SISTEMA PARA MÚLTIPLES MEDICIONES DE PULSIOXIMETRÍA ORIENTADO AL MONITOREO DE PACIENTES PARA APOYO EN ZONA DE TRIAGE EN LA SALA DE ESPERA DE URGENCIAS.

Diego Fernando Viveros Zambrano

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Automatización

Popayán, Enero de 2017

SISTEMA PARA MÚLTIPLES MEDICIONES DE PULSIOXIMETRÍA ORIENTADO AL MONITOREO DE PACIENTES PARA APOYO EN ZONA DE TRIAGE EN LA SALA DE ESPERA DE URGENCIAS.

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Diego Fernando Viveros Zambrano

Director: MSc. Delio Eduardo Enríquez Cabrera.
Co-Directores: MSc. Jairo Alfonso Vásquez López.
PhD. Rubiel Vargas Cañas.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Automatización

Popayán, Enero de 2017

Tabla de contenido

Lista de figuras

Lista de tablas

Lista de siglas

LAN: Local Area Network.

BLE: Bluetooth Low Energy

WPAN: Wireless Personal Area Network

PDA: Personal Digital Assistant

MODEM: Modulador Demodulador

HTTP: HyperText Transfer Protocol

TCP: Transmission Control Protocol

UML: Unified Modeling Language.

IDE: Integrated Development Environment

PaD: Pulsioximetría a Distancia

MVC: Modelo-Vista-Controlador

JSP: Java Server Pages

HTML: HyperText Markup Language

WWW: World Wide Web

W3C: World Wide Web Consortium

Glosario

Pulsioximetría: Medida del pulso cardiaco y la saturación de oxígeno a través de un sensor.

Inalámbrico: Que no necesita cables para comunicarse.

Protocolo: Conjunto de reglas que permiten la comunicación entre diferentes sistemas.

Sistema: Conjunto de componentes que se relacionan he interactúan entre si.

Interfaz: Permite la comunicación entre el usuario y el sistema.

Variable: Permite guardar valores de diferentes tipos, valores que están sujetos a cambios.

Diagrama: Dibujo utilizado para representar gráficamente las soluciones al problema o funcionamiento del sistema.

Modulo: Conjunto de piezas que encajan para un mismo fin

Sistema operativo: Conjunto de órdenes y programas que controlan los procesos básicos de una computadora y permiten el funcionamiento de otros programas.

Monitoreo: Proceso de recolectar, analizar y utilizar información de signos vitales para hacer seguimiento a la salud de una persona.

Parámetro: Valores enviados desde la aplicación hacia el servicio web y sus métodos.

Servidor: Equipo que contiene una o más aplicaciones en ejecución, capaz de atender las peticiones de un cliente y devolverle una respuesta

Persistencia: Propiedad que tienen los datos para sobrevivir durante un determinado tiempo.

Saturación: La saturación de oxígeno, es la cantidad de oxígeno en el torrente sanguíneo.

Pulso Cardiaco: Número de latidos cardiacos por minuto.

CAPÍTULO 1

1. Introducción

1.1 Contexto General

Es problema evidente y generalizado en Colombia la obtención de una cita médica porque es trabajo arduo y tedioso por el tiempo empleado, no quedando otra alternativa que la utilización de urgencias. Esto origina un cuello de botella en esta área y vemos que es necesario considerarlo y podemos resolverlo mediante aplicaciones o sistemas realizados desde la ingeniería.

Gracias a los datos estadísticos entregados por el Hospital Universitario San José, ANEXO C, puede apreciarse el incremento de personas que ingresan al servicio de urgencias, entre los años 2011 y 2014, de la siguiente manera: Total de ingresos en 2011 de 8255 personas, para el 2012 de 12676 personas, para el 2013 de 15173 personas, y para el 2014 de 18625 personas. El incremento por año de pacientes en el servicio de urgencias, ocasiona congestión, lo cual satura y ralentiza la clasificación en la zona de Triage y provoca mayores tiempos de espera a las personas para recibir atención médica.
En consideración a lo anterior, es pertinente el diseño e implementación de un sistema, con tecnologías orientadas al sector salud, que permitan lecturas de varios sensores de pulsioximetría y se aplique, de forma real, al apoyo del Triage. Esto asegurará el monitoreo constante durante el tiempo de espera en la sala de urgencias.
El controlar el estado de las personas a través del seguimiento en tiempo real de signos vitales como el pulso cardiaco y la saturación de oxígeno, facilitarán la rápida identificación de los pacientes que requieran atención inmediata, garantizándose así, la atención del paciente crítico y disminuyendo la posibilidad de muertes evitables por atención no oportuna.

1.2 Definición del Problema

En nuestro medio, en ocasiones la atención del paciente en los servicios de urgencias puede ser demorada, por lo cual, muchos pacientes pueden empeorar su condición de salud, mientras esperan ser atendidos. En nuestro país este problema se presenta con alguna frecuencia, en la ciudad de Fusagasugá (Cundinamarca) se presentó un episodio de un paciente de 27 años, que murió antes de ser atendido por el personal médico del servicio de urgencias (Noticiero RCN 24 de septiembre de 2014) [1]. Este tipo de problemas derivados de la atención insuficiente de los pacientes, pueden ser causados probablemente por la gran demanda de atención en los servicios de urgencias, debido al alto volumen de pacientes que consultan por diversas condiciones como el trauma, enfermedades respiratorias, entre otras. Otras de las posibles causas de este problema es la escases de personal entrenado para la atención de las urgencias o en algunos casos la poca infraestructura.

En los centros de urgencias se realiza una clasificación inicial del paciente (triage) para luego ser atendido por el personal médico. Triage, es una palabra francesa que significa clasificación [2]. En la zona de triage de un servicio de urgencias de un centro asistencial, se maneja una clasificación de acuerdo a la gravedad del paciente, como se muestra en la figura 1, el cual se divide en cuatro etapas. Las convenciones son regidas por el código internacional de colores para atención en urgencias [3].

Cuadro I. Código internacional de colores.

Color	Prioridad	Definición
Rojo	Uno	Pacientes críticos, potencialmente recuperables, que requieren atención médica inmediata
Amarillo	Dos	Pacientes graves que requieren atención médica mediata
Verde	Tres	Pacientes con lesiones leves, que puede postergarse su atención médica sin poner en riesgo su integridad física
Negro	Cero	Pacientes con lesiones mortales por necesidad o fallecidos en el lugar

Figura 1. Código internacional de colores [3]

En esta clasificación, el paciente de acuerdo a su condición clínica, es asignado a una categoría, esta asignación es producto de la valoración rápida del paciente, mediante examen de pulso, saturación de oxígeno en la sangre, presión arterial, entre otras. Pero todas estas variables clínicas por lo general son determinadas en una sola ocasión, y si el paciente no está en prioridad uno, es puesto en espera, tiempo en el cual puede empeorar su condición o incluso morir.

La monitorización se realiza de dos maneras: Invasiva y No Invasiva. La monitorización invasiva implica una punción o solución de continuidad, mientras, la no invasiva usa sensores que se colocan en la piel o a través de orificios naturales.

La monitorización por pulsioximetría es una tecnología que permite la medición de la saturación de oxígeno de una manera no invasiva, por medio de la espectrofotometría, lo que ha acelerado su aceptación como el “quinto signo vital” (además de la temperatura, la presión arterial, el pulso y la frecuencia respiratoria) en las evaluaciones clínicas [4].

Si se aplica este tipo de monitorización en las salas de espera de urgencias como apoyo al Triage tradicional, se puede prevenir los empeoramientos de las personas durante el tiempo de espera para la atención, generando alarmas que permitan que la persona que empeora su condición de salud sea atendida con prioridad.

Como se observa en el estado del arte no se ha encontrado un sistema que permita el monitoreo simultaneo de varios pacientes a través pulsioximetria mediante un único sistema de monitoreo, que sea capaz de generar alarmas visuales y sonoras. Entre los

sistemas existentes solo se ha encontrado uno que se propone aplicarlo a la sala de urgencias en la zona de Triage, pero solo cuenta con una única silla con la cual solo se puede atender a un paciente a la vez, siguiendo el método tradicional del Triage hospitalario.

Teniendo en cuenta lo mencionado anteriormente se plantea la siguiente pregunta de investigación: ¿Es posible implementar un sistema que permita monitorear simultáneamente varios pacientes a través de sensores de pulsioximetría en la sala de espera de urgencias como apoyo a la clasificación Triage?

La hipótesis que se platea es que es posible desarrollar un sistema que permita el monitoreo en tiempo real de varios pacientes mediante pulsioximetría con la generación de alarmas visuales y sonoras, y su aplicación en la sala de espera del área de urgencias.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar un sistema que permita la lectura de múltiples sensores de pulsioximetría con el fin de generar alarmas visuales y sonoras para la zona de Triage de urgencias.

1.3.2 Objetivos Específicos

- Implementar un sistema que permita adquirir y procesar la información obtenida de las mediciones de varios sensores de pulsioximetría
- Implementar un servicio web que permita sincronizar en tiempo real la información del sistema de sensores con las aplicación móvil y dispositivo de despliegue
- Realizar pruebas de simulación del sistema desarrollado y en un ambiente real con la supervisión de un médico.

1.4 Contribuciones

- Un sistema que permite la adquisición de múltiples señales de sensores de pulsioximetría y el procesamiento de la información obtenida, dando como resultado datos de saturación de oxígeno en la sangre y pulso cardiaco, y su relación con la clasificación Triage.
- También permite la aplicación de las TIC´s para el despliegue de la información medida por los sensores con el fin de que, tanto personal médico, como usuarios, puedan observar la información de los pacientes en tiempo real.
- La proyección de integración social de este proyecto permite que las aplicaciones de la tecnología y la oportunidad de servir a todas las personas sin importar su condición en cuanto al servicio de salud que disponga, se haga realidad. La monitorización constante y en tiempo real, de las funciones vitales de los pacientes, permiten un trato más digno y acorde con los principios de conservación de la vida y la salud.

1.5 Estado Actual del Conocimiento

“Development of mHealth applications for pre-eclampsia triage” [5]

En este artículo se presenta el desarrollo de aplicaciones móviles mHealth para realizar predicción de riesgo en mujeres con preeclampsia, a través de la toma de datos mediante un pusioximetro conectado a un smarthphone, con lo cual se obtiene la saturación de oxígeno, para uso en los centros de salud y las visitas a domicilio.

“Automatic pre-hospital vital signs waveform and trend data capture fills quality management, TRIAGE and outcome prediction gaps” [6]

Este articulo presenta un sistema mediante el cual se recogen datos de signos vitales mediante formas de onda de saturación de oxígeno y señales ECG, para la predicción de la gravedad de los pacientes y mejorar la atención pre-hospitalaria, El sistema está sobre un helicóptero de trasporte de pacientes a hospital, estos datos son entregados al hospital para mejorar la calidad de atención al paciente que ingresa.

“An assistive device for congenital central hypoventilation syndrome outpatients during sleep” [7]

En este trabajo de describe un dispositivo que genera una alarma que se activa a través de los datos recogidos en tiempo real por un pusioximetro, con lo cual se controla un ventilador mecánico. Este dispositivo es usado en personas con síndrome de hipo ventilación, que generalmente se produce durante el sueño y que puede causar daños cerebrales o hasta la muerte.

Telemonitorización en tiempo real del Síndrome de Apneas-Hipopneas del Sueño (SAHS) mediante pulsioximetría domiciliaria [8].

En éste documento se resalta el aprovechamiento de las nuevas tecnologías como los dispositivos móviles por su conectividad, comunicación inalámbrica (ideal para telemedicina) y procesamiento gracias a sus sistemas operativos y como esta tecnología puede ser aplicada al área de la salud. Para este proyecto las nuevas tecnologías son aplicadas a las enfermedades respiratorias del sueño, más específicamente al síndrome de apneas-hiponeas del sueño, por medio de la pulsioximetría para la monitorización.

“Clinical use of new-generation PULSE oximeters in the neonatal intensive care unit” [9]

Este trabajo presenta un estudio acerca del uso de la pulsioximetria en la unidad de cuidados intensivos neonatales, teniendo en cuenta los pulsioximetros de la marca Masimo SET y Philips FAST, los cuales concluyen que son más resistentes al

movimiento los Masimo , por otra parte son de gran ayudan a la monitorización continua de los prematuros y recién nacidos críticamente enfermos.

Sistema Automático Digital de Triage Medico para los Niveles de Prioridad I, II y III del Servicio de Urgencias en IPS´s [10].

Este documento describe la problemática existente en las Entidades promotoras de salud o EPS´s Colombianas, en donde los usuarios se quejan frecuentemente por la atención y tiempo de generación de citas médicas, por lo cual los usuario ante la larga espera prefieren ingresar al sistema por medio del servicio de urgencias, lo cual genera que este servicio se vea afectado por la gran demanda de servicios. El área de urgencias cuenta con un sistema llamado Triage, el cual ayuda a la clasificación de los pacientes con respecto a la gravedad o grado de empeoramiento del paciente. Se busca proponer, desde un enfoque cualitativo, un sistema de triage hospitalario que facilite diagnosticar y asignar el recurso hospitalario, evitando el ingreso de datos que generalmente se toman a un paciente, lo cual se haría de forma automática antes de ser atendido por un médico y los cuales serían almacenados en una base de datos. Este método reduciría el tiempo de atención debido a que el medico podría acceder a los datos tomados del paciente. En el documento también se describen los tipos de triage que se realizan en el sistema de salud Colombiano generalmente, así como los signos que se deben tomar a los pacientes para determinar su estado de salud, tales como: pulso, temperatura, respiración, reflejo pupilar. Por otro lado se habla acerca de las normativas que deben cumplir este tipo de sistemas, las percepciones de las personas con respecto al uso o al malestar que puede causar, la usabilidad. El sistema se diseña sobre una silla la cual tiene un estudio antropométrico de acuerdo a las personas que lo usaran teniendo en cuenta su proporción y altura, así como sus posiciones al sentarse. Este trabajo muestra que la propuesta de un sistema para la ayuda en el triage es viable, indicando mediante estudios que la atención mejoraría reduciendo el cuello de botella en la atención a los pacientes.

“TELEMOLD project: OXIMETRY and exercise telemonitoring to improve long-term oxygen therapy” [11]

En este artículo se describe el proyecto llamado TELEMOLD, en el cual se busca mejorar oxigenoterapia continua domiciliaria (OCD), dirigida a pacientes con insuficiencia respiratoria crónica, a quienes se lo monitoriza mediantes un sensor de pusioximetria y un acelerómetro; estos datos de los sensores se envían a un servidor a través de tecnología bluetooth y servicio 3G, con acceso a través de un sitio de internet, con el fin de tener una prescripción adecuada de oxigeno durante las actividades diarias.

Sistema de Monitorización Inalámbrica de Sensores de SPO2 (Pulsioxímetros)[12].

Este trabajo muestra la elaboración de un sistema de monitorización remoto de pacientes mediante unas redes de sensores Ad- Hoc Bluetooth, que se interconectan entre sí mediante WiFi. Este sistema permite tener conocimiento por parte del personal médico del estado de los pacientes independiente del lugar en que se encuentren, en cierta forma como lo menciona trasladando el servicio de UCI a la calle. Se utiliza las redes MANET´s

o (Mobile ad-hoc Networks) ya que no requieren ser centralizada ni requieren infraestructura. El paciente debe llevar consigo la red de sensores, una PDA y un dispositivo GSM formando asi una red BAN, la cual se conectará mediante wifi con un Nodo Concentrador Inteligente, el cual detectara los pulsioximetros que estén en su área y procesará su información para ser enviada al sistema de control central, esta monitorización podrá hacerse en un entorno controlado y en un entorno abierto. En este trabajo para el prototipo se eligió tecnología bluetooth por su tasa de transmisión de bits, su cobertura y bajo consumo de potencia y como una de sus conclusiones está el investigar acerca de uso de dispositivos que reduzcan más aun el uso de la potencia.

"PULSE oximeter based mobile biotelemetry application" [13]

En este artículo se presenta una aplicación móvil de biotelemetría que registra los datos enviados por un pulsioximetro bluetooth hacia el teléfono inteligente, los cuales son transmitidos luego hacia un servidor web remoto. En este servidor se comparan los datos recibidos con umbrales de decisión para la saturación de oxígeno y el pulso cardiaco, y en caso de que no se cumplan los umbrales el sistema envía un SMS a un medico el cual puede enviar una ambulancia por el paciente. El sistema también permite el acceso a los datos para que un médico pueda seguir el desarrollo o el progreso de los pacientes, de esta manera se puede tener monitoreo desde cualquier lugar y en cualquier momento.

"A portable, inexpensive, wireless vital signs monitoring system" [14]

En este trabajo se menciona un dispositivo para la recopilación de datos fisiológicos fuera de un centro médico. Los datos son recolectados por biosensores que obtienen información de pusioximetria, electrocardiograma (ECG) presión arterial no invasiva (NIBP) y peso, los cuales son enviados a través de la tecnología bluetooth a un dispositivo de control, mediante el cual pueden ser almacenados o enviados a un servidor. También cuenta con interfaces con grandes botones en la aplicación para las personas con dificultades visuales o motoras. Este dispositivo permite ser usado en días, actividades o momentos específicos y permite reducir las incomodidades y el costo que generaría realizar este tipo de exámenes en forma tradicional.

1.6 Brechas

En la tabla 1 se presentan las brechas encontradas en los trabajos revisados en el Estado del Arte. Para analizar estas brechas, se tuvieron en cuenta los siguientes criterios:

- Si el sistema presentado realiza un monitoreo constante con varios sensores de pulsioximetria.
- Si el sistema realiza transmisión de datos en tiempo real.
- Si el sistema es inalámbrico.
- Si el sistema cuenta con una aplicación móvil que presente alarmas visuales y sonoras en tiempo real de los datos arrojados por los pulsioximetros.
- Si el sistema es aplicado a la zona de Triage en urgencias.

Proyecto – Articulo	**Brechas existentes**

Development of mHealth applications for pre-eclampsia triage	Presenta monitoreo constante pero no puede leer varios sensores a la vez. No esta aplicado a la zona de Triage
Automatic pre-hospital vital signs waveform and trend data capture fills quality management, triage and outcome prediction gaps	El sistema está diseñado para pacientes que necesitan atención pre-hospitalaria pero no para pacientes que se encuentran ya en la sala de Triage. No cuenta con un sistema de monitoreo en tiempo real para varios pacientes por medio de pulxioximetros, no cuenta con alarmas visuales y sonoras para el monitoreo en tiempo real de varios pacientes en la zona de triage.
An assistive device for congenital central hypoventilation syndrome outpatients during sleep	Este sistema cuenta con alarma pero no es visual ni Sonora ni usa aplicaciones móviles, solo se usa para el control mecánico del ventilador. No se aplica a la zona de Triage
Telemonitorización en tiempo real del Síndrome de Apneas-Hipopneas del Sueño (SAHS) mediante pulsioximetría domiciliaria	Este sistema no cuenta con la posibilidad de leer varios sensores de pulsioximetría al mismo tiempo. No esta aplicado a la zona de Triage.
Clinical use of new-generation pulse oximeters in the neonatal intensive care unit.	No utiliza tecnología inalámbrica. No cuenta con aplicaciones que generen alarmas visuales y sonoras. No presenta monitoreo de múltiples sensores de pulsioximetría al mismo tiempo.
Sistema Automático Digital de Triage Medico para los Niveles de Prioridad I, II y III del Servicio de Urgencias en IPS´s	No puede monitorear varios pacientes al mismo tiempo. No genera alarmas visuales y sonoras.
TELEMOLD project: oximetry and exercise telemonitoring to improve long-term oxygen therapy	No presenta un sistema capaz de leer varios sensores de pulsioximetria. No cuenta con un sistema de alarmas visuales y sonoras. No esta aplicado a la zona de Triage.

Sistema de Monitorización Inalámbrica de Sensores de SPO2	El sistema exige que la persona lleve una red de sensores BAN. No genera alarmas visuales y sonoras. No esta aplicada a la zona de Triage.
Pulse oximeter based mobile biotelemetry application	El sistema genera un mensaje de texto en caso de que los rangos de saturación y pulso cambien, pero no cuenta con un sistema de alarmas visuales y sonoras. No cuenta con un sistema único que lea la señal de varios pulsioxímetros al mismo tiempo. El sistema no ha sido aplicado al Triage.
A portable, inexpensive, wireless vital signs monitoring system	No cuenta con un sistema de alarmas visuales y sonoras. No tiene un sistema que ´permita leer varios sensores de pulsioximetría al mismo tiempo. No esta aplicado a la zona de Triage de urgencias.

Tabla 1. Brechas en trabajos relacionados con el proyecto

1.7 Estructura de la monografía

El presente trabajo de grado se estructura de la siguiente manera: en el Capítulo 2 se estudian los conceptos necesarios para el desarrollo del trabajo, así como se genera el diseño que se tendrá en cuenta para la implementación del sistema; el Capítulo 3 describe paso a paso la ejecución del proyecto ilustrándolo por medio de muchas figuras; el Capítulo 4 presenta las pruebas que se le realizaron al sistema para comprobar su funcionamiento, las conclusiones que se obtuvieron de las pruebas y los trabajos futuros que se presentan a partir de los; y finalmente, en los anexos se presenta la descripción de los códigos fuente y la documentación utilizada para el desarrollo de este proyecto. Durante el documento se intenta ser lo más específico posible y al mismo tiempo no extenderse demasiado en las descripciones.

CAPÍTULO 2

2. Estudio de prefactibilidad y Formulación del proyecto

2.1 Requerimientos de alto nivel

Luego de tener claro la definición del problema y los objetivos, se presentan a continuación los requerimientos necesarios para la implementación del sistema de monitoreo por pulsioximetría, que permitieron alcanzar desarrollo final del trabajo de grado.

Requerimiento 1: Sensor inalámbrico para monitorizar a las personas en la sala de espera de urgencias.

Requerimiento 2: Sensor cómodo y fácil de usar.

Requerimiento 3: Verificar la conexión del sistema a la red de area local del hospital (LAN).

Requerimiento 4: Asegurar la comunicación entre los sensores inalámbricos, la placa electrónica de procesamiento Raspberry y el servicio web.

Requerimiento 5: Sincronización de la información del sensor con su usuario correspondiente.

Requerimiento 6: Garantizar la gestión de alarmas audiovisuales.

Requerimiento 7: Contar con los dispositivos de despliegue que permitan la presentación de la alarmas.

2.2 Diseño

2.2.1 Metodología de desarrollo

Durante la construcción e implementación del sistema planteado, se optó por utilizar la metodología modelo de construcción de soluciones [15]. Esta metodología hace uso de cuatro fases para el desarrollo de la solución planteada, por medio del cumplimiento de los objetivos específicos

Estudio de prefactiblidad: Se centra en el análisis de los requerimientos de alto nivel, y se ponen en consideración las tecnologías a utilizar así como la arquitectura del sistema y se prioriza un listado de riegos para el proyecto.
Formulación del proyecto: En esta fase, se debe puntualizar más en los requerimientos del proyecto, mediante el uso de técnicas como plantillas extendidas de casos de uso, diagramas de secuencia, entre otras.
Ejecución del proyecto: En esta etapa se requiere el prototipo funcional de un sistema que permita adquirir los datos de varios sensores al mismo tiempo, generar alarmas

visuales y sonoras en tiempo real y desplegarlas en dispositivos móviles, además de obtenerlos modelos de descripción del sistema y de implementación de sistema.
***Validación de la solución**:* Finalmente, en esta fase se muestra la arquitectura desarrollada del sistema construido, además de esto se realizan las pruebas de funcionamiento de la solución tanto en un ambiente simulado como en un ambiente real ya que se cuenta con el aval del Hospital Universitario San José de Popayán. Se evaluará el sistema desarrollado con los profesionales de la salud, siguiendo la metodología de evaluación que se menciona a continuación.

La evaluación real del sistema se realizará con pacientes reales en la sala de urgencias del Hospital Universitario San José de Popayán. La participación de los pacientes en las pruebas del sistema se hará por medio de un consentimiento informado que se le entregará al paciente, donde cuenta con la información del sistema y el cual deberá ser firmado por el paciente.

En la actualidad las redes inalámbricas permiten la interconexión de diferentes dispositivos, permitiendo la movilidad de estos. Permiten la recolección de información de distintas fuentes como signos vitales de una persona, variables climáticas, e incluso la automatización de objetos en lo que hoy en día se llama internet de las cosas (IoT, *Internet of Things*)[1]. Por otro lado, las redes inalámbricas permiten la conexión de distintas tecnologías logrando una convergencia entre todas, además ayuda a que el sistema tenga características de escalabilidad[2]. Entre las ventajas de una red inalámbrica se encuentran:

- Estar basada en estándares y contar con certificación Wi-Fi.
- Instalación simple.
- Robusta y confiable.
- Escalabilidad.
- Facilidad de uso.
- Servidor Web para una administración más fácil.
- Seguridad.
- Una aplicación que detecte localidades.
- Costo de propiedad reducido.
- Fácil configuración para el usuario [16].

Haciendo uso de esta red y sus ventajas, se diseña el sistema de múltiples mediciones de pulsioximetría, teniendo en cuenta que los sensores utilizan tecnología Bluetooth para la comunicación de sus datos y la placa electrónica Raspberry utiliza tecnología wifi para llevar los datos hasta el servidor web.

Cabe resaltar que el sistema se caracteriza por la lectura múltiple de varios sensores inalámbricos de pulsioximetría, con un máximo de 8 sensores con transmisión Bluetooth.

[1] Es un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet.

[2] Término usado en tecnología para referirse a la propiedad de aumentar la capacidad de trabajo o de tamaño de un sistema sin comprometer el funcionamiento y calidad normales del mismo [17].

El sistema se realiza con la finalidad de mantener monitorizados a los pacientes que aguardan por atención en la sala de espera de urgencias antes de ingresar a la sala de Triage para su clasificación por el personal médico, esto con el fin de brindar un apoyo al sistema de Triage que maneja el hospital, además le da al paciente un grado de tranquilidad al observar que desde el momento que ingresa a urgencias se encuentra monitorizado y en caso de una complicación, se le atenderá inmediatamente.

