.

2.4.1 Lenguajes de programación

Según (TIOBE, 2019) empresa holandesa dedicada al desarrollo de *software*, entre los leguajes de programación más utilizados en los últimos 5 años para el desarrollo de *software* están:

- *Java*
- *C*
- *Python*
- *C++*
- *C#*
- *Visual Basic .NET*
- *JavaScript*

- *SQL*
- *PHP*
- *Objetive-C*

2.4.2 Herramientas y plataformas de programación

DBMS

Sistema gestor de base de datos, es aquel que consiste en una colección de datos interrelacionados y una colección de programas para acceder a estos datos (Korth, 2002). Entre las características, mas importantes que un *DBMS* posee, se encuentran:

- Acceso concurrente de usuarios.
- Control de transacciones.
- Permite un lenguaje de consulta para recuperar datos.
- Tiene medidas de respaldo (*backup*).
- Proporciona medidas de recuperación en el caso de fallos.
- Aporta mecanismos de seguridad para evitar consultas y modificación no autorizadas de datos.

Por otro lado, en el mercado actual existen varios tipos de *software* desarrollados bajo estas características, entre los *DBMS* más utilizados están:

- *SQL Server*
- *MySQL*
- *Oracle*
- *PostgreSQL*
- *Informix*

Visual Code

Es una herramienta considerada como editor de código fuente, que es ligera, pero mantiene la característica de ser potente y además está disponible para múltiples plataformas, además, incorpora soporte para *JavaScript*, *TypeScript*, *Node.js*, entre otros lenguajes de programación como *C++, C#, PHP*, *Java*, *Python*, (.*NET*), etc. (VisualStudio, s.f).

Android Studio

Según (AndroidStudio, s.f), Es un entorno de desarrollo integrado basado en *IntelliJ,* para soluciones y desarrollo de aplicaciones para dispositivos con sistemas operativos *Android*, también, integra varias características como:

- Un sistema de compilación flexible basado en *Gradle*.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puedes desarrollar para todos los dispositivos *Android*.
- Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación.
- Integración con GitHub y plantillas de código para ayudar a compilar funciones de aplicaciones comunes y también importar código de ejemplo.
- Variedad de marcos de trabajo y herramientas de prueba.
- Herramientas de pelusa para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros.
- Compatibilidad con C ++ y NDK.
- Compatibilidad integrada para *Google Cloud Platform*, que facilita la integración con *Google Cloud Messaging* y *App Engine*.

Herramienta *CASE*

Es un conjunto de programas desarrollados para llevar a cabo una asistencia por parte de los analistas de *software* y desarrolladores, quienes son los

encargados de realizar un seguimiento a los ciclos de investigación preliminar, análisis, diseño, implementación e instalación (León, s.f).

2.4.3 Modelo 3 Capas

(Romero, 2019), explica que este patrón surge de la necesidad de “reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos”. Por ende, lo que se consigue con esto es simplemente dar más dinamismo al desarrollo de aplicaciones o *software* de tipo *web*.

Además (Romero, 2019), explican que “al incorporar el modelo de arquitectura *MVC* a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

De forma general los componentes de un modelo de 3 capas *MVC* está compuesto por:

- Modelo. – También denominado capa de negocio, en la cual se encuentran los distintos atributos del diseño de la base de datos. Además, este se encarga de enviar los datos solicitados a la capa de vista.
- Controlador. – Es el encargado de realizar las acciones programadas y desarrolladas en la capa de vista, actuando como un intermediario que maneja la información.
- Vista. – La interfaz de usuario también conocida como vista, se encarga de presentar la información al usuario final. Es importante, mencionar que esta interfaz debe mantener características que le faciliten el manejo y sean amigables con el usuario (Carrión, 2015). En la figura 10, se puede apreciar de mejor manera como se encuentran distribuidas las distintas capas y como se relacionan entre ellas.

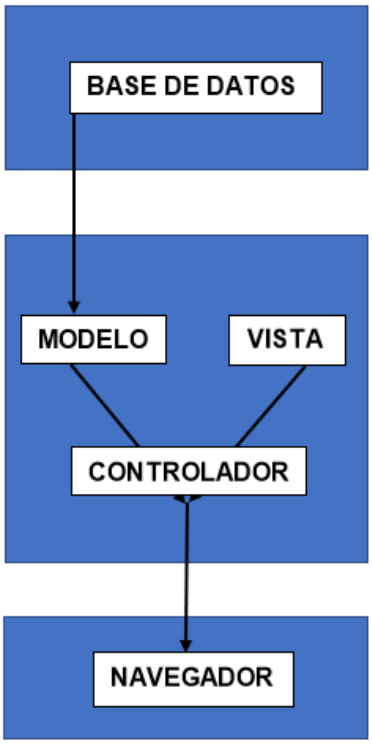


Figura 10. Arquitectura modelo 3 capas.
Adaptado de (Carrión, 2015).

2.5 *Hosting Web*

Hosting web hace referencia a un espacio de almacenamiento rentado que permite publicar distintas aplicaciones y recursos en base a las necesidades, además es fundamental mencionar que esta publicación se lo hace en servidores preparados específicamente para la demanda que este tema conlleva, de tal manera que el acceso a estas publicaciones se encuentre siempre disponible. Por otro lado, dependiendo de las necesidades del usuario existen distintos tipos de *hosting* entre los más comunes están:

- *Hosting* Compartido. – Este tipo de alojamiento comparte el servidor, así como también sus recursos con distintos usuarios que pertenecen al mismo proveedor. Es importante tener en consideración las ventajas y desventajas como se muestran en la tabla 4.

Tabla 4.

Ventajas y desventajas modelo Hosting Compartido.

VENTAJAS	DESVENTAJAS
Bajo costo	Poco o ningún control sobre la configuración del servidor
Fácil de usar para principiantes (no se requieren conocimientos técnicos específicos)	
Servidor preconfigurado	Los aumentos del tráfico en otros sitios *web* pueden ralentizar tu sitio
El mantenimiento y la administración del servidor son realizados por el proveedor	

Adaptado de (Hostinger, s.f).

- *Hosting VPS* (Servidor privado virtual). – Este servicio se caracteriza por proporcionar una partición dedicada de un servidor compartido entre varios usuarios del mismo proveedor. De la misma manera, es necesario tener en consideración los siguientes pros y contras que puede presentar en este tipo de servicio, como se muestra en la Tabla 5.

Tabla 5.

Ventajas y desventajas modelo Hosting VPS.

VENTAJAS	DESVENTAJAS
Espacio dedicado de un	Alto Costo

servidor compartido	
Acceso al servidor raíz, manteniendo altos niveles de personalización	
Escalable	Difícil administración del servidor

Adaptado de (Hostinger, s.f).

- *Hosting* en la nube (*Cloud Hosting*). – La característica principal de este servicio es el almacenamiento de los datos en diferentes servidores, de tal manera que se puede garantizar la máxima disponibilidad al usuario para acceder a su información. En la tabla 6, se muestra más detalles sobre este tipo de hosting.

Tabla 6.
Ventajas y desventajas modelo Hosting Cloud.

VENTAJAS	DESVENTAJAS
Alta disponibilidad	Difícil estimación de los costos y acceso al servidor raíz.
Se ajusta a las necesidades del usuario	
Escalabilidad	

Adaptado de (Hostinger, s.f).

- *Hosting WordPress*. – Es un tipo de hosting preparado específicamente para desarrolladores de *WordPress* con herramientas establecidas como plantillas, *plugins*, entre otras que le permitan al desarrollador arrastrar y soltar, dichas herramientas según las necesidades del usuario. También,

es fundamental considerar las ventajas y desventajas que presenta este tipo de hosting como se puede observar en la tabla 7.

Tabla 7.
Ventajas y desventajas modelo Hosting WordPress.

VENTAJAS	DESVENTAJAS
Bajo costo	Solo sitios del tipo *WordPress*
Diseñado para usuarios con poco conocimiento en el desarrollo de *software*	
Soporte técnico	
Herramientas preparadas para *WordPress*	

Adaptado de (Hostinger, s.f).

- *Hosting* con servidor dedicado. – A diferencia de los anteriores este mantiene a disposición del usuario un servidor físico dedicado para el manejo de información. Sin embargo, es fundamental tener en consideración las ventajas y desventajas que este tipo de servicio presenta, como se muestra en la tabla 8.

Tabla 8.
Ventajas y desventajas modelo Hosting con servidor dedicado.

VENTAJAS	DESVENTAJAS
Altos niveles de confiabilidad	Conocimiento necesario en el área técnica para la administración de
Acceso completo al	

servidor	servidores
Mayor seguridad	Altos costos

Adaptado de (Hostinger, s.f).

3. CAPÍTULO III. DISEÑO Y DIMENSIONAMIENTO DE LA RED MÉDICA

Las etapas de diseño y dimensionamiento son dos de los aspectos más importante a tener en consideración, debido a que estos apartados se encargan de evaluar y seleccionar los distintos tipos de tecnologías disponibles y definir las herramientas necesarias tanto de *hardware* como de *software* para llevar acabo el desarrollo del proyecto.

En términos generales, en el presente capitulo se llevará acabo el análisis de los requerimientos y la problemática presentados en el capítulo 1, para elaborar el desarrollo eficiente del dimensionamiento de la topología de la red médica a utilizarse, así como también el diseño del *hardware* y *software*, teniendo en consideración que al hablar de *hardware* este se refiere al prototipo electrónico, mientras que en el apartado de *software* se encuentran las aplicaciones *web* y móvil, las cuales se encargan del manejo de información del prototipo.

3.1 Procesamiento de datos

Envió de datos de forma directa y sin procesar

Al momento de realizar un envió de datos, los cuales no han sido procesados, el resultado va a verse afectado en parámetros como la velocidad, ya que, si se procesan los datos dentro de un entorno de desarrollo *web*, hay que considerar un proceso que se encargue de englobar los siguientes pasos:

- Validación
- Clasificación
- Recapitulación

- Agregación
- Análisis
- Información

Cada uno de estos pasos, conllevan al incremento de tiempo dentro de la presentación de información para el usuario, estos tiempos variarán dependiendo de la capacidad de procesamiento.

Por lo cual, se puede concluir que realizar este tipo de prácticas no es muy recomendable, ya que, se van a consumir recursos que son necesarios para otras acciones, además, la página *web* donde se presentarán estos datos se va a ver congelada, ya que se está haciendo dos procesos a la vez, uno el procesamiento de los datos y dos el procesamiento para mostrar estos datos, lo cual conlleva a tener una baja confiabilidad, escalabilidad, robustez y concurrencia.

Envío de datos procesados

Esta es una de las mejoras prácticas ya que se va a usar un microcontrolador dedicado, solo para poder procesar de forma correcta estos datos, a este tipo de procesamiento se lo conoce como procesamiento electrónico de datos, es decir que todos los pasos antes mencionados para realizar un óptimo proceso de datos, van a usar recursos y tiempos de una manera dedicada solo para realizar esa acción y con esto además, se consigue que haya una modularidad dentro del sistema, ya que si al momento de que se dañe la página *web*, no va a ser un problema enviar estos datos a otra página de respaldo, logrando así que exista un alto grado de disponibilidad, integridad y seguridad de dichos datos.

3.2 Red Médica

En base a los estudios realizados se ha determinado que una de las tecnologías de conexión inalámbrica más utilizada en la actualidad es *wifi*, por su velocidad, niveles de cobertura y concurrencia de usuarios, misma que se habla de forma general en el capítulo 2, además, los distintos tipos de diseños presentados en el punto 2.1 del mismo capítulo, habla acerca de las diferentes arquitecturas que se pueden utilizar en la tecnología *IoT*, teniendo en consideración todo lo mencionado anteriormente, se ha optado por utilizar la arquitectura con *Device to Gateway*, ya que esta permite conectarse a un *Gateway* como medio de propagación para enviar información por una red *LAN* hacia internet.

Una vez seleccionada la arquitectura a utilizarse para el desarrollo del proyecto se procede a realizar el diseño de la red. Como se puede apreciar en la figura 11.

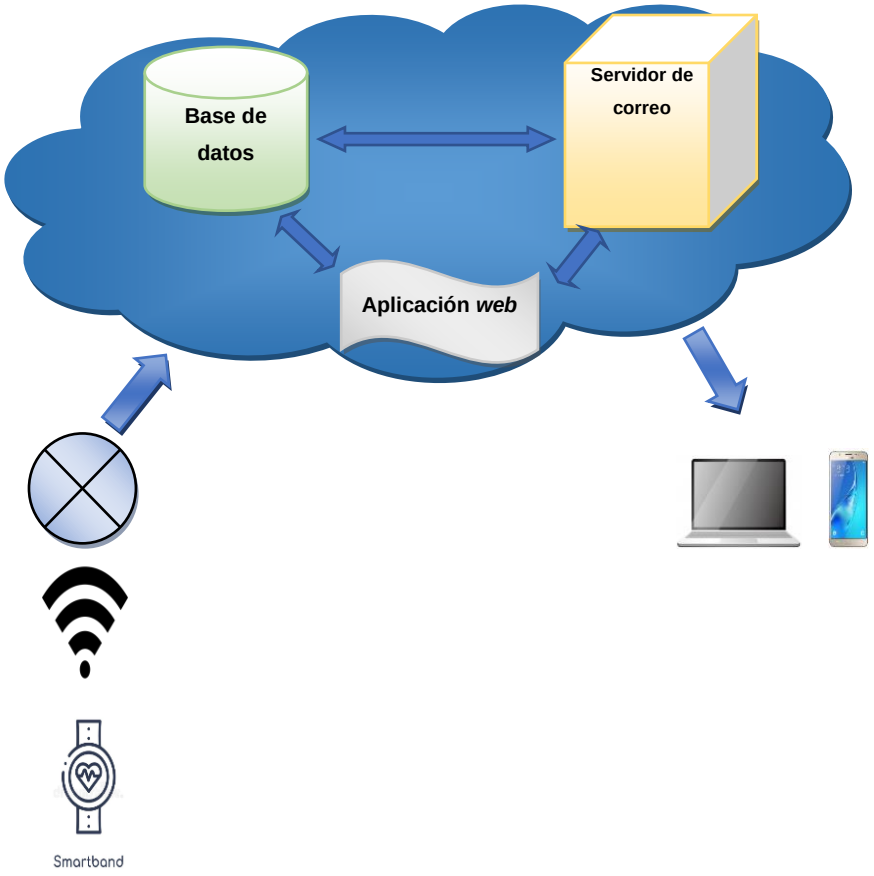


Figura 11. Topología de Red médica.

La presente topología de red está diseñada específicamente para que el prototipo pueda conectarse de forma inalámbrica con una red *LAN*, de tal manera que esta puede enviar los datos receptados por el sensor integrado en dicho prototipo electrónico hacia una base de datos alojada en la nube. De igual forma, esta contendrá en su interior más servicios como correo electrónico y una página *web*. La ventaja de usar esta arquitectura es que permite mantener disponible la información en todo momento, asegurando la integridad y disponibilidad de los datos.

El usuario podrá iniciar sesión o registrarse en la página *web*, dicha página contendrá y administrará toda la información del dispositivo o prototipo electrónico. De la misma manera, dependiendo del tipo de usuario el sistema diferenciará las cuentas de médico y paciente, debido a que la interfaz del paciente es distinta a la del médico. Además, cada vez que se registre un nuevo médico o paciente el sistema automáticamente enviará un mensaje de correo electrónico dándole la bienvenida al usuario, así como también enviándole las credenciales para iniciar sesión y registrándolo en la base de datos.

3.3 *Software*

3.3.1 Base de Datos

DBMS

Entre los sistemas de gestión de Bases de Datos o *DBMS* más utilizados para el manejo de información y desarrollo de bases de datos, se encuentran *SQL Server* y *MySQL*, entre los cuales se llevará a cabo un análisis de ventajas y desventajas, como se muestra en la Tabla 9. El proceso de elección del *DBMS* es fundamental para el desarrollo del proyecto, debido que es el *software* encargado de gestionar toda la información del sistema médico.

Tabla 9.

Ventajas y desventajas SQL Server vs MySQL.

	SQL Server	**MySQL**
Empresa	*Microsoft*	*MySQL AB*
Sistema Operativo	*Windows, Linux*	Múltiples SO
Lenguaje *underlying*	C++ en su mayor parte, con pocas integraciones en C.	C, C++
Idioma	Varios idiomas	Ingles
Sintaxis	Simple y fácil	Poco compleja
Copias de Seguridad	Completas, parciales e incrementales	Diferenciales, completas, a nivel de archivo e incrementales
Licencia	Comercial	Código abierto

Adaptado de (EDUCBA, s.f).

En base al análisis de las características que presenta cada tipo de *RDBMS*, se ha optado por seleccionar *MySQL* para el desarrollo de la base de datos del sistema médico, debido a que este sistema de gestión para base de datos posee una licencia de código abierto, así como también soporta múltiples sistemas operativos, lo que lo hace una solución óptima para proyectos con características escalables. De la misma manera, este *RDBMS* permite trabajar en un entorno de tipo cliente-servidor.

Antes de realizar el desarrollo de la base de datos, es fundamental efectuar un estudio de los requerimientos especificados en el capítulo 1, para representar de manera gráfica en forma de cuadros las distintas entidades y atributos que

presenta este sistema, así como también sus respectivas relaciones. Este tipo de gráfico se lo conoce como diseño lógico.

Diagrama Lógico

Por otro lado, para la realización del diseño lógico, como se puede ver en la figura 12, se utilizará la herramienta *CASE* denominada *Dezing*, la cual es una herramienta de diseño para bases de datos que proporciona una ayuda para modelar, crear y mantener bases de datos eficientes y con buen rendimiento. También, utiliza diagramas de entidades relacionales y ofrece un ambiente de modelaje sofisticado, de tal manera que el desarrollo de una base de datos sea más sencillo.

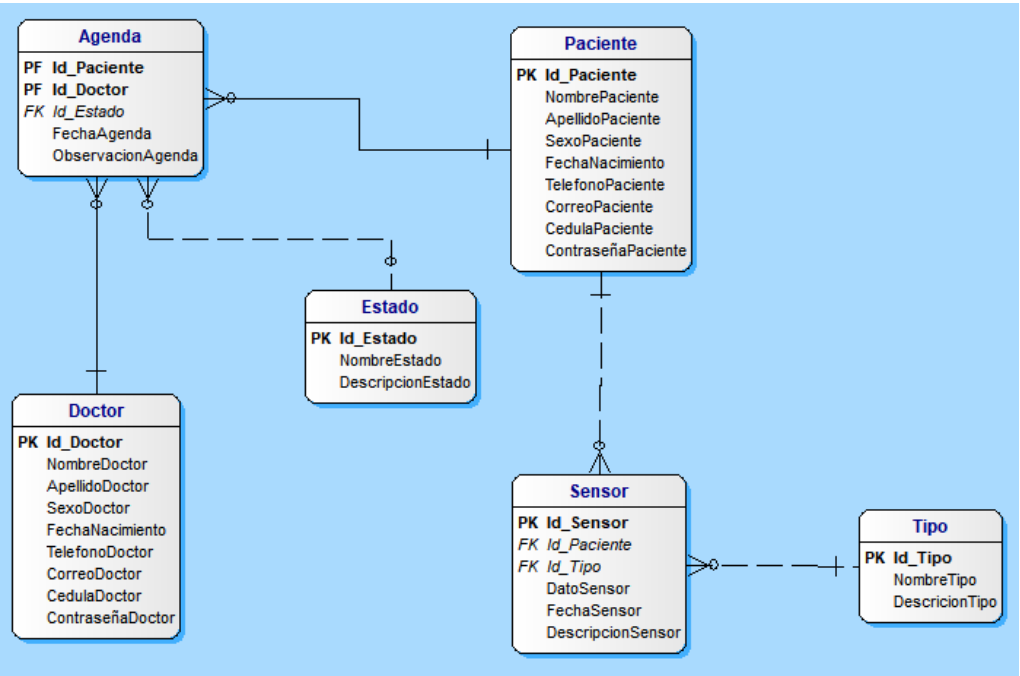


Figura 12. Diseño lógico del sistema médico.

El diseño de la base de datos se encuentra realizado con base al modelo de funcionamiento del aplicativo. Las entidades manejadas dentro de la base de datos se encuentran representando a todos los partícipes del proceso, siendo estos pacientes, doctores, agenda, estado, sensor y tipo de sensor. El registro

médico que se lleva a cabo en la tabla sensor se encarga de almacenar los datos enviados a través de un dispositivo electrónico, además este registrará toda la información correspondiente acerca del estado de salud de un paciente. El paciente se relaciona con la agenda tomando en cuenta que un paciente puede tener diversos registros, además de encontrarse con varios médicos a través de una tabla intermedia que rompe una relación de muchos a muchos entre ambas entidades, teniendo como resultado o formando la entidad agenda.

3.3.2 Aplicación *Web*

La aplicación *web* es la encargada de gestionar toda la información enviada por el prototipo electrónico, esta aplicación debe ser amigable con el usuario tanto en la manipulación como en la interfaz gráfica. Por lo que es fundamental llevar a cabo una evaluación de los distintos lenguajes de programación, entornos de desarrollo y herramientas presentes en la actualidad. En la Tabla 10, se muestran varios lenguajes de programación cada uno con sus respectivas características.

Tabla 10.
Lenguajes de programación para desarrollar aplicaciones web.

Lenguajes de programación				
	Características	**Ventajas**	**Desventajas**	**Sistema Operativo**
Java	Lenguaje de programación orientado a objetos.	Aplicaciones de escritorio, con ejecución independient	Actualizaciones. Necesita de u interprete.	Multiplataform a.

	Independiente de la plataforma.	e. Soluciones para entornos de desarrollo *web*. Desarrollo de aplicaciones distribuidas. *Garbage collector*. Soporte para aplicaciones móviles.		
C#	Lenguaje de programación orientado a objetos. Orientado a componentes. Desarrollado por *Microsoft*.	Declarar una o varias clases en un mismo espacio de nombres. Multihilo. Mayor rango para definir tipos de datos Atributos del	Inconvenientes de ejecución si no se utiliza un *framework* de *Microsoft*.	*Windows*. Proyecto mono multiplataform a

		tipo: público, protegido, interno, interno protegido y privado. Adaptable para desarrollar aplicaciones de escritorio, *web* y móvil.		
Visual Basic .NET	Lenguaje de programación basado en eventos.	Formularios de *Windows*. Acceso a gran parte de las Api y librerías de *Windows*. Soporte para ejecutar scripts. Librerías DDL y componentes ActiveX.	Problemas con versiones anteriores a vb.net. Poco soporte para el desarrollo de programación orientada a objetos.	*Windows*.

JavaScript	Lenguaje de programación scripting.	Lenguaje de programación sencillo. Desarrollo de aplicaciones dinámicas y *web*. *FullStack*. Buenos efectos visuales. El código se ejecuta desde el servidor hacia el cliente.	Código visible en el *FrontEnd*. Ejecución distinta por compatibilidad en diferentes navegadores.	Multiplataforma.
PHP	Lenguaje de programación interpretado.	Aplicaciones *web* Dinámicas. Código abierto. Lenguaje interpretado en	No se puede ocultar el código fuente. Ofuscación.	Multiplataforma.

ejecución.