Para realizar este sistema es necesario el uso de los sensores inalámbricos, porque permiten llevarlo hasta el paciente en cualquier lugar de la sala de espera evitando incomodidades. Existe muchos tipos de dispositivos en el mercado que realizan monitorización de signos vitales del cuerpo humano, pero no todos cuentan con transmisión Bluetooth, ni todos son fáciles de manipular por su tamaño y complejidad. Otro punto que se debe resaltar es que la gran mayoría de sensores biomédicos actualmente leen un solo signo vital lo que implicaría el uso de un sensor por signo lo que incrementaría las transmisiones Bluetooth reduciendo la capacidad de lectura de la placa electrónica Raspberry.

De acuerdo a lo mencionado anteriormente, se opta por monitorizar a los pacientes con un sensor de pulsioximetria, debido a que puede medir dos signos vitales: Saturación de oxígeno en la sangre y pulso cardiaco, mediante un solo dispositivo y contiene transmisión Bluetooth, además es un dispositivo muy pequeño y fácil de usar.

Cada sensor puede transmitir sus datos por medio de la tecnología Bluetooth, pero se necesita recibir todos estos datos por un solo dispositivo y a su vez el dispositivo receptor tenga la capacidad de procesar los datos recibidos, decidir un tipo de alarma y llevarla hasta el servicio web. También, se debe seguir con requerimientos elegidos en el sensor como lo son tamaño para su fácil ubicación y manejo y su baja complejidad en cuanto al uso. Para lograr cumplir con estas especificaciones se realiza una búsqueda de dispositivos que se acoplen a estas condiciones entre los cuales se destaca una placa electrónica perteneciente a la empresa Raspberry[3]. Esta placa tiene una alta capacidad de procesamiento, además cuenta con un bluetooth de versión 4.0 que permite leer ocho conexiones Bluetooth, además de utilizar baja energía o LE(Low Energy)[18], con lo cual permite la lectura de todos los sensores de pulsioxietría. Esta placa también cuenta con un módulo WiFi con el cual se logra conectar a la red de área local del hospital para el envío de datos al servidor web.

Se ha mencionado un servicio web donde llegan los datos procesados por placa electrónica Raspberry, este servicio web es necesario debido que allí se debe realizar la sincronización entre los datos de cada sensor y el usuario que lo está usando. Se decide utilizar el servicio web debido a que permite la implementación de servicios dentro de éste, que pueden ser accedidos desde cualquier parte de la red local del hospital y desde cualquier dispositivo que tenga una conexión a esta red. Por otro lado, este servicio web recoge la información del usuario, que para el caso de este sistema se usa el nombre de la persona como identificador y el sensor numerado del “1” al “8”.

[3] Considerada como un computador pequeño personal.

La recolección de los datos de las personas son recogidos por una aplicación móvil Android[4] por su facilidad de manejo y uso común en la actualidad. La aplicación debe ser amigable al usuario y permitir mediante el teclado del dispositivo ingresar el nombre que identifica a la persona y el número del sensor (ejemplo: nombre: Diego, sensor: 1). Al momento de ingresar los datos en la aplicación se dará inicio a la sincronización con datos que se recolectaron en la placa Raspberry y así su presentación en los dispositivos de despliegue. Finalmente por medio de la misma aplicación en Android se logra detener la lectura de los signos una vez el paciente sea atendido en la zona de Triage, para iniciar la lectura con un paciente diferente.

2.3 Tecnologías a utilizar

Con respecto al diseño planteado para la realización del sistema, a continuación se realiza una breve descripción las tecnologías necesarias para el desarrollo del sistema:

Bluetooth:

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.[19] Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo.

Los dispositivos que incorporan este protocolo pueden comunicarse entre sí cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión es suficiente. Estos dispositivos se

[4] Siendo Android el sistema más usado en la actualidad

clasifican como “Clase 1”, “Clase 2” o “Clase 3” en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una caja de ordenador. [19]

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~5-10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 2. Clase, Potencia y Alcance aproximado Bluetooth [19]

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte la mayor sensibilidad del dispositivo de clase 1 permite recibir la señal del otro pese a ser más débil.

Los dispositivos con Bluetooth también pueden clasificarse según su capacidad de canal [18]:

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	32 Mbit/s

Tabla 3. Versión / Ancho de banda. [19]

Pulsioxímetro Bluetooth:

Dispositivo con el cual se realiza la adquisición de datos que representan los signos del paciente a través de pulsioximetría, en cuanto a saturación de oxígeno y pulso cardiaco. Se escoge un sensor que cuente con la propiedad del Bluetooth con el fin de evitar el uso de cable en la sala de espera de urgencias y facilitar la ubicación del paciente que use el sistema. La ***Pulsioximetría*** es un método no invasivo, que permite determinar el porcentaje de saturación de oxígeno de la hemoglobina en sangre de un paciente con ayuda de métodos fotoeléctricos. Para realizar esta técnica, se coloca el pulsioxímetro, en una parte del cuerpo que sea relativamente translúcida y tenga un buen flujo sanguíneo, por ejemplo los dedos de la mano o del pie o el lóbulo de la oreja. El pulsioxímetro emite luces con longitudes de onda, roja e infrarroja que pasan secuencialmente desde un emisor hasta un fotodetector a través del paciente. Se mide la absorbancia de cada longitud de onda causada por la sangre arterial (componente pulsátil), excluyendo sangre venosa, piel, huesos, músculo, grasa. Con estos datos será posible calcular la saturación de oxígeno en sangre. [20]

Figura 2. Pulsioxímetro Bluetooth

[https://www.aliexpress.com/bluetooth-pulse-oximeter_reviews.html]

Raspberry Pi 3:

Siendo el modelo más actual de Raspberry es usado por su capacidad de procesamiento y por sus puertos, además de los nuevos módulos introducidos para esta versión como lo son un módulo Bluetooth y un módulo wifi, como se indica a continuación:

A 1.2GHz 64-bit quad-core ARMv8 CPU

802.11n Wireless LAN

Bluetooth 4.1

Bluetooth Low Energy (BLE)

Figura 3. Raspberry pi 3

[https://www.raspberrypi.org/weekly/connected/]

Bluetooth v4.0 introdujo Bluetooth de Bajo Consumo (también llamado Bluetooth de Baja Energia) en 2014, oficialmente conocido como Bluetooth Smart. Esta especificación introduce una radio completamente diferente utiliza menos energía y es más barata, para satisfacer las necesidades de estas nuevas aplicaciones. El amplio soporte de BLE en Smartphones y tabletas que hace que las aplicaciones BLE sean fáciles de implementar y fáciles de usar para los consumidores. [21]

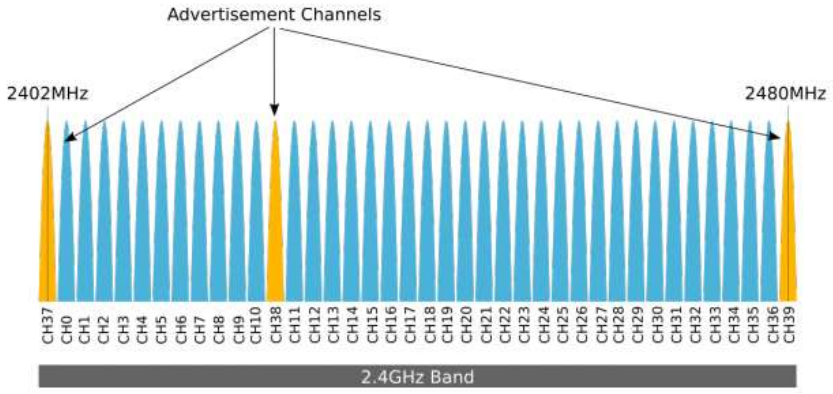


Figura 4. Capa física BLE

[https://www.argenox.com/es/bluetooth-baja-energia-ble-desarrollo/library/introduction-bluetooth-bajo-consumo/]

WiFi

En Wi-Fi un punto de acceso inalámbrico (access point) transmite y recibe datos a través de ondas de radio y los equipos remotos, que cuentan con un transceptor (transmisor-receptor) en una tarjeta de acceso, se comunican con él como se muestra en la figura 2. El punto de acceso inalámbrico (access point) se conecta a un MODEM que se comunica

de manera cableada con el núcleo de la red. Por cuestiones de seguridad, mediante un esquema llamado WEP (Wired Equivalent Privacy) los datos reciben un tratamiento criptográfico con códigos de 128 bits y solo los usuarios con contraseña pueden acceder a la red. Hoy en día, se utiliza un esquema más robusto llamado WPA: Wi-Fi Protected Access [22]

Figura 5. Logo WIFI

[https://es.wikipedia.org/wiki/Wifi#/media/File:Wi-Fi_Logo.svg]

Java

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

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

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, ylibrerías de clases en 1991 y las publicó

por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como elCompilador de Java de GNU y el GNU Classpath. [23]

Protocolo de Transferencia de Hipertexto (http):

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL [24].

Figura 6. Protocolo http

(http://roble.pntic.mec.es/jprp0006/tecnologia/bachillerato_tic/unidad01_navegadores/navegadores3.htm)

Tableta Electrónica

Es un tipo de computadora portátil, en la cual es posible la instalación de aplicaciones, para el caso de este trabajo de grado se usa la tableta para el ingreso de datos del paciente que accede a usar el sistema, así como el despliegue de la información de los signos del paciente.

La tableta funciona como una computadora, solo que más ligera en peso y más orientada al multimedia, lectura de contenidos y a la navegación web que a usos profesionales. Para que pueda leerse una memoria o disco duro externo USB, debe contar con USB On-The-Go, también denominado USB Host.

Dependiendo del sistema operativo que implementen y su configuración, al conectarse por USB a un ordenador, se pueden presentar como dispositivos de almacenamiento, mostrando solo la posible tarjeta de memoria conectada, la memoria flash interna, e incluso la flash ROM. Por ejemplo en Android el usuario debe de activar el modo de dispositivo de almacenamiento, apareciendo mientras como una ranura sin tarjeta.

Algunas tabletas presentan conectores minijack de 3.5, VGA o HDMI para poder conectarse a un televisor o a un monitor de computadora [25].

Figura 7. Tableta electrónica Samsung

[http://www.heraldo.es/noticias/sociedad/samsung_podra_exponer_tablet_berlin_por_pleito_con_apple.html]

2.4 Arquitectura inicial del sistema

La arquitectura del sistema presentado en la figura 8, está compuesta por los sensores de pusioximetría inalámbricos que se conectan mediante tecnología Bluetooth con una placa electrónica Raspberry, donde se recogen los datos obtenidos por los sensores. Esta placa se conecta al servidor mediante tecnología Wifi por medio del área local del hospital. En el servidor se encuentra el servicio web donde se almacenan temporalmente los datos de cada sensor. Por otro lado un dispositivo con sistema operativo Android (tableta electrónica), recoge los datos personales del usuario y los asocia a un sensor, para luego conectarse mediante la tecnología Wifi con el servidor web y enviarles estos datos mencionados. Por último el servidor se encarga de acoplar los datos personales con los signos obtenidos por los sensores y generar alarmas las cuales se desplegaran en dispositivos como tablets o televisores. En la figura 8, lo que se encuentra encerrado en rectángulos azules, hace alusión a las tecnologías usadas para la comunicación de información entre los dispositivos.

Figura 8. Arquitectura de despliegue del sistema.

2.5 Riesgos del proyecto

En el presente trabajo de grado se tienen en cuenta dos tipos de riegos, los que generan complicaciones a la salud de los usuarios o bioéticos y los riesgos por manejo de la información de los usuarios. En cuanto a los riesgos que generen alguna complicación a la salud de los pacientes, se debe tener en cuenta que el proyecto no presenta efectos adversos al ser humano, ya que la medición hecha por el sensor maneja un proceso no invasivo, debido a que se trabaja con un diodo emisor de luz infrarroja y otro emisor de luz roja a muy baja potencia ya que los diodos trabajan a voltajes entre 2 y 5 voltios. Por otro lado se tienen en cuenta los 4 principios bioéticos: principio de autonomía, de justicia, de beneficencia y de no maleficencia.

El Principio de Autonomía, el cual es la aceptación del otro como agente moral responsable y libre para tomar decisiones, no se violara con la implementación de este proyecto, debido a que cada persona será informada acerca del sistema, a para que sea

la misma persona quien decida si se aplica en ella las funcionalidades del sistema, ninguna persona será obligada a hacer uso de este.

El Principio de Justicia, que debe ejercer la sociedad a través de sus instituciones de salud. Frente a este principio, todo paciente que de consentimiento, podrá hacer uso del sistema y acceder a sus beneficios sin importar su institución de salud, raza, procedencia, estado social o económico.

Principio de Beneficencia, el cual se enfoca en hacer bien a las personas en todo sentido, es un principio fundamental en el diseño de este proyecto, debido a que el diseño del sistema está enfocado a ayudar en la atención de las personas, prevenir empeoramientos de condición de salud.

En cuanto al principio de No Maleficencia, que obliga a no dañar ni hacer mal a las personas, es tenido en cuenta desde el comienzo del diseño de este proyecto, realizando una correcta investigación de las tecnologías que se usarán, tanto en las que tendrán contacto directo con las personas como en las que se usarán para las comunicaciones, siempre pensando en que ninguna afecte la salud de los pacientes ni interfiera con los procesos actuales que se realizan en la sala de espera de los urgencias, siempre buscando mejorar la calidad de la atención no el empeoramiento o deterioro.

Para el control de la información del paciente, se toma como referencia los principios éticos ya mencionados, se realizará un consentimiento informado del paciente para las pruebas del sistema, el cual llevará datos del paciente en cuanto nombre y edad y antecedentes personales de complicaciones de salud que podrían alterar los datos captados por los sensores. A esta información solo tendrá acceso el estudiante encargado del diseño del proyecto y el doctor Jairo Alfonso Vásquez López, quien es el asesor de proyecto, pero solo con fines de diseño y pruebas de funcionamiento. Esta información no podrá ser difundida acogiéndose al derecho que tiene cada paciente a la privacidad que mediante la Ley 1581 de 2012 se expidió el Régimen General de Protección de Datos Personales, el cual, de conformidad con su artículo 1°, tiene por objeto “(...) desarrollar el derecho constitucional que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos, y los demás derechos, libertades y garantías constitucionales a que se refiere el artículo 15 de la Constitución Política; así como el derecho a la información consagrado en el artículo 20 de la misma” [26].

2.6 Casos de Uso

Mediante el uso de UML(Unified Modeling Language o en español Lenguaje unificado de modelado) el cual se define como el resultado de un esfuerzo dirigido a obtener una notación gráfica unificada para representar los modelos de sistemas desarrollados con el paradigma de orientación a objetos[27], se representan las actividades que realiza el sistema y los actores que las pueden desarrollar. En la figura 9, el administrador, el portero y el medico representan los actores, quienes tienen acceso al sistema y realizan

actividades sobre este, por otro lado se encuentran las actividades o casos de uso que se realiza en el sistema, para este caso se tienen 5 principales: Iniciar servicio web, Inicializar variables, Pruebas de sistema, Visualizar signos y Agregar / Detener Lectura.

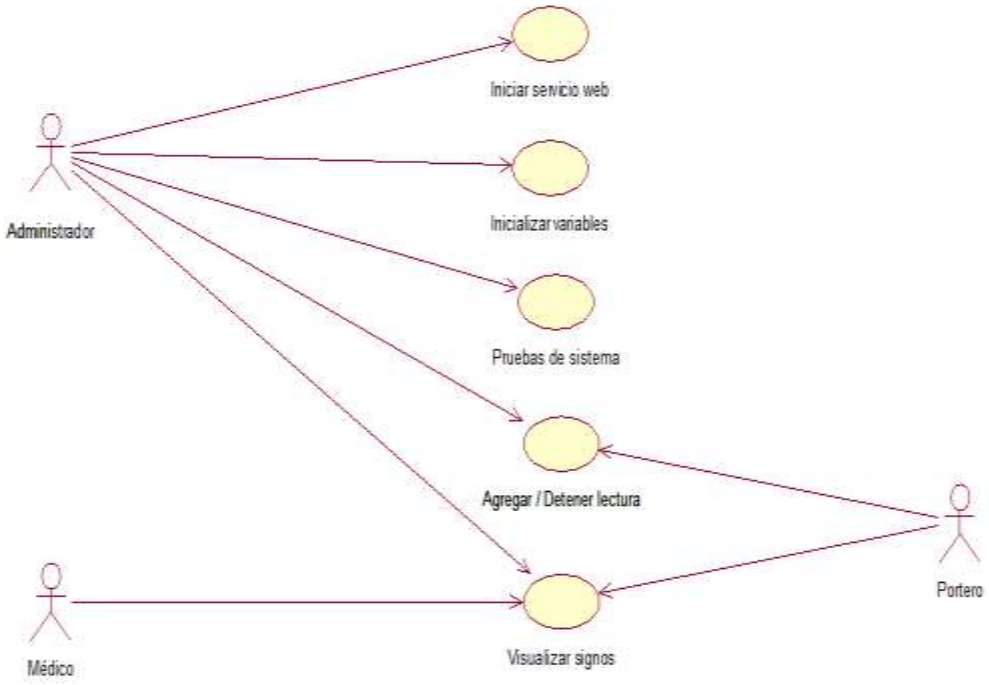


Figura 9. Casos de Uso

2.6.1 Casos de uso extendidos

Describen las interacciones entre actores y el sistema de manera detallada, enumerando paso a paso los eventos que se presentan durante una ocurrencia típica del caso de uso.

Información general

Caso de Uso:	**Iniciar servicio web**
Actores:	Administrador
Propósito:	Colocar en funcionamiento el servicio web
Resumen:	El administrador habilita el servicio web desde el servidor del hospital, el cual empieza a recibir las peticiones realizadas por el dispositivo Android, por la placa Raspberry y el dispositivo de despliegue.
Tipo:	Primario.

Precondiciones

El servicio web debe estar instalado en el servidor conectado al área local del hospital. El servicio web debe contar con una interfaz web de administración a la cual se accede mediante un usuario y contraseña, los cuales debe poseer el administrador.

Flujo principal
-Este caso de uso inicia cuando el administrador ejecuta el servicio web dentro del servidor. -El sistema presenta al administrador una interfaz de administración el cual le permite realizar sus funciones como inicializar las variables, pruebas del sistema, Agregar / Detener lectura, ver los signos del paciente.

Información general

Caso de Uso:	**Inicializar variables.**
Actores:	Administrador.
Propósito:	Asignarle un valor inicial a las variables del sistema.
Resumen:	Cuando el administrador elige la opción inicializar variables en la página de administración, el sistema automáticamente le asigna valores por defecto a las variables del sistema como nombre de usuario, sensor, valor de saturación y pulso cardiaco, nivel y estado.
Tipo:	Primario.

Precondiciones
El administrador debe haber ingresado mediante su usuario y contraseña. El servidor debe estar conectado al área local del hospital.

Flujo principal
-Este caso de uso inicia cuando el administrador elige la opción de inicializar variables en la página de administración del servicio web. -El sistema asigna los valores por defecto a las variables que se usan en el sistema. -Se muestra un mensaje indicando que las variables fueron inicializadas con éxito. -El sistema permite el despliegue de información de los signos del paciente. -El sistema queda pendiente a las peticiones al servicio web.

Información general

Caso de Uso:	**Pruebas de sistema.**
Actores:	Administrador.
Propósito:	Comprobar el correcto funcionamiento del servicio web.
Resumen:	El administrador puede elegir la opción de realizar las pruebas a l sistema, en donde podrá simular el envío de peticiones desde el dispositivo con sistema Android al servicio web y el envío de peticiones desde la placa electrónica Raspberry al servicio web.

Tipo:	Secundario.

Precondiciones El servicio web debe estar previamente ejecutado. Las variables deben estar inicializadas.

Flujo principal **-**Este caso de uso inicia cuando el administrador elige la opción de realizar las pruebas al sistema. -El sistema presenta al administrador dos opciones de pruebas: pruebas de envío desde el sistema con sistema Android o pruebas de envío desde la placa electrónica Raspberry. -Una vez el administrador elige la opción, se despliega una página web. -El administrador llena los campos en la página web correspondientes a lo que se envía al servidor. -El sistema muestra un mensaje informando si los datos fueron cargados exitosamente.

Información general

Caso de Uso:	**Agregar / Detener lectura.**
Actores:	Administrador, Portero
Propósito:	Agregar una nueva lectura al sistema y detener la lectura de los signos vitales.
Resumen:	El administrador puede elegir la opción en la página web de administración de realizar pruebas, por medio de esta opción tiene la posibilidad de agregar una lectura nueva y el sistema comenzara a recibir la información de los sensores que vienen desde la placa electrónica Raspberry. Así mismo por medio de la misma opción puede detener la lectura de los signos y colocar por defecto todas las variables. El portero realiza un papel importante por medio de este caso de uso, debido a que es quien se encarga de realizar principalmente el inicio de lectura en el sistema agregando un nuevo usuario o la detención de la lectura, todo esto a través del dispositivo móvil mediante la aplicación Android.
Tipo:	Secundario (Administrador), Primario (Portero).

Precondiciones Para agregar un nuevo usuario y comenzar la lectura, primero debe ser colocado el sensor al usuario o paciente. Para finalizar la lectura, primero se debe haber iniciado previamente.

Flujo principal

Administrador
-Este caso de uso inicia cuando el administrador elige la opción de realizar las pruebas al sistema.
-Del mismo modo que en el caso de uso de Realizar pruebas, se presentan las opciones al administrador.
-El administrador ingresa los datos del usuario y da inicio a la lectura.
-El sistema despliega los datos de los signos vitales y las posibles alarmas.

Portero
-Este caso de uso inicia cuando el portero incia la aplicación móvil.
-La aplicación despliega una pantalla inicial de conexión y luego despliega una interfaz donde se muestran los signos leídos o los valores por defecto en caso de no haber iniciado ninguna lectura.
-En la interfaz se encuentran dos botones para agregar y para detener lectura respectivamente.
-Al agregar o detener lectura vuelve la interfaz donde se despliegan los valores de los signos vitales.

Información general

Caso de Uso:	**Visualizar signos**
Actores:	Administrador, Portero, Médico
Propósito:	Observar los signos vitales y posibles alarmar en los dispositivos de despliegue.
Resumen:	El administrador puede elegir la opción en la página web de administración de Página principal / Despliegue de datos de usuario que le permitirá visualizar signos e inmediatamente se desplegará un página web en donde se podrán observar los signos junto a su respectivo usuario y se podrá observar las posibles alarmas generadas. El portero puede acceder a este caso de uso mediante la aplicación Android en el dispositivo móvil, en la pantalla principal de la aplicación donde se desplegaran los mismos datos que en la página web del administrador. El médico puede acceder al caso de uso visualizar signos a través de la misma aplicación Android en un dispositivo o móvil, también se dispone desplegar la información en un televisor con conexión a internet ubicado en la estación de enfermería.
Tipo:	Secundario (Administrador), Primario (Portero), Primario (Médico).

Precondiciones

El sistema debe estar ejecutándose.

Para un correcto funcionamiento de este caso de uso, se debe haber iniciado al menos un usuario en el sistema y haber colocado un sensor para comenzar con la lectura y

poder visualizar los datos.

Flujo principal

Administrador
-Este caso de uso inicia cuando el administrador elige la opción de Página principal / Despliegue de datos de usuario.
-Se despliega la página web donde se puede visualizar los signos de los usuarios conectados a los sensores.

Portero
-Este caso de uso inicia cuando el portero inicia la aplicación móvil.
-La aplicación despliega una pantalla inicial de conexión y luego despliega una interfaz donde se muestran los signos leídos o los valores por defecto en caso de no haber iniciado ninguna lectura.

Médico
-Este caso de uso inicia cuando el médico inicia la aplicación móvil.
-La aplicación despliega una pantalla inicial de conexión y luego despliega una interfaz donde se muestran los signos leídos o los valores por defecto en caso de no haber iniciado ninguna lectura.
-Se conecta el televisor Smart a internet y se ingresa a la dirección web del servicio mediante el cual se despliega en la pantalla los datos de los signos de los pacientes.