La mayoría de los lenguajes de programación presentados anteriormente en la tabla 10, poseen varias características interesantes para el desarrollo de la aplicación *web*, sin embargo, se ha optado por utilizar como lenguaje de programación principal *PHP*, ya que es de código abierto y está diseñado específicamente para páginas *web* dinámicas. De la misma forma, se utilizará en el lado del servidor el lenguaje de programación *JavaScript* como un complemento para la conexión con la base de datos y el grafico estadístico, también se utilizará hojas de estilos o *Css* para un mejor acabado del diseño.

3.3.2.1 Diagramas de Flujo

Un diagrama de flujo representa de forma gráfica, los procesos que se pueden llevar acabo en la aplicación *web*. Los diagramas referentes a cada clase de la aplicación se los puede mostrar a continuación en los siguientes literales.

Clases Paciente y Médico

Los procesos de pacientes y médicos son similares, por lo que se ha optado por realizar un diagrama de flujo de forma general para usuarios. En la figura 13, se muestra de forma gráfica el diseño del diagrama de flujo con los distintos procesos que se pueden llevar acabo en paciente y médico. Por otro lado, en la figura 14, se puede observar como el usuario puede iniciar sesión en la aplicación, de tal manera que dependiendo del tipo de usuario se dirige a una pestaña distinta, también, en la misma figura se muestra el proceso a efectuar para realizar una búsqueda de usuarios, es importante tener en consideración que para la búsqueda es necesario ingresar el número de cédula.

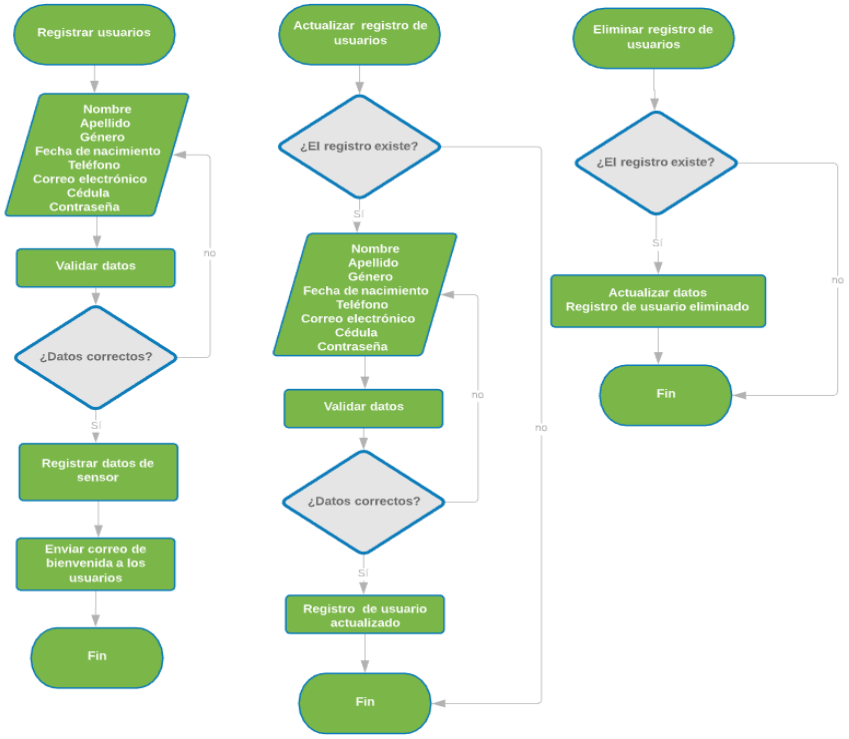


Figura 13. Diagramas de flujo para Pacientes y Médicos.

En la figura 14, se pueden apreciar las distintas acciones y procesos que la aplicación *web* puede realizar por medio de los diagramas de flujo.

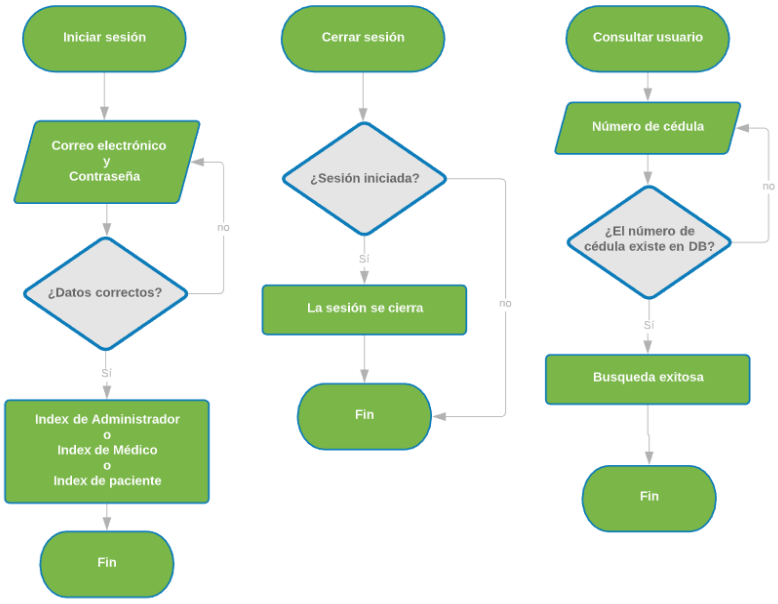


Figura 14. Diagramas de flujo de procesos adicionales para usuarios.

Clase Agenda

La entidad agenda definida en la base de datos puede llevar acabo procesos como:

- Registrar
- Actualizar
- Eliminar

Todos estos procesos se los puede percibir de mejor manera en la figura 15, donde se muestra de forma gráfica y mediante un diagrama de flujo, las distintas etapas que componen estos procesos.

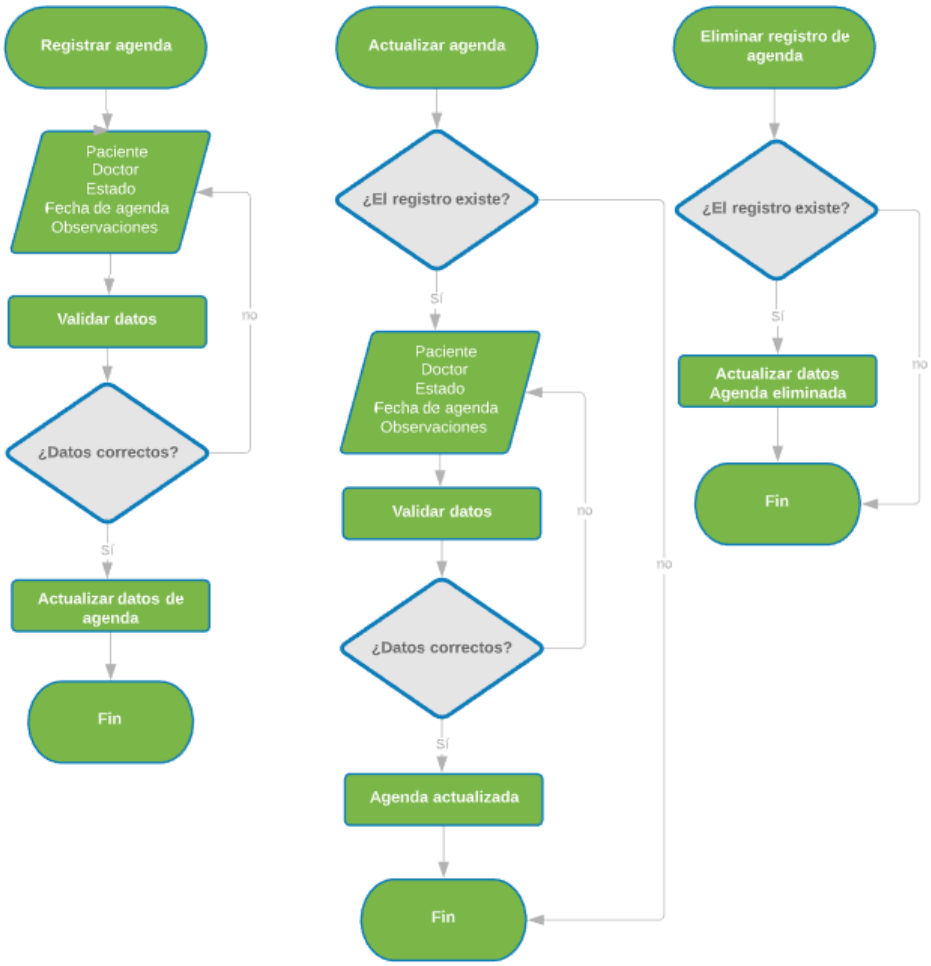


Figura 15. Diagramas de flujo de Agenda.

Clase Estado

La entidad estado, se encarga de registrar, modificar y eliminar los datos del estado de los pacientes, como complemento a la información de la agenda. Estos procesos se los puede ver de mejor manera en la figura 16.

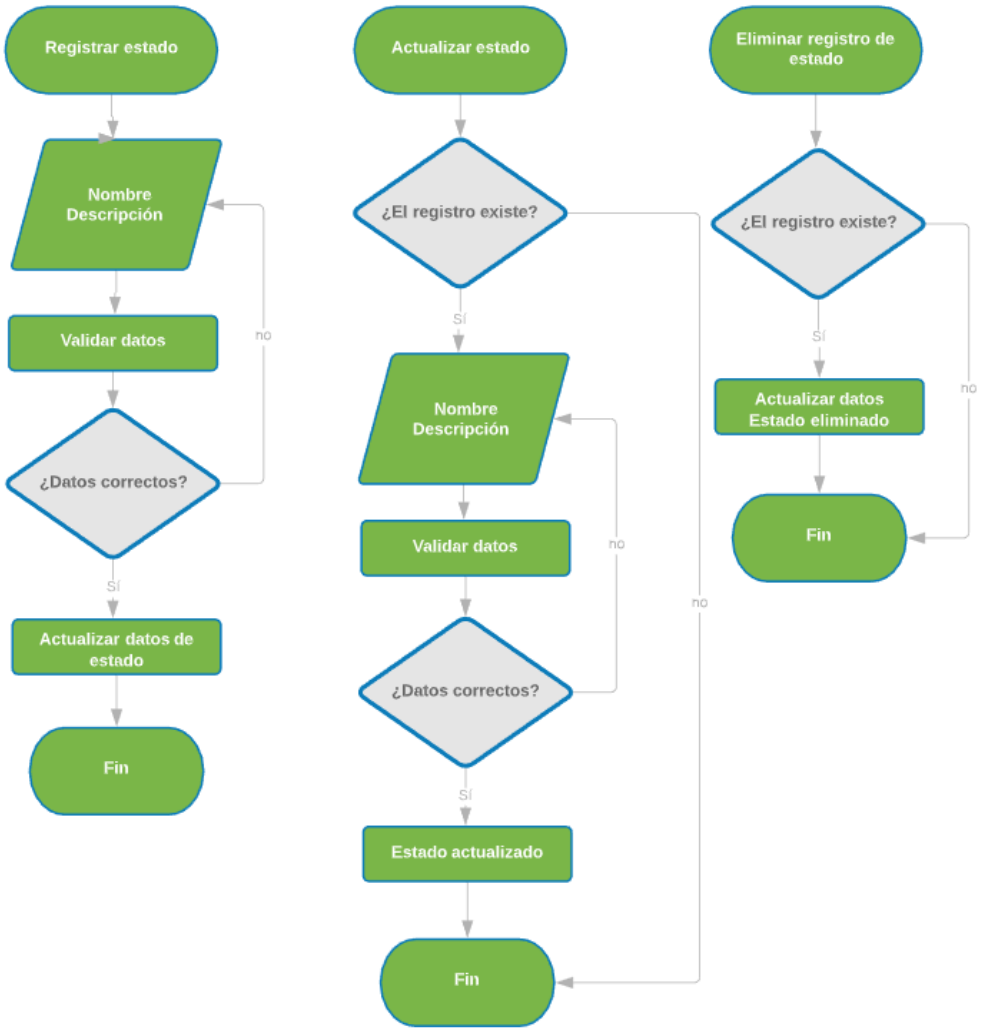


Figura 16. Diagramas de flujo de Estado.

Clase Tipo Sensor

El tipo sensor se encarga de gestionar los sensores mediente los procesos: registrar, modificar o eliminar. En la figura 17, se puede apreciar de mejor manera estos procesos en un diagrama de flujo.

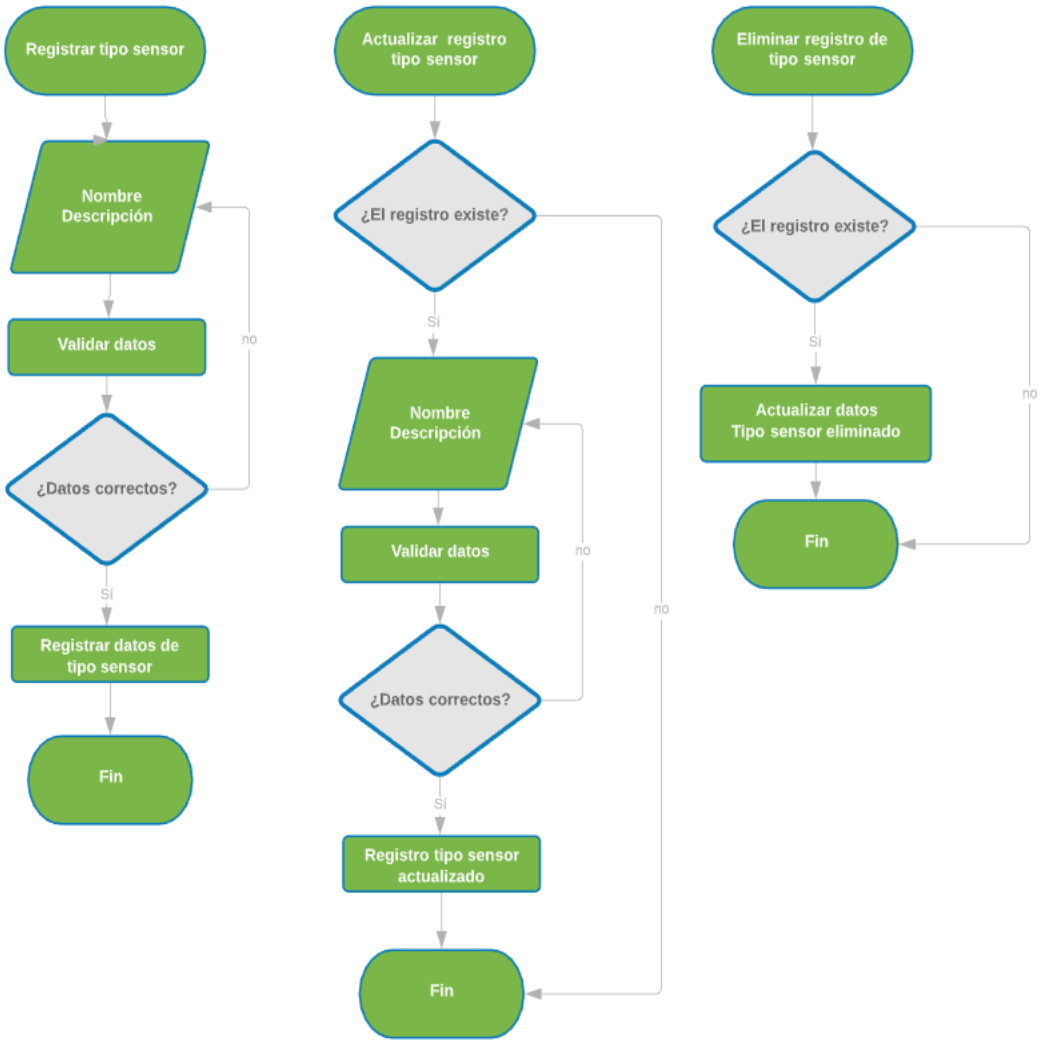


Figura 17. Diagramas de flujo de Tipo Sensor.

Clase Sensor

La entidad sensor a diferencia de las demás, se encarga de administrar toda la información emitida por los sensores del prototipo electrónico, mediante procesos como:

- Registrar
- Modificar
- Eliminar

Estos procesos se los puede percibir con más detalles en la figura 18.

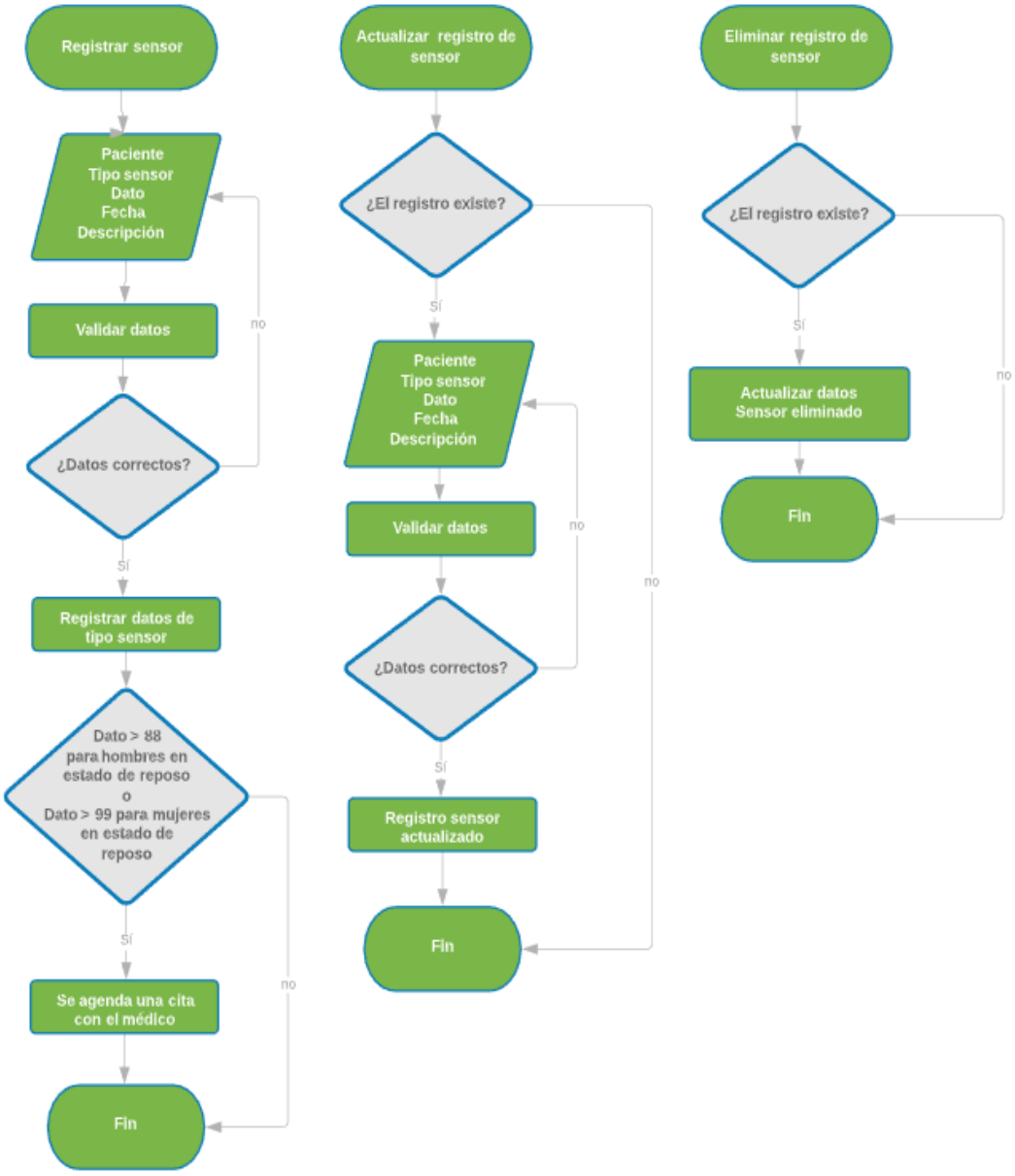


Figura 18. Diagramas de flujo de Sensor.

3.3.3 Aplicación Móvil

El desarrollo de una gran variedad de aplicaciones móviles en la actualidad ha tenido un gran impacto en la población mundial, esto conlleva a la creación y distribución de distintas herramientas para el desarrollo de estas aplicaciones,

en la tabla 11 se presenta una comparativa entre las plataformas más utilizadas hoy en día.

Tabla 11.
Herramientas para desarrollar aplicaciones móviles.

Herramientas para desarrollar aplicaciones móviles en *Android*.		
	Android Studio	***Xamarin***
Lenguaje de programación.	*Java y C++*	*C#*
Código compartido.	No	El mismo código se puede ejecutar en distintas plataformas.
Tipo de aplicación.	Nativa.	Multiplataforma.
Diseño	Mejor diseño para aplicaciones nativas.	Diseño aceptable al ser multiplataforma.

Según la evaluación de la tabla 11, se ha optado por seleccionar como entorno de desarrollo a *Android Studio*, ya que permite la creación de aplicaciones nativas con interfaces amigables para el usuario, utilizando lenguajes de programación como *Java y C++.*

Diagrama de flujo

La figura 19 representa de forma gráfica el proceso que ejecuta la aplicación móvil, mediante la utilización de un diagrama de flujo donde se detallan las distintas etapas que debe cumplir dicha aplicación.

Figura 19. Diagramas de flujo de la aplicación móvil.

3.3.4 *Web Hosting*

000webhost

Este *web hostig* tiene como característica principal la compatibilidad directa con el lenguaje de programación *PHP*, además, posee un *cPanel* personalizado y fácil de utilizar para el usuario, también, en el ámbito de seguridad ofrece de forma gratuita el certificado *SSL* que se encarga de mantener los sitios web seguros y acceder a estos mediante *HTTPS*.

Entre las desventajas que más llaman la atención del paquete gratuito de este *hosting* es que durante una hora el sitio se mantiene inactivo, sin embargo, este

presenta la ventaja de poder seleccionar la hora de inactividad del sitio como tal. En la figura 20, se muestra de forma más detalla los distintos tipos de planes que posee este *web hosting*.

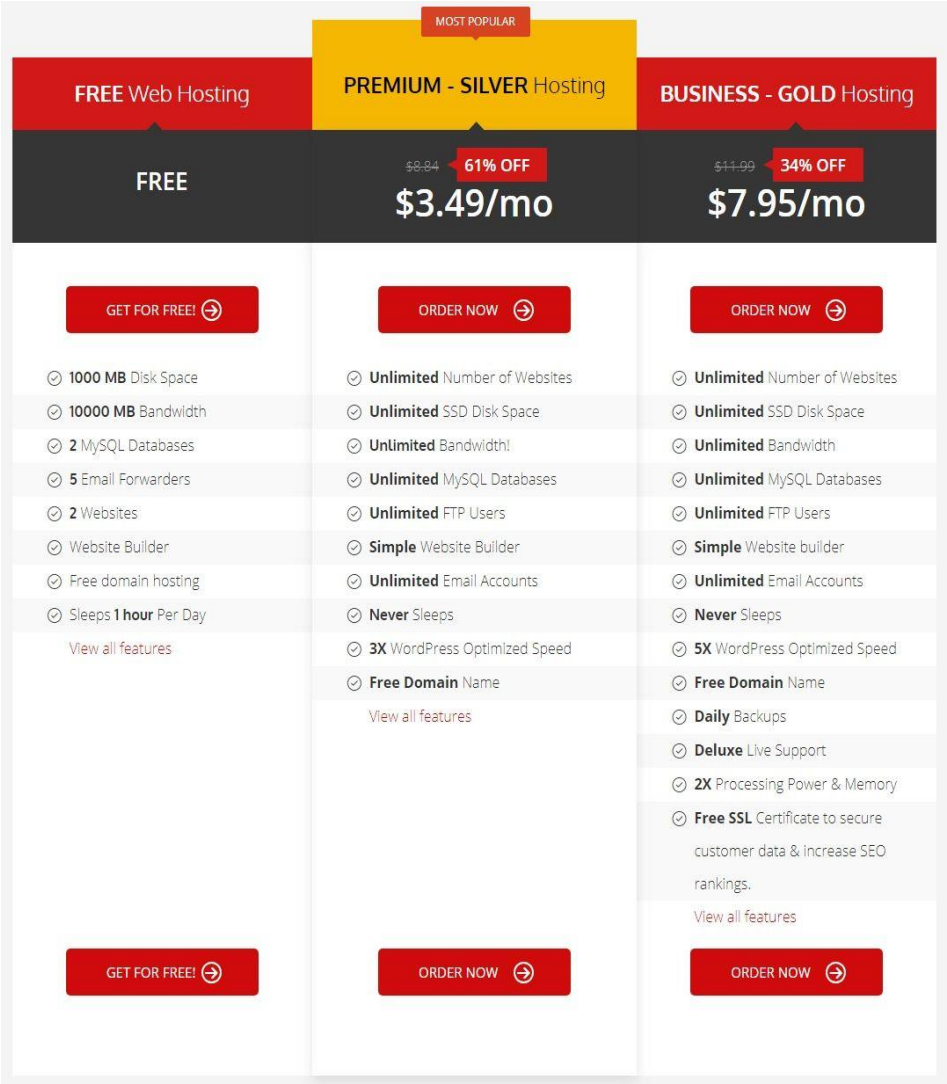


Figura 20. Planes de *000WebHost*.
Tomado de (Hostinggratis, 2019).