2.7 Diagrama de Clases Servicio Web

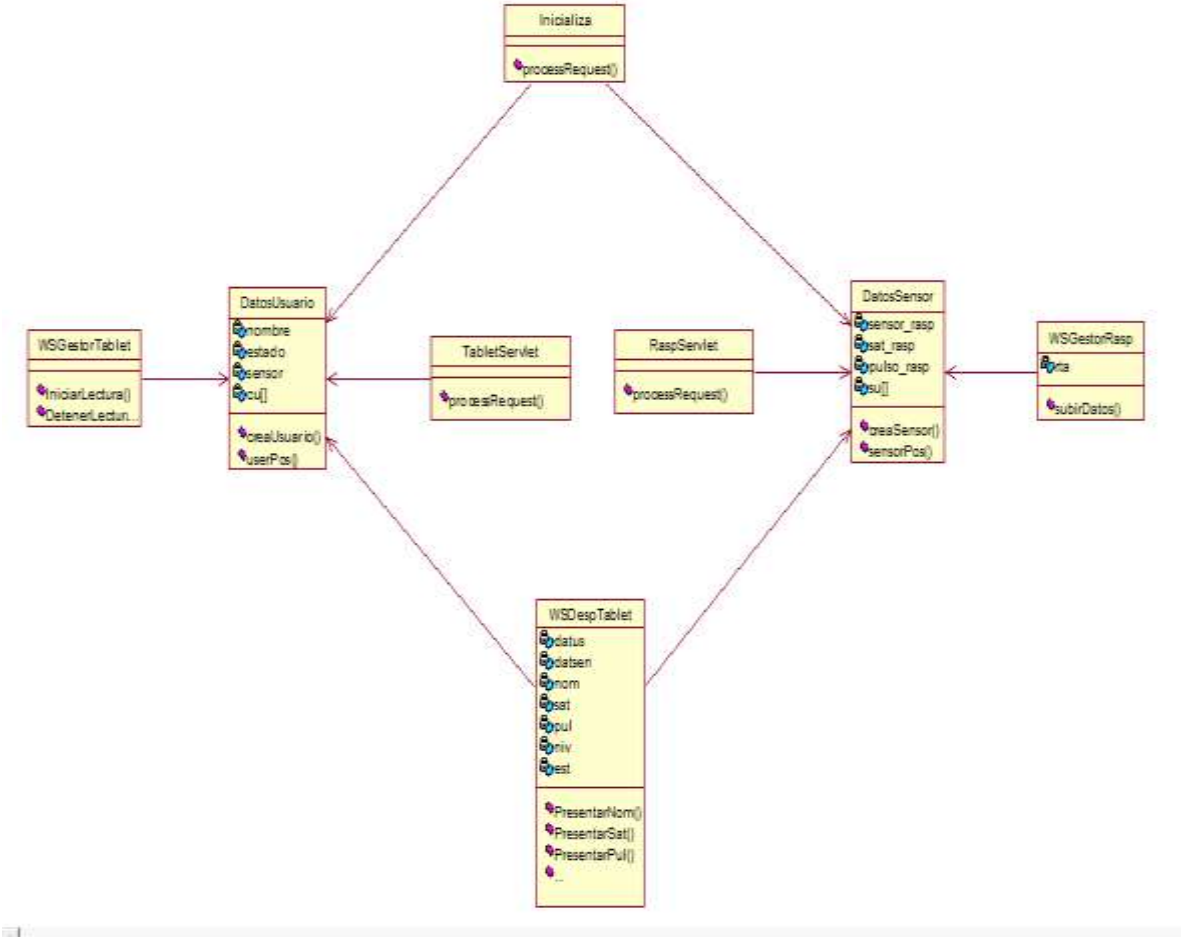


Figura 10. Diagrama de Clases.

Descripción del diagrama de clases

El diagrama de la figura 10, representa las clases utilizadas en el servicio web, a continuación se hace una pequeña descripción de lo que contiene cada clase:

-Clases DatosUsuario y DatosSensor:

Estas clases son las principales en el programa, debido a que son utilizadas para la creación y consulta de los datos del usuario o paciente y sus respectivos signos vitales. Estas clases se encuentran conectadas con todas las demás clases del programa y son usadas para la sincronización entre paciente y sensor. La clase DatosUsuario contiene como atributos generales el *nombre* del usuario que va a usar el sensor, el *estado* de la lectura que varía entre off(apagado) y on (encendido), el *sensor* que representa el número[5] de sensor que está usando y por último la variable *cu*[], la cual es una variable

[5] Este número representa el id del sensor para que el sistema sepa que sensor se está usando.

que contiene un arreglo de variables de tipo DatosUsuario en la que se guardaran las ocho lecturas de los diferentes datos de usuario.

La clase DatosSensor también tiene sus atributos: *sensor_rasp* que representa el número de sensor que se está usando, *sat_rasp* que representa la saturación que se envía desde el sensor a través de la placa electrónica Raspberry, *pulso_rasp* la cual representa el valor del pulso cardiaco enviado a través de la placa electrónica Raspberry y por último la variable *su*[], la cual es una variable que contiene un arreglo de variables de tipo DatosSensor en la que se guardaran las ocho lecturas de los diferentes sensores.

La *variable sensor* y *sensor_rasp* representan un mismo valor en ambas clases con lo que se logra sincronizar datos de usuario con datos provenientes de un sensor.

-Clases TabletServlet y RaspServlet:

Estas dos clases son dos servlets[6]. Son utilizadas para responder a las peticiones hechas por las páginas web contenidas en el servidor, mediante las cuales se realizan las pruebas del sistema a través del envío de datos con un formulario web. Estas peticiones son atendidas gracias a que las dos clases implementan un método de java llamado processRequest(). Las dos clases reciben los datos del dispositivo móvil (Tablet) y de la placa electrónica Raspberry respectivamente y los guardan en las variables locales del sistema, para luego ser desplegados. La clase TabletServlet guarda sus datos utilizando la clase DatosUsuario y la clase RaspServlet guarda sus datos utilizando la clase RaspServlet.

-Clase Inicializa:

Esta clase es otro servlet, el cual también contiene el método processRequest() que atiende las peticiones realizadas por la página web que envía los datos para inicializar los datos de las variables locales del sistema correspondientes a los signos vitales de saturación de oxígeno y pulso cardiaco, nombre de usuario, sensor, estado y nivel, con el fin de iniciar el sistema. Esta clase hace uso de las clases DatosUsuario y DatosSensor

-Clases WSGestorTablet y WSDespTablet:

Son dos servicios web (WebServices) que contienen métodos web, para responder peticiones realizadas desde el dispositivo móvil (Tablet), mediante una aplicación Android. Estas dos clases hacen uso de las clases DatosUsuario y DatosSensor

EL servicio web WSGestorTablet se encargar de recibir peticiones realizadas desde el dispositivo móvil que se encargaran de dar inicio a una lectura y también las enviadas para detener una lectura. Cuenta con dos métodos web para realizar estas tareas llamados: IniciarTarea() y DetenerTarea().

El servicio web WSDespTablet se encarga de recibir peticiones realizadas desde el dispositivo móvil para enviar los datos al dispositivo correspondientes a los signos vitales, niveles y nombres, para que puedan ser desplegados en el dispositivo móvil. Con el

[6] Clases pertenecientes al lenguaje java, que permiten mejoran las capacidades del servidor.

propósito de responder a estas peticiones cuentas con 5 métodos web llamados: PresentarNom(), PresentarSat(), PresentarPul(), PresentarNiv() y Estado().

-Clase WSGestorRasp

Es un servicio web que recibe las peticiones hechas por la placa electrónica Raspberry, en la que se envían los datos leídos desde los sensores inalámbricos de pulsioximetría, el sensor utilizado, además del nivel que resulta del procesamiento de los datos de los sensores. El servicio web a través de este método debe almacenar los datos en las variables locales del sistema utilizando las clases DatosUsuario y DatosSensor.

2.8 Diagramas de secuencia

Se presentan los diagramas de secuencia, teniendo en cuenta los casos de uso presentados. Se inicia, considerando que el caso de uso Iniciar servicio web, ya está funcionando y a partir de este se presentan los demás casos de uso.

Diagrama de secuencia que explica el funcionamiento dinámico del caso de uso Inicializar variables:

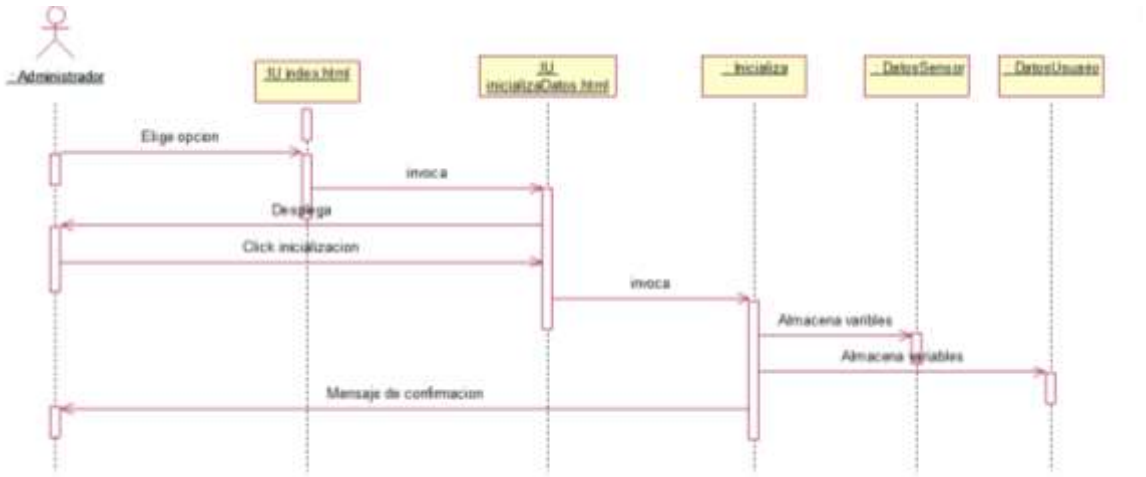


Figura 11. Diagrama de secuencia caso de uso inicializar.

En el anterior diagrama de secuencia, figura 11, se puede apreciar los pasos que realizará el sistema para inicializar las variables. La secuencia es iniciada por el administrador, eligiendo la opción de inicializar variables en la interfaz de usuario index.html, la cual hace el llamado a la siguiente interfaz de usuario inicializarDatos.html y esta se muestra al administrador. Luego de leer datos de cómo se inicializaran datos se deberá elegir una opción con la cual el sistema o servicio web invoca al servlet llamado inicializa, quien se encarga de almacenar los datos por defecto en las variables del sistema por medio de las

clases DatosSensor y DatosUsuario, para finalizar mostrando un mensaje de confirmación al administrador, indicando que se almacenaron los datos con éxito.

Diagrama de secuencia que explica el funcionamiento dinámico del caso de uso Pruebas de sistema:

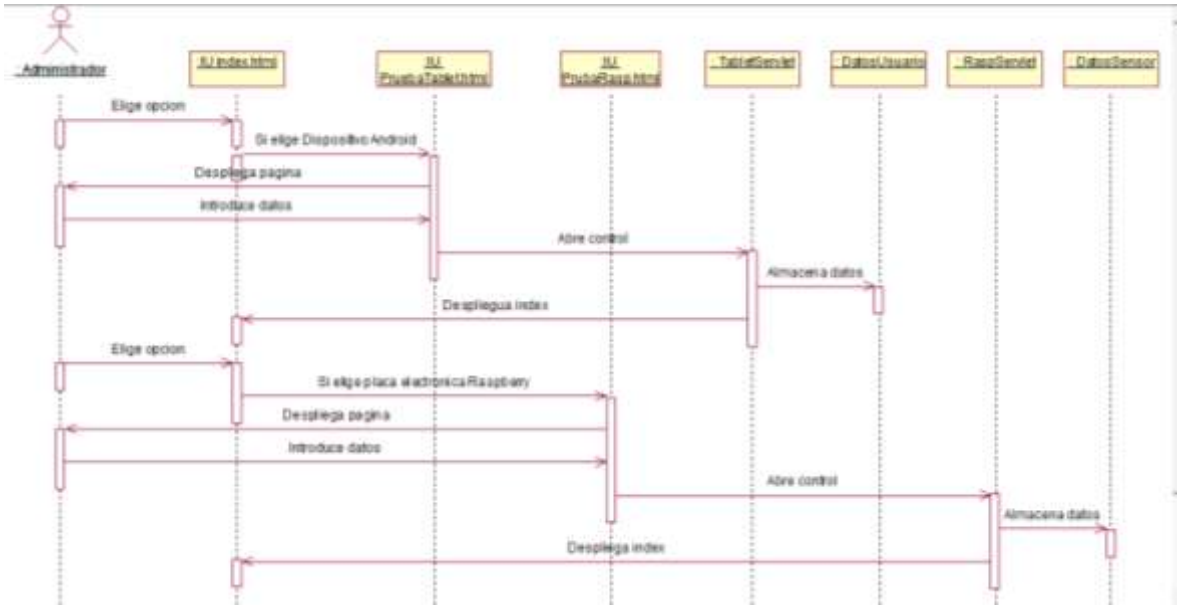


Figura 12. Diagrama de secuencia caso de uso Pruebas de sistema.

El anterior diagrama de secuencia, figura 12, muestra como el administrador debe elegir la opción de realizar pruebas de envio de datos desde el dispositivo móvil a lo cual el sistema responde desplegando la interfaz de usuario PruebaTablet.html al administrador. En esta página web el administrador debe ingresar los datos solicitados, que serán enviados a través de un servlet llamado TabletServlet, quien almacena los datos ingresados en las variables del sistema a través de la clase DatosUsuario, finalizando el primer proceso volviendo a desplegar la interfaz de usuario o página web index.html al administrador. El administrador también puede elegir la opción de hacer pruebas de envio desde la placa electrónica Raspberry, a lo cual se despliega al administrador la interfaz de usuario PruebaRasp.html, que utiliza el servlet llamado RaspServlet para almacenar los datos utilizando la clase DatosUsuario, en las variables del sistema. Finalmente vuelve a desplegar la interfaz de usuario index.html al administrador.

Diagrama de secuencia que explica el funcionamiento dinámico del caso de uso Agregar / Detener Lectura:

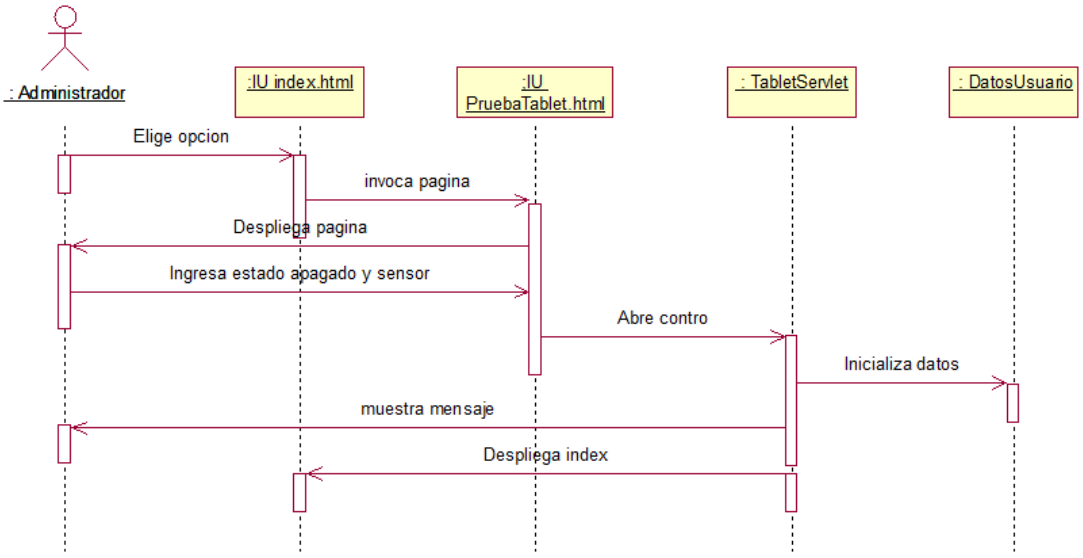


Figura 13. Diagrama de secuencia caso de uso Agregar / Detener Lectura.

El diagrama anterior, figura 13, se muestra la secuencia del servicio web para agregar un usuario para su monitoreo y detener un monitoreo. El administrador debe elegir la opción de ingresar datos que vendrían desde el dispositivo móvil (Tablet), de la misma manera como se realiza para hacer las pruebas vista anteriormente, la diferencia es que se debe ingresar un estado de apagado (off) en los datos que solicita la página web, con lo cual la página a través del servlet llamado TabletServlet y esté utilizando la clase DatosUsuario almacena los datos en las variables del servicio web.

Diagrama de secuencia que explica el funcionamiento dinámico del caso de uso Visualizar signos:

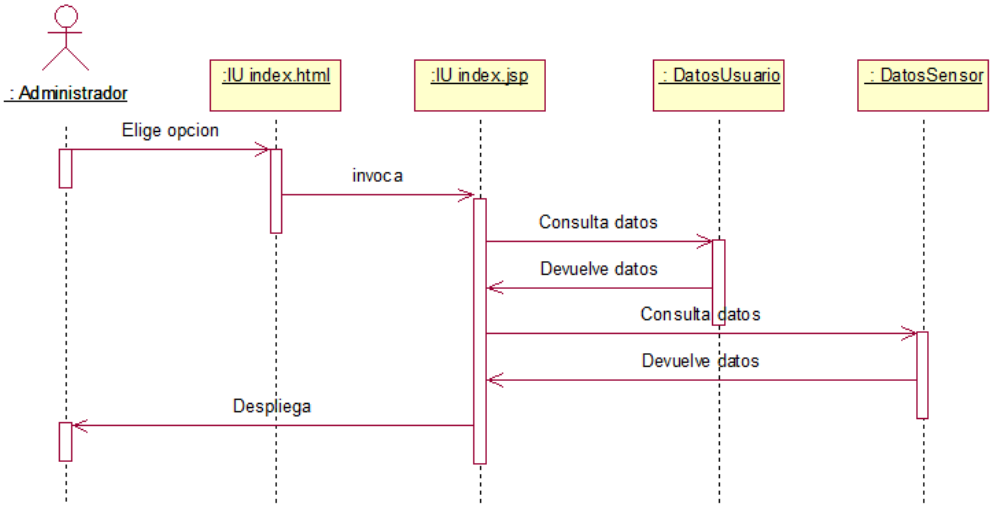


Figura 14. Diagrama de secuencia caso de uso Visualizar signos.

En el anterior diagrama, figura 14, se puede observar la secuencia que debe tomar el sistema para desplegar los datos al administrador. El administrador elige la opción de visualizar en la interfaz de usuario index.html, esta despliega una nueva interfaz de usuario llamada index.jsp. Esta nueva interfaz consulta los datos almacenados en las variables por medio de las clases DatosUsuario y DatosSensor, y termina la secuencia mostrando los datos de los signos al administrador. De la misma manera trabaja el sistema para desplegar los datos en el dispositivo de despliegue representado por un televisor con conexión wifi, donde puede visualizar los datos el médico.

CAPÍTULO 3

3. Ejecución del proyecto

3.1 Dispositivos para uso en el proyecto

Teniendo en cuenta los módulos necesarios para el proyecto y los dispositivos encontrados en el mercado, se eligieron los siguientes 3 dispositivos, teniendo en cuenta su costo, accesibilidad y compatibilidad con el sistema:

I. Sensor de Pulsioximetria

Después de un larga búsqueda de un sensor apropiado para el proyecto, se eligieron dos opciones, debido a que el resto de sensores inalámbricos encontrados tenían su sistema cerrado para desarrolladores o atados a un sistema no compatibles con este proyecto. La

primera opción, el sensor de pulsioximetria o pulsioximetro inalámbrico Bluetooth Jumper JDP-500F mostrado en la figura 15, fabricado por Shenzhen Jumper Medical Equipment. Es un sensor portátil que cuenta con un sistema de medición de saturación de oxígeno en la sangre y pulso cardiaco, Pantalla LED de fácil lectura que muestra SpO2 y frecuencia del pulso. Se conecta de forma inalámbrica a través de Bluetooth, trabaja offline y online. Cuenta con una App gratuita integrada oxigen pressure.

Figura 15. Pulsioxímetro inalámbrico Jumper.

Esta primera opción fue descartada debido a que al realizar pruebas con este sensor se encontró que a pesar de que en la página principal del fabricante no lo decía, este sensor solo trabaja con su aplicación de propietario y no permite conexión libre desde otro sistema ni tampoco presenta sdk para la obtención de sus datos.

La segunda opción, fue hacer uso de la placa e-Health Sensor Platform V2.0, figura 15.1, esta placa desarrollada por Cooking Hacks, permite la lectura de 9 tipos de sensores diferentes de signos del cuerpo humano, entre los cuales cuenta con un pulsioxímetro, figura15.2 y es posible el uso de un módulo bluetooth ya que esta placa electrónica cuenta con pines de entradas y salidas analógicos y digitales, además la universidad del Cauca cuenta con esta placa y es posible su utilización en las pruebas de este proyecto.

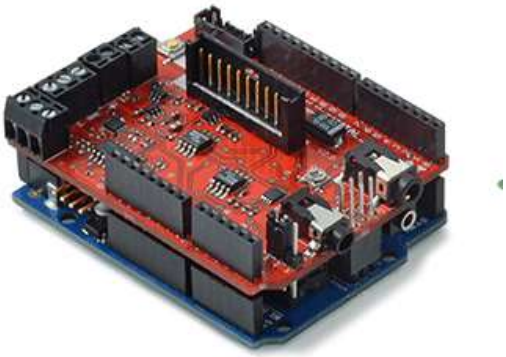


Figura 15.1 e-Health Shield

[https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical#img_6]

Figura 15.2 Sensor de Pulsioximetría.

[https://www.cooking-hacks.com/pulse-and-oxygen-in-blood-sensor-spo2-ehealth-medical]

Esta segunda opción fue la elegida y utilizada para la realización de este proyecto.

II. Placa procesadora de datos de signos vitales

Para el presente trabajo se eligió la placa electrónica Raspberry PI 3, mostrada en la figura 16, un computador miniatura, que cuenta con módulos como:

A 1.2GHz 64-bit quad-core ARMv8 CPU

4 USB ports

40 GPIO pins

Full HDMI port

Ethernet port

Combined 3.5mm audio jack and composite video

Camera interface (CSI)

Display interface (DSI)

Micro SD card slot (now push-pull rather than push-push)

VideoCore IV 3D graphics core

802.11n Wireless LAN

Bluetooth 4.1

Bluetooth Low Energy (BLE)

Destacando los módulos Bluetooth y wifi, lo cuales son importantes para este proyecto, ya que con estos módulos se harán las conexiones con los sensores y con el servidor.

Figura 16. Raspberry Pi 3, usada en el proyecto junto con su caja en acrílico para su protección.

III. Dispositivos para el despliegue de datos

Para este proyecto se consideran dos dispositivos para el despliegue de los datos y las alarmas.

El primer dispositivo, debido a que debe soportar la aplicación Android se eligió una tableta, figura 17, con sistema operativo Android igual o suerior a la versión 4.0, con conexión wifi , pantalla de 7 pulgadas , para un mejor manejo de la aplicación por parte del usuario y una mejor visualización de los signos presentados en su interfaz principal y las posibles alarmas.

El segundo dispositivo utilizado para desplegar los datos es un televisor Smart tv del tipo encontrado en la figura 18, el televisor es ubicado en frente de la estación de enfermería, con el fin de que los médicos y las enfermeras puedan observar los signos vitales de los usuarios conectados a los diferentes sensores mientras esperan por su atención antes de pasar por la sala de Triage.

Figura 17. Tableta electrónica Android marca Increa, usada en el proyecto.

Figura 18. Televisor Samsung.

3.2 Módulos del sistema y comunicación

3.2.1 Módulo de medición y procesamiento de signos vitales

La placa Raspberry por su gran capacidad y velocidad de procesamiento permite la instalación de un sistema operativo, para el caso de este trabajo el sistema operativo Raspbian Jessie versión de kernel 4.1, el cual es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian Wheezy (Debian 7.0) para la placa computadora (SBC) Raspberry Pi, orientado a la enseñanza de informática. El lanzamiento inicial fue en junio de 2012 [28]

Para el procesamiento de los datos en necesario primero que el sensor este encendido y se coloque en el dedo del paciente al que se le va a realizar la medida y monitoreo de sus signos vitales

La codificación en la placa Raspberry se realiza utilizando el lenguaje de programación de Java. Para que este lenguaje pueda compilar y ejecutarse en el sistema operativo Raspbian es necesario el uso de un paquete llamado OpenJdk, siendo este una versión libre de la plataforma de desarrollo de java.

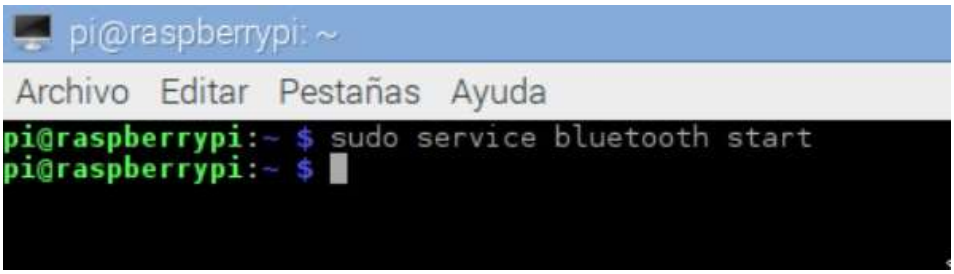
Este módulo no cuenta con una interfaz gráfica desarrollada, debido a que solo cuenta con código, el cual se ejecuta para recibir y transmitir los datos de signos vitales y no hay necesidad que el usuario o administrador del sistema introduca dato alguno, solo ejecutarlo una vez para iniciar el programa, pero a través de comandos por terminal de sistema.

Para lograr utilizar el Bluetooth interno de la placa Raspberry, fue necesaria la instalación de un paquete llamado Bluez, que cueneta con herramientas para la administración y configuración del Blueetooth. La herramienta más utilizada fue *gatttool*, usada para la conexión con el sensor remoto, luego se utilizan los comandos mostrados a continuación para la conexión, emparejamiento y recepción de datos:

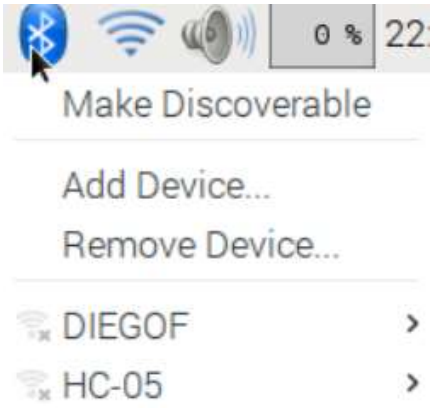
Otra herramienta utilizada fue *Bluetoothctl*, con la cual se logra realizar un escaneo de los dispositivos cercanos y ver su dirección física única Mac.

Con el fin de utilizar datos reales de pacientes y como se mencionó anteriormente se utilizó la placa e-Health Sensor Platform V2.0, a la cual se adicionó un módulo Bluetooth HC-05 y esta placa a su vez es montada sobre una placa Arduino Uno para su fácil programación. La función de la placa Arduino es recibir los datos de los signos provenientes del sensor de pulsioximetría y enviarlos a través del módulo Bluetooth. Para finalizar el módulo de sensor inalámbrico, se adaptó una batería de 9 voltios a la placa para mejor manejo al momento de realizar las pruebas en ambiente real. El código y su descripción realizado para el funcionamiento de la placa Arduino se encuentra en el anexo A.

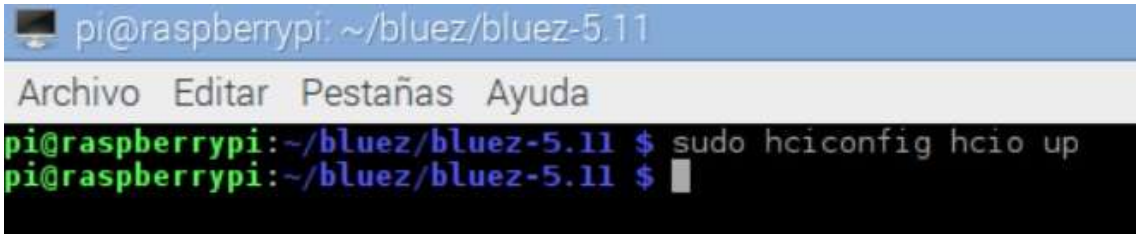
Una vez listo el sensor inalámbrico, se procede a iniciar el servicio de Bluetooth en la placa electrónica Raspberry. Esto se hace mediante el siguiente comando, en modo super usuario (sudo):



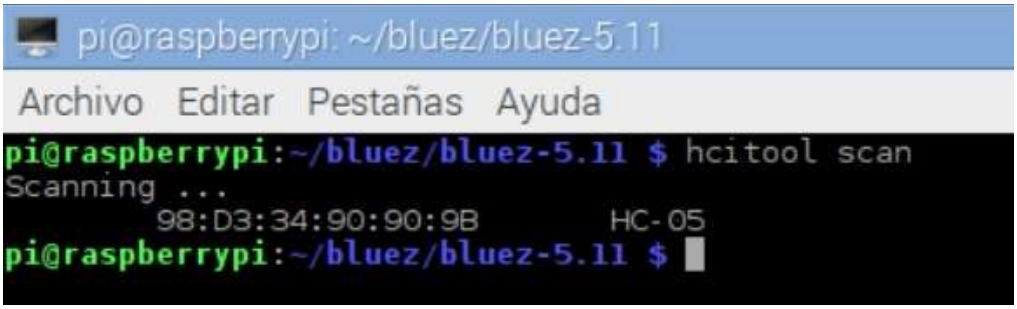
Una vez iniciado el servicio Bluetooth, se procedió a la verificación de la detección del módulo HC-05 por la placa Raspberry:



Al verificar que se detecta el modulo Bluetooth, se colocó en alto la interfaz con la cual se realizará lecturas:



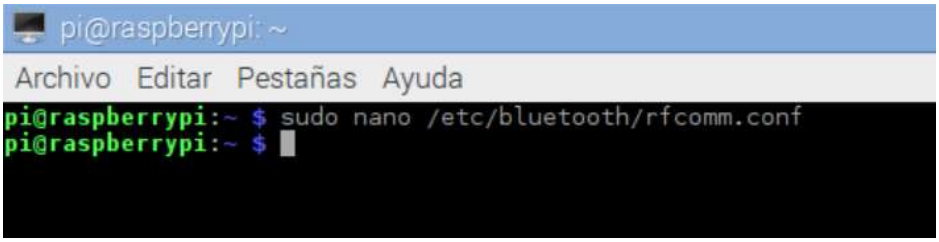
Se realiza un escaneo del módulo mediante el siguiente comando:



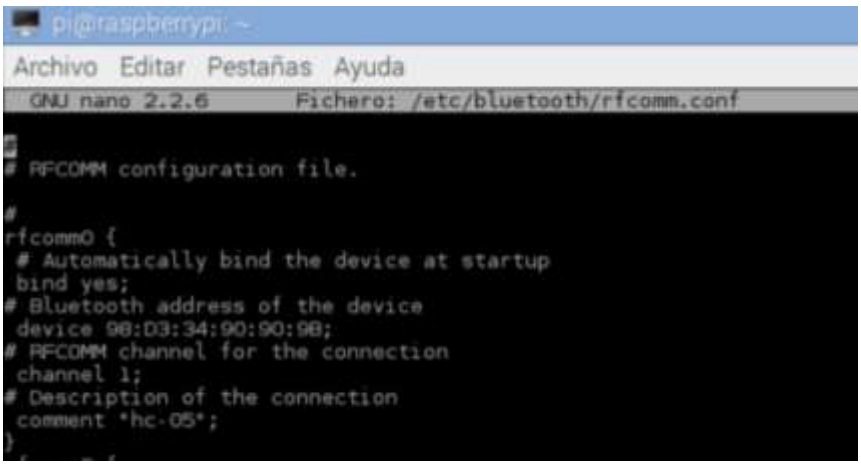
Mediante este escaneo es posible visualizar el nombre del módulo Bluetooth así como su dirección MAC, la cual es única para cada dispositivo y esta es usada para la configuración en el puerto mediante el protocolo RFCOMM, con lo cual se garantiza que el sistema no va a aceptar transmisiones de datos de dispositivos extraños al sistema,

Paso seguido se empareja el sensor con la placa electrónica Raspberry, y se ingresa el número de código del dispositivo (para este caso 1234):

Este mismo emparejamiento se debe realizar con cada uno de los sensores que se quiera agregar al sistema hasta un máximo de 8. Luego de esto se configuro el archivo rfcomm.conf, en donde se va a configurar cada dispositivo con su respectiva dirección MAC con el fin de garantizar la seguridad del sistema en cuanto a intromisiones de dispositivos extraños al sistema. Para ingresar a este archivo se digita el siguiente código en la terminal del sistema de Raspberry así:



Con lo cual se obtiene el siguiente archivo:



Donde rfcomm0 hace referencia al primer sensor (el segundo sensor seria rfcomm1, el tercero sería rfcomm2, etc.), se coloca la dirección MAC o física de cada dispositivo, un

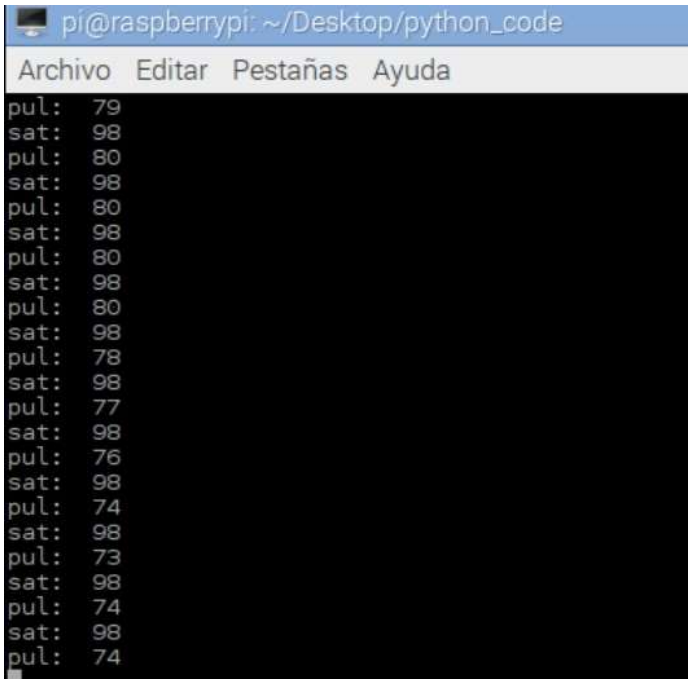
canal de transmisión y el nombre del dispositivo, esto para cada dispositivo agregado al sistema.

Una vez agregado el dispositivo utilizado en nuestro proyecto, se procedió a la realización del código para la lectura y el procesamiento de los datos recibidos desde los sensores. Para el desarrollo de estos códigos se tomó dos lenguajes: Lenguaje JAVA para el código necesario para realizar las pruebas, los cuales son DeteccionSignosSemi.java con el cual se reciben datos reales en forma manual y de ahí en adelante los datos son procesados por el sistema hasta su despliegue y el segundo código llamado DeteccionSignosSim.java el cual es utilizado para el procesamiento de datos a travez de una aplicación móvil que puede modificar el rango de datos de los signos de saturación de oxígeno y pulso cardiaco con el fin de desplegar las alarmas en la prueba de ambiente simulado. El segundo lenguaje utilizado es Python, siendo este el lenguaje nativo en en el sistema operativo Raspbian y por su facilidad de uso, fue utilizado para crear el código principal llamado deteccionSignosAuto.py con el cual se leen los datos de cada uno de los sensores inalámbricos y su posterior procesamiento de datos y envío al servidor, todo esto de forma automática.

El codigo python mencionado anteriormente se encuentran descrito en el anexo A, los códigos java no se encuentran en este documento debido a que solo fueron utilizados en las pruebas en ambiente simulado, pero no hacen parte del sistema final, pero están anexados en el CD entregado junto con este docuemento.

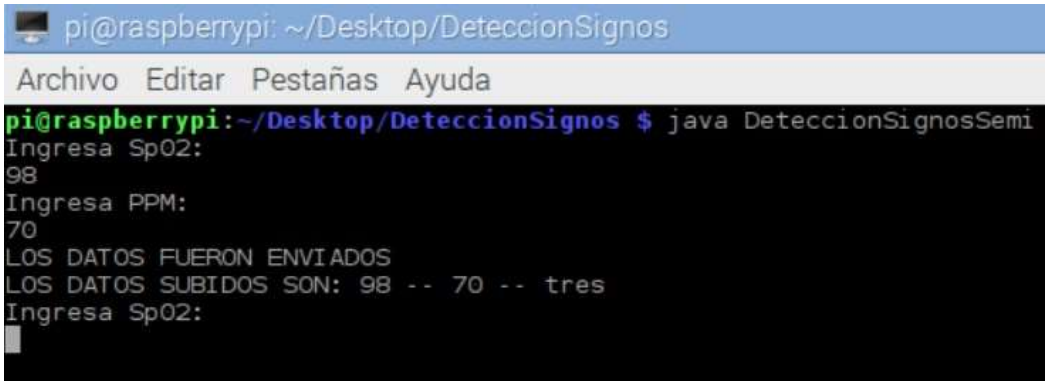
Una vez creados los códigos y compilados, se procedió a probarlos. Se inició con el código principal deteccionSignosAuto.py, para esto primero se ejecuta el código mediante el siguiente comando:

El sistema detecta automáticamente los signos, los procesa para detectar alarmas y se envían al servidor. Para comprobar la lectura de los datos se imprimen los signos en el termina de sistema como se muestra a continuación:



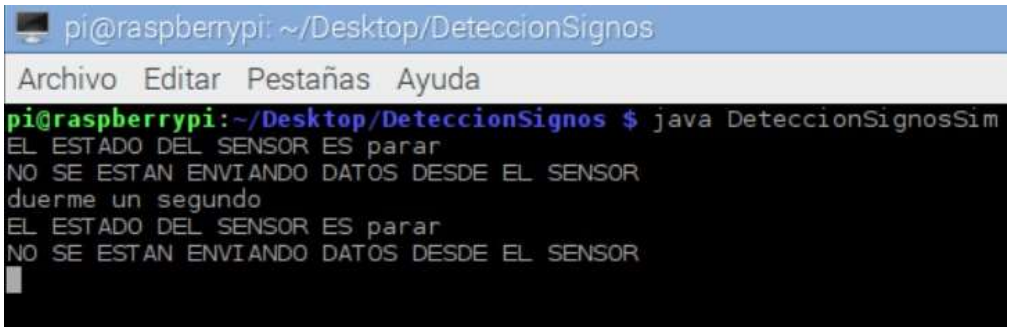
Al mostrar los datos leídos desde el sensor y al aparecer en el servicio web junto a su tipo de Triage se probó el correcto funcionamiento de la lectura de los datos y procesamiento por parte de la placa Raspberry, además tampoco genero ningún tipo de error.

Una vez probado el código principal, se procedió a ejecutar el primer código para las pruebas del sistema DeteccionSignosSemi.java, que se ejecuta mediante el comando:

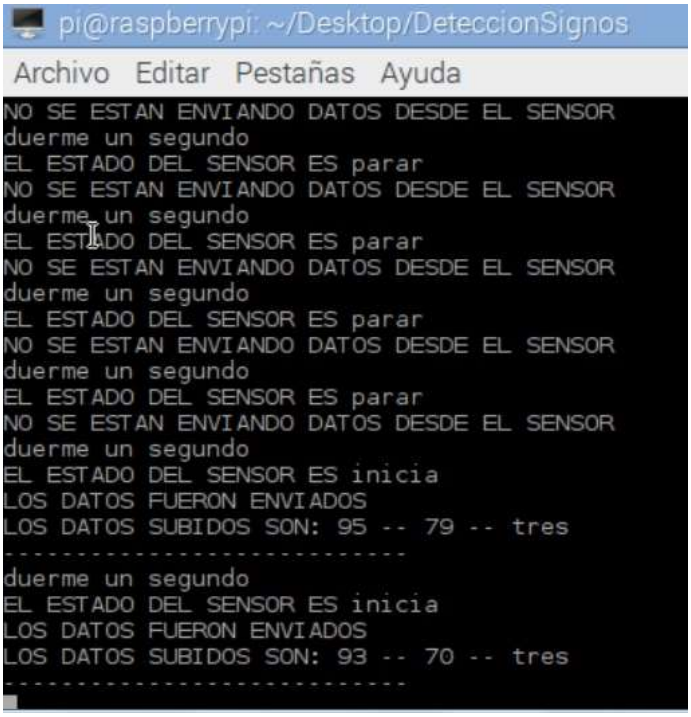


Donde se puede apreciar que el sistema pide el valor de saturación de oxigeno leído en un pulsioximetro externo sin conexión Bluetooth y luego el valor de pulso cardiaco y una vez ingresados el sistema automáticamente procesa los datos, genera un tipo de Triage y envía los datos al servidor web. Además imprime los datos necesarios para comprobar su funcionamiento.

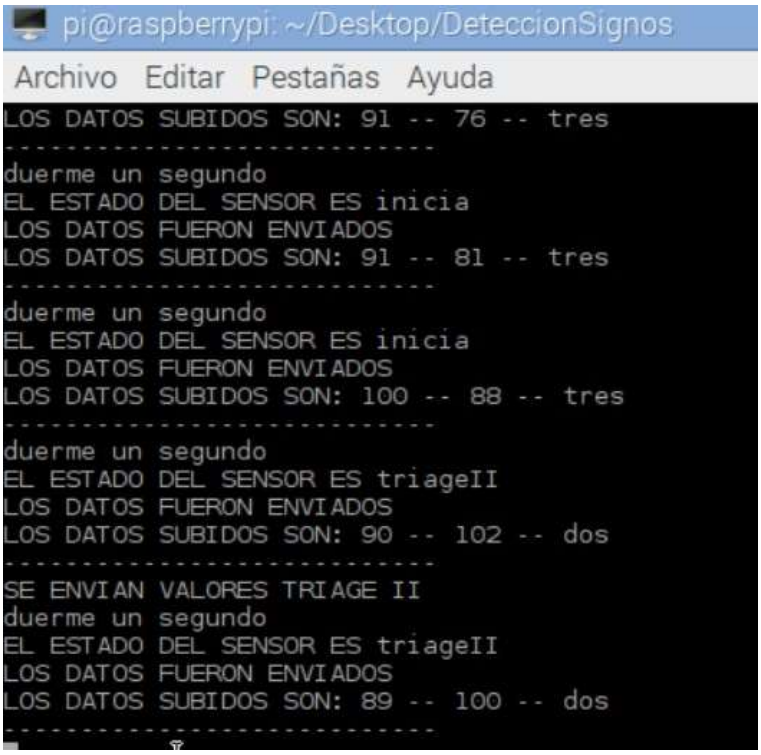
El segundo código para pruebas DeteccionSignosSim.java se ejecuta mediante el siguiente comando:



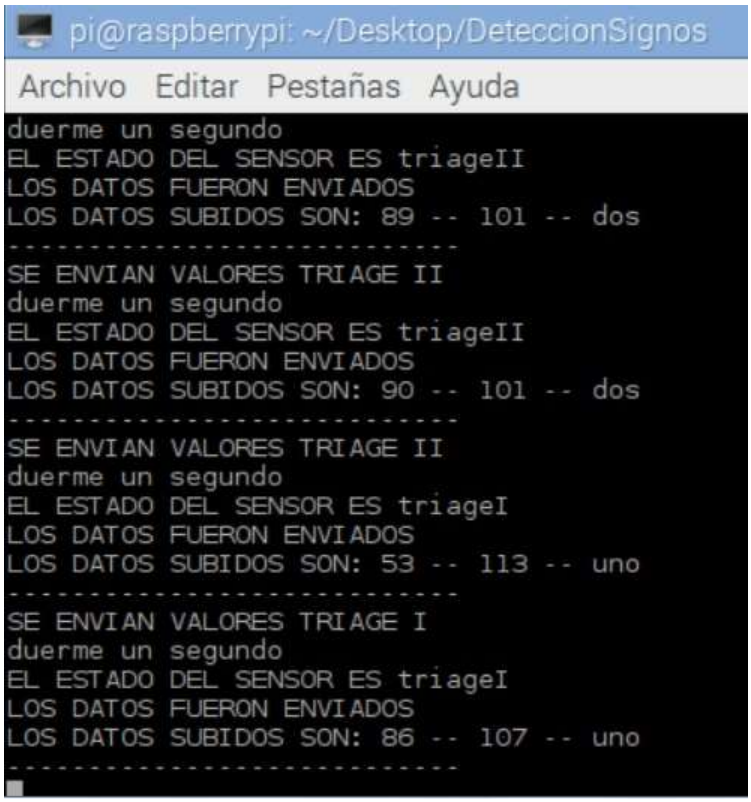
Este código esta pendiente de la una aplicación móvil que simula la transmisión de un pulsioximetro Bluetooth, como se puede apreciar mientras la aplicación no se inicie no detectara ningún sensor. Una vez se inicia la transmisión en la aplicación de simulación el sistema inicia el procesamiento de datos y su envío al servidor web como se observa a continuación:



Desde la aplicación se puede generar una primera alarma tipo Triage II (dos) como se observa a continuación:



Así mismo, se puede generar una alarma de tipo Triage I (uno) de la siguiente manera:



De esta forma, se completa el módulo de medición y procesamiento de signos vitales.

3.2.1.1 Clasificación de medidas para generación de alarmas

Se considera que el porcentaje adecuado de oxígeno en sangre es de entre el 96 y 99 por ciento. Por debajo del 90 por ciento de saturación se produce hipoxemia, es decir disminución anormal de la presión parcial de oxígeno en sangre arterial [29].

Por regla general, la frecuencia cardiaca normal en reposo oscila entre 50 y 100 latidos por minuto. Sin embargo hay que detallar algunos aspectos que alteran su estado:

- Cuando nacemos tenemos una frecuencia cardíaca elevada porque la actividad del organismo es muy intensa. A partir del primer mes de vida, va disminuyendo hasta llegar a la edad adulta, manteniéndose estable después de los 20 años.
- Varía a lo largo del día y la noche y en respuesta a diversos estímulos, por lo que su medición tiene gran variabilidad.
- Al realizar ejercicio físico el corazón produce una respuesta normal que es la taquicardia (la frecuencia cardíaca en reposo está por encima de 100 latidos por minuto -lpm-).
- También puede producirse bradicardia (la frecuencia cardíaca está por debajo de 50 lpm).[30]

Dato importante:

¿Cómo calcular la frecuencia cardiaca máxima?

La frecuencia máxima que puede alcanzar el corazón ante un ejercicio físico alto depende de la edad y puede calcularse mediante esta fórmula:

Frecuencia cardiaca máxima = 220 lpm – edad. [30]

Se realizan mediante las siguientes tablas:

Tabla 2. Clasificación de las desaturaciones	
Clasificación	**Saturación**
Normosaturación	> 95%
Desaturación leve	93%-95%
Desaturación moderada	88%-92%
Desaturación grave	< 88%

Tabla 4. Clasificación de desaturaciones (http://www.efdeportes.com/efd79/ergoes.htm)

Tabla 5. Clasificación pulso cardiaco

(http://es.slideshare.net/JorgeLuisMolinaMoreno/fisiologia-expo-frecuencia-cardiaca)

3.2.2 Módulo de entrada de datos de usuario

El módulo de entrada de datos se compone de una aplicación realizada para el sistema operativo Android. Esta aplicación se realizó para su despliegue en un dispositivo móvil (Tablet), la cual se utiliza para ingresar el nombre del usuario a quien se va a monitorizar y el número de sensor que está utilizando con el fin de sincronizarlos con los datos que vienen desde la placa electrónica Raspberry.

Para la implementación de la aplicación móvil se hace uso del IDE (entorno de desarrollo integrado) Android Studio para la plataforma Android, en la versión 2.1.2. A continuación se describe la aplicación que se implementó para el proyecto. El código de la aplicación móvil y su descripción se encuentran en el ANEXO B.2.

Los datos a ingresar por el portero, quien es el encargado de ingresar a los usuarios en la aplicación móvil son los siguientes:

- Nombre del usuario
- Sensor

El nombre del usuario debe ser corto (Por ejemplo: ingresar el primer nombre y la inicial de su primer apellido), con el fin de agilizar el proceso de monitoreo. El sensor, es un dato que se usa para indicarle al sistema cuales datos pertenecen a un respectivo usuario.

La aplicación se realizó de la siguiente manera: inicia con la interfaz en la cual se comprueba la conexión a la red de área local (Para el caso de la Tablet, verifica la conexión a una red wifi), figura 19.1, si no existe conexión se indicará un pequeño mensaje informando al usuario de la aplicación como se puede observar en la figura 19.2.

Figura 19.1. Interfaz inicial de la aplicación Android.

Figura 19.2. Interfaz inicial con mensaje de No hay conexión.

Si la conexión existe, se programó la aplicación para presentar al usuario una nueva interfaz, en la cual se puede apreciar un listado junto con dos botones, esto se realizó con el fin de seguir el diseño planteado en los casos de uso Agregar/Detener Lectura (Por medios de dos botones, uno para agregar una nueva lectura y otro para detenerla) y Visualizar signos (Con una lista de visualización de los signos del paciente y las alarmas visuales). En el momento de desplegarse la interfaz se presenta en blanco los valores y solo se pueden observar los símbolos de porcentaje (%) de saturación de oxígeno y los símbolos de pulso cardiaco de pulsaciones por minuto (PPM), figura 20.1, si no se ha conectado con el servicio web. En el momento en que la aplicación se conecta con el servicio web, se muestra en lista los valores por defecto para cada sensor como se observa en la figura 20.2, además se colocan las filas de color gris para indicar que no se han iniciado lecturas de los sensores.

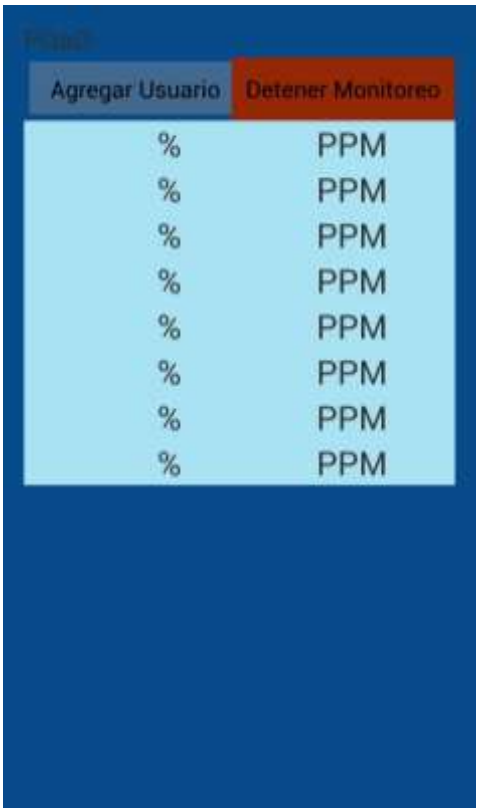


Figura 20.1. Interfaz principal de la aplicación para despliegue, sin conexión con el servicio web.

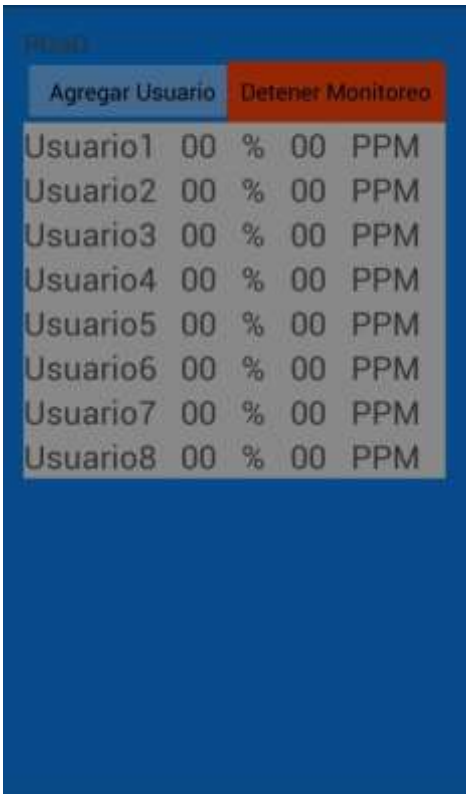


Figura 20.2. Interfaz principal de la aplicación para despliegue, con conexión con el servicio web y valores por defecto.

Continuando con lo que se planteó en el diseño del proyecto, se debía crear una aplicación que permita ingresar el nombre del usuario y el sensor que se está utilizando, para esto la interfaz anteriormente presentada tiene un botón con el texto *Agregar Usuario*, al presionar este botón se despliega la interfaz de la figura 21.1. Esta interfaz se diseñó con una entrada de texto para escribir el nombre del usuario, al dar click sobre esta entrada aparecerá el teclado del teléfono para ingresarlo, figura 21.2.

Dentro de esta interfaz se colocó un Spinner (Lista desplegable), para que se pueda seleccionar el número del sensor que tiene el usuario como se indica en la figura 21.3. Para concluir con esta interfaz se agregó un botón verde grande, con el fin de hacer la aplicación amigable al usuario[7], con la palabra Iniciar.