Microsoft Azure

Azure es una plataforma que ofrece servicios en la nube, este posee varias características que lo vuelven una solución óptima para proyectos que necesitan de alta disponibilidad y accesibilidad a sus datos, además, es

compatible con cualquier sistema operativo y lenguajes de programación, de tal forma que los distintos servicios pueden funcionar en cualquier dispositivo. De la misma manera, este brinda un servicio personalizado y a la medida del usuario, debido a que los costos se relacionan directamente con los servicios utilizados (Fajardo, 2017).

Azure* vs *000WebHost

Para seleccionar la plataforma más adecuada para el desarrollo del proyecto, es necesario tener en consideración varios factores, los cuales se muestran en la tabla 12.

Tabla 12.
Comparativa entre Microsoft Azure y 000WebHost.

	Microsoft Azure	***000WebHost***
Compatibilidad	Soporta varios S.O y lenguajes de programación	Soporta lenguaje de programación *PHP*
Disponibilidad	Siempre disponible	99.9% de disponibilidad
Costo	En base a los servicios utilizados	Depende del plan
Seguridad	Certificados SSL	Certificados SSL gratis

En base a todos los aspectos mostrados anteriormente, la mejor solución es *000WebHost*, debido a los bajos costos de sus planes en comparación con *Azure*, así como también, la garantía de certificados *SSL* de forma gratuita y su compatibilidad con el lenguaje de programación *PHP*.

3.4 *Hardware*

3.4.1 Microcontrolador

En el presente literal se llevará a cabo la comparativa entre los distintos tipos de dispositivos, entornos de desarrollo y tecnologías a nivel de *hardware* que pueden ser utilizados para realizar proyectos teniendo como base la tecnología *IoT*.

El objetivo de esta comparativa es determinar cuál es la tecnología o dispositivo que proporciona una mejor solución para el desarrollo del proyecto. Teniendo en consideración lo anterior, las presentes comparativas están evaluadas mediante una escala como se puede observar en la tabla 13, la cual va desde el número 1 que significa o califica como fácil, hasta el número 3 que se considera como difícil. Estos números tienen sentido en los literales Accesibilidad, Manejo y control de las comparativas mencionadas anteriormente.

Tabla 13.
Escala de calificación para literales de accesibilidad, manejo y control.

Calificación	**Numero**
Fácil	1
Medio	2
Difícil	3

La primera comparativa a llevar acabo se da entre la clasificación de los dispositivos que presenta la plataforma o entorno de desarrollo *Arduino*. En la tabla 14, se puede apreciar como el *Arduino* nano posee un tamaño menor que el *Arduino* micro, este es uno de los factores más importantes que se busca

para desarrollar el proyecto, además presenta un menor costo. Por otro lado, las demás características son similares en cuento a la su alimentación, accesibilidad, manejo y control.

Tabla 14.

Comparativa entre Arduino micro y Arduino nano.

***Arduino* micro vs *Arduino* nano**		
	Arduino micro	*Arduino* nano
Tamaño	4.8 cm x 1.77 cm	4.5 cm x 1.8 cm
Costo	$ 9	$ 7
Accesibilidad	1	1
Manejo y control	1	1
Alimentación	5 v	5 v

En la actualidad se han presentado una variedad de tecnologías y marcas para desarrollar proyectos relacionados con *IoT*, una de las tecnologías que más se ha pronunciado en los últimos años es la proporcionada por la marca *Raspberry*, esto conlleva a un análisis entre el *Arduino* nano y dicha marca. En la tabla 15. se puede evidenciar como el *Arduino* nano sigue siendo una buena opción para desarrollar el proyecto en comparación a uno de los dispositivos más pequeños de la marca *Raspberry*. De la misma manera, los aspectos como el costo, manejo y control son esenciales a tener en consideración, debido a que *arduino* trabaja con un lenguaje de programación que permite añadir sentencias *C++,* el mismo que es fácil e intuitivo, mientras que *Raspberry* utiliza como lenguaje *Python* que posee cierto grado de dificultad.

Tabla 15.

Comparativa entre Arduino nano y Raspberry.

Arduino* nano vs *Raspberry		
	Arduino nano	*Raspberry Pi*
Tamaño	4.5 cm x 1.8 cm	85.6 mm x 56.5 mm
costo	$ 7	$ 60
Accesibilidad	1	1
Manejo y control	1	2
Alimentación	5 v	5 v – 300 mA

Uno de los dispositivos que más lleva en el mercado hasta la actualidad es el microcontrolador *PIC*, este posee la ventaja de tener un tamaño más reducido que su competidor *Arduino* nano. Sin embargo, también posee ciertas desventajas, debido a factores como manejo y control con respecto a la plataforma de *Arduino*, ya que este trabaja con lenguajes de programación Basic y ensamblador que lo hace poco manejable en proyectos con un grado alto de complejidad, además necesita de un quemador de *PIC's* para cargar el *software* en dicho microcontrolador. Estas ventajas y desventajas se pueden apreciar de mejor manera en la tabla 16.

Tabla 16.

Comparativa entre Arduino nano y Arduino nano.

Arduino* nano vs *Pic		
	Arduino nano	*Pic*
Tamaño	4.5 cm x 1.8 cm	Depende del modelo

costo	$ 7	$ 3
Accesibilidad	1	1
Manejo y control	1	2
Alimentación	5 v	5 v

El siguiente análisis por llevar a cabo es el microcontrolador *VoCore2*, este dispositivo no tiene mucho tiempo de salir al mercado, sin embargo, posee varias características que lo vuelven una óptima solución para proyectos con altos grados de complejidad que desean integrar parámetros de *IoT*, además, tiene como características principales un tamaño muy reducido que se asemeja al de una moneda e incorpora *wifi*, también utiliza como base el sistema operativo de *Linux* mediante el uso de lenguaje de programación C. Es importante mencionar que este nuevo microcontrolador puede acoplarse con dispositivos *arduino*. En la tabla 17 se puede apreciar de mejor manera las ventajas y desventajas que este microcontrolador presenta.

Tabla 17.
Ventajas y Desventajas de VoCore2.

Microcontrolador *VoCore2*		
	Ventaja	Desventaja
Módulos o Sensores	Permite la integración de sensores por medio de un microprocesador de *arduino*.	Al integrar este microcontrolador a un dispositivo *Arduino* no se obtiene un buen resultado en cuanto a la reducción de tamaño.

Finalmente, la solución óptima para el desarrollo del proyecto es el microcontrolador denominado *CheapDuino*, el cual posee un tamaño relativamente pequeño, además, este dispositivo puede considerarse como un elemento que pertenece a la rama de *Arduino*, debido a que es compatible con la mayoría de sus módulos y sensores. Estas y más observaciones se las puede apreciar de mejor manera en la tabla 18.

Tabla 18.
Características de CheapDuino.

	CheapDuino
Tamaño	2cm x 2cm x 0.2cm
Alimentación	3 – 5 v
Microcontrolador	Atmel AVR ATmega8
Pines	3 pines digitales 3 pines analógicos 3 pines pwm
Interfaz	I2C UART

Adaptado de (DFROBOT, 2019).

3.4.2 Sensores

3.4.2.1 Sensor de Ritmo Cardíaco

El sensor de ritmo cardíaco a utilizarse debe estar en contacto con la piel como se muestra en la figura 21, además, este cuenta con un led y un sensor de luz ambiental.

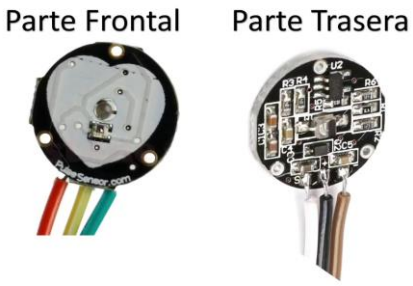


Figura 21. Sensor de Ritmo cardíaco.
Tomado de (Tdrobotica, 2019).

Su funcionamiento consta de 2 etapas, la etapa de amplificación y una etapa para filtrar el ruido, de tal manera que la señal analógica sea amplificada y posteriormente normalizada logrando que dicha señal tenga como resultado mayor estabilidad, el esquema electrónico de este sensor se lo puede apreciar de mejor manera en la figura 22. Por otro lado, al hablar de la luz led que compone este sensor, es importante mencionar la intensidad de dicha luz, como esta se refleja y cambia durante cada impulso, ya que, si la luz incidente en el sensor permanece de forma constante, esto significa que el valor de la señal se mantiene aproximadamente entre 512, mismo que es un valor medio si tiene en consideración el rango de ADC. De la misma manera, a mayor cantidad de luz existe mayor señal y a menor cantidad de luz el valor de la señal analógica es menor.

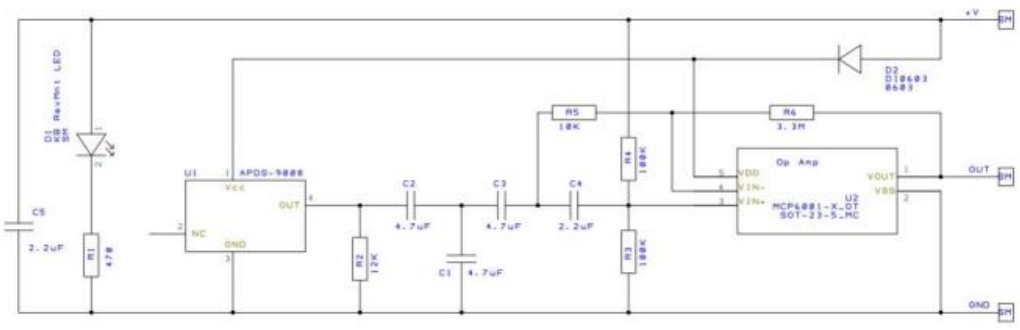


Figura 22. Esquema electrónico del sensor de pulse cardíaco.
Tomado de (Velázquez, Villagrán ,2016).

El objetivo principal es determinar en la señal los momentos sucesivos que se presentan en los latidos instantáneos del corazón y de esta manera medir el intervalo de tiempo trascurrido entre estos. En la figura 23 se puede apreciar el pulso de la onda tomado a partir de una muestra de la señal.

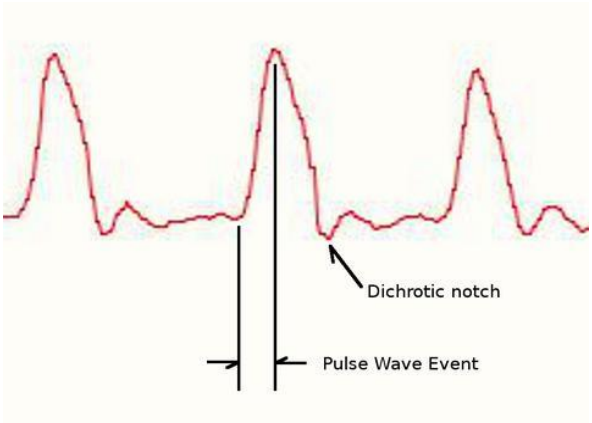


Figura 23. Pulso de onda.
Tomado de (Bolaños, 2019).

Otras de las características importantes de este sensor es que no necesita de muchas variables para funcionar, ya que es necesario una alimentación de 3 - 5v, 4mA y utilizar un dispositivo que se encargue de procesar sus datos. En la figura 24, se puede observar a configuración de los pines del sensor de pulso cardíaco. De la misma manera, este sensor posee un diámetro de 16 mm y un espesor de 3 mm.

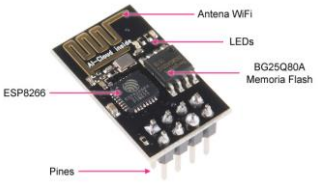
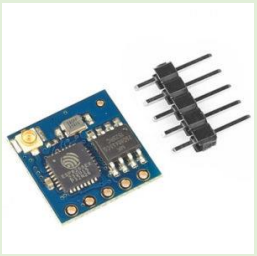
Figura 24. Sensor de pulso cardíaco.
Tomado de (Lletí, 2015).

3.4.2.2 Módulos *wifi* comparativas

Los módulos *wifi* utilizan un chip para conectarse de manera inalámbrica a una red de datos, mantiene la compatibilidad con el protocolo *TCP/IP* y además facilita el desarrollo de proyectos tecnológicos en el área de *IoT*. En la tabla 19, se puede observar una comparativa de las distintitas versiones de dicho módulo.

Tabla 19.

Comparativa de módulos wifi Esp.

Comparativa módulos Esp		
Módulo	**Características**	**Diagrama**
Esp-01	Voltaje = 3.3 – 3.6 v Corriente = 200 mA Potencia = 0.684 w Comunicación I2C	
Esp-05	No dispone de puerto GPIO. Numero de pines = 5 Voltaje 3.3 – 5 v Corriente = 500 mA Potencia = 1.65 w	

Esp-12	Numero de puertos GPIO = 11 Conexión no muy amigable Voltaje = 3.3 v Corriente de fuga = 10 uA	

Adaptado de (Hernández, 2019).

Modulo *wifi Esp8266*

El módulo wifi es el encargado de enviar los datos de forma inalámbrica hacia la aplicación *web*. Para el desarrollo del proyecto se utilizará el módulo *Esp8266*, en la tabla 20 se presentan las características de este módulo.

Tabla 20.
Características del módulo Esp8266.

Modulo *Esp8266*	
Compatibilidad	802.11 b/g/n
Tamaño	25mm x 15mm x 1mm
Tiempo de transferencia de paquetes	Menor a 2 ms
Consumo de potencia	1 mW
Alimentación	3.3 v

Adaptado de (Rambal, 2019).

La figura 25, muestra la configuración de los pines que presenta el módulo *Esp8266*.

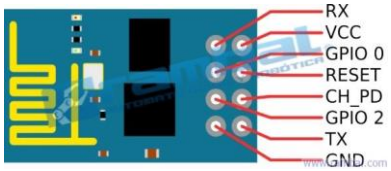


Figura 25. Pines del módulo *Esp8266*.
Tomado de (Rambal, 2019).

Módulo de Carga

La etapa de alimentación es muy importante para el diseño correcto de un prototipo electrónico, ya que dicha etapa se encarga de asegurar un óptimo funcionamiento de la parte electrónica y toma de datos confiables.
Por otro lado, al utilizar baterías de litio es importante considerar que estas deben tener un control de carga, de tal manera que se pueda evitar la sobrecarga y los posibles daños que se pueden presentar en las celdas de las baterías.

Teniendo en consideración lo comentado anteriormente, se ha optado por utilizar el módulo de carga *TP4056*, el cual está diseñado específicamente para controlar y mitigar el inconveniente de sobrecarga asegurando una mayor vida útil de las baterías. En la figura 26, se puede apreciar los distintos pines de conexión que posee el módulo de carga.

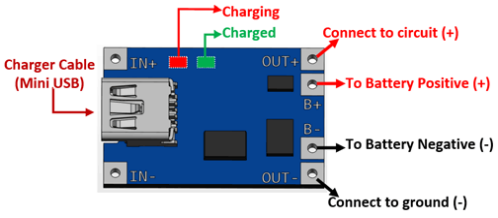


Figura 26. Pines del módulo *TP4056*.
Tomado de (Components101, 2018).

La tabla 21, presenta las características técnicas del módulo de carga *TP4056*.

Tabla 21.

Características del módulo TP4056.

Módulo *TP4056*	
Corriente de carga	1 A
Voltaje de entrada	4.5 v a 5.5 v
Voltaje de carga completa	4.2 v
Tamaño	25 x 19 x 10 mm
Interfaz de Entrada	Micro *USB*

Adaptado de (Components101, 2019).

Diagrama del Prototipo electrónico

El prototipo electrónico está compuesto de distintas etapas, las cuales deben ejecutarse completamente para asegurar un óptimo rendimiento del prototipo y toma de datos, en la figura 27, se puede apreciar de mejor manera dichas etapas, las cuales son:

- Etapa de alimentación: En esta primera etapa se define la alimentación de todo el prototipo, mediante la utilización de baterías de litio recargables y un módulo *Tp4056*, el cual se responsabiliza de la protección de las baterías y de su carga.
- Recopilación de datos: El encargado de realizar la recolección de todos los datos es el sensor de ritmo cardíaco.
- Procesamiento de datos: Esta etapa es fundamental, debido a que se encarga de tomar los valores analógicos del sensor y pasarlos a digital, utilizando un microcontrolador.
- Envió de datos: Una vez procesados los datos, estos se envían hacia la aplicación *web* utilizando como medio de propagación un módulo *wifi*.

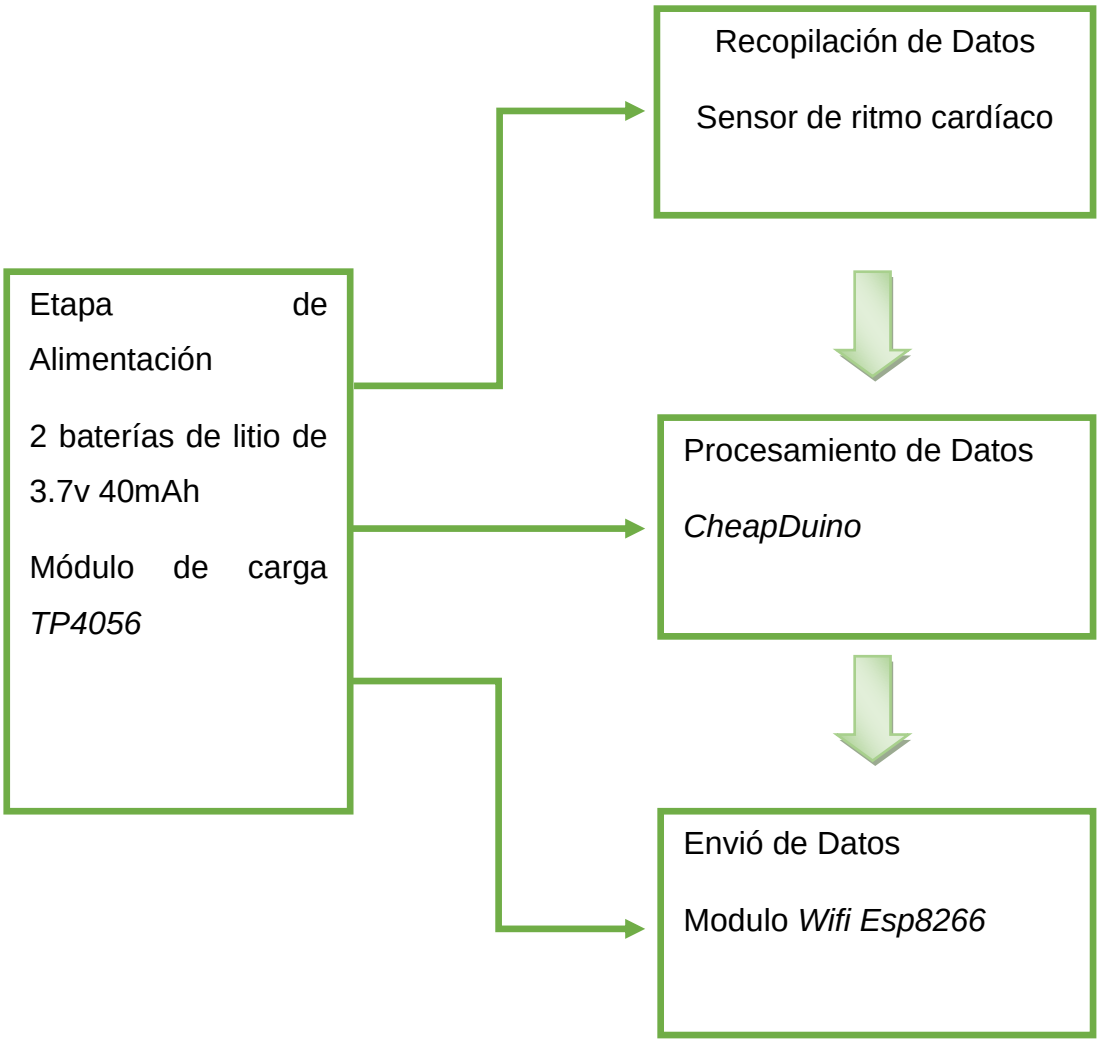


Figura 27. Diagrama del prototipo.

4. CAPÍTULO IV. IMPLEMENTACIÓN DE LA RED MÉDICA

En el presente capítulo se describe la implementación de la red médica y los distintos módulos que componen este sistema de monitoreo. Estos módulos luego de su implementación pasaran a unirse y a complementarse, para finalizar la implementación de la red de monitoreo de frecuencia cardíaca que utiliza conceptos de *IoTM*.

4.1 *BackEnd*

En la siguiente sección se presenta la parte lógica de la red médica, *DBMS* y *hosting web*.

4.1.1 *MySQL*

Uno de los puntos importantes al hablar de los lenguajes de consulta estructurados y de la manipulación de los datos, es la relación que mantienen con el algebra relacional, la cual se basa en la teoría de conjuntos. Es importante este punto porque, sirve como base para el tratamiento de las consultas que se llevan a cabo de manera interna en los *DBMS*.

En la plataforma de *MySQL* es necesario crear una base de datos y posteriormente importar el código .*SQL*. Una vez realizado estos pasos, se obtiene como resultado final la base de datos con todas sus entidades, atributos y relaciones. La figura 28, presenta la base datos implementada en la plataforma *MySQL*.

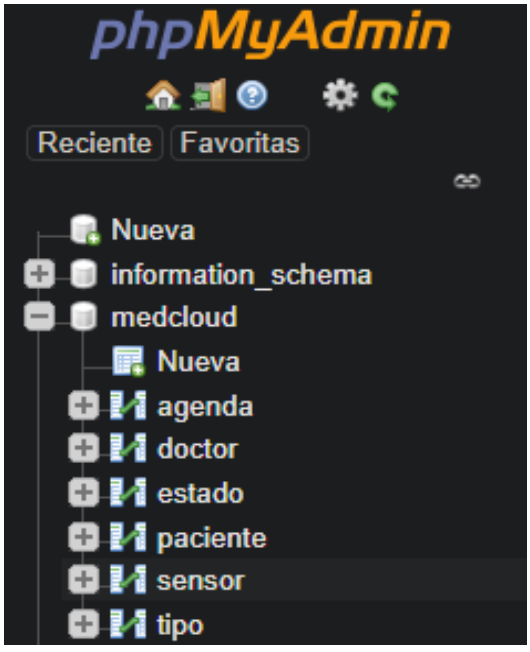


Figura 28. Base de datos en *MySQL*.