[7] Amigable al usuario nos referimos a que la aplicación Android

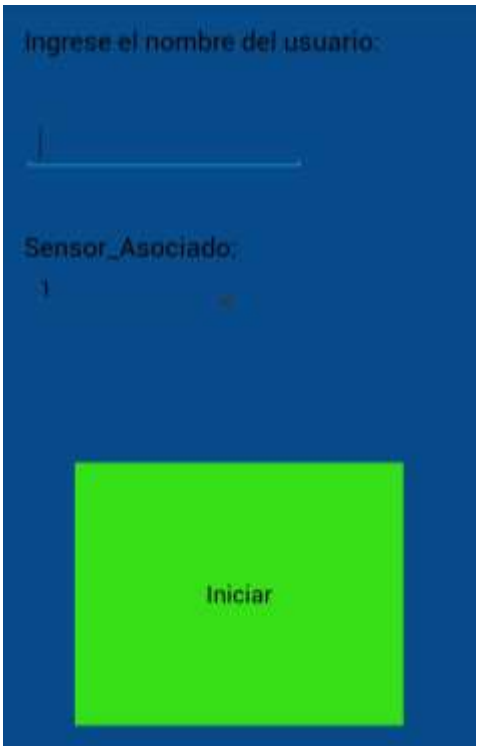


Figura 21.1. Interfaz para agregar nuevo usuario.

Figura 21.2. Interfaz para agregar nuevo usuario y teclado.

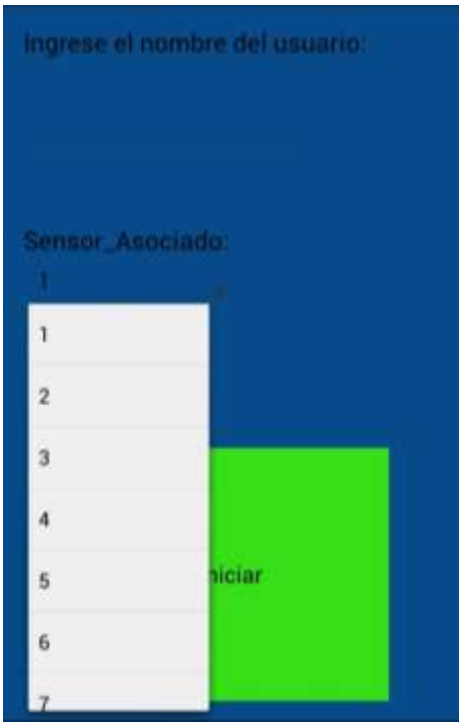


Figura 21.3. Spinner desplegable para elegir sensor.

Cuando se presiona el botón la aplicación envía una petición al servidor informando que ese usuario agregado se encuentra activo para el comienzo del monitoreo y así el servicio web puede comenzar a recibir los datos para ese usuario y empezar a desplegarlos. Al mismo tiempo la aplicación regresa a la interfaz donde se despliegan los datos donde aparecerá el nombre agregado, figura 22.

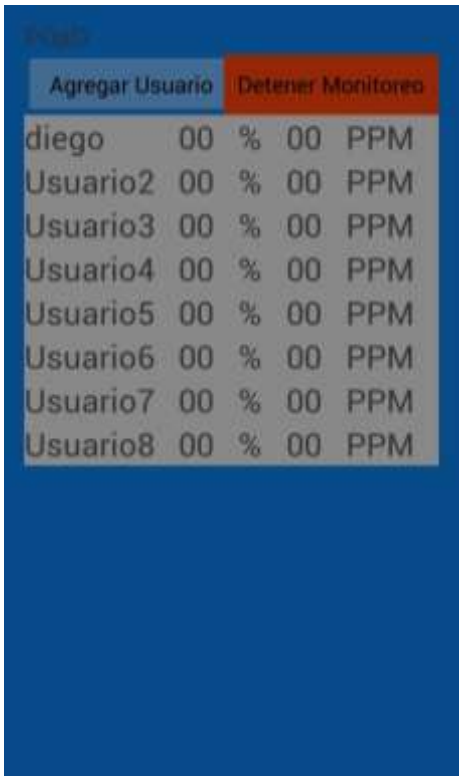


Figura 22. Interfaz desplegando el nombre dl usuario agregado.

Cuando el servicio web comience a recibir los datos de los signos del usuario, la aplicación mostrara los datos en la interfaz. La aplicación móvil Android fue diseñada para indicar por medio de colores el nivel o clasificación de Triage, esto por medio de un dato recibido desde el servicio web. La aplicación maneja cuatro colores para indicar el estado de cada usuario de la siguiente manera:

- Color gris: Indica, que el usuario aún no ha sido agregado o que aún no se ha recibido datos de los signos vitales desde la placa electrónica Raspberry, figura 22.
- Color verde: La fila se coloca en color verde, figura 23.1, cuando la aplicación recibe una datos de clasificación de Triage igual o superior a tres (Clasificación de Triage III) desde el servicio web.
- Color naranja: La fila se coloca en color naranja, figura 23.2, para indicar que existe una alarma de Triage nivel dos (Clasificación de Triage II), desde el servicio web, que indica que se debe tener precaución con el usuario, debido a que sus signos de saturación de oxigeno o pulso cardiaco están saliendo de los rangos normales.
- Color rojo: La fila se coloca en nivel rojo, figura 23.3, cuando se recibe desde el servicio web un dato de clasificación de Triage nivel uno (Clasificación de Triage I), indicando que es una alarma urgente y que el usuario debe ser atendido inmediatamente. Además la aplicación al recibir un datos de Triage nivel uno genera un sonido de alarma, para completar la alarma audio_visual.

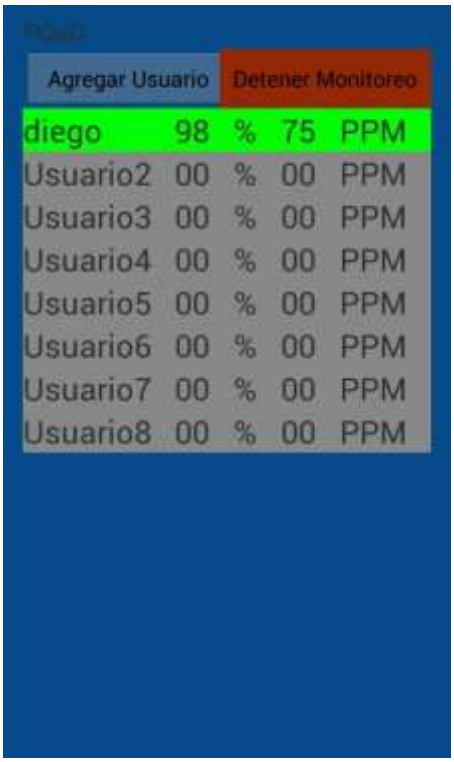


Figura 23.1. Interfaz desplegando datos normales de signos sin alarmas (Color verde).

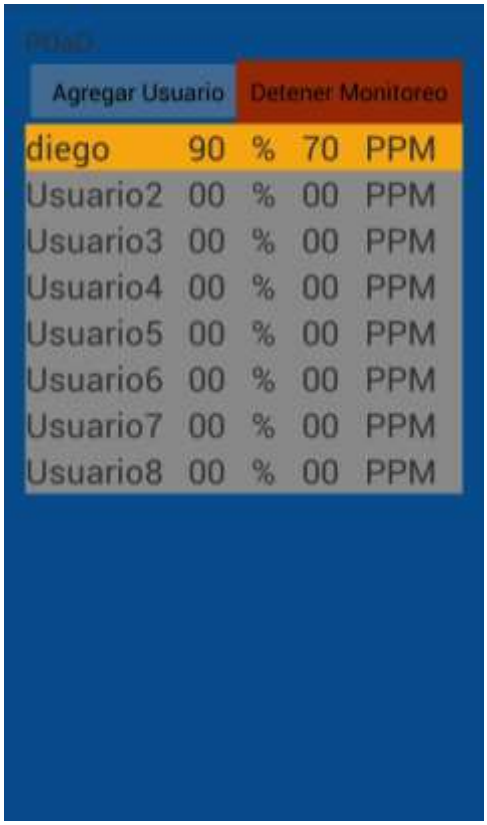


Figura 23.2. Interfaz desplegando datos representando una primera alarma nivel II (Color naranja).

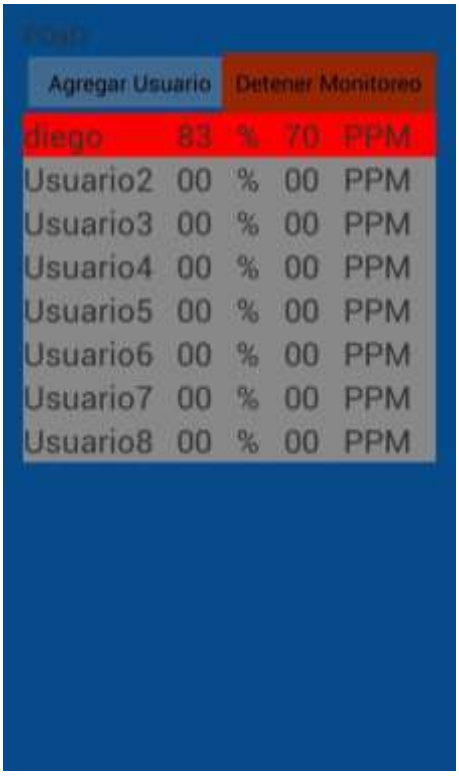


Figura 23.3. Interfaz desplegando datos representando una alarma de nivel I (Color rojo).

De forma análoga se puede ir agregando más usuario de acuerdo a cada sensor, e irán apareciendo en la lista de acuerdo al sensor usado que indicara su posición en la lista, figura 24.

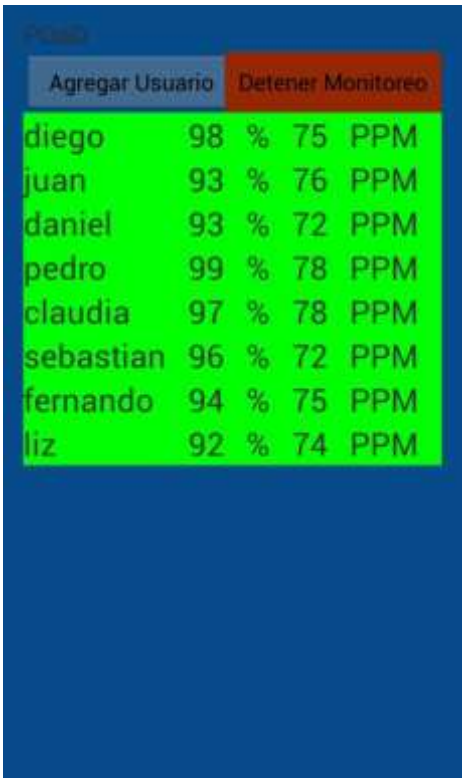


Figura 24. Interfaz con varios usuarios agregados y sus signos.

En el momento en que el usuario es ingresado al consultorio de Triage del hospital para ser valorado por el médico, es necesario retirarle el sensor y detener la lectura del sistema. Para esto se coloca en la interfaz de despliegue un botón de color rojo, con el texto Detener Monitoreo, el cual se programó la aplicación para desplegar una nueva interfaz, figura 25.1, en la cual se colocó un Spinner para que el usuario de la aplicación pueda elegir el sensor que se debe dejar de monitorear por el sistema, figura 25.2. Al presionar el Spinner se despliega una lista con número que indican cada sensor de pulsioximetría y se debe dar click sobre el que se desea detener, por ultimo coloco un botón grande de color rojo, para ser amigable la aplicación al usuario, con el cual se enviar una petición al servicio web para indicar que se debe detener la lectura en el sensor indicado. La aplicación móvil regresa a la interfaz de despliegue de datos y detiene el monitoreo del sensor elegido, volviendo a sus valores por defecto.

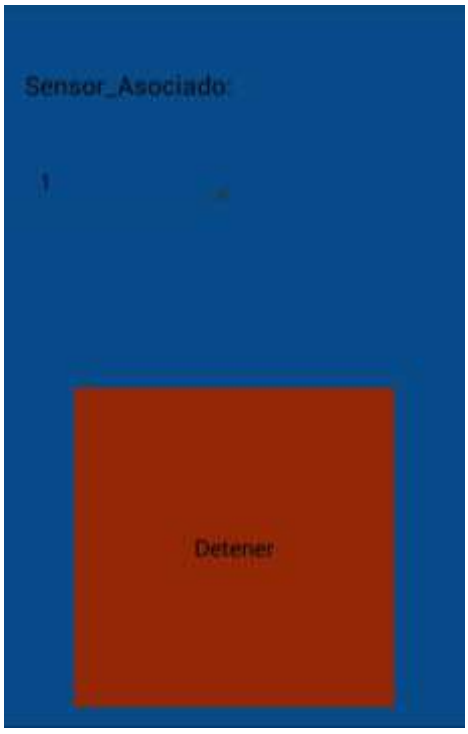


Figura 25.1. Interfaz para detener monitoreo.

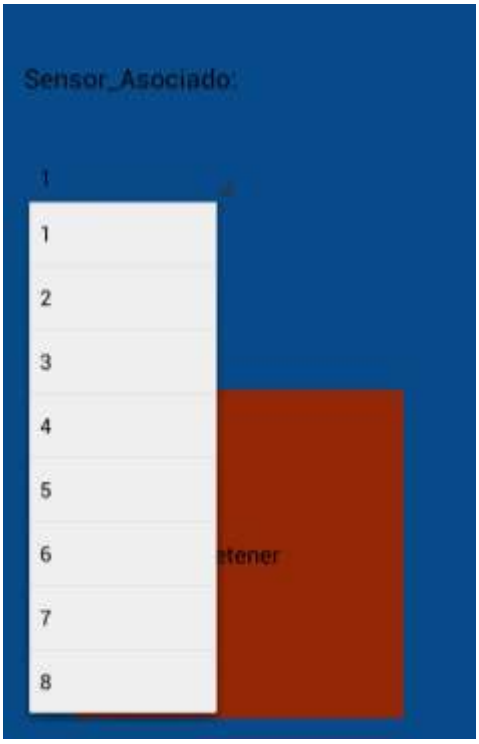


Figura 25.2. Spinner para elegir el sensor al que se detendrá el monitoreo.

3.2.3 Modulo sincronización y despliegue de datos

Se implementa un servicio web, el cual debe cumplir la tarea de sincronizar los datos ingresados del usuario con los respectivos datos recibidos y procesados de los sensores de pulsioximetría en la placa Raspberry. El servicio web se comunica con la placa electrónica Raspberry PI 3 y el dispositivo móvil por medio de la tecnología Wifi. Esta tarea de sincronización se realiza por una serie de pasos descritos a continuación.

Para la implementación del servicio web, se utiliza el entorno de desarrollo Netbeans IDE 7.1.1, debido a que permite la realización de proyectos web con muchas herramientas de desarrollo y se decide utilizarlo ya que se utilizó durante muchas materias cursadas durante la carrera de Ingeniería Electrónica y Telecomunicaciones para la realización de diferentes proyectos.

Descripción breve del servicio web

En un primer paso el servicio web recibe la información del usuario desde el dispositivo móvil que corresponde a nombre del usuario y sensor asociado, en un segundo paso el servicio web recibe los datos desde la placa electrónica Raspberry a través de una petición, que contiene los datos procesados de los sensores junto con la clasificación o nivel de Triage que representa. Como cuarto paso el servidor sincroniza la información enviada desde la placa Raspberry y el dispositivo móvil y la muestra en los dispositivos de despliegue.

El servidor web llamado, para este proyecto ServicioWebPaD (donde las siglas PaD vienen de **P**ulsioximetría **a** **D**istancia), es implementado en el lenguaje de programación Java, tomando como referencia el MVC (Modelo-Vista-Controlador)[31], donde el *modelo* está compuesto por una serie de clases[8] que gestionan variables temporales del sistema, la *vista* está compuesta por paginas dinámicas JSP (Java Server Pages) y el *controlador* que está compuesto por métodos web y Servlets para la gestión de las peticiones web.

3.2.3.1 Modelo

El modelo se compone de dos clases principales que permiten realizar una persistencia temporal, de los datos recibidos y desplegados por el sistema, las cuales son:

- Clase DatosUsuario
- Clase DatoSensor

Una clase en el lenguaje de programación Java, contiene un conjunto de variables y métodos en los cuales se puede realizar acciones sobre las variables. Estas clases son creadas teniendo en cuenta el diagrama de clases que se encuentra en el diseño de este proyecto

[8] Una clase java representa una plantilla para la creación de cada objeto necesario en el sistema.

Clase DatosUsuario:

Esta clase está conformada por 4 variables que permiten almacenar temporalmente los datos recibidos desde el dispositivo móvil: nombre, estado y sensor, y una variable cu[] de tipo DatosUsuario que contiene un arreglo para almacenar en ella los 8 posibles usuarios del sistema en un mismo tiempo. Además cuenta con métodos que permiten que esta clase pueda ser instanciada desde otras clases o paquetes del sistema como son un metodo constructor vacio y un constructor con variables para inicializar las variables de la clase, métodos get y set[9], método crearUsuario() y método userPos().

Clase DatosSensor:

La clase DatosSensor se conforma por 5 variables que permiten el almacenamiento temporal de los datos recibidos desde la placa Raspberry pi 3: sensor_rasp, sat_rasp, pulso rasp, nivel_triage y su[] que como en la anterior clase, representa un arreglo que permite almacenas temporalmente los datos de los ocho sensores. Así como la clase DatosUsuario cuenta con los métodos constructor vacío y un constructor con variables para inicializar las variables de la clase, métodos get y set para ser instanciada, método creaSensor() y método sensorpPos().

El código de las clases y su descripción se encuentran en el ANEXO B.1.

En las clases descritas anteriormente se habla de persistencia temporal, debido a que el sistema monitorea al paciente mientras espera su turno en la sala de espera de urgencias, y una vez es llamado para su respectiva atención, el sensor es retirado del usuario para monitorear a un nuevo usuario, por tanto, el sistema no tiene necesidad de realizar una persistencia de la medida de los signos vitales en una base de datos. Las variables locales almacenan los datos en memoria durante el tiempo en que el dato es recibido hasta que el dato es desplegado en los dispositivos, luego de esto el nuevo dato es recibido y se repite el proceso. Los datos que se reciben de los sensores realizan este proceso muy rápido debido a que los signos vitales pueden variar cada segundo, más las variables que almacenan los datos del usuario realizan este proceso más lento debido a que solo se modifican cuando se retira el sensor a un paciente y se le es colocado a un nuevo paciente.

3.2.3.2 Controlador

El control del sistema es realizado por medio de los Servlets y métodos web. Los servlets son programas de java que corren dentro de un servidor de aplicaciones java EE o contenedor. Cuando a un contenedor le llega una petición http, comprueba si hay algún servlet registrado para responder a dicha petición; en caso afirmativo, invocara dicho código [33]. Los métodos web funcionan de manera similar a los servlets, estos se encuentran dentro de un webservice, con el cual debe ser llamado.

[9] El método get en java, es usado extraer el valor de una variable privada desde otra clase. El método set se usa para almacenar un valor en una variable privada.

Para este proyecto se hace uso de los servlets para gestionar las peticiones que se realizan desde las páginas desde las que se hacen las pruebas a sistema, la inicialización de variables del sistema y para el despliegue en el televisor. Los métodos web son utilizados para la gestión de peticiones desde la su aplicación Android ubicada en el dispositivo móvil o Tablet, ya que a través de estos métodos web es posible devolver una respuesta a la petición realizada. A continuación se describen de forma general los servlets y métodos implementados en este proyecto. El código y su descripción más específica se encuentran en el ANEXO B.1.

Servlets:

- **TabletServlet:** Responde a las peticiones http realizadas por la página web de prueba en la que se introducen los datos que corresponden al nombre, sensor y estado que agregan un nuevo usuario al sistema, simulando el envío de datos desde el dispositivo móvil (Tablet).

- **RaspServlet:** Responde a las peticiones http realizadas por la página web de prueba en la que se introducen los datos correspondientes a sensor, saturación, pulso y nivel, simulando el envio de datos desde la placa electrónica Raspberry.

- **Inicializa:** Responde a las peticiones http realizadas por la página web que muestra al usuario como se van a inicializar los valores. Este servlet da un valor inicial a las variables del sistema con el fin de evitar errores o excepciones al ejecutar el servicio debido a que encuentre valores vacíos de las variables.

Métodos web:

Estos métodos son contenidos dentro clases, las cuales tienes el nombre de cada web service, como se describe a continuación:

- Web service WSGestorTablet, contiene dos métodos web utilizados para agregar a un usuario nuevo y detener su monitoreo, mediante las peticiones realizadas desde la aplicación Android del dispositivo móvil (Tablet). Contiene los siguientes métodos web:

    - **IniciarLectura:** Este método recibe los datos enviados desde el dispositivo móvil(Tablet), con los datos que corresponden a nombre, estado y sensor, y mediante estos datos realiza un llamado a la clase DatosUsuario para agregar al nuevo usuario a quien se le realiza el monitoreo, esto para cada usuario que se ingresa al sistema. Además devuelve a la Tablet un mensaje de confirmación.

    - **DetenerLectura:** Recibe el parámetro sensor enviado desde la Tablet, con el cual avisa al sistema cual es el usuario a quien ya no se debe monitorizar.

- Web service WSDespTablet, contiene 5 métodos web utilizados para llevar los datos desde el servicio web hacia el dispositivo móvil (Tablet) por medio de peticiones desde el mismo dispositivo. Los métodos son los siguientes:

    - **PresentarNom:** Método que recibe un parámetro desde el dispositivo móvil, que corresponde al sensor, debido a que se usa el número del sensor como posición en la variable que contiene el arreglo de datos del usuario, de esta manera, regresa el nombre solicitado mediante la petición.

    - **PresentarSat:** Recibe el datos sensor, usado para ubicar la posición del valor de la variable saturación en el arreglo de datos de los datos del sensor. Este método devuelve al dispositivo móvil el valor de saturación solicitado.

    - **PresentarPul:** Este método recibe desde el dispositivo móvil el valor del sensor, que se usa como posición para buscar en el arreglo de datos que contienen los datos del sensor el valor del pulso solicitado. Devuelve el valor del pulso al dispositivo móvil (Tablet)
      .
    - **PresentarNiv:** Recibe el parámetro sensor, con el cual ubica la posición en el arreglo de datos del sensor de la variable que contiene el nivel de Triage solicitado y devuelve este valor al dispositivo móvil (Tablet).

    - **Estado:** Método que recibe el parámetro sensor, para utilizarlo como posición y buscar en el arreglo de datos del usuario la variable estado, y retornar su valor al dispositivo móvil (Tablet).

- Web service WSGestorRasp, contiene un método web, encargado de recibir los datos enviados desde la placa electrónica Raspberry.

    - **subirDatos:** Este método recibe los parámetro enviados desde la placa electrónica Raspberry correspondientes a: sensor, saturación, pulso y nivel. El método se encarga de almacenar estos datos en las variables del sistema mediante una instancia a la clase DatosSensor.

Tanto en los servlets como en los métodos web, se menciona la variable *sensor* como posición, estos se hace debido a que en el sistema se usa un número del 0 al 7 para identificar cada uno de los 8 sensores y saber cuál se está utilizando, y a su vez, este mismo valor es usado para almacenar los datos en una posición especifica en una variable de tipo arreglo dentro del sistema, luego mediante estas posiciones poder sincronizar los datos obtenidos por la aplicación web y los datos obtenidos por la placa electrónica Raspberry. Así mismo se utiliza esta misma posición para consultar los datos y enviarlos a al dispositivo móvil (Tablet) o desplegarlos en el televisor a través de una página web.

3.2.3.3 Vista

La vista es conformada por lo que se despliega al usuario, en este caso se usan páginas estáticas html y paginas dinámicas jsp, contenidas dentro del servicio web. A continuación se muestra una breve descripción de lo que es html y jsp.

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado [34].

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

Para desplegar y correr JavaServer Pages, se requiere un servidor web compatible con contenedores servlet como Apache Tomcat o Jetty.

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, ASP o los CGIs, programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas.

Para empezar, los JSPs y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él. Cada servlet (o JSP, a partir de ahora lo usaremos de forma indistinta) se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa +intérprete). Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo [35].

Para este proyecto se generaron diferentes vistas, amigables al usuario, que permiten interactuar con facilidad con el sistema diseñado. Entre las vistas, se implementaron primero las paginas html, iniciando con la página principal llamada index.html, figura 26, creada para el administrador del servidor, la cual cuenta con accesos a varios servicios del sistema.

Figura 26. index.html

Se implementó esta página, con una pequeña descripción de lo que hace el servicio web y cuatro enlaces a los servicios que contiene, entre los que están: Inicializar variables de sistema, Página de Despliegue de datos y Pruebas de ingresos de datos al servicio (Dispositivos Android y Placa electrónica Raspberry / Sensor). Al dar click sobre cada uno de los enlaces se despliegan nuevas páginas web que dan acceso directo a los servicios web.

Al dar click sobre el primer enlace, se abre la página mostrada en la figura 27.1, de nombre inicializaDatos.html. Esta página se implementó con el fin de mostrarle al administrador los valores y la forma en que las variables del sistema serán inicializadas, además, en la parte inferior de la página se introdujo unas notas que dan información de los valores que se van a introducir. En la parte central de la página se localizó un botón de color gris, con el cual se da inicio al proceso de inicializar las variables.

Una vez inicializadas, se programó la página para que muestre una ventana pequeña emergente, figura 27.2, en la que se confirma que se inicializo las variables con éxito y al darle aceptar a esta pequeña ventana se regresa a la página principal.