4.1.2 Hosting Web

En el caso de la selección del *hosting* para poder alojar la página *web* se ha optado por el uso de *000webhost* ya que implementarlo resulta más económico en comparación a diferentes tipos *hosting* presentes en el mercado actual, además brinda características que se acoplan a los requerimientos del sistema. En la figura 29, se puede apreciar el panel principal de dicho *hosting*.

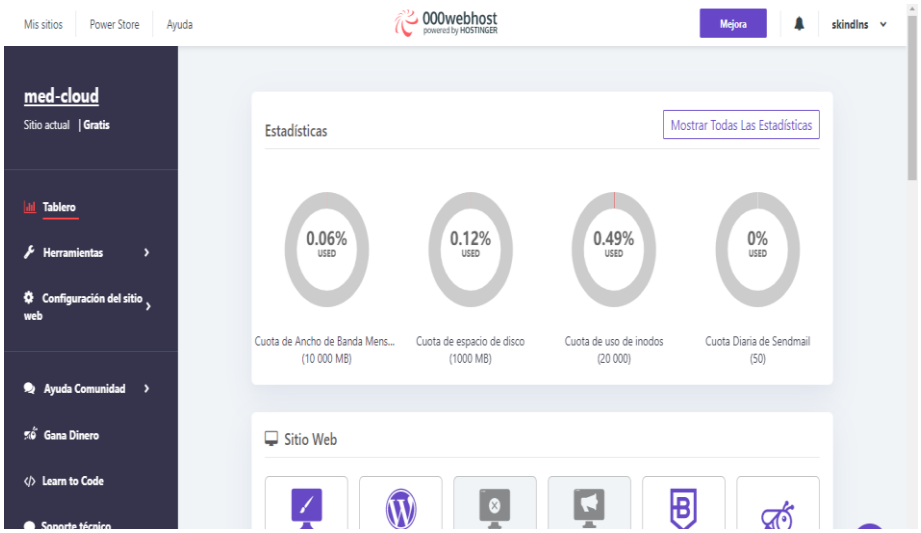


Figura 29. Panel *000webhost*.

Dentro de este *hosting* se implementa todo el desarrollo de la página *web*, para lo cual se utiliza lo siguiente:

- El lenguaje de programación *PHP* para el desarrollo backend.
- *Mysql* como motor de base de datos para el almacenamiento de la información obtenida por el sensor.
- Para el apartado de frontend se optó por el uso de *HTML, CSS* y *JavaScript*.

4.2 FrontEnd

4.2.1 Aplicación *web*

El primer paso para llevar acabo el desarrollo de la aplicación *web*, es pasar el diagrama del diseño lógico de la base de datos relacional a código, mediante la utilización de los leguajes de consulta estructurados *Data Manipulation Language, Data Definition Language y Transactional Control Language*, para ejecutarlo en la plataforma de *MySQL*.

4.2.1.1 Conexión con la base de datos

El siguiente paso, es unir la base de datos con la aplicación *web*, para esto es necesario crear un archivo con la extensión *.php* y definir en variables las credenciales para conectarse con la base de datos, seguido de las líneas de código para establecer dicha conexión. En la figura 30 se muestran las credenciales y la línea de conexión con la base de datos, de forma local.

```php
<?php
$host = 'localhost';
$basededatos = 'medcloud';
$usuario = 'root';
$contraseña = '';

$conexion = new mysqli($host, $usuario,$contraseña, $basededatos);
if ($conexion -> connect_errno)
{
    die("Fallo la conexion:(".$conexion -> mysqli_connect_errno().")".$conexion-> mysqli_connect_error());
}

?>
```

Figura 30. Conexión con la base de datos.

Para el desarrollo de los formularios es fundamental conocer los tipos de usuarios como son:

- Administrador. – El usuario administrador es el único usuario que puede gestionar toda la información de los médicos.
- Médico. – Este usuario puede gestionar la información de sus pacientes, así como también registrarlos en el sistema.

- Paciente. – El paciente tiene acceso solo a un histograma donde se presenta de manera gráfica los datos receptados por el prototipo de ritmo cardíaco.

Cada uno de estos usuarios tiene sus permisos definidos y acceso a distintos formularios. De la misma manera, el acceso y navegación por los diferentes formularios depende principalmente de la primera pantalla de navegación o formulario.

También, se debe considerar sentencias del tipo *Data Retrivel Language* para consultar los datos alojados en la base de datos y utilizarlos para el desarrollo del *back-end* en la aplicación *web*.

4.2.1.2 Interfaz de Inicio

Dentro de la primera interfaz se implementa un formulario que es utilizado por los usuarios para iniciar sesión. En la figura 31, se muestra el formulario *Login*.

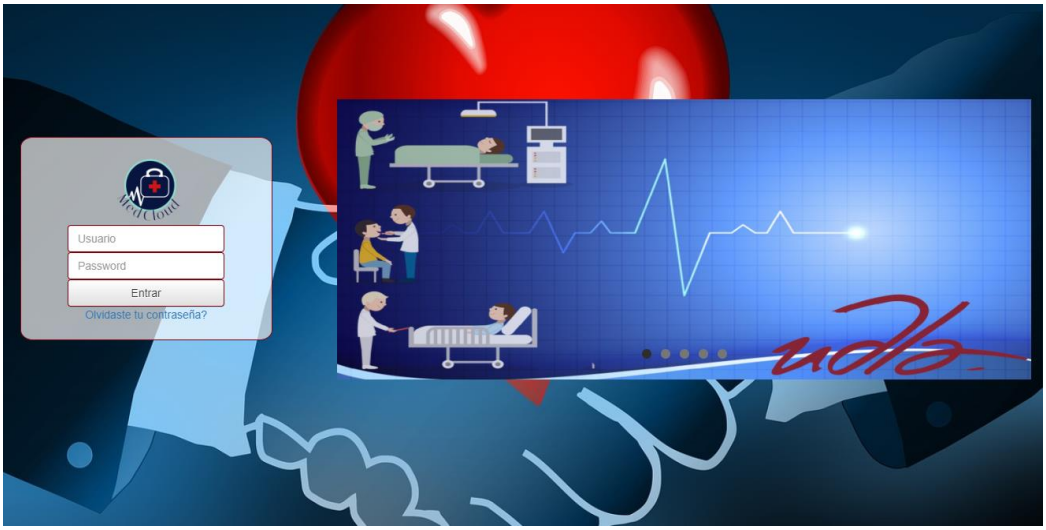


Figura 31. Interfaz de inicio de la página *web*.

4.2.1.3 Rol de Administrador

En este literal se mostrarán los distintos formularios, así como también los permisos establecidos y desarrollados para el usuario Administrador.

Manejo médico

Cuando un usuario inicia sesión con las credenciales de administrador tiene acceso a todos los formularios del sistema. El administrador, es el encargado de registrar a los médicos en la base de datos de la aplicación. La figura 32, presenta el formulario de registro para médicos.

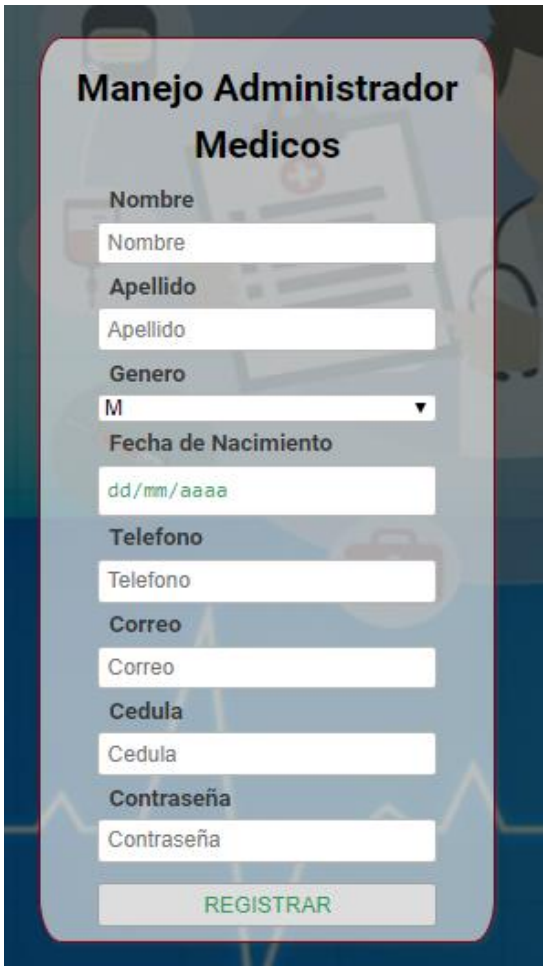


Figura 32. Formulario registro médicos.

Seguido de un formulario se encuentra una tabla donde se puede visualizar la información alojada en la base de datos, así mismo este apartado posee un

campo que le otorga al usuario la posibilidad de buscar un registro, dependiendo del atributo que el mencionado usuario considere más importante. Por ejemplo, la tabla que contiene la información de los médicos permite una búsqueda por Nombre, Apellido, Genero, Fecha de Nacimiento, Numero de Cedula, Teléfono y Correo. Estos atributos se los puede apreciar de mejor manera en la figura 33.

Figura 33. Tabla de médicos.

Manejo Paciente

El usuario administrador poseerá los privilegios para visualizar a todos los usuarios del tipo paciente. En la figura 34, se puede apreciar el formulario de registro para este tipo de usuarios, con su respectiva tabla de información consultada de la base de datos, mediante sentencias *Data Retrivel Language.*

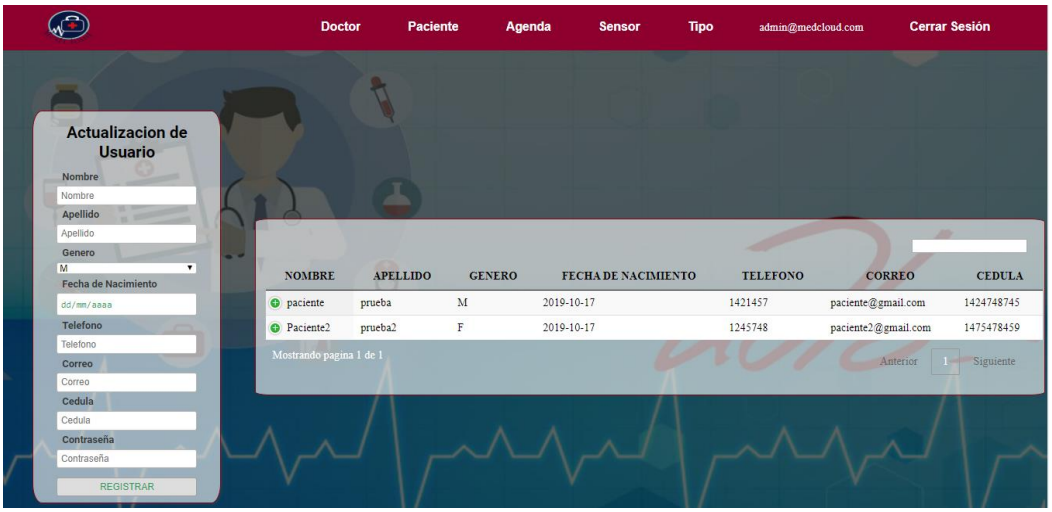


Figura 34. Formulario e información de los pacientes.

Manejo Agenda

De la misma manera, el usuario administrador también puede gestionar toda la información alojada en la entidad de la base de datos agenda. En la figura 35, se puede apreciar de mejor manera la implementación de esta vista.

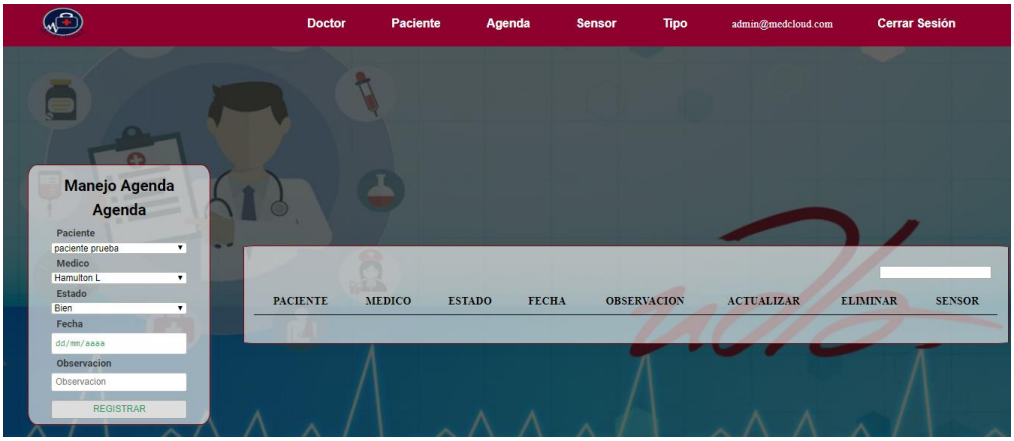


Figura 35. Formulario e información de la clase agenda.

Manejo Estado

El usuario administrador es el encargado de registrar los datos de la tabla estado en la base de datos, mediante la utilización del formulario desarrollado en la aplicación móvil. En la figura 36, se muestra el formulario de registro y la información referente a la clase estado.

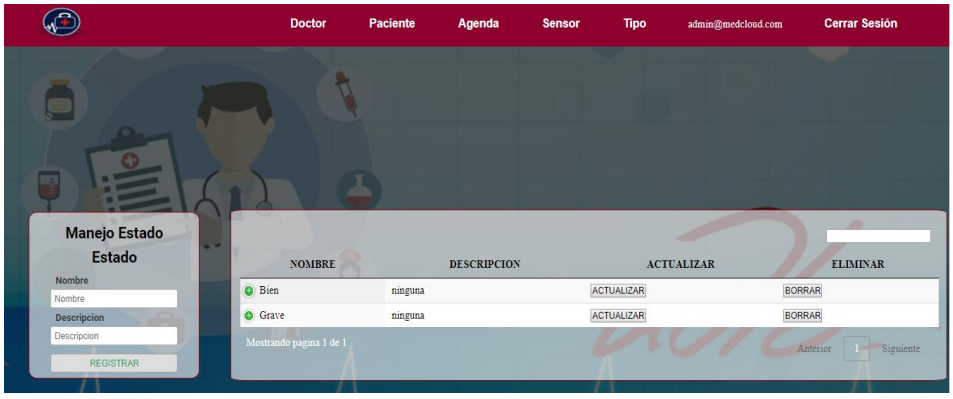


Figura 36. Formulario e información de la clase estado.

Manejo Sensor

El administrador tiene acceso a los datos del sensor para supervisar la funcionalidad del sistema y él envió de datos desde la pulsera de frecuencia cardíaca. En la figura 37, se puede apreciar la información que contiene la clase sensor.

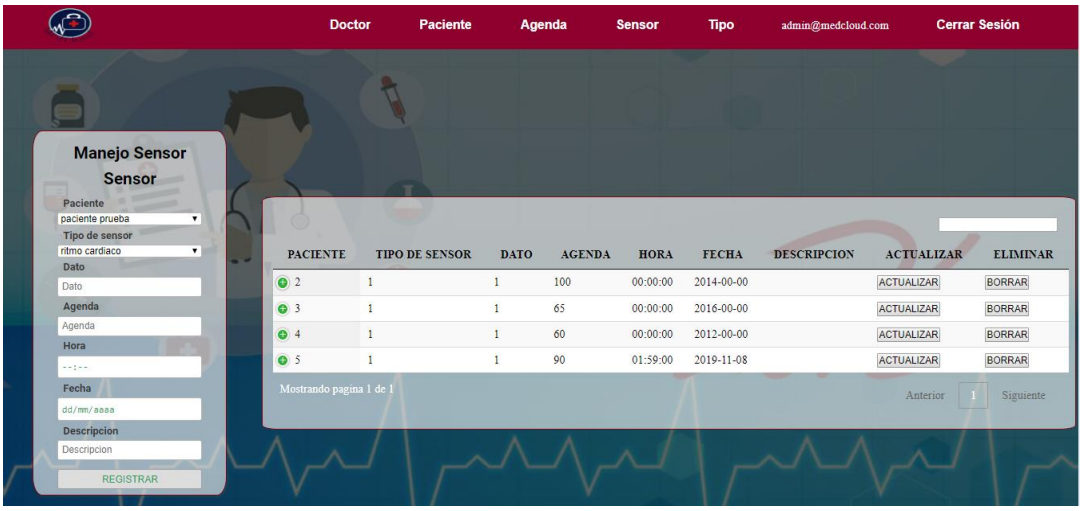


Figura 37. Información de la clase sensor.

Manejo Tipo

La clase tipo se implementará específicamente para que el diseño de la aplicación y el sistema médico sean escalables. El usuario administrador, es el encargado de registrar y administrar toda la información de los nuevos sensores que se pueden adaptar al prototipo electrónico, sin alterar el diseño actual del *software*. En la figura 38, se muestra el formulario de registro y la tabla dinámica con la información consultada desde la base de datos.

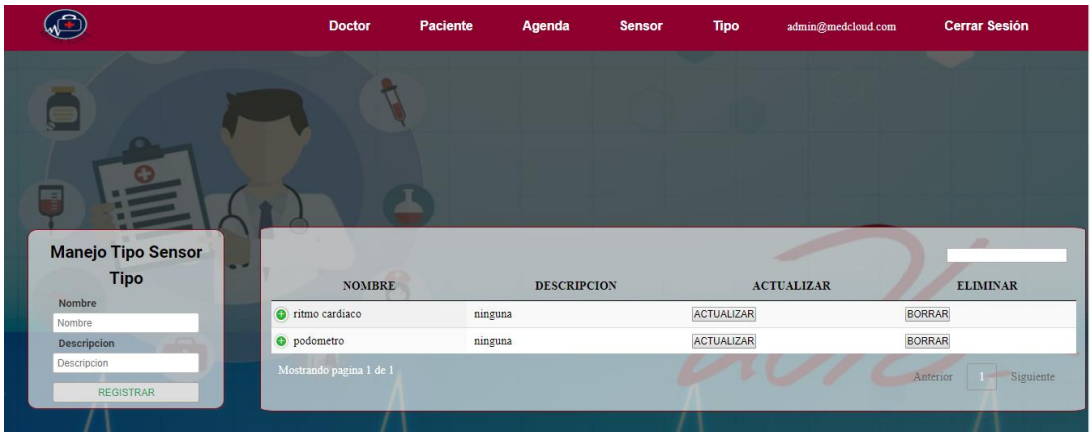


Figura 38. Formulario e Información de la clase tipo.

4.2.1.4 Rol de Médico

Manejo Paciente

Por otro lado, el médico es el encargado de ingresar y gestionar toda la información referente a sus pacientes, en la figura 39 se muestra el formulario de registro para pacientes y una tabla con la información personal de los pacientes registrados por dicho médico.

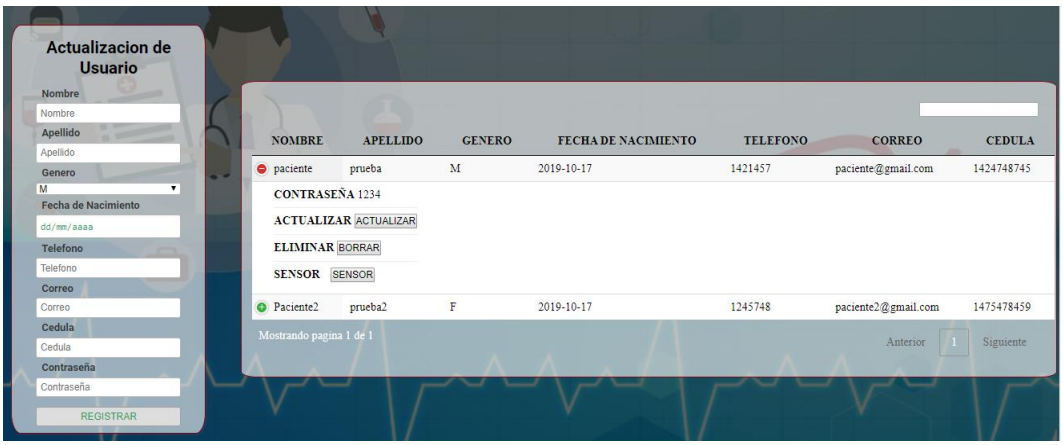


Figura 39. Formulario e información de clase paciente.

Manejo Agenda

El médico puede acceder a la pestaña de agenda para poder visualizar las citas que tiene pendientes, estas citas médicas se agendan de manera automática en caso de que el ritmo cardíaco de algún paciente se salga de los límites establecidos como normales en él *software*. La figura 40, presenta un ejemplo de como se muestra la agenda de un médico en la aplicación.

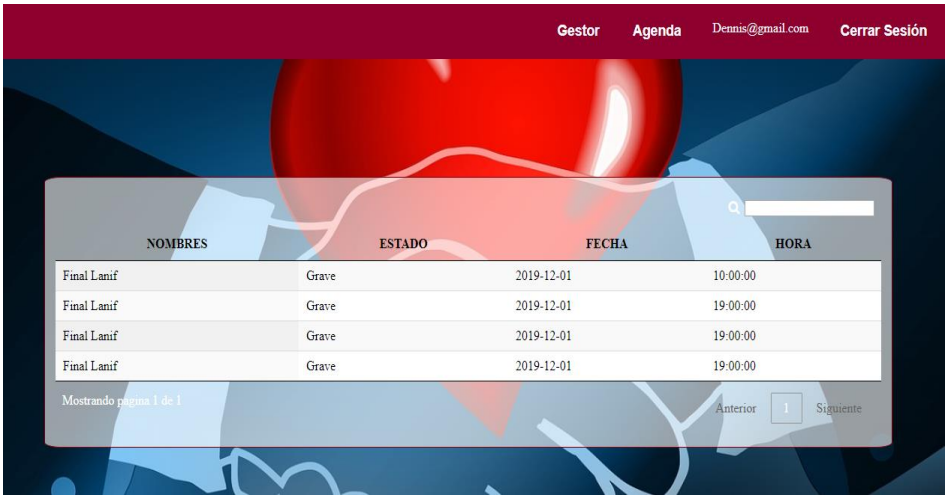


Figura 40. Formulario e información de clase agenda.

Manejo Sensor

La pestaña sensor aloja información recopilada por el prototipo electrónico, el usuario del tipo médico podrá acceder a esta pestaña, para poder visualizar en un histograma los datos de frecuencia cardíaca de su paciente. En la figura 41, se puede apreciar de mejor manera la pestaña sensor a la cual el médico podrá tener acceso.

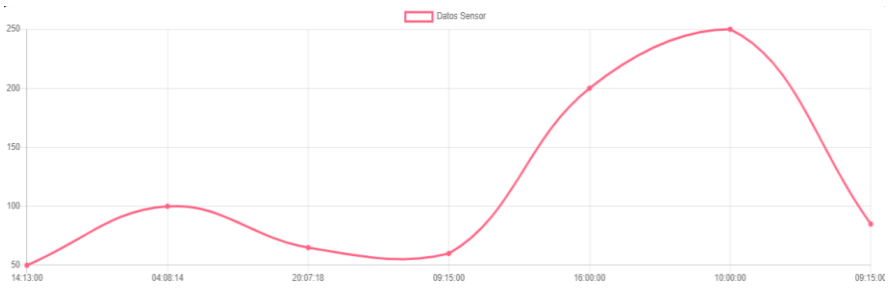


Figura 41. Histograma de paciente.

4.2.1.5 Rol de Paciente

A diferencia de los anteriores tipos de usuarios, el paciente puede tener acceso solo a un histograma donde se presenta la información receptada por el prototipo de frecuencia cardíaca. En la figura 42, se muestra un usuario de ejemplo con su respectiva información representada en el histograma.

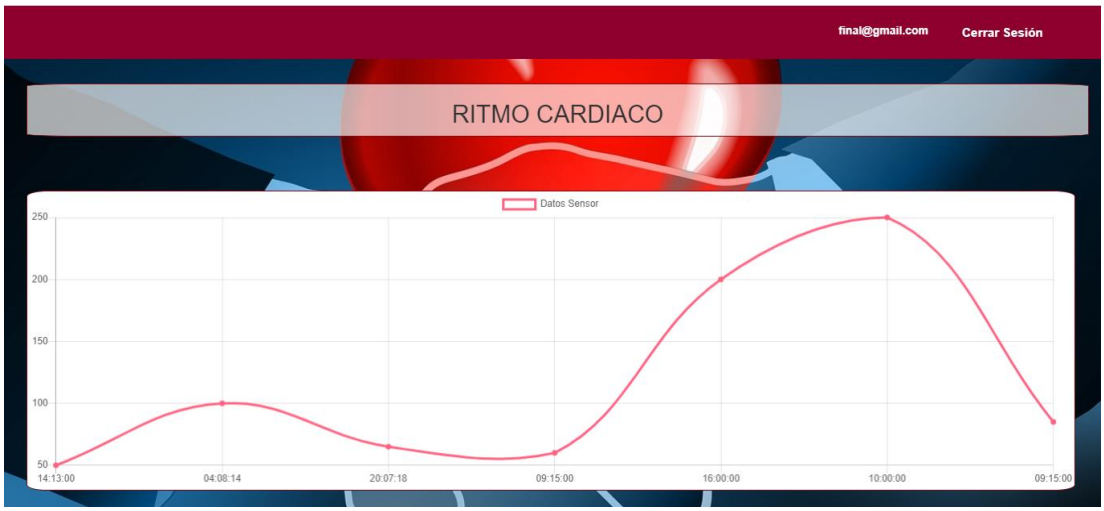


Figura 42. Histograma de paciente.

Para el desarrollo del histograma de la figura 42, es necesario aplicar los lenguajes de programación *JavaScript* y hojas de estilo *Css*, para consultar la información de la base de datos y proporcionarle un aspecto amigable y atractivo para el usuario.

4.2.2 Aplicación móvil

La aplicación móvil se implementará utilizando conceptos de *UX (User Experience),* que se refiere a aplicaciones móviles basadas en la experiencia del usuario. Esta aplicación consta de 2 pestañas, la primera es una pestaña con un formulario para iniciar sesión con las credenciales del usuario, como se puede apreciar en la figura 43.

Figura 43. Pestaña para iniciar sesión.

Por otro lado, la segunda pestaña muestra los datos tomados por la pulsera de frecuencia cardíaca de forma gráfica, mediante un histograma, como se puede ver en la figura 44. Es importante mencionar que dicha aplicación móvil, es utilizada para testear los datos que se envían hacia el servidor alojado en la nube.

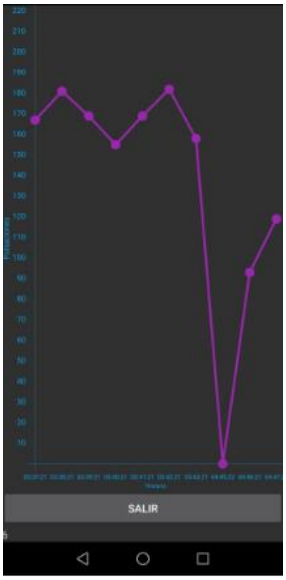


Figura 44. Histograma del paciente.

Para llevar a cabo el desarrollo de la conexión de la aplicación móvil con la página *web*, se debe considerar las credenciales para una conexión remota con la base de datos alojada en el *hosting web*. Esas credenciales, son necesarias para la cadena de conexión, así mismo estas credenciales se las puede observar de mejor manera en la figura 45.

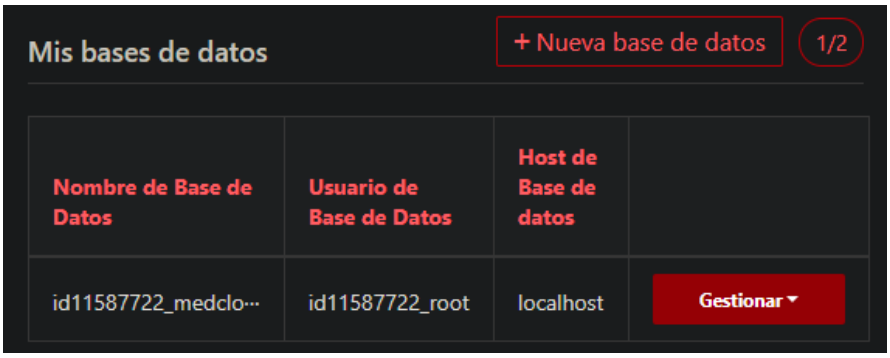


Figura 45. Credenciales para conexión remota.

4.2.3 *Hardware*

4.2.3.1 Prototipo

En este literal se describe el desarrollo del prototipo electrónico, así como también los materiales utilizados, teniendo en consideración el diseño realizado con anterioridad.

Para el desarrollo del dispositivo electrónico es fundamental tener en consideración la alimentación de cada uno los elementos que lo componen, como se puede apreciar en la figura 46, se tratan de elementos sumamente sensibles, en los cuales si se presenta alguna variación fuerte de alimentación puede provocar un daño irreparable en dichos elementos.

El puerto A4 del microcontrolador *cheapduino*, es el encargado de recolectar y procesar los datos receptados por el sensor de ritmo cardíaco, posteriormente para el envío y la recepción de datos mediante el módulo *Wifi*, es fundamental

utilizar los pines *TX* y *RX* que utilizan un protocolo de comunicación serial asíncrono.

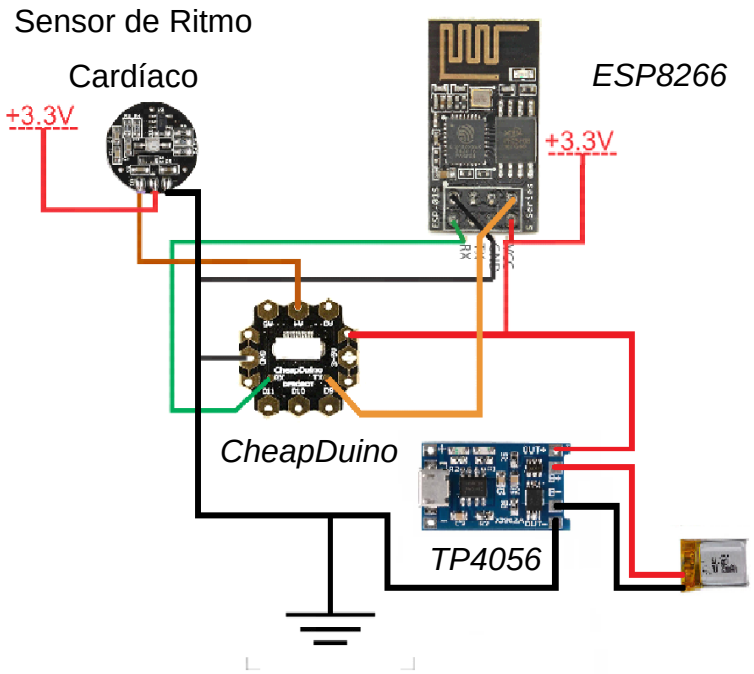


Figura 46. Conexión y alimentación de elementos electrónicos.

El módulo *wifi ESP8266* y el microcontrolador son programados por separado, esto debido a que el módulo *wifi* es el encargado de recibir y enviar las tramas hacia la aplicación *web* y base de datos, mientras que el microcontrolador *CheapDuino* es el encargado de recolectar los datos del sensor de ritmo cardíaco.

Es importante mencionar que para el envío de variables desde el microcontrolador *cheapduino* al módulo *wifi*, es necesario utilizar la librería *WiFiClient.h*. Sin embargo, para poder utilizar dicha librería es necesario instalar el paquete del *ESP8266* versión 2.6.4.

La librería *WiFiClient.h*, es la encargada de realizar la conexión del módulo *wifi* con la red *WLAN*, para esto es necesario establecer en la programación las

credenciales de la red con la cual se desea establecer una comunicación, como se muestra en la figura 47.

```
const char* ssid     = "Nombre de la Red";
const char* password = "Contraseña de la Red";
```

Figura 47. Conexión del módulo *wifi*.

De la misma manera, para realizar la conexión con la aplicación *web* alojada en la nube, es necesario desarrollar un fichero del tipo escucha, utilizando parámetros *Json* de *JavaScript*. Este archivo, se ubica en el directorio raíz del servidor, de tal forma que siempre este a la espera de un dato. Como se puede apreciar en la figura 48.

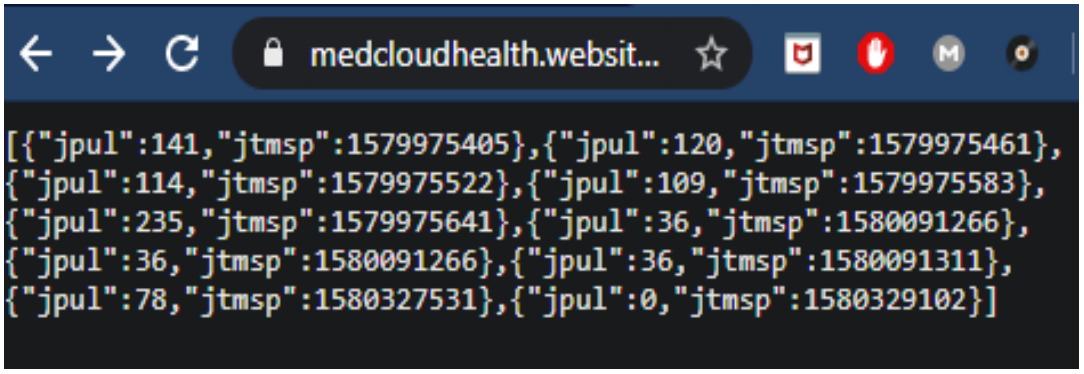


Figura 48. Captura de datos con *Json*.

Por otro lado, para insertar un registro en la tabla de la base de datos, se crea un archivo del tipo *php*, en el cual se utilizar el dato enviado por el sensor y el id, para realizar un *insert* completando los atributos de dicha tabla. Estos archivos se los puede apreciar de mejor manera en la figura 49.

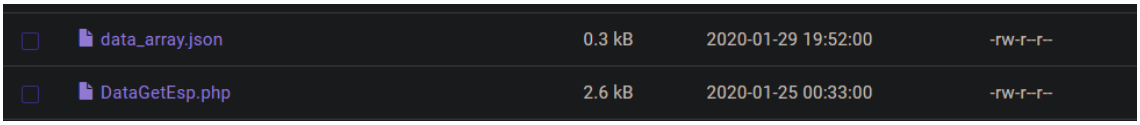


Figura 49. Ficheros para la conexión con *MySQL*.

Para cargar la programación desarrollada en la plataforma *Arduino* al microntrolador *cheapduino*, es necesario utilizar un convertidor serial a *USB*, en el cual se puedan identificar los pines *DTR*, *RX*, *TX* y *GND*, como se puede ver en la figura 50.

Figura 50. Pines del convertidor Serial-*USB*.

La figura 51, muestra la parte frontal del convertidor Serial-*USB*.

Figura 51. Convertidor Serial-*USB*.

La figura 52, muestra la conexión entre el adaptador propio del *cheapduino* y el convertidor Serial-*USB*.

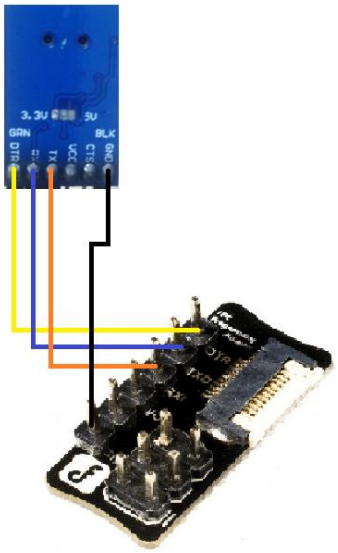


Figura 52. Conexión de adaptador *CheapDuino* con Serial-*USB*.

Finalmente, el prototipo es plasmado de forma compacta con la ayuda del *software EAGLE 8.0.1*, el mismo que permite realizar el diseño de una baquelita, en la cual se puedan integrar todos los elementos, garantizando un tamaño con dimensiones reducidas. El diseño de la baquelita y las pistas se las puede apreciar de mejor manera en la figura 53.

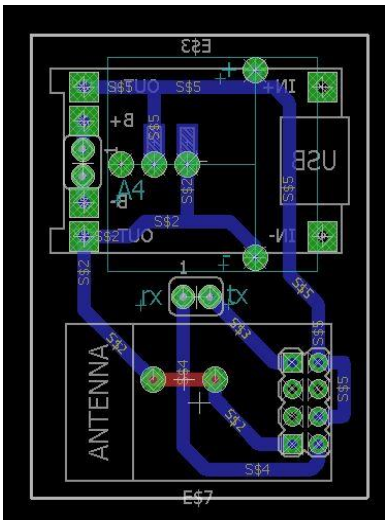


Figura 53. Diseño electrónico la baquelita.

Una vez terminado todo el proceso anterior, se obtiene como resultado el prototipo o dispositivo electrónico final, el cual, se puede apreciar de mejor forma en la figura 54.

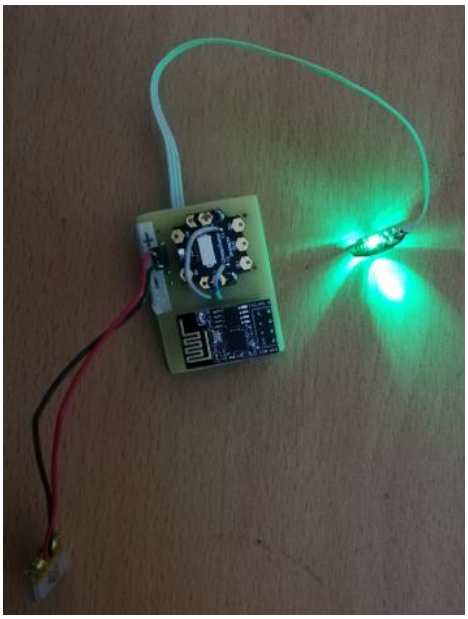


Figura 54. Prototipo electrónico final – parte delantera.

La figura 55 muestra el prototipo desde una vista posterior, donde se encuentra ubicado el módulo de carga y las pistas del diseño electrónico.

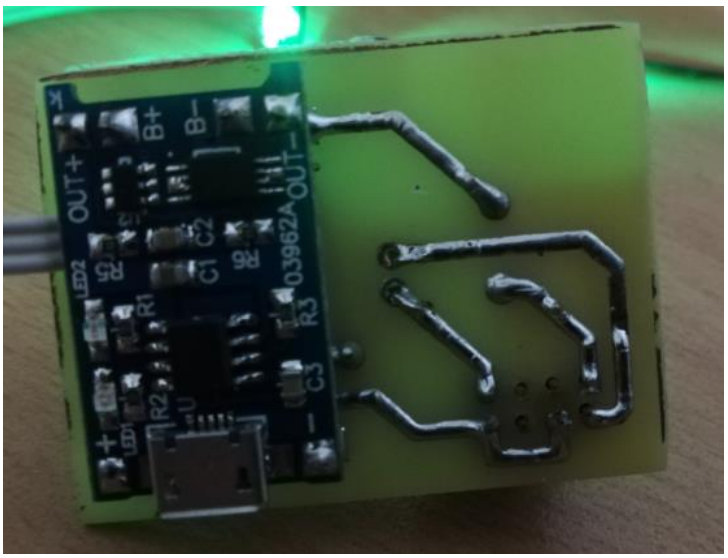


Figura 55. Prototipo electrónico final – parte posterior.

5. CAPÍTULO V. PRUEBAS DE FUNCIONAMIENTO

5.1 *Software*

La figura 56, muestra los distintos registros de prueba alojados en la base de datos, mediante una consulta realizada a la tabla paciente.

Figura 56. Consulta a la tabla paciente de la base de datos.

Por otro lado, la figura 57, despliega de forma gráfica la información almacenada en la entidad doctor mediante una consulta básica.

Figura 57. Consulta a la tabla doctor de la base de datos.

Para registrar un nuevo médico, es necesario iniciar sesión en la aplicación móvil con las credenciales de administrador, en este caso son:

- Usuario: admin@medcloud.com
- Contraseña:1234

Posteriormente se ingresan los datos del nuevo médico en el formulario, como se puede apreciar en la figura 58.

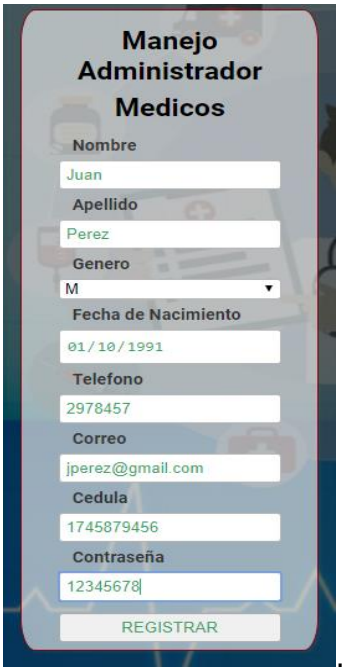
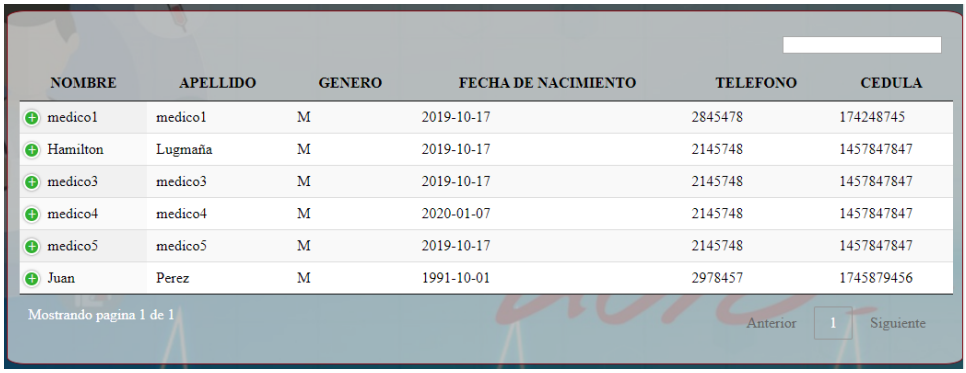


Figura 58. Registro de un nuevo Médico.

finalmente se registra esta información, misma que se muestra en una tabla como se observa en la figura 59.



NOMBRE	APELLIDO	GENERO	FECHA DE NACIMIENTO	TELEFONO	CEDULA
medico1	medico1	M	2019-10-17	2845478	174248745
Hamilton	Lugmaña	M	2019-10-17	2145748	1457847847
medico3	medico3	M	2019-10-17	2145748	1457847847
medico4	medico4	M	2020-01-07	2145748	1457847847
medico5	medico5	M	2019-10-17	2145748	1457847847
Juan	Perez	M	1991-10-01	2978457	1745879456

Mostrando pagina 1 de 1 Anterior 1 Siguiente

Figura 59. Tabla de registros de Médicos.

El proceso para registrar nuevos usuarios del tipo paciente es similar al del médico, sin embargo, dichos pacientes son registrados únicamente por el

usuario del tipo Médico, de tal manera que se pueda asegurar una eficiente confidencialidad entre estos 2 tipos de usuarios.

Una vez terminado el registro tanto de médicos como de pacientes el sistema automáticamente enviara un correo de bienvenida, en el cual se incluyen las credenciales necesarias para que los usuarios puedan iniciar sesión en la aplicación *web*. En la figura 60, se muestra un ejemplo de correo electrónico enviado automáticamente a un nuevo Médico registrado en el sistema.

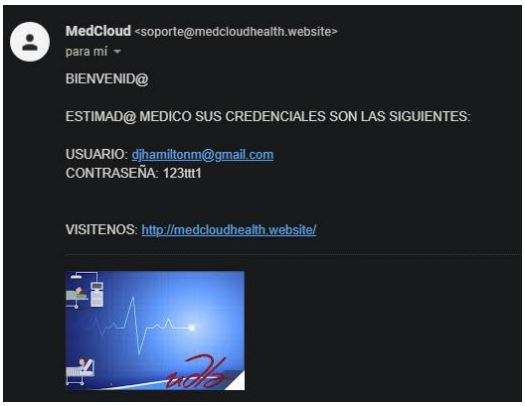


Figura 60. Ejemplo de correo electrónico.

El sistema es escalable, debido a que permite el registro e integración de nuevos sensores sin la necesidad de realizar un cambio en el diseño del *software*. Para esto es fundamental efectuar un nuevo registro en el formulario Tipo, mismo que se muestra en la figura 61.

Figura 61. Formulario de registro de nuevos sensores.

Luego de haber realizado el registro se muestra automáticamente en la tabla de sensores, como se observa en la figura 62.

Figura 62. Tabla de sensores.

5.2 *Hardware*

Antes de validar la conexión del prototipo de frecuencia cardíaca a una red *WLAN*, es necesario cambiar las credenciales de dicha red, como se muestra a continuación:

- *SSID* = Latidos_Web
- *PASSWORD* = 1234abcd#20

Posteriormente, se utiliza el monitor serie de *arduino*, mismo que imprime en pantalla la dirección *IP*, como se muestra en la figura 63.

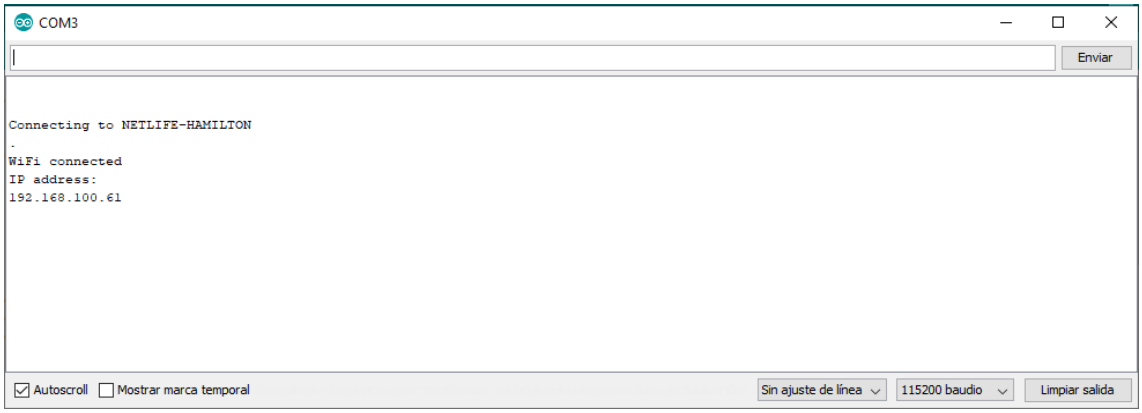


Figura 63. Dirección *IP* del prototipo.