Inicializacion de datos

Los datos con los que se inicializaran las variables locales son los siguientes:

Sensor	Usuario	Estado	Saturacion	Pulso	Nivel
0	Usuario1	0ff	00	00	cero
1	Usuario2	0ff	00	00	cero
2	Usuario3	0ff	00	00	cero
3	Usuario4	0ff	00	00	cero
4	Usuario5	0ff	00	00	cero
5	Usuario6	0ff	00	00	cero
6	Usuario7	0ff	00	00	cero
7	Usuario8	0ff	00	00	cero

Nota (Posiciones):

La varible que corresponde al sensor se inicializa de 0 a 7 y no de 1 a 8, debido a que se utiliza la variable sensor como posiciones de variables locales con el fin de sincronizar los valores que son enviados del dispositivo movil Android con los valores enviados de la placa electronica Raspberry.

Nota (Nivel):

El servicio web trabaja con tres niveles de triage. Se hace referencia al nivel "uno", el cual representa nivel Triage I, la variable nivel "dos", el cual representa nivel de Triage II, la variable nivel "tres", el cual representa un nivel normal de signos para el Triage y la variable nivel "cero", que representa la inactividad del sensor (sensor no se encuentra transmitiendo o tamando datos.)

Figura 27.1. inicializaDatos.html

Figura 27.2. Ventana de mensajes de confirmación de inicialización de variables.

Al dar click sobre el segundo enlace, se despliega una nueva página llamada index.jsp, figura 28, una página web dinámica que interactúa con los datos de usuario y sensor a través de un servlet, funcionando así como página de despliegue de datos.

Figura 28. index.jsp

Esta página de despliegue de datos contiene en su parte central, la información del usuario y sus signos de saturación de oxígeno y pulso cardiaco, además el sensor que está usando y su nivel de Triage. Se programó la página para que se vayan mostrando los datos a medida que se van agregando usuarios y monitoreando sus signos, por otro lado, está programada para mostrar mediante colores el estado de cada paciente teniendo en cuenta los colores usados para la clasificación del Triage de la siguiente manera.

- Color gris: La fila tiene un fondo de color gris, figura 28, cuando no se ha agregado un usuario para su lectura y pueda iniciar su monitoreo.

- Color verde: La fila de datos se coloca con un fondo verde, figura 29.1, cuando se ha iniciado el monitoreo. También se usa este color de fondo para indicar una clasificación de Triage de nivel tres (Clasificación de Triage III) o superior, indicando que los signos del paciente están en un rango normal o que no necesitan de una atención inmediata.

- Color naranja: La fila de datos se coloca con un fondo de color naranja, figura 29.2, cuando los signos del paciente se alteran, saliendo de los rangos considerados normales indicando una clasificación de Triage de nivel dos (Clasificación de Triage II), lo cual indica al personal médico que es una paciente al cual se debe tener mayor cuidado debido a que puede presentar un posible empeoramiento repentino de su salud.

- Color rojo: La fila de datos se coloca con un fondo de color rojo, figura 29.3, cuando los signos del paciente se alteran hasta llegar a un punto grave, lo que indica una clasificación de Triage de nivel uno (Clasificacion de Triage I), estado en el cual el paciente no puede esperar y debe ser atendido de manera inmediata ya que su vida está en riego, además de la alarma sonora, que hace que se preste mas atención a lo que le está sucediendo a la persona.

Figura 29.1. Fila de color verde indicando signos normales sin alarma.

Figura 29.2. Fila de color naranja indicando una primera alarma de nivel II.

Figura 29.3. Fila de color rojo indicando una alarma nivel I.

Teniendo en cuenta que esta página es utilizada para desplegar los datos en el televisor y este es ubicado para las pruebas en la estación de enfermería, lugar en el cual se encuentra el personal de enfermería y personal médico, se dejó un espacio en el lado derecho, para colocar los diez derechos de los pacientes considerando su importancia y diez deberes de enfermería para la administración de medicamentos.

Continuando con la página principal para el administrador, vista en la figura 26, si se da click sobre el tercer enlace (Dispositivo Android), se abrirá una nueva página web llamada PruebaTablet.html, figura 30.1, que contiene varios campos de texto, en los que se ingresan los datos que enviaría la Tablet, que son: nombre, estado y sensor, simulando la petición que hace la aplicación Android desde la Tablet al servicio web. Cabe aclarar, que en la aplicación, como ya se mostró el usuario ingresa solo el nombre y el sensor y no el estado, debido a que se programó la aplicación Android para que envíe este dato automáticamente al servicio web una vez el usuario presiona el botón verde iniciar. En la página de prueba el estado si se ingresa manualmente permitiendo al administrador detener la lectura de un usuario ingresando un estado de apagado (off).

El estado funciona de la siguiente manera:

Estado	Descripción
on	Estado de encendido. Cuando la variable toma este valor, se agrega el usuario al sistema para que inicie su lectura, y se sincronice con los datos que vienen de los

	sensores desde la placa electrónica Raspberry.
off	Estado de desconectado. Cuando la variable estado toma este valor desconecta del sistema al usuario y se colocan todas las variables que estaban asociadas a él, a sus valores por defecto.

Tabla 6. Estados de usuario.

En la parte inferior de los campos de texto, se encuentra un botón con texto *Enviar,* que al ser presionado está programado para enviar los datos al servicio web, que son recibidos por un servlet y almacenados en las variables del sistema, simulando el envío de datos desde el dispositivo móvil (Tablet). Finaliza este proceso con un mensaje de confirmación de que los datos fueron enviados de forma correcta, figura 30.2, y regresa a la página principal del administrador.

Figura 30.1. PruebaTablet.html

Figura 30.2. Mensaje de confirmación de ingreso de datos de prueba de envío desde Tablet.

En la anterior imagen se puede observar un ejemplo de datos ingresados con los valores: nombre = diego, estado = on, sensor = 0. Con sensor 0 se hace referencia al sensor numero 1 (teniendo en cuenta que la aplicación móvil maneja los sensores de 0 a 7).

El cuarto enlace de la página principal del administrador, figura 26, despliega una nueva página web llamada PruebaRasp.html, figura 31, se implementó esta página con campos de texto en los cuales se ingresan los datos que se serían enviados desde la placa electrónica Raspberry, con el fin de probar la respuesta del servicio web. Los datos que se ingresan son: sensor, saturación, pulso y nivel. Estos son los datos que envía normalmente y automáticamente la placa electrónica Raspberry al servicio web. En la parte inferior de los campos de texto se ubicó un botón con el texto *Enviar,* que se programó de forma análoga al botón de la página de prueba anterior, y de la misma forma una vez confirmado que se ingresaron con éxito los datos, vuelve a la página principal del administrador.

Figura 31. PruebaRasp.html

El código de las páginas web html y jsp mencionadas anteriormente y su descripción se encuentra en el ANEXO B.1.

CAPÍTULO 4

4. Pruebas, conclusiones y trabajos futuros

En este capítulo se describe el modo en que se realizaron las pruebas y lo necesario para estas. Las pruebas están divididas en dos secciones: Las pruebas en un ambiente simulado y las pruebas en un ambiente real.

4.1 Prueba en ambiente simulado

La prueba en ambiente simulado se realiza con la finalidad de probar que el sistema funciona de manera adecuada, en condiciones ideales. El sistema descrito a lo largo de este documento, está diseñado para recibir la información proveniente de 8 sensores de pulsioximetría inalámbricos, que se comunican a través de tecnología Bluetooth al mismo tiempo y procesar su información generando alarmas audiovisuales cuando sea necesario.

Al inicio del proyecto se intentó conseguir financiamiento para la compra de todos los sensores ya que su precio es elevado y lograr realizar pruebas con el sistema completo, pero no fue posible encontrarla debido a que es un proyecto de tipo social y no genera ningún tipo de retribución económica. A partir de este problema nace la idea de realizar la prueba en un ambiente simulado que se describe a continuación.

Fue necesario adquirir al menos un sensor de pulsioximetría, con el fin de comprobar su conexión con la placa electrónica Raspberry Pi 3, y la forma del envío de datos hacia ella y lograr comprobar que el sistema si se puede comunicar con el sensor. Para esto, se demuestra como el sistema detecta al sensor, figura 32; una vez detectado, se pasa a la lectura de los datos provenientes del sensor, figura 33.

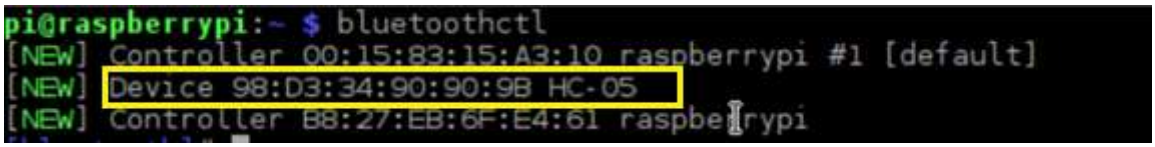


Figura 32. Detección del sensor.

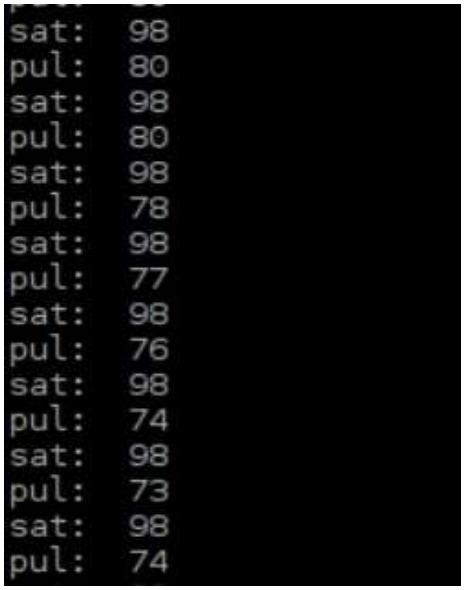


Figura 33. Lectura de datos desde el sensor.

Si bien con esto se demuestra que el sistema completo se comunica con el sensor de pulsioximetría inalámbrico Bluetooth de manera correcta, se debe probar que el sistema responde a un máximo de 8 sensores al mismo tiempo. Al tener un solo sensor para el proyecto, se decide simular los demás sensores a través de una aplicación móvil Android.

Teniendo en cuenta que el sensor envía datos a través de Bluetooth, correspondientes a saturación de oxígeno y pulso cardiaco, se realizó una aplicación Android que simula el envío de datos de saturación de oxígeno y pulso cardiaco a través de Bluetooth, simulando ser un sensor de pulsioximetría inalámbrico que está monitorizando a una persona y enviando datos Bluetooth a la paca Raspberry. Al conectarse la aplicación Android con la placa electrónica Raspberry, la placa asimila que está recibiendo datos de un sensor.

Una de las ventajas de utilizar sensores simulados, es que se puede recrear la situación de una alarma, que para el caso del sensor real se debería encontrar una persona que esté en un estado grave, lo cual haría más complicada la prueba de las alarmas.

Se diseñó la aplicación de prueba que simula un sensor de pulsioximetría, con una interfaz, figura 34, con nombre SimuladorPO (PO, iniciales de PulsiOximetro).

Figura 34. Interfaz principal de aplicación Android que simula un sensor de pulsioximetría inalámbrico.

Se ubicó en la interfaz de la aplicación 4 botones y 2 espacios para información de los datos que funcionan de la siguiente manera:

- En la parte izquierda de la interfaz de la aplicación se puede notar un cuadro de color gris, acompañado del título Saturación y de un símbolo de porcentaje (%), figura 35. Este cuadro de color gris tiene la función de indicar los valores de saturación que se están enviando a través de Bluetooth hacia la placa electrónica Raspberry, de la misma manera que el sensor indica a través de su pequeña pantalla los datos medidos y enviados por Bluetooth. Cuando no se están enviando datos el cuadro muestra el valor “00” y permanece de color gris, simulando que el sensor está apagado o sin monitorizar a nadie.

Figura 35. Valores de saturación enviados mediante Bluetooth.

- En la parte inferior de la interfaz de la aplicación se puede apreciar un segundo cuadro de color gris, acompañado del título Pulso (Haciendo alusión a la medida del pulso cardiaco) y de un símbolo de pulsaciones por minuto (PPM), figura 36. Este cuadro de color gris tiene la función de indicar los valores de pulso cardiaco que se están enviando simulando la transmisión Bluetooth hacia la placa electrónica Raspberry, de la misma manera que el sensor indica a través de su pequeña pantalla los datos medidos y enviados por Bluetooth. Cuando no se están enviando datos el cuadro muestra el valor “00” y permanece de color gris, simulando que el sensor está apagado o sin monitorizar a nadie.

Figura 36. Valores de pulso enviados mediante Bluetooth.

- Botón Iniciar: Este botón fue programado, para que una vez sea presionado, inicie la simulación de la transmisión Bluetooth de los datos aleatorios de saturación de oxígeno y pulso cardiaco hacia la placa Raspberry, en un rango normal o nivel de Triage tres (Clasificación de Triage III). Esta transmisión se realiza mediante la activación de una variable en el servicio web, la cual es detectada por la placa y se hace el inicio de la transmisión de la placa hacia el servicio web de los signos

aleatorio. Al mismo tiempo la aplicación está programada para mostrar los datos que se están enviando, colocándolos en los cuadros de información vistos, además de cambiar el color del cuadro a un fondo verde, indicando que son datos de rangos normales que no generan alarmas en el sistema, figuras 37.1, 37.2, 37.3. Se envían datos aleatorios en un rango normal, para comprobar que el sistema funciona ante los cambios de los valores de los signos enviados por los sensores.

Figura 37.1

Figura 37.2

Figura 37.3

Como se puede observar en las anteriores imágenes (Tomadas en diferentes tiempos), al presionar el botón Iniciar se envían datos aleatorios de saturación y pulso.

- Los botones con nombre Triage I y Triage II, simulan el envío de datos que están en los rangos para generar una alarma de Triage nivel uno y Triage nivel dos respectivamente. A presionar el botón Triage II, el cuadro de información de valores en la interfaz de la aplicación, se coloca de fondo naranja, indicando la alarma de Triage nivel dos (Clasificación de Triage II) y así mismo se envían datos de forma aleatoria, figuras 38.1, 38.2, 38.3. Si se presiona el botón Triage I, el cuadro de información de valores en la interfaz de la aplicación, se coloca de color rojo, indicando la alarma de Triage nivel uno (Clasificación de Triage I), enviando datos en forma aleatoria, figuras 39.1, 39.2, 39.3.

Figura 38.1.

Figura 38.2.

Figura 38.3.

Figura 39.1.

Figura 39.2.

Figura 39.3.

Una vez realizada la aplicación Android (Código Android y descripción en ANEXO B.3), se procede a la realización de la prueba simulada, realizada en el laboratorio de Telemática de la Universidad del Cauca, figura 40, ya que el laboratorio cuenta con lo necesario para realizar la prueba simulada. Para testificar la realización de la prueba, se encuentra presente en el laboratorio el Ingeniero Civil Andres Fernando Benavides Ruiz, identificado

con cedula 1.061.727.521, contratista de la Universidad del Cauca, quien da testimonio mediante el documento firmado que se presenta en el ANEXO E, a quien agradecemos su colaboración. A continuación se lista lo que se necesita para la prueba:

- Red de área local que permite la conexión wifi.
- Equipo que funciona como servidor de aplicaciones web.
- Placa electrónica Raspberry Pi 3.
- Sensor de pulsioximetría inalámbrico Bluetooth.
- Dispositivos móviles Smart Phone.
- Aplicación Android de prueba instalada sobre los dispositivos móviles.
- Televisor Smart con conexión a Wifi.

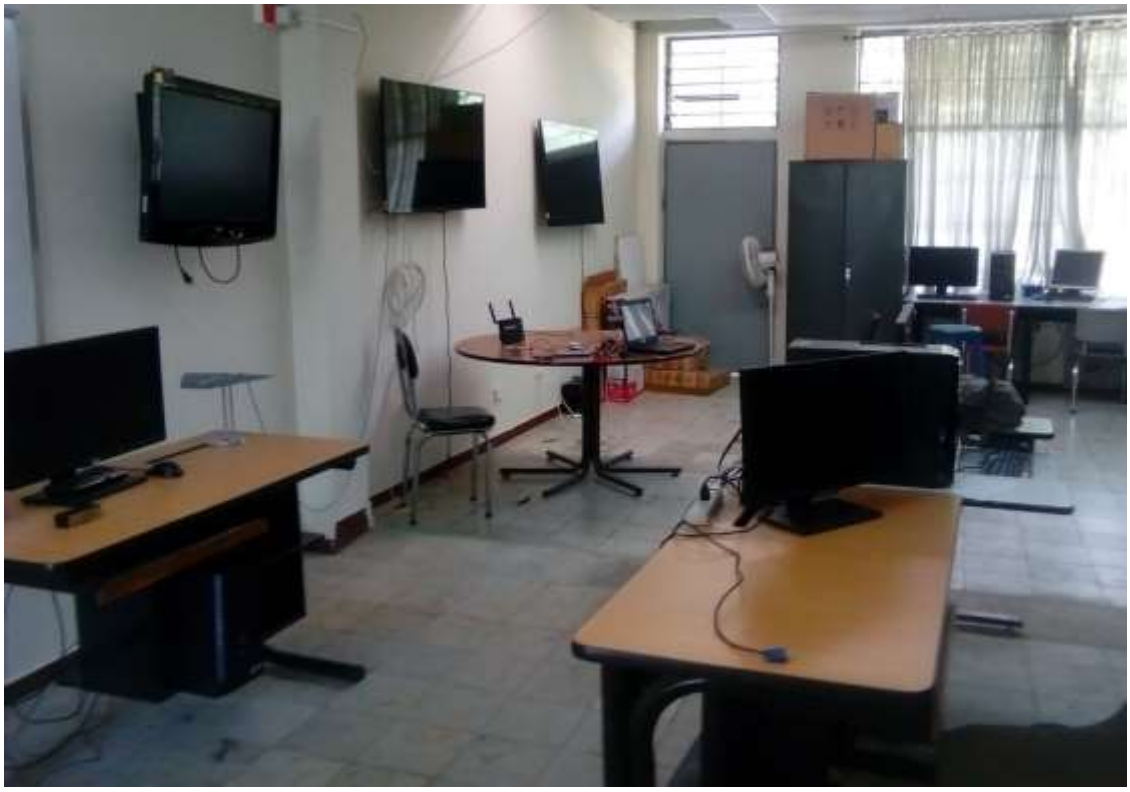


Figura 40. Laboratorio de Telemática – Universidad del Cauca.

Para colocar el sistema en funcionamiento, se ejecutó el servicio web, se inicializaron las variables, luego se ingresó a la página de despliegue desde el televisor, figura 41, se inició el sistema de la placa electrónica Raspberry junto con el programa para la lectura de los sensores y procesamiento de sus datos, por último se agregó todos los sensores que intervienen en la prueba (Agregar sensores al sistema, ANEXO A), tanto el sensor real como los simulados con la aplicación Android. De esta manera el sistema queda listo para comenzar con el monitoreo.

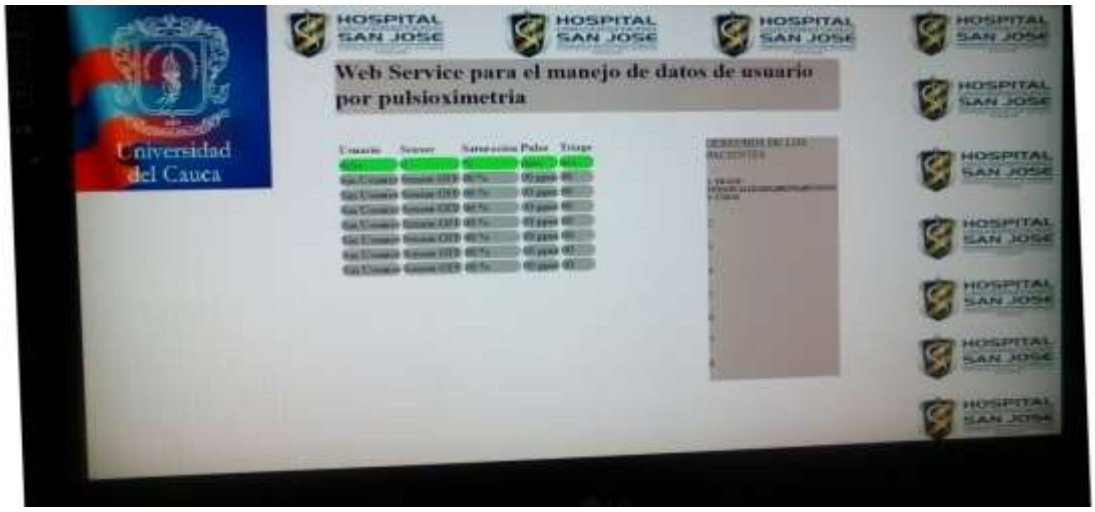


Figura 41. Página de despliegue iniciada en el televisor.

Se inicia agregando un nuevo usuario al sistema a través de la aplicación Android para su monitoreo, figura 42.1, luego se ubica el sensor de pulsioximetría en el dedo de una de los presentes en la prueba de simulación, figura 42.2, así mismo una vez puesto el sensor se puede apreciar como comienzan a aparecer los datos en el televisor y en la interfaz principal de la aplicación, figuras 43.1 y 43.2, mostrando como sincroniza los datos el sistema.

Figura 42.1.

Figura 42.2.

Figura 43.1.

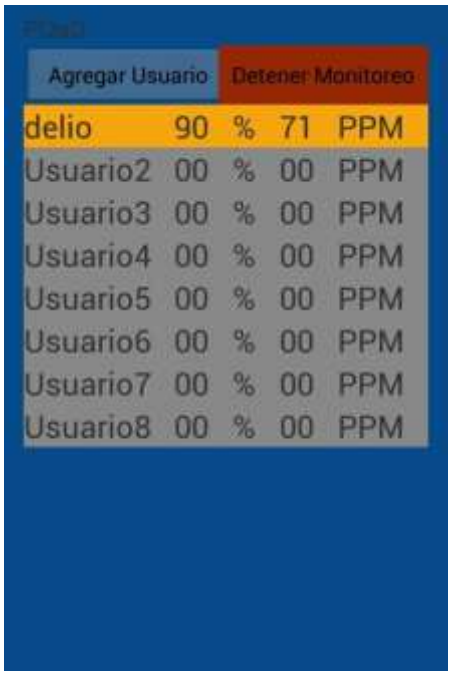


Figura 43.2.

De forma análoga, se ingresa un segundo usuario al sistema y se inicia la transmisión desde la aplicación Android hacia la placa Raspbery y estos se puede observar en el televisor y en la página principal de la aplicación Android, figuras 44.1 y 44.2.

Figura 44.1.

Figura 44.2.

Para probar como genera las alarmas el sistema, se presiona el botón Triage II en la aplicación Android para que envíe datos de signos más bajos hacia la placa y se puede apreciar cómo cambia el color tanto en lo que se muestra en el televisor, figura 45.1, como en la interfaz principal de la aplicación Android, figura 45.2. indicando que están disminuyendo los signos, generando una primera alarma.

Figura 45.1.

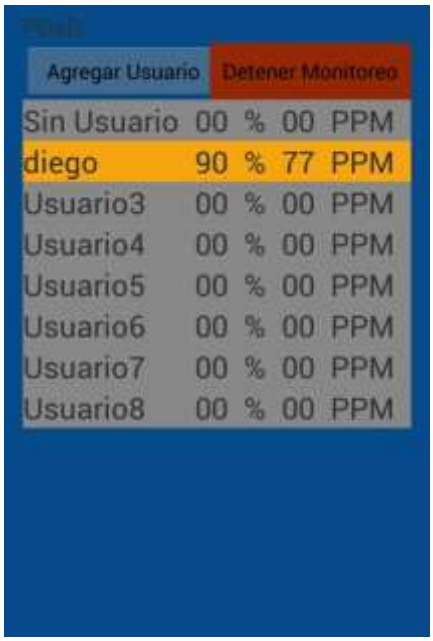


Figura 45.2.

Se presiona a continuación el botón Triage I, en la aplicación Android, con lo cual la aplicación comienza a enviar datos muy bajos de los signos de saturación de oxígeno y pulso cardiaco, simulando una persona que está demasiado grave y peligro de muerte. Así mismo, se observa como el sistema responde desplegando la alarma en forma de color rojo junto con un sonido de alarma tanto en televisor como en la aplicación Android, figuras 46.1 y 46.2. Con esto se comprueba la generación de las alarmas visuales y sonoras automáticas.

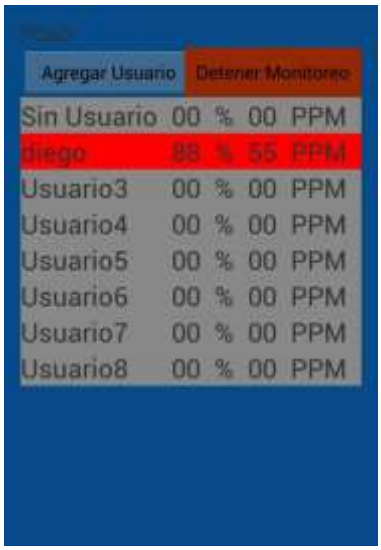
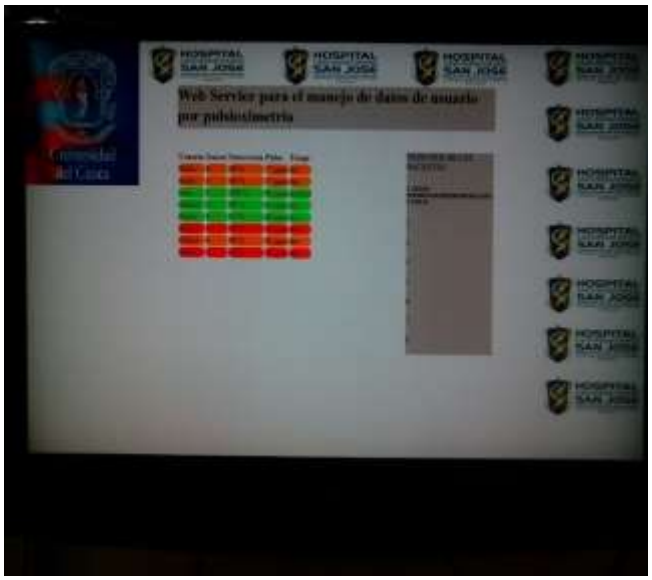


Figura 46.1. Figura 46.2.