De la misma manera, se puede verificar la dirección *IP* desde el dispositivo que se comparte internet en modo *hotspot*, desde el apartado de configuraciones, en la opción Dispositivos conectados, como se puede ver en la figura 64.

Figura 64. Dirección *IP* del prototipo en un dispositivo móvil.

Una vez realizada la prueba de conexión del prototipo, el siguiente paso es realizar una prueba del envió de la frecuencia cardíaca, para esto utilizaremos el archivo tipo *JSON* alojado en el servidor, el cual muestra cada minuto un dato nuevo, el mismo que es insertado en la base de datos de *MySQL* y mostrado posteriormente en la página *web* y aplicación móvil. Estos datos se pueden apreciar de mejor manera en la figura 65.

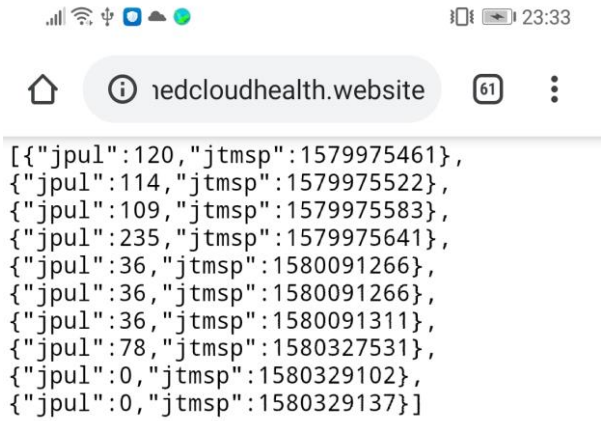


Figura 65. Datos de frecuencia cardíaca.

Los datos del paciente se ven reflejados directamente en la base de datos de *MySQL* del sistema, como se puede apreciar en la figura 66.

Figura 66. Datos de frecuencia cardíaca en base de datos *MySQL*.

Así mismo, la información mostrada en la base de datos de *MySQL* es representada de manera gráfica en la pestaña del paciente, como se puede apreciar en la figura 67.

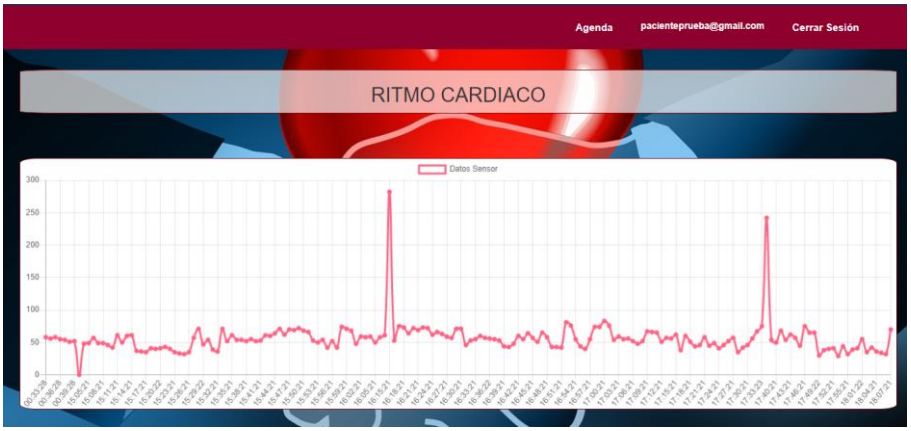


Figura 67. Datos de frecuencia cardíaca en histograma.

Por otro lado, es fundamental considerar los niveles de frecuencia cardíaca, debido a que, si estos datos se salen del rango preestablecido como normal en el desarrollo de la programación, se almacena automáticamente un registro en la tabla agenda con la fecha y la hora actual del sistema, de tal forma que el paciente pueda ser atendido por un médico de forma inmediata.

Para las pruebas respectivas de este proceso es necesario ingresar datos de frecuencia cardíaca mayores a 86 en la tabla sensor, como se puede observar en la figura 68.

Figura 68. Frecuencia cardíaca mayor a 86.

De la misma manera, este procedimiento se facilita gracias a un *Trigger* implementado en la base de datos que permite tomar el dato receptado por el sensor, compararlo y almacenar una agenda automáticamente con un médico en caso de ser necesario, esto se puede apreciar de mejor manera en la figura 69.

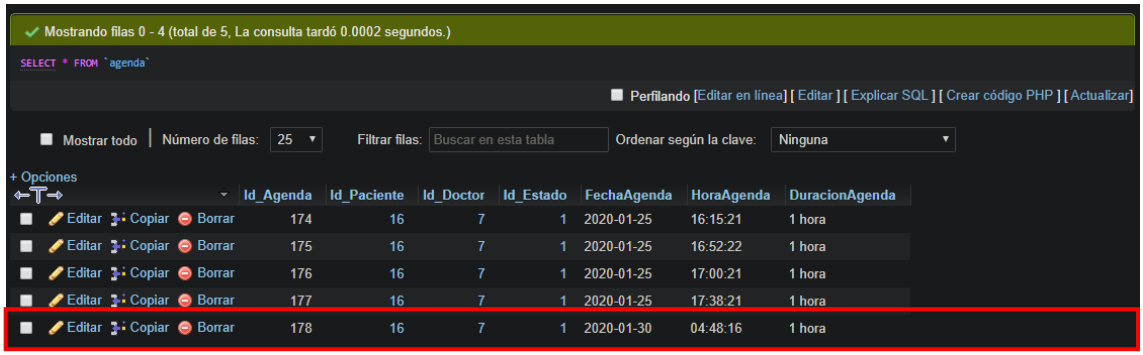


Figura 69. Registro de agenda automática por valor de frecuencia alto.

Esta información se la puede apreciar de mejor manera en la interfaz gráfica del paciente y del médico como se pueden apreciar en la figura 70.

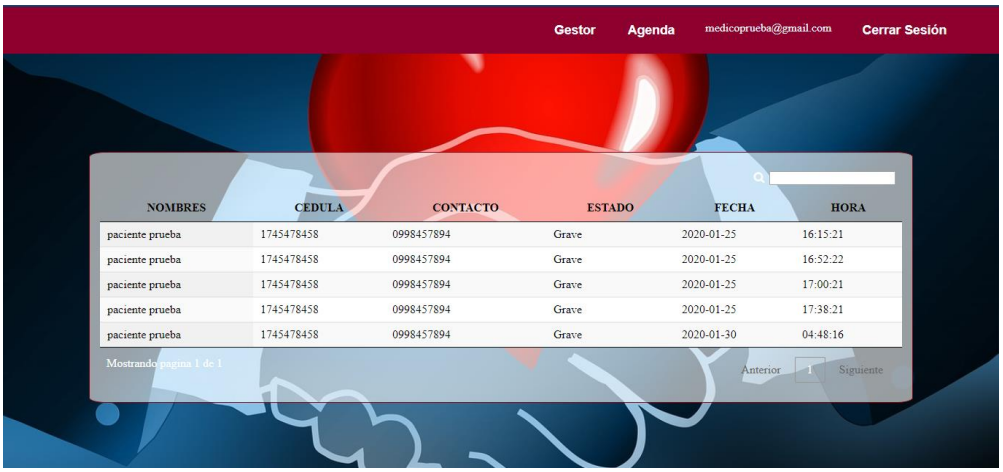


Figura 70. Vista de agenda Médico y Paciente.

Un punto importante para tener en consideración es la ubicación del servidor donde se va a alojar el sistema, ya que dicho sistema se encuentra configurado y programado para tomar el horario del servidor actual.

Por lo que al realizar las pruebas de funcionamiento del envió de datos del prototipo hacia la aplicación en la nube, la información se almacena con una diferencia de 5 horas, debido a la zona horaria donde se ubica el servidor.

Para solucionar este inconveniente, es necesario configurar la programación definiendo la zona horaria más cercana a nuestra ubicación. En este caso se ha optado por utilizar la de “America/*New-York*” ya que su horario se encuentra sincronizado con el Ecuador, como se puede apreciar en la figura 71.

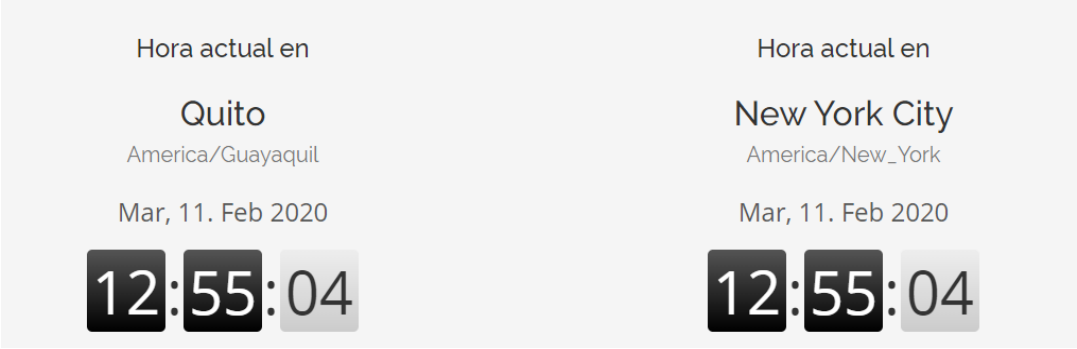


Figura 71. Zona horaria Quito – *New York City*.

Las líneas de código necesarias para configurar la zona horaria de nuestra necesidad se encuentran a continuación en la figura 72.

```
$fecha = new DateTime();
$Timestamp = $fecha->getTimestamp();
date_default_timezone_set("America/New_York");
$hora1 = date("His");
echo $hora1;
$date = date('ymt');
echo $date;
```

Figura 72. Funciones para configurar la zona horaria.

Por otro lado, el cálculo de la frecuencia cardíaca se lo lleva a cabo mediante una comparativa entre el número de latidos máximos que una persona en reposo puede tener y el dato de frecuencia cardíaca máxima.

Por otro lado, al realizar las pruebas en un escenario donde el paciente no se encuentra en un estado de reposo la aplicación muestra un valor de frecuencia cardíaca superior a 80. Como se puede apreciar en la figura 73.

Figura 73. Frecuencia cardíaca de un paciente en un estado distinto a reposo.

Para validar la información obtenida por el sensor de ritmo cardíaco se utilizó un monitor de ritmo cardíaco comercial, de tal manera que se puedan comprar los valores obtenidos en cada uno de los dispositivos. En las tablas 22, 23 y 24

se pueden apreciar los distintos datos obtenidos por el prototipo del sistema y el monitor de ritmo cardíaco juntamente con su error.

Tabla 22.

Datos de frecuencia cardíaca prueba 1.

	Monitor de ritmo cardíaco	**Prototipo(pulsera)**	**%error**
Primera lectura	72	74	2.7%
Segunda lectura	72	75	4%
Tercera lectura	72	74	2.7%

Tabla 23.

Datos de frecuencia cardíaca prueba 2.

	Monitor de ritmo cardíaco	**Prototipo(pulsera)**	**%error**
Primera lectura	62	65	4.61%
Segunda lectura	62	64	3.13%
Tercera lectura	61	63	3.17%

Tabla 24.

Datos de frecuencia cardíaca prueba 3.

	Monitor de ritmo cardíaco	**Prototipo(pulsera)**	**%error**
Primera lectura	67	69	2.89%
Segunda	66	69	4.34%

lectura			
Tercera lectura	67	68	2.94%

6. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

Una vez analizado una alta variedad de componentes y con base a sus características, se determinó que los elementos utilizados en este prototipo cumplen con las condiciones óptimas para ser integrados dentro de la arquitectura de este aplicativo.

El costo económico de los componentes evaluados es relativamente bajo, lo mismo que aumentaría su demanda en caso de ser comercializado, además es importante recalcar que las características de los elementos mencionados anteriormente son propias de la tecnología *IoT*, la cual busca automatizar la mayor cantidad de procesos y evitar en gran parte la interacción humana.

El estudio efectuado a las distintas herramientas de programación permitió seleccionar los lenguajes de programación más eficientes, estos a su vez se complementan para el desarrollo óptimo de *back-end y front-end*, independientemente del entorno de desarrollo, asegurando una aplicación amigable con el usuario sin descuidar su funcionalidad.

La red posee un modelo Cliente-Servidor permitiendo que el sistema sea escalable de forma vertical u horizontal. Adicionalmente, el dispositivo que mide los pulsos cardíacos (pulsera) se conecta a una red inalámbrica mediante un módulo *Wi-Fi*. Sin embargo, esto está condicionado a que exista una red inalámbrica disponible.

El diseño electrónico de la pulsera o dispositivo consta de cuatro etapas, las cuales están relacionadas y dependen la una de la otra, debido a que cada etapa del diseño cumple con un trabajo específico; es decir, para que las etapas de recopilación, procesamiento y envió de datos funcionen, necesitan de una etapa de alimentación.

Mediante el diseño e implementación del prototipo electrónico, se logró tomar varias muestras de frecuencia cardíaca en 1 minuto en la etapa de procesamiento y en un estado de reposo por parte del usuario, de tal manera que esta información resulte ser más confiable y además facilite el diagnóstico por parte del profesional médico.

El sistema utiliza una arquitectura *Device to Gateway* tomado de los modelos de conectividad de la tecnología *IoT*, misma que resalta una conectividad *wi-fi* para el envío de información, en este caso el prototipo electrónico o pulsera se conectan con un proveedor de servicios en la nube, de tal forma que se pueda garantizar una mayor disponibilidad y accesibilidad a los datos.

Una de las ventajas de desarrollar el sistema informático utilizando lenguajes de programación para *back-end y front-end* como *PHP, JavaScript, Css*, entre otros. Es la compatibilidad con varios *Hosting Web de* bajos costos y gratuitos, logrando de esta manera facilitar su alojamiento y la accesibilidad a los datos ya sea por parte de los médicos como también de los pacientes.

Al realizar las pruebas pertinentes tanto en estado de reposo como en actividad física, a partir de las tablas de la sociedad española de cardiología, se logra concluir que los rangos del prototipo cumplen con el umbral establecido por dicha sociedad, demostrando de esta manera la efectividad del prototipo.

Las pruebas realizadas con el prototipo electrónico y el monitor de ritmo cardíaco comercial presentan ciertos valores con un rango o margen de error aproximado entre 2.7 a 4.61%, los cuales indican una pequeña variación, proporcionado un nivel de confianza mayor al 95% que se considera como aceptable, debido a que el *datasheet* del sensor no proporciona información acerca de este valor, se usó el modelo experimental para llegar a este resultado.

Se logro concluir que al utilizar *JSON/JavaScript* el intercambio de datos entre el dispositivo y el sistema resulta en un procesamiento de información más rápido ya que este lenguaje de programación tiene un formato de texto sencillo, evitando así el uso de sintaxis complejas que ralentizan el procesamiento de información.

6.2 Recomendaciones

Para mantener una mayor portabilidad y movilidad de la pulsera o prototipo electrónico, sin que esta se desconecte de la red, se recomienda conectarse a una red wi-fi de un teléfono móvil en modo *hotspot*, es decir compartir acceso a internet, de tal manera que dicha pulsera pueda mantener un mayor rango de cobertura para su funcionalidad.

Es fundamental considerar las distintas etapas que componen el prototipo electrónico ya que cada una de estas depende de las otras. Además, se recomienda un estudio previo de los elementos y herramientas para el desarrollo de un proyecto donde se incluyan comparativas que faciliten la elección más adecuada de dichos elementos para proporcionar una solución optima

Utilizar *Triggers* en la base de datos es de gran ayuda para automatizar los procesos en las distintas tablas que componen un diseño lógico, de tal manera, que se pueda minimizar el código en el desarrollo de la página *web*, logrando minimizar de esta manera el tiempo de ejecución de esta. En el desarrollo de este trabajo, fue necesario un *Trigger* para automatizar el registro de una nueva agenda en caso de ser necesario.

Tener la consideración la estructura de los lenguajes de programación a utilizarse, ya sean para el desarrollo del *back-end*, como también para el *front - end*, de tal manera, que se puedan evitar errores por sintaxis. De la misma manera, es importante mantener una buena organización del código para una fácil compresión, en caso de algún cambio. Por otro lado, también se

recomienda la utilización de diagramas de flujo, ya que son de gran ayuda para determinar los distintos procesos, así como también el orden en el que se ejecutan estos.

Para futuras mejoras del sistema se recomienda integrar una solución que asegure una mayor flexibilidad al momento de agendar automáticamente una cita médica, teniendo en consideración la disponibilidad del médico y el estado del paciente, de tal manera, que si el médico no se encuentra disponible por alguna razón, se pueda agendar automáticamente una cita con otro médico registrado previamente en el sistema.

REFERENCIAS

Arias, O. (2017). *Diseño e implementación de un sistema de monitoreo para la medición del pulso cardiaco y saturación de oxígeno en la sangre*. Recuperado el 14 de abril de 2019 de http://dspace.udla.edu.ec/bitstream/33000/8359/1/UDLA-EC-TIRT-2017-36.pdf

AndroidStudio, C. (s.f). *Introducción a Android Studio*, Recuperado el 02 de diciembre de 2019 de https://developer.android.com/studio/intro

Arduino. (s.f). *¿Qué es Arduino?* Recuperado el 02 de noviembre de 2019 de https://arduino.cl/que-es-arduino/

Bolaños, D. (2018). *Sensor de pulso cardiaco*, Recuperado el 13 de noviembre de 2019 de http://www.bolanosdj.com.ar/MOVIL/ARDUINO2/TeoriaSensorPulsos.pdf

Boutros, N., Saikali, K., Abou, R. (2018). *An IoMT platform to simplify the development of healthcare monitoring applications*. Recuperado el 20 de mayo de 2019 de https://ieeexplore.ieee.org/document/8357124

Bret, A. (2018). *Samsung Health*. Recuperado el 14 de abril de 2019, de https://www.tuexpertoapps.com/2018/09/13/samsung-health-asi-la-renovada-aplicacion-salud-samsung/

Carrión, V. (2015). *Desarrollo de una aplicación web basada en el modelo vista controlador para la gestión de las historias clínicas de los pacientes en el centro de salud de San Jerónimo*. Recuperado el 17 de octubre de 2019 de http://repositorio.unajma.edu.pe/bitstream/handle/123456789/177/10-2015-EPIS-%20Carrion%20Abollaneda%20Victor-desarrollo%20de%20una%20apliccacion%20web%20modelo%20vista.pdf?sequence=1&isAllowed=y

Castro, R. (2019). *Going forward more Secure WIFI Networks*. Recuperado el 16 de octubre del 2019 de http://www.rediris.es/difusion/publicaciones/boletin/73/ENFOQUE1.pdf

Cemeblog. (2019). *Tipo de estandares Wi-fi*. Recuperado el 08 de noviembre de 2019 de http://blog.cemebe.info/tag/wifi/

Components101. (2019). *Módulo de carga / descarga de batería de iones de litio TP4056A*. Recuperado el 20 de noviembre del 2019 de https://components101.com/tp4056a-li-ion-battery-chargingdischarging-module

DFROBOT. (2019). *Cheapduino*. Recuperado el 13 de noviembre de 2019 de https://www.dfrobot.com/product-899.html

Editorial Equipo. (2018). IoMT: *Internet de las cosas médicas y sus aportes a la salud*. Recuperado el 15 de octubre de 2019 de https://reportedigital.com/iot/internet-de-las-cosas-medicas-iomt/

EDUCBA. (s.f). *MySQL vs SQL Server*. Recuperado el 07 de noviembre de 2019 de https://www.educba.com/mysql-vs-mssql/

EP. (2019). *Toma de Constantes Vitales*. Recuperado el 02 de noviembre del 2019 de https://enfermeriapractica.com/procedimientos/toma-de-constantes-vitales

Fajardo, V. (2017). *5 ventajas de Micrsoft Azure en su empresa*. Recuperado el 13 de noviembre de 2019 de https://www.migesamicrosoft.com/5-ventajas-micrsoft-azure-empresa

Gaur, N., Kurma, P., Thakur, S. (2017). *An IoMT based medical-aid-alarm system with online server assistance*. Recuperado el 20 de mayo del 2019 de https://ieeexplore.ieee.org/document/7943155

Gesport. (s.f). *La frecuencia cardiaca*. Recuperado el 17 de octubre de 2019 de http://www.elromeralcastejon.com/archivos/FORMACION/LA%20FRECUENCIA%20CARDIACA.pdf

Hernández, L. (2019). *ESP8266 todo lo que necesitas saber del módulo WiFi para Arduino*. Recuperado el 04 de diciembre de 2019 de https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

Hostinger. (s.f). *¿Qué es un hosting?*. Recuperado el 05 de noviembre de 2019 de https://www.hostinger.es/tutoriales/que-es-un-hosting

Hostinggratis, D. (2019). *000WebHost*. Recuperado el 13 de noviembre de 2019 de https://hostinggratis.xyz/opiniones/000webhost

Korth, H. (2002). *introducción*. En Fundamentos de Bases de Datos (p. 24). España: McGraw-Hill Inc.

León, I. (s.f). *Definición e historia de las herramientas CASE*. Recuperado el 02 de diciembre de 2019 de https://sites.google.com/site/ingenierialeosw/unidad-1-fundamentos-de-ingenieria-de-software/1-5-definicion-e-historia-de-las-herramientas-case

Lletí, Fidel, V. (2015). *Comunicación Bluetooth entre Arduino UNO y Android aplicado a un detector de mentiras*. Recuperado el 17 de noviembre de 2019 de https://riunet.upv.es/bitstream/handle/10251/57549/Memoria.pdf?sequence=1

Llordachs, F. (2016). *E-Health, Internet de las cosas médicas (IoMT)*. Recuperado el 14 de abril de 2019 de http://www.sumandohistorias.com/reportajes/e-health-internet-cosas-medicas/

Lui, Y., Niu, J., Yang, L., Shu, L. (2014). *An IoT-based system for NCD patient's homecare in China*. Recuperado el 12 de mayo de 2019 de https://ieeexplore.ieee.org/document/7037175

NCI. (2019). *frecuencia cardíaca*. Recuperado el 17 de octubre de 2019, de https://www.cancer.gov/espanol/publicaciones/diccionario/def/frecuencia-cardiaca

Nuñez, V. (2016). *Internet of Things and Home-Centered Health*. Recuperado el 16 de octubre de 2019 de http://rcientificas.uninorte.edu.co/index.php/salud/article/viewArticle/7580/9870

OMG. (2015). *Informe sobre la salud en el mundo*. Recuperado el 14 de abril de 2019 de https://www.who.int/whr/2006/overview/es/

Rambal, V. (2019). *Modulo Wifi ESP8266*. Recuperado el 13 de noviembre de 2019 de https://rambal.com/wi-fi/544-modulo-wifi-esp8266.html

Romero, Y. (2019). *Patrón Modelo-Vista-Controlador*. Telem@tica, 11, pp.48-51.

Rossano, V. (2009). *Electrónica & Microcontroladores PIC*. Argentina: Users.

Samsung. (s.f). *Samsung Health*. Recuperado el 14 de abril de 2019 de https://www.samsung.com/latin/apps/samsung-health/

SEC. (2019). *Controla tu riesgo: Frecuencia cardiaca*. Recuperado el 17 de octubre de 2019 de https://fundaciondelcorazon.com/images/stories/file/controla-tu-riesgo-frecuencia-cardiaca.pdf

Tamgno, J., Rokhaya, N., Lishou, C. (2018). *IoT-based medical control system.* Recuperado el 20 de mayo de 2019 de https://ieeexplore.ieee.org/document/8323772

Tdrobotica. (2019). *Sensor de pulso*. Recuperado el 13 de noviembre de 2019 de http://tdrobotica.co/sensor-de-pulso/356.html

Thaler, D. (2015). *Architectural Considerations in Smart Object Networking*. Recuperado el 16 de octubre de 2019 de https://www.ietf.org/rfc/rfc7452.txt

Velázquez, P. Villagrán, L. (2016). *Monitor de ritmo cardiaco para dispositivos Android mediante un enlace Bluetooth*. Recuperado el 21 de noviembre de 2019 de

https://tesis.ipn.mx/bitstream/handle/123456789/20455/I.C.E.%2012-16%20-%2019-CD28.pdf?sequence=1&isAllowed=y

VisualStudio, C. (s.f). *Visual Studio Code en acción*. Recuperado el 02 de diciembre de 2019 de https://code.visualstudio.com/docs

Vocore2. (s.f). *Vocore2*. Recuperado el 05 de noviembre de 2019 de https://vocore.io/v2.html

Zhang, G., Li, C., Zhang, Y. (2012). *SemanMedical: A kind of semantic medical monitoring system model based on the IoT sensors*. Recuperado el 20 de mayo de 2019 de https://ieeexplore.ieee.org/document/6379414

ANEXOS

ANEXO 1. Script de la base de datos

```
-- phpMyAdmin SQL Dump

-- version 4.8.2

-- https://www.phpmyadmin.net/

-- Servidor: 127.0.0.1

-- Tiempo de generación: 14-11-2019 a las 21:57:54

-- Versión del servidor: 10.1.34-MariaDB

-- Versión de PHP: 7.2.8

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time_zone = "+00:00";

/*!40101 SET
        @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIE
        NT */;

/*!40101 SET
        @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_R
        ESULTS */;

/*!40101 SET
        @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTI
        ON */;

/*!40101 SET NAMES utf8mb4 */;

-- Base de datos: `medcloud`

-- Estructura de tabla para la tabla `doctor`

CREATE TABLE `doctor` (
```

```
  `Id_Doctor` int(11) NOT NULL AUTO_INCREMENT,
  `NombreDoctor` varchar(20) NOT NULL,
  `ApellidoDoctor` varchar(20) NOT NULL,
  `SexoDoctor` char(1) NOT NULL,
  `FechaNacimiento` date NOT NULL,
  `TelefonoDoctor` varchar(10) NOT NULL,
  `CorreoDoctor` varchar(40) NOT NULL,
  `CedulaDoctor` varchar(10) NOT NULL,
  `ContrasenaDoctor` varchar(20) NOT NULL,
  PRIMARY KEY (`Id_Doctor`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;

-- Volcado de datos para la tabla `doctor`

INSERT INTO `doctor` (`Id_Doctor`, `NombreDoctor`, `ApellidoDoctor`,
    `SexoDoctor`, `FechaNacimiento`, `TelefonoDoctor`, `CorreoDoctor`,
    `CedulaDoctor`, `ContrasenaDoctor`) VALUES
(1, 'Carmelino', 'Cartagena', 'M', '2019-10-17', '2845478',
    'Carmelino@gmail.com', '174248745', '1234'),
(2, 'Dennis', 'Lara', 'M', '2019-10-17', '2145748', 'Dennis@gmail.com',
    '1457847847', '1234');

-- Estructura de tabla para la tabla `estado`

CREATE TABLE `estado` (
  `Id_Estado` int(11) NOT NULL AUTO_INCREMENT,
  `NombreEstado` varchar(100) NOT NULL,
  `DescripcionEstado` varchar(100) NOT NULL,
  PRIMARY KEY (Id_Estado)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;

-- Volcado de datos para la tabla `estado`

INSERT INTO `estado` (`Id_Estado`, `NombreEstado`, `DescripcionEstado`)
        VALUES

(1, 'Bien', 'ninguna'),

(2, 'Grave', 'ninguna');

-- Estructura de tabla para la tabla `paciente`

CREATE TABLE `paciente` (

  `Id_Paciente` int(11) NOT NULL AUTO_INCREMENT,

  `NombrePaciente` varchar(20) NOT NULL,

  `ApellidoPaciente` varchar(20) NOT NULL,

  `SexoPaciente` char(1) NOT NULL,

  `FechaNacimiento` date NOT NULL,

  `TelefonoPaciente` varchar(10) NOT NULL,

  `CorreoPaciente` varchar(40) NOT NULL,

  `CedulaPaciente` varchar(10) NOT NULL,

  `ContrasenaPaciente` varchar(20) NOT NULL,

   PRIMARY KEY (`Id_Paciente`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;

-- Volcado de datos para la tabla `paciente`

--Inserts de prueba

INSERT INTO `paciente` (`Id_Paciente`, `NombrePaciente`, `ApellidoPaciente`,
        `SexoPaciente`, `FechaNacimiento`, `TelefonoPaciente`,
        `CorreoPaciente`, `CedulaPaciente`, `ContrasenaPaciente`)
        VALUES
```

```
(1, 'pacienteP', 'prueba', 'M', '2019-10-17', '1421457', 'paciente@gmail.com',
        '1424748745', '1234'),

(2, 'prueba2', 'prueba2', 'M', '2019-10-17', '1245748', 'prueba2@gmail.com',
        '1475478459', '1234');

-- Estructura de tabla para la tabla `sensor`

CREATE TABLE `sensor` (
  `Id_Sensor` int(11) NOT NULL AUTO_INCREMENT,
  `Id_Paciente` int(11) NOT NULL,
  `Id_Tipo` int(11) NOT NULL,
  `DatoSensor` int(11) NOT NULL,
  `AgendaSensor` int(11) NOT NULL,
  `HoraSensor` time NOT NULL,
  `FechaSensor` date NOT NULL,
  `DescripcionSensor` varchar(100) NOT NULL,
  PRIMARY KEY (`Id_Sensor`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;

-- Volcado de datos para la tabla `sensor`

INSERT INTO `sensor` (`Id_Sensor`, `Id_Paciente`, `Id_Tipo`, `DatoSensor`,
        `FechaSensor`, `DescripcionSensor`) VALUES

(1, 1, 1, 50, '12:00:00', 'Fortttt!!!!'),

(2, 1, 1, 100, '14:00:00', 'Maimeeee!!!'),

(3, 1, 1, 65, '16:00:00', 'Richy Fort'),

(4, 1, 1, 60, '12:00:00', 'ritmo cardiaco');

-- Estructura de tabla para la tabla `agenda`
```

```
CREATE TABLE `agenda` (
  `Id_Agenda` int(11) NOT NULL AUTO_INCREMENT,
  `Id_Paciente` int(11) NOT NULL,
  `Id_Doctor` int(11) NOT NULL,
  `Id_Estado` int(11) DEFAULT NULL,
  `FechaAgenda` datetime NOT NULL,
  `ObservacionAgenda` varchar(40) NOT NULL,
PRIMARY KEY (`Id_Agenda`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;
-- Volcado de datos para la tabla `agenda`
-- Estructura de tabla para la tabla `tipo`
CREATE TABLE `tipo` (
  `Id_Tipo` int(11) NOT NULL AUTO_INCREMENT,
  `NombreTipo` varchar(60) NOT NULL,
  `DescricionTipo` varchar(100) NOT NULL,
  PRIMARY KEY (`Id_Tipo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;
-- Volcado de datos para la tabla `tipo`
INSERT INTO `tipo` (`Id_Tipo`, `NombreTipo`, `DescricionTipo`) VALUES
(1, 'ritmo cardiaco', 'ninguna'),
(2, 'podometro', 'ninguna');
-- Índices para tablas volcadas
-- Indices de la tabla `sensor`
```

```
ALTER TABLE `sensor`
  ADD KEY `Paciente_Sensor` (`Id_Paciente`),
  ADD KEY `Tipo_Sensor` (`Id_Tipo`);
-- Restricciones para tablas volcadas
-- Filtros para la tabla `agenda`
ALTER TABLE `agenda`
  ADD CONSTRAINT `Doctor_Agenda` FOREIGN KEY (`Id_Doctor`) REFERENCES `doctor` (`Id_Doctor`) ON DELETE CASCADE,
  ADD CONSTRAINT `Estado_Agenda` FOREIGN KEY (`Id_Estado`) REFERENCES `estado` (`Id_Estado`) ON DELETE CASCADE,
  ADD CONSTRAINT `Paciente_Agenda` FOREIGN KEY (`Id_Paciente`) REFERENCES `paciente` (`Id_Paciente`) ON DELETE CASCADE;
-- Filtros para la tabla `sensor`
ALTER TABLE `sensor`
  ADD CONSTRAINT `Paciente_Sensor` FOREIGN KEY (`Id_Paciente`) REFERENCES `paciente` (`Id_Paciente`) ON DELETE CASCADE,
  ADD CONSTRAINT `Tipo_Sensor` FOREIGN KEY (`Id_Tipo`) REFERENCES `tipo` (`Id_Tipo`) ON DELETE CASCADE;
COMMIT;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET
```

```
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;
```

ANEXO 2. Triggers para registrar la una cita médica automaticamente

```
DROP TRIGGER IF EXISTS `sensorAgenda`;CREATE DEFINER=`root`@`loca
lhost` TRIGGER `sensorAgenda` AFTER INSERT ON
`sensor` FOR EACH ROW if(NEW.DatoSensor>=84) THEN INSERT
INTO `agenda` (`Id_Agenda`, `Id_Paciente`, `Id_Doctor`, `Id_Estado`
, `FechaAgenda`, `HoraAgenda`, `DuracionAgenda`) VALUES (NULL
, NEW.Id_Paciente, '1', '1', NOW(), now(), '30
minutos'); UPDATE `paciente` SET `ocupadoPaciente` = '1' WHERE
`paciente`.`Id_Paciente` = NEW.Id_Paciente; ELSEIF(NEW.DatoSen
sor>100) THEN INSERT INTO `agenda` (`Id_Agenda`, `Id_Paciente`
, `Id_Doctor`, `Id_Estado`, `FechaAgenda`, `HoraAgenda`, `Duracion
Agenda`) VALUES (NULL, NEW.Id_Paciente,'1', '2', NOW(), now(), '1
hora'); UPDATE `paciente` SET `ocupadoPaciente` = '1' WHERE `pa
ciente`.`Id_Paciente` = NEW.Id_Paciente; END IF
```

```
DROP TRIGGER IF EXISTS `ActualizarAgenda`;CREATE DEFINER=`root`@`l
ocalhost` TRIGGER `ActualizarAgenda` AFTER UPDATE ON
`paciente` FOR EACH ROW UPDATE `agenda` SET `Id_Doctor` =
NEW.Id_Doctor WHERE
`agenda`.`Id_Agenda` = (SELECT Id_Agenda from agenda INNER J
OIN paciente ON agenda.Id_Paciente=paciente.Id_Paciente WHERE
paciente.Id_Paciente=NEW.Id_Paciente LIMIT 1)
```

ANEXO 3. Código de la aplicación Web

```
//Resgitro de nuevos usuarios
<?php
require_once '../conexionBD/conexionmysqli.php';
if(isset($_POST['submitPD'])){
    $nombre = $_POST['nombre'];
    $apellido = $_POST['apellido'];
    $genero = $_POST['genero'];
    $fecha = $_POST['fecha'];
    $telefono = $_POST['telefono'];
    $correo = $_POST['correo'];
    $cedula = $_POST['cedula'];
    $contrasena = $_POST['contrasena'];
    $consulta="INSERT INTO paciente(NombrePaciente, ApellidoPaciente,
            SexoPaciente,FechaNacimiento, TelefonoPaciente, CorreoPaciente,
            CedulaPaciente,ContrasenaPaciente)
    VALUES('$nombre', '$apellido', '$genero', '$fecha','$telefono',
            '$correo','$cedula', '$contrasena')";
    if( mysqli_query($conexion, $consulta)){
    echo "<script language='JavaScript'>";
    echo "location = '../roles/doctor.php'";
    echo "</script>";
    }else{
       echo "ERROR: Could not able to execute $consulta. " .
            mysqli_error($conexion);
```

```
    }
}else{
    echo "Error";
}
?>
//Actualización de pacientes de usuarios
<?php
require_once '../conexionBD/conexionmysqli.php';
$id = isset($_GET['id'] )? $_GET['id'] : 0;
?>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<style>
*{
    padding: 0;
    margin: 0;
}
.formularioContenedor{
```

```
    display: flex;
    position: relative;
    justify-content: center;
    background-image: url("../img/img1.jpg");
    background-size: cover;
}
.formulario{
    display: flex;
    flex-flow: column wrap;
    border: 1px solid rgba(129, 2, 14, 1);
    box-shadow: 2px 2px 2px rgba(89, 89, 89, 0.2);
    font-family: 'Roboto', sans-serif;
    background-color: rgba(254, 254, 254, 0.6);
    font-weight: lighter;
    width:30% ;
}
.formularioContenedor .formulario label{
    display: flex;
    color: #343535;
    font-size: 90%;
    margin-left: 28px;
    padding: 5px;
    font-weight: bold;
```

```
}
.formulario input{
    border: 1px solid rgba(70, 70, 70, 0.4);
    border-radius: 3px;
    padding: 5px;
    width: 80%;
    color: rgba(0, 142, 55, 0.8);
    align-self: center;
}
.formulario button{
    margin:15px;
    background-color: #81020e;
    padding: 5px;
    color: white;
    border: black;
    border-radius: 3px;
    cursor: pointer;
    font-size: 15px;
    font-weight: bold;
    width: 190px;
    height: 30px;
    align-self: center;
}
```

```
.formulario button:hover{
    background-color: #CD5C5C;
    color:#81020e;
    font-size: 13px;
    border: 2px solid rgba(129, 2, 14, 1);
}
h1{
    font-size: 145%;
    align-self: center;
    margin: 3%;
}
</style>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
        <div class="formularioContenedor">
                <div class="formulario">
                        <h1>Actualizacion de Usuario</h1>
```

<label>Nombre</label>

<label>Apellido</label>

<label>Genero</label>

<label>Fecha de Nacimiento</label>

<label>Telefono</label>

<label>Correo</label>

<label>Cedula</label>

<label>Contraseña</label>

<form id="formularioModificar" action="update.php" method="POST" enctype="multipart/form-data">

<?php

$consulta = "SELECT * FROM paciente WHERE Id_Paciente = '$id'";

$resultado = $conexion->query($consulta);

//Ejemplo para imprimir los datos. El bucle recorre todos los registros.

if($fila = $resultado->fetch_assoc()){

echo "<input style='display:none' readonly='readonly' name='id' type='text' value=$id>";

echo "<input class='ingresar' onkeyup='mayus(this);' name='nombre' type='text' value = '".$fila['NombrePaciente']."' placeholder='Nombre'></input>";

echo "<input class='ingresar' onkeyup='mayus(this);' name='apellido' type='text' value = '".$fila['ApellidoPaciente']."' placeholder='Apellido'></input>";

echo "<input class='ingresar' onkeyup='mayus(this);'

```
name='genero' type='text' value = '".$fila['SexoPaciente']."' placeholder='Genero'></input>";

        echo "<input class='ingresar' onkeyup='mayus(this);' name='fecha' type='date' value = '".$fila['FechaNacimiento']."' placeholder='Fecha de Nacimiento'></input>";

        echo "<input class='ingresar' onkeyup='mayus(this);' name='telefono' type='text' value = '".$fila['TelefonoPaciente']."' placeholder='Telefono'></input>";

        echo "<input class='ingresar' onkeyup='mayus(this);' name='correo' type='text' value = '".$fila['CorreoPaciente']."' placeholder='Correo'></input>";

        echo "<input class='ingresar' onkeyup='mayus(this);' name='cedula' type='text' value = '".$fila['CedulaPaciente']."' placeholder='Cedula'></input>";

        echo "<input class='ingresar' onkeyup='mayus(this);' name='contrasena' type='text' value = '".$fila['ContrasenaPaciente']."' placeholder='Contraseña'></input>";

        echo "<input class='subir btn-sucess' name='submit' type='submit' value='ACTUALIZAR'>";

    }else{

        echo "Aborte mision papi";

    }

    ?>

    </form>

  </div>

</div>

</body>
```

```
</html>

//Eliminar usuarios

<?php

require_once '../conexionBD/conexionmysqli.php';


$id = isset($_GET['id'] )? $_GET['id'] : 0;

$consulta = "DELETE FROM paciente WHERE Id_Paciente = '$id'";

if(mysqli_query($conexion, $consulta)){

    echo "<script language='JavaScript'>";

    echo "location = '../roles/doctor.php'";

    echo "</script>";

   /* echo "Factible";*/

}else{

    echo "ERROR: Could not able to execute $consulta. " .
            mysqli_error($conexion);

}

?>

//Login

<?php

if(isset($_POST['enviar'])) {

    require_once '../conexionBD/conexionmysqli.php';

     $loginGeneral = $_POST['usuario'];

     $loginPasswordG = $_POST['password'];

     $consultaDoc = "SELECT * FROM doctor WHERE CorreoDoctor =
```

```
            '$loginGeneral' AND ContrasenaDoctor = '$loginPasswordG'";
    $consultaUser = "SELECT * FROM paciente WHERE CorreoPaciente =
            '$loginGeneral' AND ContrasenaPaciente = '$loginPasswordG'";
   $resultDoc = $conexion->query($consultaDoc);
   $resultUser = $conexion->query($consultaUser);
   if($resultDoc->num_rows > 0){
     while($row = $resultDoc->fetch_assoc()){
         $doctok = $row['CorreoDoctor'];
         $passDok = $row['ContrasenaDoctor'];
     }
   }elseif($resultUser->num_rows > 0){
      while($row = $resultUser->fetch_assoc()){
         $userok = $row['CorreoPaciente'];
         $passUok = $row['ContrasenaPaciente'];
     }
   }
//    $resultDoc->close();
  //  $resultUser->close();
//$conexion->close();
 if(!empty($loginGeneral) && !empty($loginPasswordG)) {
      if(!empty($doctok) && !empty($passDok) && empty($userok) &&
            empty($passUok)){
       if($loginGeneral == $doctok && $loginPasswordG == $passDok) {
          session_start();
```

```
     $_SESSION['correo_doctor']= $doctok;
     //alert($doctok);
     header("Location: ../roles/doctor/doctor.php");
    // echo '<script language="javascript">alert("'.$doctok.'");</script>';
 }else{
    header("Location: ../index.php");
    header("Location: ../index.php?error=login");
}
 }elseif(empty($doctok) && empty($passDok) && !empty($userok) &&
        !empty($passUok)){
    if($loginGeneral == $userok && $loginPasswordG == $passUok){
        session_start();
        $_SESSION['correo_usuario']= $userok;
        //alert($userok);
        header("Location: ../roles/paciente/paciente.php");
     }else{
        header("Location: ../index.php");
        header("Location: ../index.php?error=login");
    }
 }elseif(empty($doctok) && empty($passDok) && empty($userok) &&
        empty($passUok)){
    if($loginGeneral == 'admin@medcloud.com' && $loginPasswordG ==
        '1234'){
       session_start();
```

```
            $_SESSION['correo_admin']= 'admin@medcloud.com';
            //alert($userok);
            header("Location: ../roles/administrador/admin.php");
          }else{
            header("Location: ../index.php");
            header("Location: ../index.php?error=login");
          }
        }
          else{
            header("Location: ../index.php?error=login");
        }
    }
    else
    {
       header("Location: ../index.php");
    }
 } else {
    header("Location: ../index.php");
 }
?>
//Cerrar sesion
<?php
              session_start();
```

```
            session_unset();
            session_destroy();
            header ("location:../index.php");
            /*@session_start();
            session_destroy();
            session_commit();
            header("Location: ../index.php");*/
 ?>
//Index
<?php
if(isset($_GET["error"]) && $_GET["error"] != "login") {
                header("Location: index.php");
            }
 ?>
<!DOCTYPE html>
<html lang="es">
<head>
            <meta charset="UTF-8">
            <title>LOGIN</title>
            <!--<meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0">-->
            <!-- Latest compiled and minified CSS -->
            <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
```

```
        .min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">

        <!-- Optional theme -->

        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css" integrity="sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r" crossorigin="anonymous">

        <link rel="stylesheet" href="css/estilos.css">

        <link rel="stylesheet" href="css/flexslider.css" type="text/css">

        <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

</head>

<body>

        <div class="container">

            <div class="left">

            <?php

 if(isset($_GET["error"])) {

             echo "<p class='error'>Usuario y / o Contrasena erroneos. Intentelo de nuevo.</p>";

 }


?>

<div class="contenedorLogin">
```

```
<form action="login/login.php" method="post">
<div class="form-group">
<div class="bg-primary">
 <img class="col-md-6 col-md-offset-3 col-xs-3 col-xs-offset-4" src="img/logo.png">
</div>
</div>
 <div class="form-group">
 <input type="text" class="form-control" name="usuario" placeholder="Usuario">
 </div>
 <div class="form-group">
 <input type="password" class="form-control" name="password" placeholder="Password">
 </div>
 <button type="submit" name="enviar" class="btn btn-default">Entrar</button>
 <div class = "form-group">
 <a id="registrarse" href="#">Olvidaste tu contraseña?</a>
 </div>
</form>
</div>
 </div>
 <div class="right">
 <div class="slider">
```

```
                <!--Meter codigo SLIDER AUTOMATICO-->
                 <div class="flexslider">
<ul class="slides">
        <li>
                <img src="img/img4.jpg" alt="">
                <section class="flex-caption">
                        <p></p>
                </section>
        </li>
        <li>
                <img src="img/img2.jpg" alt="">
                <section class="flex-caption">
                        <p></p>
                </section>
        </li>
        <li>
                <img src="img/img3.jpg" alt="">
                <section class="flex-caption">
                        <p></p>
                </section>
        </li>
        <li>
                <img src="img/img4.jpg" alt="">
```

```
                    <section class="flex-caption">
                        <p></p>
                    </section>
                </li>
                <li>
                    <img src="img/img5.jpg" alt="">
                    <section class="flex-caption">
                        <p></p>
                    </section>
                </li>
            </ul>
        </div>
        </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
    <script src="js/jquery.flexslider.js"></script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>
</body>
<script>
    $(document).ready(function(){
```

```
$('.flexslider').flexslider({
        touch: true,
        pauseOnAction: false,
        pauseOnHover: false,
        });
        });
        </script>
</html>
//Slider imagenes
<!DOCTYPE html>
<html lang="es">
<head>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="flexslider.css" type="text/css">
        <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js">
        </script>
        <meta name="viewport" content="width=device-width, user-
        scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-
        scale=1.0">
        <script src="js/jquery.flexslider.js"></script>
</head>
<body>
        <div class="flexslider">
            <ul class="slides">
```

```
<li>
        <img src="imagenes/img4.jpg" alt="">
        <section class="flex-caption">
                <p></p>
        </section>
</li>
<li>
        <img src="imagenes/img2.jpg" alt="">
        <section class="flex-caption">
                <p></p>
        </section>
</li>
<li>
        <img src="imagenes/img3.jpg" alt="">
        <section class="flex-caption">
                <p></p>
        </section>
</li>
<li>
        <img src="imagenes/img4.jpg" alt="">
        <section class="flex-caption">
                <p></p>
        </section>
```

```
                    </li>


                    <li>
                        <img src="imagenes/img5.jpg" alt="">
                        <section class="flex-caption">
                            <p></p>
                        </section>
                    </li>
                </ul>
            </div>
</body>
<script>
</html>
```

ANEXO 4. Datasheet CheapDuino

RB-Dfr-226
Cheapduino Microcontroller (5pk)

Introduction
CheapDuino is the most affordable Arduino compatible processor in the world. The price for each cheapDuino is almost 1/5 price of the Arduino UNO.It's aimed to supply an extremely low cost solution for any disposable projects, such as DIY projects, workshop, gifts projects, E-Textiles and education. For those students from second or third world countries who can not afford the official Arduino, cheapduino opens the door to the physical world.