Para finalizar la prueba simulada, se agregan todos los usuarios, junto con todos los dispositivos que simulan los sensores, al sistema y se observa tanto en la pantalla del televisor, figura 47.1, como en la interfaz principal de la aplicación Android, figura 47.2, como aparecen todos los usuarios junto con sus signos sincronizados. Tambien se hace uso de un ingreso de datos reales pero de forma manual al sistema para hacer pruebas de funcionamiento con mas sensores.



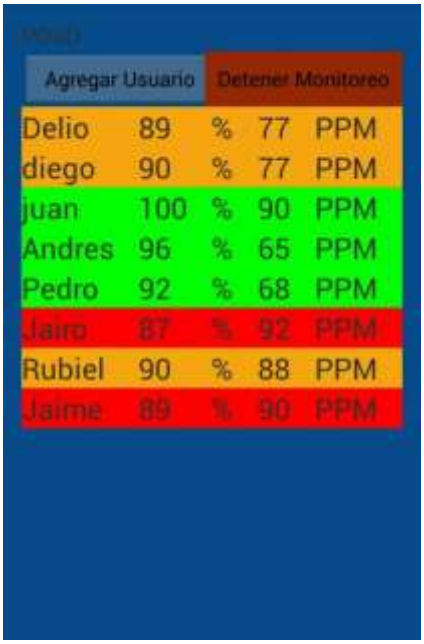


Figura 47.1. Figura 47.2.

Así, se da por cumplida la prueba simulada del sistema, logrando verificar que el sistema cumple con las condiciones de funcionamiento de recepción de datos y generación de alarmas visuales y sonoras, dando paso a la prueba en un ambiente real.

4.2 Prueba en ambiente real

La prueba que corresponde a un ambiente real, se realizó en el hospital Universitario San José de Popayán, en la sala de espera de urgencias. Para que el hospital permitiera realizar la prueba, hubo la necesidad de enviar una solicitud antes de iniciar con este proyecto por medio de un informe, al cual, el hospital respondió dando el aval correspondiente, ANEXO G, para realizar las pruebas necesarias.

La prueba se realiza de manera similar a lo que se realizó en la prueba en ambiente simulado. Lo necesario para realizar la prueba en el hospital, se lista a continuación:

- Servidor de aplicaciones web.
- Red de área local con conexión wifi.
- Placa Raspberry Pi 3.
- Televisor Smart con conexión wifi.
- Sensor de pulsioximetría inalámbrico Bluetooth.
- Dispositivo móvil (Tablet), con aplicación Android principal instalada.

Se contó con la colaboración del doctor Arnulfo Orobic, Urgentólogo del Hospital Universitario San José de Popayán, con numero de cédula 10.546.896, quien testifico la realización de las pruebas en ambiente real realizadas en la sala de espera de urgencias, mediante el documento firmado que se encuentra en el ANEXO E, a quien agradecemos su colaboración.

Como primer paso, se ejecutó servicio web realizado en este trabajo en el servidor webl, y luego fue ejecutado y probado por medio de las páginas de prueba del servicio web. Además, se inicializaron todas las variables del sistema.

Se ubicó el televisor en frente del área de la estación de Triage como se puede apreciar en la figura 48, con el fin de no interrumpir los procedimientos realizados en esta área y se accedió a la página web de despliegue de datos, figura 49,. Luego, se ubicó la placa electrónica Raspberry en la zona central de la sala de espera de urgencias y se inició el programa que recibe los datos de los sensores, agregando el sensor de pulsioximetría. De esta manera el sistema está listo para iniciar la prueba en ambiente real.

Figura 48. Ubicación del televisor en la estación de enfermería.

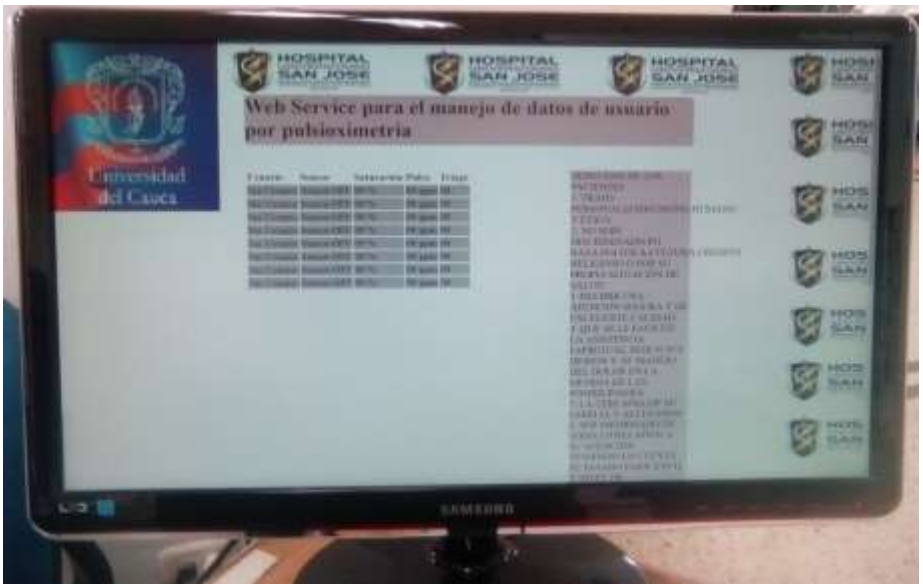
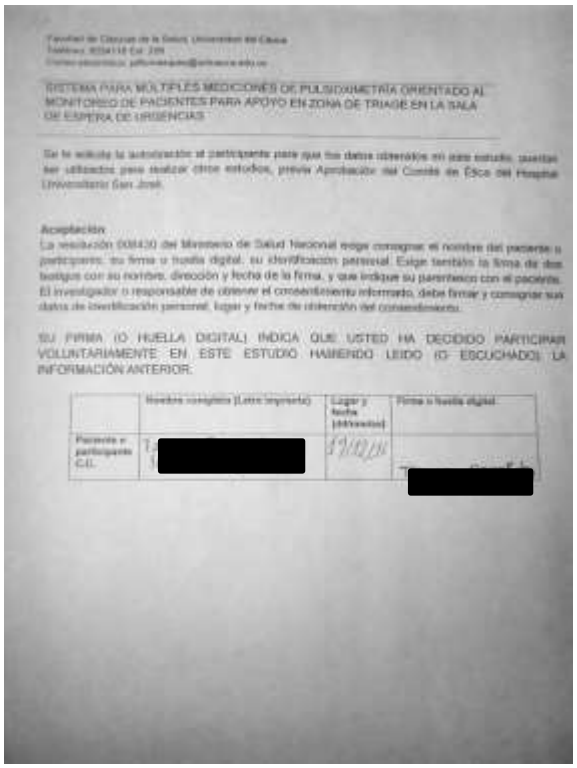


Figura 49. Página principal de despliegue de datos.

Se eligió un paciente de la sala de espera de urgencias, al cual, primero se le hace entrega del documento Consentimiento informado, ANEXO D, en el cual se explica brevemente en lo que consiste el proyecto, la prueba y las complicaciones. Este

consentimiento informado debe ser firmado por el paciente para poder hacer parte de la prueba, figura 50.



SISTEMA PARA MÚLTIPLES MEDICIONES DE PULSIOXIMETRÍA ORIENTADO AL MONITOREO DE PACIENTES PARA APOYO EN ZONA DE TRIAGE EN LA SALA DE ESPERA DE URGENCIAS

Se le solicita la autorización al participante para que los datos obtenidos en este estudio, puedan ser utilizados para realizar otros estudios, previa Aprobación del Comité de Ética del Hospital Universitario San José.

Aceptación

La resolución 008430 del Ministerio de Salud Nacional exige consignar el nombre del paciente o participante, su firma o huella digital, su identificación personal. Exige también la firma de dos testigos con su nombre, dirección y fecha de la firma, y que indique su parentesco con el paciente. El investigador o responsable de obtener el consentimiento informado, debe firmar y consignar sus datos de identificación personal, lugar y fecha de obtención del consentimiento.

SU FIRMA (O HUELLA DIGITAL) INDICA QUE USTED HA DECIDIDO PARTICIPAR VOLUNTARIAMENTE EN ESTE ESTUDIO HABIENDO LEIDO (O ESCUCHADO) LA INFORMACIÓN ANTERIOR.

	Nombre completo (Letra imprenta)	Lugar y fecha (dd/mm/aa)	Firma o huella digital
Paciente o participante C.C.	T. [redacted]	[redacted]	[redacted]

Figura 50. Firma del consentimiento informado.

Una vez firmado el documento se agrega al paciente por medio de la aplicación Android instalada en el dispositivo móvil (Tablet) y se coloca el sensor en el dedo del paciente, figura 51, momento en el cual los datos empiezan a aparecer en la interfaz principal de la aplicación Android, figuras 52.1, asi mismo en la pantalla, comprobando el funcionamiento en un ambiente real. Luego de monitorizar al paciente hasta que es llamado a la sala de Triage, se le quita del dedo el sensor de pulsioximetría, terminando el proceso de monitoreo y se detiene la lectura por medio de la aplicación Android. Para terminar la prueba con el paciente se le hace una breve encuesta acerca del proyecto, ANEXO F.

Figura 51. Colocación del sensor al paciente.

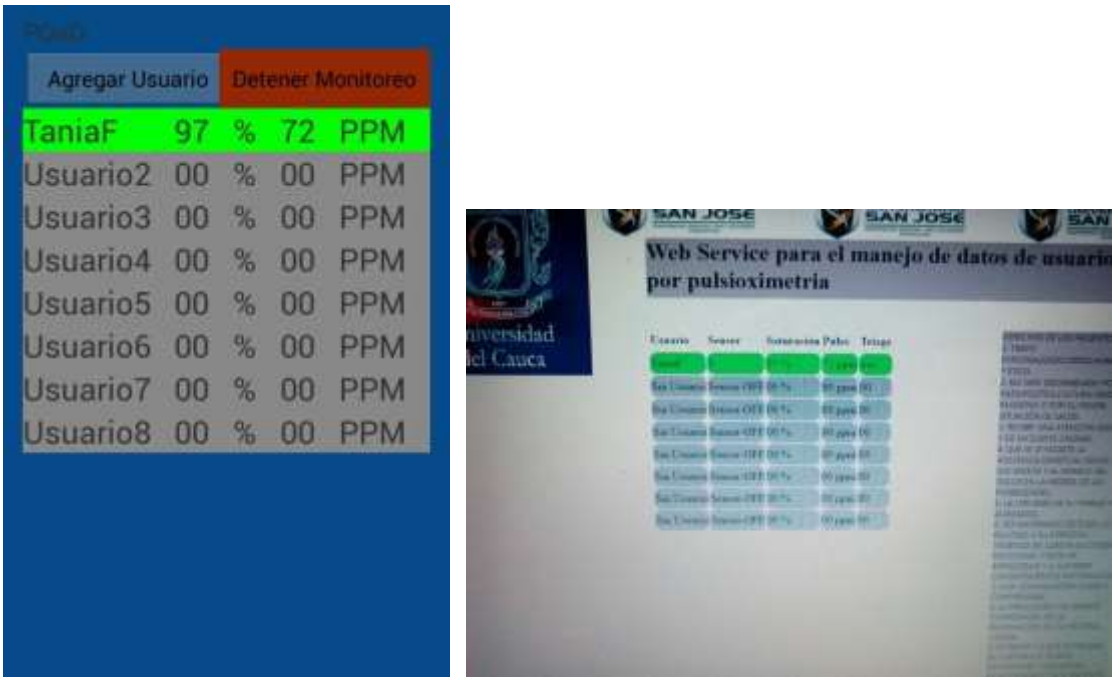


Figura 52.1

Esta misma prueba se repite a 30 pacientes de la sala de espera de la misma manera, en diferente horarios de atención del hospital.

Resultados de la Encuesta Realizada a los Pacientes.

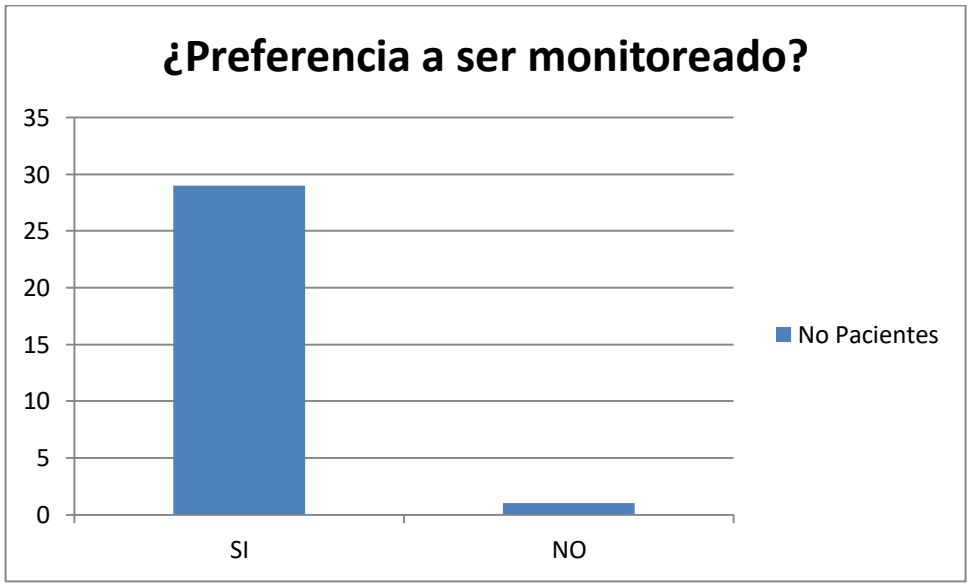


Figura 53.1. Primera pregunta

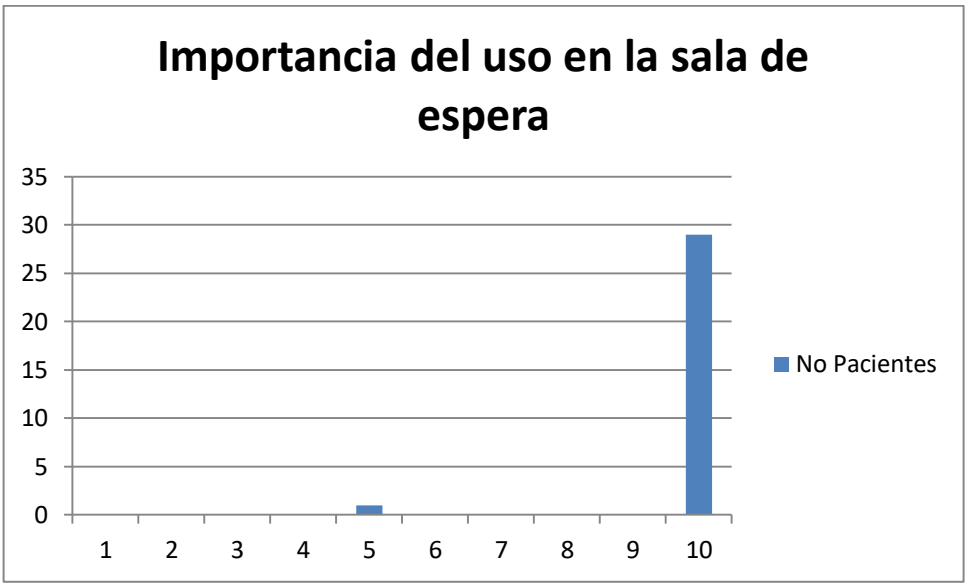


Figura 53.2. Segunda pregunta

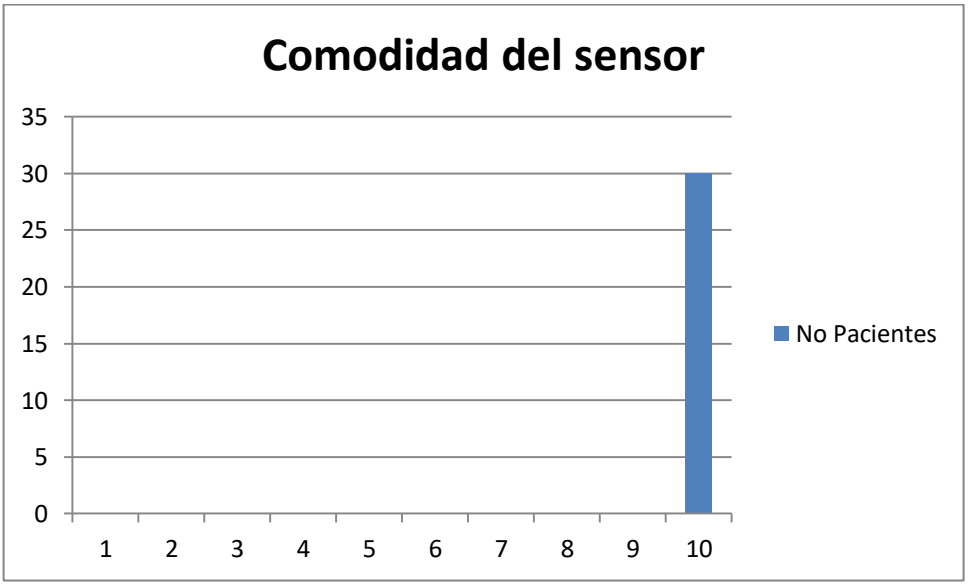


Figura 53.3. Tercera pregunta

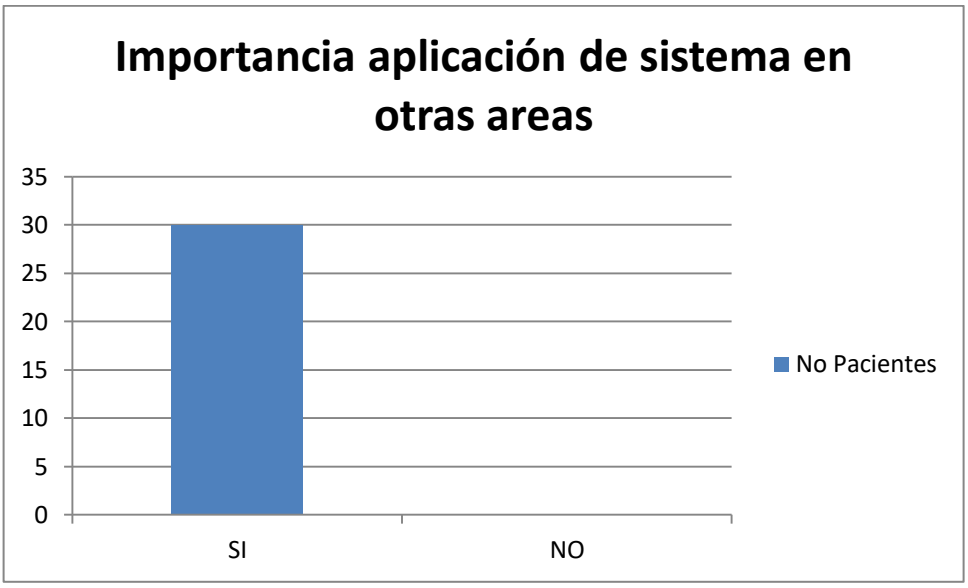


Figura 53.4. Cuarta pregunta.

Del anterior grafico se observa la aceptación de los pacientes al proyecto, además de mejorar la confianza en la atención prestada por el hospital hacia ellos. Se notó que de los 30 pacientes participantes en la prueba, solamente uno respondió que no le gustaría ser monitoreado constantemente durante el tiempo de espera y le daba importancia baja al sistema, los 29 pacientes restantes mostraron total aceptación ante el sistema como se observa en las gráficas, lo que indica que el sistema fue altamente aceptado por los pacientes en la sala de espera de urgencias del hospital.

Con la prueba en ambiente real, finaliza el proceso de pruebas del sistema habiendo obtenidos buenos resultados y buena aceptación por parte de los pacientes que participaron en la prueba en ambiente real. Un video realizado durante las pruebas en ambiente simulado se anexa en el CD entregado junto con este trabajo. Con el cumplimiento de las pruebas, se da por completo este trabajo de grado.

4.3 Conclusiones

La información de ocho sensores de pulsioximetría (saturación de oxígeno, pulso e identificador del sensor) se obtuvo de manera inalámbrica y en tiempo real utilizando como transmisor, tecnología Arduino y comunicación Bluetooth. La identificación del sensor se asignó a partir de la dirección MAC generada por el modulo Bluetooth, dando un identificador a cada sensor entre 0 y 7. La recepción y procesamiento de los datos se realizó utilizando una tarjeta Raspberry Pi 3, la cual recibió la información por medio de su módulo Bluetooth.

A partir de los datos obtenidos se realizó una reclasificación, cada dos segundos del estado del paciente, basados en parámetros de la norma actual de Triage. Durante la realización de este trabajo, inicialmente se planteó utilizar sensores de pulsioximetría

comerciales con tecnología Bluetooth incluida. Sin embargo, no fue posible debido a que estos contienen código cerrado y por tanto fue necesario utilizar un prototipo que permitiera la adquisición de los datos. Al liberar el código de lectura de los pulsioxímetros comerciales, con tecnología Bluetooth harían más sencillo el desarrollo de proyectos similares por cuanto mejoraría su portabilidad, comodidad para el paciente y además facilitaría su implementación

La sincronización de los datos provenientes de la tarjeta Raspberry Pi 3(Saturación de oxígeno, Pulso, clasificación de Triage, identificación del sensor), y los provenientes de la aplicación móvil (Datos del paciente: Nombre e identificador del sensor asignado) se realizó utilizando el identificador del sensor y el protocolo SOAP por medio de un servicio web basado en el modelo vista-controlador (MVC) que permitió observar los datos a través de páginas web (html y jsp) en monitores o Smart TV ubicados en las salas de urgencias, así como también en la pantalla de la aplicación móvil mediante el protocolo SOAP.

El sistema fue comprobado en condiciones de laboratorio y en un ambiente clínico: En ambiente simulado, mediante la App móvil de prueba SimuladorPO, la cual permitió comprobar la respuesta del sistema en términos de lectura, procesamiento de los datos y generación de alarmas (Reclasificación de Triage), inducidas por varios sensores de pulsioximetría. El sistema también fue probado directamente en la sala de urgencias del Hospital Universitario San José de Popayán, previo aval ético, permiso y consentimiento informado. En esta prueba participaron 30 paciente voluntarios que estaban presentes en la sala de espera, próximos a recibir la valoración médica definitiva. Al comienzo se obtuvieron algunas objeciones por parte de algunos pacientes a los cuales se les solicitó su participación; sin embargo, posteriormente decidieron participar en esta prueba. Es probable que esta conducta sea debida a la facilidad de aplicación y que no presentaba incomodidad. Durante la prueba los signos de los pacientes no presentaron alteraciones sensibles en su estado de salud que generaran la reclasificación del estado del Triage y, por ende, de las alarmas del sistema.

Se logró diseñar e implementar un sistema de soporte en la clasificación de Triage en un servicio de urgencias, que permitió monitorear, en tiempo real, variables fisiológicas como saturación de oxígeno y el pulso de múltiples pacientes presentes en la sala de espera, utilizando sensores inalámbricos de pulsioximetría. Los resultados de esta prueba piloto, demostraron la confiabilidad del sistema para realizar el monitoreo constante de los pacientes durante el tiempo de espera dentro del servicio de urgencias del hospital.

Si este tipo de dispositivos son implementados, es posible reducir las complicaciones médicas que pueden aparecer durante el tiempo de espera de atención médica. Este sistema, además de ser bajo costo y seguro para el paciente, es de fácil implementación en las áreas de urgencias.

4.4 Trabajos futuros

Teniendo en cuenta el proyecto realizado y los resultados obtenidos, debe constituirse como referencia para trabajos futuros, que busquen implementar nuevos sistemas de monitoreo más completos y con más innovación. Los trabajos futuros propuestos son los siguientes:

Aumentar la muestra de la población de pacientes monitorizados incluyendo diferentes grupos etarios para mejorar la validez interna del trabajo.

Adicionar más sensores para obtener datos de otros signos vitales, con el fin de hacer un monitoreo completo en tiempo real, y poder llevar estos datos a historias clínicas digitales.

Hacer un diseño y desarrollo propio, que disminuya los costos de instrumentación. Teniendo en cuenta que es posible realizar la lectura del sensor mediante una placa Arduino nano o micro directamente, obviando la placa eHealth..

Hacer un estudio de usabilidad por parte del cuerpo médico del servicio de urgencias.

Desarrollo de sistemas de bajo costo aplicados a salud, con el fin de cumplir el objetivo de que todos muramos de viejos.

REFERENCIAS

[1] Noticias RCN, [Online], Ultimo Acceso: 4 de Junio de 2015 http://www.noticiasrcn.com/nacional-regiones-centro/joven-grabo-falta-atencion-medica-morir

[2] Manosalva Murillo, J. (2005). Rol del enfermero en el área de triage. *Av. enferm*, *23*(1), 82-89.

[3] "Triage: atención y selección de pacientes", Dr. Gerardo José Illescas Fernández, Vol. 9, No. 2, Mayo-Agosto de 2006, Medigraphic Artemisa en línea.

[4] "SMA Bluetooth® Wireless Technology", Descripción técnica, versión 1.1

[5] Dunsmuir DT, Payne BA, Cloete G, Petersen CL, Görges M, Lim J, von Dadelszen P, Dumont GA, Ansermino JM. (2014). IEEE J Biomed Health Inform. Noviembre de 2014.

[6] Mackenzie CF, Hu P, Sen A, Dutton R, Seebode S, Floccare D, Scalea T. (2008). Automatic pre-hospital vital signs waveform and trend data capture fills quality management, triage and outcome prediction gaps. AMIA Annu Symp Proc. Novimbre de 2008.

[7] Biffi E, Piazza C, Cavalleri M, Taddeo P, Carcano A, Morandi F, Reni G. (2014). An assistive device for congenital central hypoventilation syndrome outpatients during sleep. Ann Biomed Eng. Octubre de 2014 .