The cheapDuino integrates a ATmega8 microcontroller as Arduino NG. So it's available to program it directly with Arduino IDE via the FTDI programmer or USB Serial Light Adapter. The cheapDuino is powerful in compact size, which is only 2cm x 2cm dimension.

Although the cheapduino has extremely low price tag, we do spent lots of time to consider every detail of the product. The processor has 3 pwm digital pins, 3 analog pins and power supply interface with hexagonal pads around the board, which make it really easy to solder for the beginners. Of course, the I2C interface and serial port is also available to extend the 2 wire peripheral devices directly. The adoptation of micro FPC connector saves extra space compared with regular Arduino USB interface. "Cheapduino - Not just cheap"

Note:

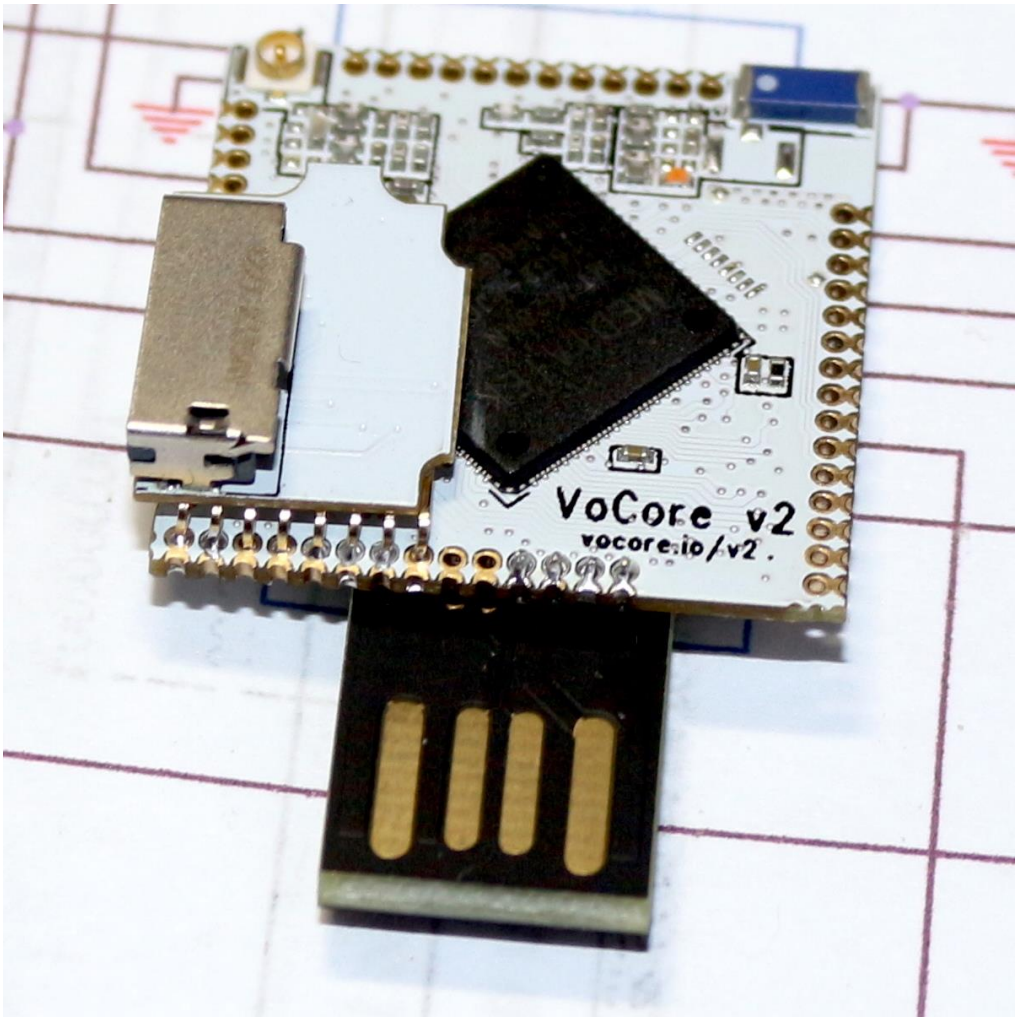
- When plugin the fpc programming cable to DFRobot FPC programmer and cheapDuino, please the blue side facing upward.
- This kit comes with 5 cheapDuino, 2 DFRobot standard fpc programming cable and 1 fpc programmer adapter.The FTDI programmer is not included in this kit.
- We recommended you to buy one of the programmer below and mini USB cable if you don't have it.
- FTDI Basic Breakout 3.3/5V (Arduino Compatible)
- USB Serial Light Adapter (Arduino Compatible)

Specification

- Working voltage: 3~5 volts
- Recommended power supply: 5v

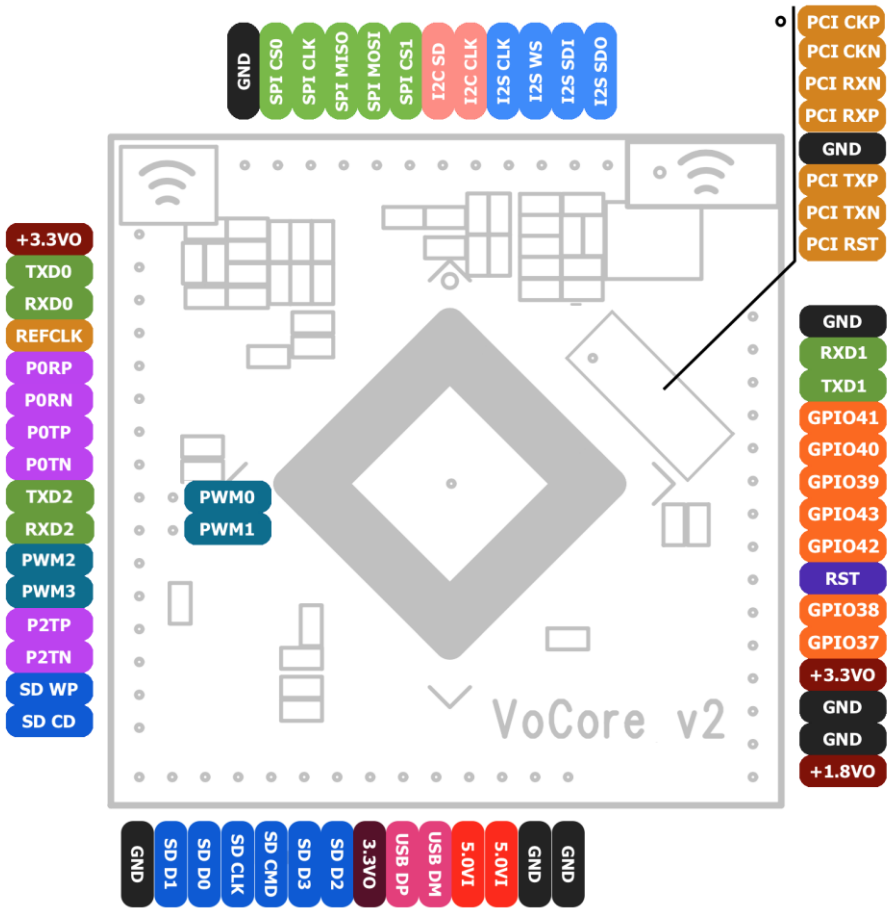
- Microctonroller: Atmel AVR ATmega8
- bootloader(Board option in Arduino IDE): Arduino NG / w ATmega8
- 3 digital pins, 3 analog pins with easy-to-solder hexagonal pads
- Integrate 3 pwm pins,I2C interface and UART interface
- Suitable for workshop,education usage,DIY or compact size projects and E-Textiles
- Low cost Arduino compatible controller
- Dimensions: 2cm x 2cm x 0.2cm

ANEXO 5. Datasheet Vocore2



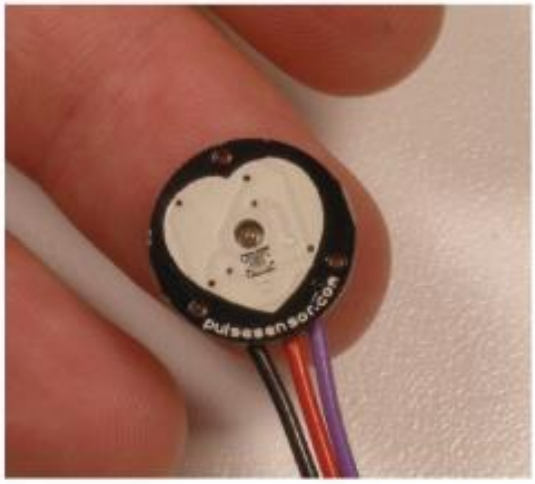
Parámetros

	Detalles
TALLA	25,6 mm x 25,6 mm x 3,0 mm
UPC	MT7628, 580 MHz, MIPS 24K
MEMORIA	128 MB, DDR2, 166 MHz
ALMACENAMIENTO	16M NOR a bordo, admite SDXC de hasta 2TB
INALÁMBRICO	802.11n, 2T2R, velocidad de hasta 300Mbps.
ANTENA	Una ranura U.FL, una antena a bordo.
ETHERNET	1 puerto / 5 puertos, hasta 100Mbps.
USB	Admite USB 2.0 (solo host), hasta 480 MBit / s.
GPIO	alrededor de 40 (pinmux)
UART	x3 (UART2 para la consola de depuración)
PWM	x4
PCIe (opción)	x1 (opción)
TEMPERATURA	0C ~ 40C (carga completa) o -10C ~ 70C
FUENTE DE ALIMENTACIÓN	3.6V ~ 5.5V, 500mA
EL CONSUMO DE ENERGÍA	74mA wifi en espera, 230mA wifi a toda velocidad, entrada de 5V.



ANEXO 6. Datasheet Sensor de frecuencia cardiaca

Pulse Sensor Getting Started Guide



Introduction:

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart-rate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some jumper cables. It also includes an open-source monitoring app that graphs your pulse in real time.

1) A 24-inch Color-Coded Cable, with (male) header connectors. You'll find this makes it easy to embed the sensor into your project, and connect to an Arduino. No soldering is required.

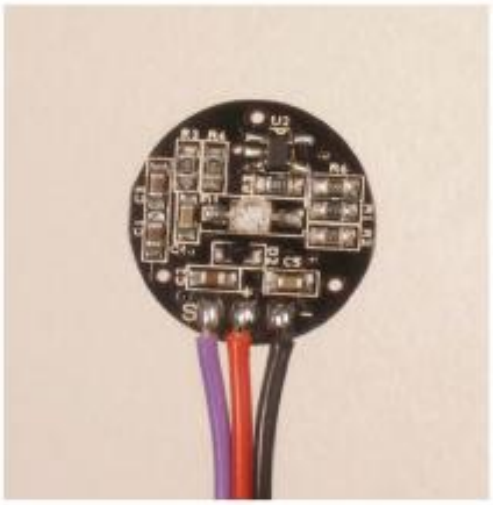
2) An Ear Clip, perfectly sized to the sensor. We searched many places to find just the right clip. It can be hot-glued to the back of the sensor and easily worn on the earlobe.

3) 2 Velcro Dots. These are 'hook' side and are also perfectly sized to the sensor. You'll find these velcro dots very useful if you want to make a velcro (or fabric) strap to wrap around a finger tip.

4) Velcro strap to wrap the Pulse Sensor around your finger.

4) 3 Transparent Stickers. These are used on the front of the Pulse Sensor to protect it from oily fingers and sweaty earlobes.

5) The Pulse Sensor has 3 holes around the outside edge which make it easy to sew it into almost anything.

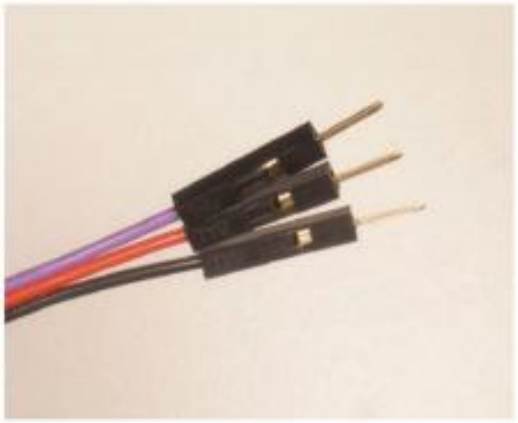


Let's get started with Pulse Sensor Anatomy

The front of the sensor is the pretty side with the Heart logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines through from the back, and there is also a little square just under the LED. The square is an ambient light sensor, exactly like the one used in cellphones, tablets, and laptops, to adjust the screen brightness in different light conditions. The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back. The back of the sensor is where the rest of the parts are mounted. We put them there so they would not get in the way of the of the sensor on the front. Even the LED we are using is a reverse mount LED. For more about the circuit functionality, check out the Hardware page.[needs link]

The cable is a 24" flat color coded ribbon cable with 3 male header connectors.

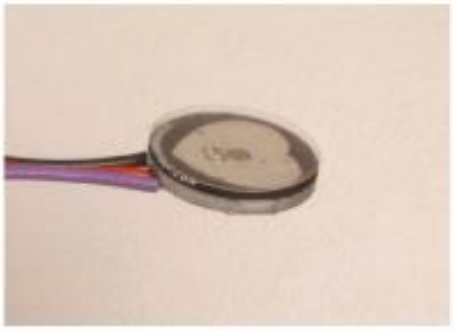
RED wire = +3V to +5V
BLACK wire = GND
PURPLE wire = Signal



The Pulse Sensor can be connected to arduino, or plugged into a breadboard. Before we get it up and running, we need to protect the exposed circuitry so you can get a reliable heart beat signal.

Preparing the Pulse Sensor

Before you really start using the sensor you want to insulate the board from your (naturally) sweaty/oily fingers. The Pulse Sensor is an exposed circuit board, and if you touch the solder points, you could short the board, or introduce unwanted signal noise. We will use a thin film of vinyl to seal the sensor side. Find the small page of four clear round stickers in your kit, and peel one off. Then center it on the Pulse Sensor. It should fit perfectly.



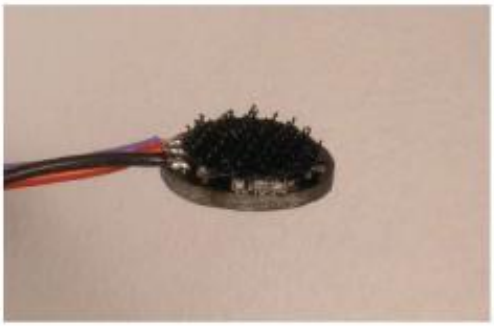
When you are happy with the way it's lined up, squeeze it onto the face all at once! The sticker (made of vinyl) will kind of stretch over the sensor and give it a nice close fit. If you get a wrinkle, don't worry, just press it down really hard and it should stick. We gave you 4, so you can replace it if necessary.

That takes care of the front side. The vinyl sticker offers very good protection for the underlying circuit, and we rate it 'water resistant'. meaning: it can stand to get splashed on, but don't throw it in the pool!

If this is your first time working with Pulse Sensor, you're probably eager to get started, and not sure if you want to use the ear-clip or finger-strap (or other thing). The back of the Pulse Sensor has even more exposed contacts than the front, so you need to make sure that you don't let it touch anything conductive or wet.

The easiest and quickest way to protect the back side from undesireable shorts or noise is to simply stick a velcro dot there for now. The dot will keep your parts away from the Pulse Sensor parts enough for you to get a good feel for the sensor and decide how you want to mount it. You'll find that the velcro dot comes off easily, and stores back on the little strip of plastic next to the other one.

Notice that the electrical connections are still exposed! We only recommend this as a temporary setup so you can get started. We show you how to better seal the Pulse Sensor later in this document.



Running The Pulse Sensor Code

Get the latest Arduino and Processing Pulse Sensor software here http://pulsesensor.com/downloads/

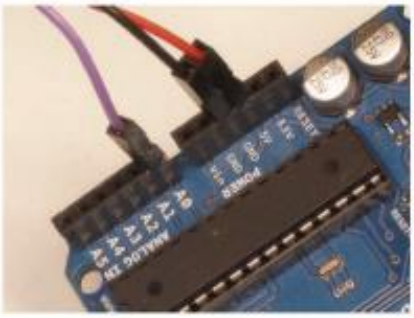
Arduino code is called "PulseSensorAmped_Arduino-xx"
The Processing code is called "PulseSensorAmped_Processing-xx"

We strongly advise that you DO NOT connect the Pulse Sensor to your body while your computer or arduino is being powered from the mains AC line. That goes for charging laptops and DC power supplies. Please be safe and isolate yourself from from the power grid, or work under battery power.
Connect the Pulse Sensor to: +V (red), Ground (black), and Analog Pin 0 (purple) on your favorite Arduino, or Arduino compatible device, and upload the 'PulseSensoAmped_Arduino-xx' sketch.

note: If you want to power Pulse Sensor Amped with low voltage (3.3V for example), make sure you have this line of code in the setup()

```
analogReference(EXTERNAL);
```

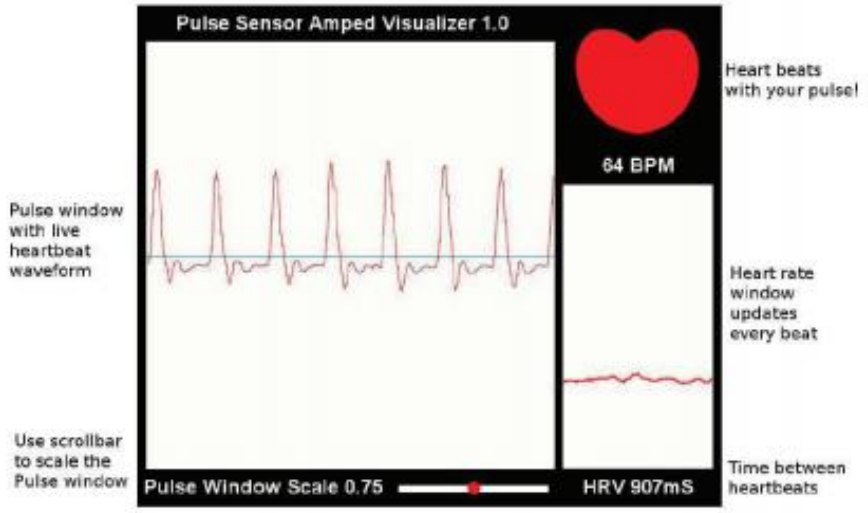
Also, make sure that you apply the lower voltage to the Arduino Aref pin (next to pin 13).



After it's done uploading, you should see Arduino pin 13 blink in time with your heartbeat when you hold the sensor on your fingertip. If you grip the sensor too hard, you will squeeze all the blood out of your fingertip and there will be no signal! If you hold it too lightly, you will invite noise from movement and ambient light. Sweet

Spot pressure on the Pulse Sensor will give a nice clean signal. You may need to play around and try different parts of your body and pressures. If you see an intermittent blink, or no blink, you might be a zombie or a robot.

To view the heartbeat waveform and check your heart rate, you can use the Processing sketch that we made. Start up Processing on your computer and run the 'PulseSensorAmped_Processing-xx' sketch. This is our data visualization software, and it looks like this.



note: If you get an error when starting this code, you may need to make sure you are selecting the right serial port. Check the Troubleshooting section below..

The large main window shows a graph of raw sensor data over time. The Pulse Sensor Data Window can be scaled using the scrollbar at the bottom if you have a very large or very small signal. At the right of the screen, a smaller data window graphs heart rate over time. This graph advances every pulse, and the Beats Per Minute is updated every pulse as a running average of the last ten pulses. The big red heart in the upper right also pulses to the time of your heartbeat. When you hold the Pulse Sensor to your fingertip or earlobe or (fill in body part here) you should see a nice heartbeat waveform like the one above. If you don't, and you're sure you're not a zombie, try the sensor on different parts of your body that have capillary tissue. We've had good results on the side of the nose, middle of the forehead, palm, and lower lip. We're all different, original organisms. Play around and find the best spot on you and your friends. As you are testing and getting used to the sensor. You may find that some fingers or parts of fingers are better than others. For example, I find that when I position the sensor so that the edge of the PCB is at the bottom edge of my earlobe I get an awesome signal. Also, people with cold hands or poor circulation may have a harder time reading the pulse. Run your hands under warm water, or do some jumping-jacks!

Arduino and Processing programming environments available for download here:
www.arduino.cc www.processing.org

ANEXO 7. Datasheet Modulo ESP8266

Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

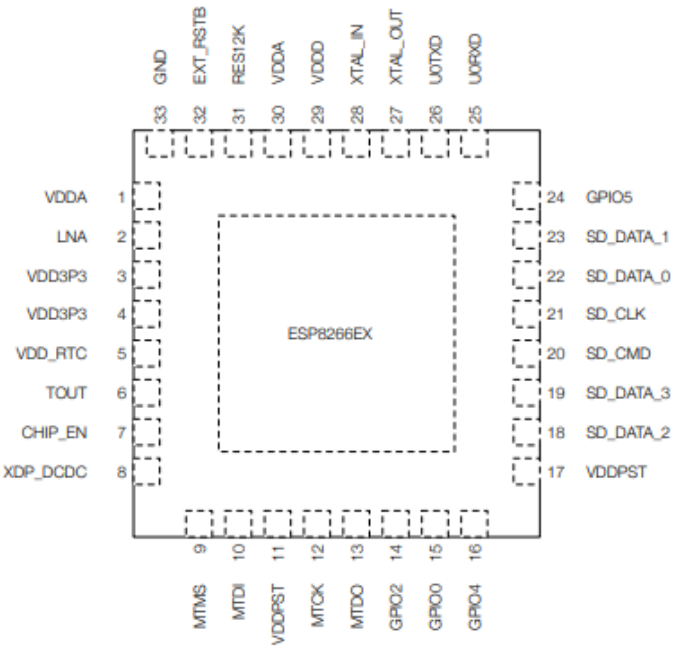


Figure 2-1. Pin Layout (Top View)

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.

Pin	Name	Type	Function
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

Electrical Characteristics

Table 5-1. Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Operating Temperature Range		-	–40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020	-	-	260	°C
Working Voltage Value		-	2.5	3.3	3.6	V
I/O	V_{IL}	-	–0.3	-	0.25 V_{IO}	V
	V_{IH}		0.75 V_{IO}		3.6	
	V_{OL}	-	-	-	0.1 V_{IO}	
	V_{OH}		0.8 V_{IO}		-	
	I_{MAX}	-	-	-	12	mA
Electrostatic Discharge (HBM)		TAMB = 25 °C	-	-	2	KV
Electrostatic Discharge (CDM)		TAMB = 25 °C	-	-	0.5	KV