[8] Burgos Llamo, A. (2014). *Telemonitorización en tiempo real del Síndrome de Apneas-Hipopneas del Sueño (SAHS) mediante pulsioximetría domiciliaria*. Servicio Editorial de la Universidad del País Vasco/Euskal Herriko Unibertsitatearen Argitalpen Zerbitzua.

[9] Workie FA, Rais-Bahrami K, Short BL. (2005). Workie FA, Rais-Bahrami K, Short BL. Am J Perinatol. Octubre de 2005

[10] Montoya Camayo, L. M. (2011). Sistema automático digital de triage médico para los niveles de prioridad I, II y III del servicio de urgencias en IPS´ S.

[11] Faria I, Gaspar C, Zamith M, Matias I, das Neves RC, Rodrigues F, Bárbara C. (2014). TELEMOLD project: oximetry and exercise telemonitoring to improve long-term oxygen therapy. Telemed J E Health. Juio de 2014

[12] Morón, M., Casilari, E., & Gázquez, J. (2004). Sistema de Monitorización Inalámbrica de Sensores de SPO2 (pulsioxímetros).

[13] Işik AH, Güler I. (2012). Pulse oximeter based mobile biotelemetry application. Stud Health Technol Inform. 2012

[14] Kaputa D, Price D, Enderle Jd. (2010). A Portable, Inexpensive, Wireless Vital Signs Monitoring System. Biomed instrum technol. julio de 2010

[15] Serrano, C. (2008). Modelo para la Construcción de Soluciones. Modelo Integral para el Profesional en Ingeniería, p. 43-58.

[16] Sánchez, Jesús Ramírez, and José Vicente Díaz Martínez. "Las redes inalámbricas, más ventajas que desventajas." (2004).

[17] Cisco y la escalabilidad, Último acceso: 22 de noviembre de 2016,https://colombiadigital.net/actualidad/articulos-informativos/item/7734-cisco-y-la-escalabilidad-ahorros-al-invertir-en-ti.html

[18] Bluetooth Low Energy: Argenox, Último acceso 01 de mayo de 2016, http://www.argenox.com/es/bluetooth-baja-energia-ble-desarrollo/library/introduction-bluetooth-bajo-consumo/

[19] Bluetooth: Wikipedia, Último accedo:01 de mayo de 2016, https://es.wikipedia.org/wiki/Bluetooth
[20] Pusioximetria: Wikipedia, Ultimo acceso: 01 de mayo de 2016, https://es.wikipedia.org/wiki/Pulsioximetr%C3%ADa
[21] Bluetooth Low Energy: Argenox, Último acceso 01 de mayo de 2016, http://www.argenox.com/es/bluetooth-baja-energia-ble-desarrollo/library/introduction-bluetooth-bajo-consumo/
[22] Análisis comparativo de tecnologías inalámbricas para una solución de servicios de telemedicina. César Viloria Núñez, Jairo Cardona Peña, Carlos Lozano Garzón. Revista Científica Ingeniería y Desarrollo, No 25 (2009)
[23] Java (Lenguaje de programación), Último acceso: 22 de noviembre de 2016, https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación
[24] Protocolo http: Protocolos de comunicación, Último acceso 01 de mayo de 2016, http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html
[25] Tableta:Wikipedia, https://es.wikipedia.org/wiki/Tableta_(computadora)
[26] Decreto 1377 de 2013 nivel nacional, Colombia, último acceso: agosto 16 2016.
[27] El lenguaje Unificado de Modelado (UML). Álvaro Rendón Gallón. Universidad del Cauca. Popayan, marzo de 2002.
[28] Raspbian: Wikipedia: https://es.wikipedia.org/wiki/Raspbian
[29] Saturación de oxígeno y niveles normales, http://www.mimoonline.es/pregunta.php?idP=48
[30] Frecuencia cardiaca: Fundación española del corazón, Último acceso: 01 de mayo de 2016, http://www.fundaciondelcorazon.com/prevencion/riesgo-cardiovascular/frecuencia-cardiaca.html
[31] Modelo Vista Controlador. Último acceso: 22 d noviembre de 2016, https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador
[32] Tutorial básico de Java EE. Jose Villalba Cortazzo. 2010
[33] HTML. Último acceso: 01 de noviembre de 2016. https://es.wikipedia.org/wiki/HTML
[34] JavaServerPages. Último acceso: 25 de octubre de 2016. https://es.wikipedia.org/wiki/JavaServer_Pages

ANEXOS

ANEXO A. CODIGOS FUENTE UTILIZADOS EN LA PLACA RASPBERRY PI 3.

A continuación se presenta el código principal utilizado en la placa electrónica Raspberry pi para la medición y procesamiento de signos vitales y una breve descripción. Se aclara que el sistema puede realizar ocho lecturas en tiempo real, pero se presenta el código utilizado para la lectura de un solo sensor con el fin de no extender el documento y debido a que las lecturas son análogas con respecto al código. También se presenta el código realizado en Arduino para la lectura y envío de datos por Bluetooth y una breve descripción.

Por otro lado se aclara que, la lectura se hace en tiempo, realizando la lectura del primer sensor, luego se conecta al segundo sensor y hace su lectura, luego se conecta al tercer sensor y hace su lectora y así sucesivamente hasta completar los 8 sensores y gracias al alto procesamiento de la placa esto es indetectable por el usuario. El código completo se encuentra en el CD entregado junto con este documento.

A.1. detecciónSignosAuto.py

```
import serial
import suds


def obtener_niv(sat, pul):
        niv = ""
        if(sat > 90 and pul > 63 and pul < 98):
                niv = "normal"
        if((sat > 88 and sat < 91) or (pul > 97 and pul < 113) or (pul
                niv = "dos"
        if(sat < 89 or pul < 60 or pul > 112):
                niv = "uno"
        return niv


def cambio_ent(cadena):
        cadena = int(cadena)
        return cadena

url = "http://192.168.100.3:8080/ServicioWebPaD/WSGestorRasp?WSDL"
client = suds.client.Client(url)
```

```
btSerial = serial.Serial("/dev/rfcomm0", baudrate=9600, timeout=2)

while True:
        spo2 = btSerial.readline()[:-1]
        if (len(spo2)==0):
                alt = btSerial.readline()[:-1]

                print "variable vacia"
        bpm = btSerial.readline()[:-1]
        if (len(spo2)==0):

                print "variable vacia"
        if (len(spo2)==0):
                print "variable vacia prueba"
        else:
                nivel = ""
                nivel = obtener_niv(spo2, bpm)

        print "sat: ", spo2
        print "pul: ", bpm
        nivel = obtener_niv(spo2, bpm)
```

```
        respuesta = client.service.subirDatos('0','98','70',nivel)
        spo2 = 0
        bpm = 0
```

Figura A.1.1 detecciónSignosAuto.py

Del anterior código, destacamos la primera función con la cual se procesan los rangos de los signos para lograr generar un Triage. Esta función recibe como parámetros la saturación de oxígeno y el pulso cardiaco y devuelve el nivel o tipo de Triage. En la variable url se encuentra la dirección donde se localiza el servicio web. En la variable btserial se muestra la localización de los parámetros de lectura de los datos que se reciben por Bluetooth desde el sensor por medio de rfcomm, esta variable varia para cada sensor utilizando cada uno de los perfiles configurados en rfcomm.conf.

Se lee cada una de las líneas que se enviaron por Bluetooth desde el sensor a través del método readline() y se verifica si la variable no se encuentra vacía, si no está vacía se almacena en las variables spo2 y bpm y estas a su vez se envían a la función obtener_niv() para obtener su nivel. Para finalizar se envía la saturación el pulso y el nivel al servidor a través del método web subirDatos(). Luego se repite el ciclo con cada sensor hasta llegar al octavo sensor o btserial8.

A.2. RXTXDatosPO

```
#include <PinChangeInt.h>
#include <eHealth.h>
int cont = 0;
int num = 2;

void setup() {
  Serial.begin(9600);
  eHealth.initPulsioximeter();
  //Attach the inttruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}

void loop() {
  int sat;
  int pul;
  sat = eHealth.getOxygenSaturation();
  pul = eHealth.getBPM();
  if (sat > 10 && pul > 10)
     {
      Serial.println(sat);
      Serial.println(pul);
     }

  delay(2000);
}



//Include always this code when using the pulsioximeter sensor
//========================================================================
void readPulsioximeter(){

  cont ++;

  if (cont == 50) { //Get only of one 50 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

Figura A.2.1. RXTXDatosPO

El anterior código realizado para Arduino, se basa en el código de ejemplo contenido en la librería eHealth para lectura del pulsioxÍmetro con modificaciones para este proyecto. Del código anterior se destaca la tasa de lectura y envio de 9600 baudios, se inicia el proceso de lectura del pulsioximetro mediante el método eHealth.initPulsioximeter(), se realizan interrupciones a través del pin 6, todo esto dentro del método setup(). Ya en el método loop() se inicializan las variables que van a almacenar los signos vitales, luego se llaman las funciones eHealth.getOxygenSaturation() y eHealth.getBPM() que traen los datos de saturación de oxígeno y pulso cardiaco. A continuación se envían las variables a través de Bluetooth mediante el método println().

ANEXO B. CÓDIGOS FUENTE Y DESCRIPCIÓN.

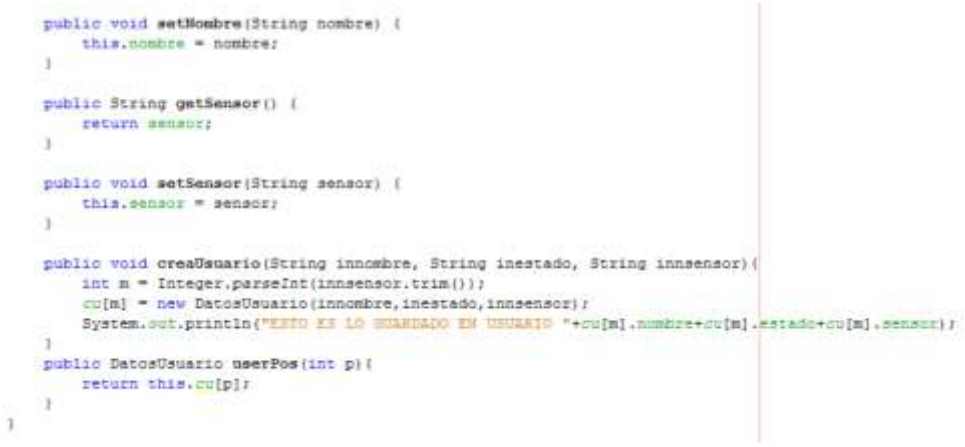
A continuación se presentan los códigos fuente utilizados en el proyecto y una descripción breve de cómo funcionan y contenido. El sistema, a pesar de leer ocho diferentes sensores, al leer uno a uno lo hace de la misma manera que los demás, por lo tanto en este anexo se mostrara como se lee para un sensor, ya que se entiende que el resto es similar o análogo.

B.1. CÓDIGO SERVICIO WEB.

El servicio web se compone de dos clases principales encargadas de almacenar temporalmente los signos en variables locales, estas dos clases se encuentran dentro del paquete llamado *datos*, las clases son las siguientes:

Clase DatosUsuario:

```
package datos;

/*
 * @Diego Fernando Viveros Zambrano
 */
public class DatosUsuario {
    private String nombre,estado,sensor;
    private static DatosUsuario du[] = new DatosUsuario[8];

    public DatosUsuario(){

    }

    public DatosUsuario(String nombre, String estado, String sensor){

        this.nombre = nombre;
        this.estado = estado;
        this.sensor = sensor;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getNombre() {
        return nombre;
    }
```



```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getSensor() {
        return sensor;
    }

    public void setSensor(String sensor) {
        this.sensor = sensor;
    }

    public void creaUsuario(String innombre, String inestado, String innsensor){
        int m = Integer.parseInt(innsensor.trim());
        cu[m] = new DatosUsuario(innombre,inestado,innsensor);
        System.out.println("ESTO ES LO GUARDADO EN USUARIO "+cu[m].nombre+cu[m].estado+cu[m].sensor);
    }
    public DatosUsuario userPos(int p){
        return this.cu[p];
    }
}
```

Figura B.1.1. Clase DatosUsuario.

En la anterior imagen, figura B.1.1, se puede observar el código que permite almacenar los datos que vienen desde el dispositivo móvil (Tablet), comenzando de arriba hacia abajo, se aprecia en la primera línea el paquete llamado *datos*, que contiene almacenado el código de esta clase. Continuando se encuentra el nombre de la clase *DatosUsuario,* luego las variables de la clase *nombre, estado, sensor* y una variable del mismo tipo de la clase pero de tipo arreglo *cu[]*, que permite almacenar los datos de los 8 sensores, más una posición para pruebas.

Se continúa con un constructor vacío (Sin parámetros), para instanciar la clase con el mismo nombre de la clase y un constructor con parámetros de entrada que serán almacenados en las variables de la clase. Para diferenciar las variables de la clase con las variables que están entrando por parámetro se hace uso de la palabra *this* antes de la variable de la clase como se puede apreciar en la anterior figura.

Luego se encuentra los métodos get y set; con los métodos get como se puede observar se retorna cada una de las variables de la clase donde están almacenados los datos del usuario, estos métodos get pueden ser utilizados desde otras clases para acceder a las variables de la clase DatosUsuario. Los métodos set reciben como parámetros cada una de las variables por separado y permiten almacenar su valor en cada variable de la clase por separado. Continuando hacia abajo, se encuentra el método llamado *creaUsuario*, el cual es quien recibe directamente las variables del usuario y utiliza los demás métodos de la clase para almacenarlas en cara posición del arreglo *cu[]*. Para finalizar con los métodos de la clase, se encuentra el método *userPos*, quien recibe como parámetro una variable de tipo entero, con el fin de identificar una posición dentro del arreglo *cu[]* y retornar los valores del usuario almacenados en esa posición.

Clase DatosSensor:

```java
package datos;

/*
 * @author Diego Fernando Viveros Zambrano
 */
public class DatosSensor {
    private String sensor_rasp,sat_rasp,pulso_rasp;
    private String nivel_triage;
    private static DatosSensor su[] = new DatosSensor[8];

    public DatosSensor(){

    }
    public DatosSensor(String sensor,String sat,String pulso,String nivel){
        sensor_rasp = sensor;
        sat_rasp = sat;
        pulso_rasp = pulso;
        nivel_triage= nivel;

    }

    public String getNivel_triage() {
        return nivel_triage;
    }

    public void setNivel_triage(String nivel_triage) {
        this.nivel_triage = nivel_triage;
    }

    public String getPulso_rasp() {
        return pulso_rasp;
    }
    public void setPulso_rasp(String pulso_rasp) {
        this.pulso_rasp = pulso_rasp;
    }

    public String getSat_rasp() {
        return sat_rasp;
    }

    public void setSat_rasp(String sat_rasp) {
        this.sat_rasp = sat_rasp;
    }

    public String getSensor_rasp() {
        return sensor_rasp;
    }

    public void setSensor_rasp(String sensor_rasp) {
        this.sensor_rasp = sensor_rasp;
    }
    public void creaSensor(String insensor,String insat,String inpulso,String innivel){
        int n = Integer.parseInt(insensor.trim());
        su[n] = new DatosSensor(insensor, insat, inpulso, innivel);
        System.out.println("ESTO ES LO GUARDADO "+su[n].sensor_rasp+su[n].sat_rasp+su[n].pulso_rasp+su[n].nivel_triage);
    }
    public DatosSensor sensorPos(int p){
        return su[p];
    }
}
```

Figura B.1.2. Clase DatosSensor.

En la anterior imagen, figura B.1.2, se puede observar el código que permite almacenar los datos que vienen desde la placa electrónica Raspberry Pi 3. Comenzando de arriba hacia abajo, se aprecia en la primera línea el paquete llamado *datos*, que contiene almacenado el código de esta clase al igual que la clase anterior. Continuando se encuentra el nombre de la clase *DatosSensor,* luego las variables de la clase *sensor_rasp, sat_rasp, pulso_rasp, nivel_triage* y una variable del mismo tipo de la clase

pero de tipo arreglo *su[]*, que permite almacenar los datos de los 8 sensores, más una posición para pruebas.

Se continúa con un constructor vacío (Sin parámetros), para instanciar la clase con el mismo nombre de la clase y un constructor con parámetros de entrada que serán almacenados en las variables de la clase. Para diferenciar las variables de la clase con las variables que están entrando por parámetro se hace uso de la palabra *this* antes de la variable de la clase como se puede apreciar en la anterior figura.

Luego se encuentra los métodos get y set, que se utilizan de la misma manera como se explicó en la clase DatosUsuario. Continuando hacia abajo, se encuentra el método llamado *creaSensor*, el cual es quien recibe directamente las variables de los sensores y utiliza los demás métodos de la clase para almacenarlas en cara posición del arreglo *su[]*. Para finalizar con los métodos de la clase, se encuentra el método *sensorPos*, quien recibe como parámetro una variable de tipo entero, con el fin de identificar una posición dentro del arreglo *su[]* y retornar los valores del sensor almacenados en esa posición.

En las clases anteriores, en los métodos *creaUsuario* y *creaSensor*, se agrega una línea System.out.println, con la cual se imprime un mensaje en la consola ser servidor para saber cuándo se ha agregado un nuevo usuario y nuevos datos de sensor, con sus respectivos valores.

Las siguientes clases representan los servlets del sistema y se encuentran dentro del paquete llamado *servicios,* como se muestra a continuación:

Servlet TabletServlet:

```
package servicios;

import datos.*;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Diego Fernando Viveros Zambrano
 */
@WebServlet(name = "TabletServlet", urlPatterns = {"/tabletServlet"})
public class TabletServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String nom_tab = request.getParameter("nombre_tab");
        String estado = request.getParameter("estado");
        String sensor_tab = request.getParameter("sensor_tab");

        DatosUsuario du = new DatosUsuario();
        du.creaUsuario(nom_tab,estado,sensor_tab);
        System.out.println(nom_tab+" "+estado+" "+sensor_tab);
        PrintWriter out = response.getWriter();
        response.setContentType("text/html;charset=UTF-8");
        try {
```

Figura B.1.3. Servlet TabletServlet.

El anterior código, figura B.1.3, representa el servlet que responde a las peticiones realizadas por la página de prueba de envío de datos del dispositivo móvil y a las pruebas de agregar usuario y detener lectura. El código de arriba hacia abajo, comienza con el nombre del paquete que lo contiene llamado *servicios,* continuando con las importaciones de los paquetes necesarios para el funcionamiento del código. Continua con la declaración de la clase como servlet y la url con la que será llamado /*tableServlet.* Luego, se hace la declaración de la clase junto con la extensión a la clase superior *HttpServlet* para poder usar sus métodos. A continuación se implementa el método *processRequest,* donde se reciben los parámetros enviados a través de *request.getParameter*(), y almacenados en variables tipo String, para luego almacenarlos en las varibles del sistema por medio de la instancia a la clase DatosUsuario y su método creaUsuario.

Servlet RaspServlet:

```
package servicios;

import datos.DatosSensor;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Diego Fernando Viveros Zambrano
 */
@WebServlet(name = "RaspServlet", urlPatterns = {"/raspServlet"})
public class RaspServlet extends HttpServlet {


    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String id_rasp = request.getParameter("id_rasp");
        String sensor_rasp = request.getParameter("sensor_rasp");
        String saturacion_rasp = request.getParameter("sat_rasp");
        String pulso_rasp = request.getParameter("pulso_rasp");
        String nivel_rasp = request.getParameter("nivel_rasp");
        DatosSensor dt = new DatosSensor();
        dt.creaSensor(sensor_rasp, saturacion_rasp, pulso_rasp, nivel_rasp);
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
```

Figura B.1.4. Servlet RaspServlet.

El anterior código, figura B.1.4, representa el servlet que responde a las peticiones realizadas por la página de prueba de envío de datos desde la placa electrónica Raspberry. El código de arriba hacia abajo, comienza con el nombre del paquete que lo contiene llamado *servicios,* continuando con las importaciones de los paquetes necesarios para el funcionamiento del código. Continua con la declaración de la clase como servlet y la url con la que será llamado /*raspServlet.* Luego, se hace la declaración de la clase junto con la extensión a la clase superior *HttpServlet* para poder usar sus métodos. A continuación se implementa el método *processRequest,* donde se reciben los parámetros enviados a través de *request.getParameter*(), y almacenados en variables tipo String, para luego almacenarlos en las varibles del sistema por medio de la instancia a la clase DatosSensor y su método creaSensor.

Servlet Inicializa:

```
package servicios;

import datos.DatosSensor;
import datos.DatosUsuario;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Diego Fernando Tiverez Zambrano
 */
@WebServlet(name = "Inicializa", urlPatterns = {"/inicializa"})
public class Inicializa extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String act = request.getParameter("activar");
        if(act.equals("ok")){
            DatosUsuario du = new DatosUsuario();
            DatosSensor dt = new DatosSensor();
            du.creaUsuario("Usuario1", "off", "0");
            dt.creaSensor("0", "00", "00", "cero");
            du.creaUsuario("Usuario2", "off", "1");
            dt.creaSensor("1", "00", "00", "cero");
            du.creaUsuario("Usuario3", "off", "2");
            dt.creaSensor("2", "00", "00", "cero");
            du.creaUsuario("Usuario4", "off", "3");
            dt.creaSensor("3", "00", "00", "cero");
            du.creaUsuario("Usuario5", "off", "4");
            dt.creaSensor("4", "00", "00", "cero");
            du.creaUsuario("Usuario6", "off", "5");
            dt.creaSensor("5", "00", "00", "cero");
            du.creaUsuario("Usuario7", "off", "6");
            dt.creaSensor("6", "00", "00", "cero");
            du.creaUsuario("Usuario8", "off", "7");
            dt.creaSensor("7", "00", "00", "cero");


        }
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
```

Figura B.1.5. Servlet Inicializa.

El código anterior, figura B.1.5, de la misma forma que los anteriores servlets inicia con el paquete que lo almacena llamado *servicios*, este código se diferencia a los dos anteriores en que solo recibe un parámetro llamado *activar,* el cual decide a través de un *if* si el valor que contienes es *ok,* si es asi, crea dos intancias a las clases DatosSensor y DatosUsuario y almacena valores por defecto en las variables del sistema a través de los métodos creaUsuario y creaSensor.

Las clases que se presentan a continuación, corresponden a los métodos web que se encuentran dentro de webservices, encargados de comunicarse con la aplicación Android instalada en el dispositivo móvil (Tablet) y con la placa electrónica Raspberry:

Clase WSGestorTablet:

```
package servicios;

import datos.DatosSensor;
import datos.DatosUsuario;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Diego Fernando Vivenes Zambrano
 */
@WebService(serviceName = "WSGestorTablet")
public class WSGestorTablet {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "IniciarLectura")
    public String IniciarLectura(@WebParam(name = "nombre_tab") String nombre_tab,
    @WebParam(name = "estado") String estado, @WebParam(name = "sensor_tab") String sensor_tab) {
        DatosUsuario du = new DatosUsuario();
        String senElegidoon = "";
        if(sensor_tab.equals("1")){senElegidoon = "0";}
        if(sensor_tab.equals("2")){senElegidoon = "1";}
        if(sensor_tab.equals("3")){senElegidoon = "2";}
        if(sensor_tab.equals("4")){senElegidoon = "3";}
        if(sensor_tab.equals("5")){senElegidoon = "4";}
        if(sensor_tab.equals("6")){senElegidoon = "5";}
        if(sensor_tab.equals("7")){senElegidoon = "6";}
        if(sensor_tab.equals("8")){senElegidoon = "7";}
        System.out.println("EL SENSOR A ENCENDER ES: "+senElegidoon);
        du.creaUsuario(nombre_tab,estado,senElegidoon);
        String resul = "Usuario Sincronizado con sensor: "+sensor_tab;
        return resul;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "DetenerLectura")
    public String DetenerLectura(@WebParam(name = "sensor_tab") String sensor) {
        DatosUsuario dud = new DatosUsuario();
        DatosSensor dt = new DatosSensor();
        String senElegidooff = "";
        if(sensor.equals("1")){senElegidooff = "0";}
        if(sensor.equals("2")){senElegidooff = "1";}
        if(sensor.equals("3")){senElegidooff = "2";}
        if(sensor.equals("4")){senElegidooff = "3";}
        if(sensor.equals("5")){senElegidooff = "4";}
        if(sensor.equals("6")){senElegidooff = "5";}
        if(sensor.equals("7")){senElegidooff = "6";}
        if(sensor.equals("8")){senElegidooff = "7";}
        System.out.println("EL SENSOR A APAGAR ES: "+senElegidooff);
        dud.creaUsuario("Sin Usuario","off",senElegidooff);
         dt.creaSensor(senElegidooff, "00", "00", "cero");
        String sal = "Sensor "+sensor+" Apagado";
        return sal;
    }
}
```

Figura B.1.6. Clase WSGestorTablet.

Este código presentado anteriormente, figura B.1.6, representa un webservices que contiene dos métodos web *IniciarLectura* y *DetenerLectura.* El código inicia mencionando el paquete que lo contiene *servicios,* continuando con las importaciones a los paquetes necesarios para el código. Prosigue con la declaración de la clase como webservice y la declaración de la clase. Más abajo, se encuentra la declaración del primer método web *IniciarLectura* junto con los parámetro de entrada y sus respectivos tipos. Se sigue con la instancia a la clase Datos sensor y una comparación del valor de la variable sensor debido a que la variable como se explicó en este documento se maneja en 8 valores del 0 al 7 para manejarlo como posición, pero del dispositivo llegan los valores del 1 al 8 y deben ser acomodados al manejo que se hace en el servicio web. Por último, se utilizan

los métodos de la clase DatosUsuario y se imprime un mensaje en consola confirmando que la acción se realizó correctamente. Para cerrar el método se retorna al dispositivo móvil un mensaje en una variable String un mensaje de que se hiso de forma correcta.
Se continúa con la declaración de segundo método *DetenerLectura* como método web, luego la declaración como clase junto con sus parámetros de entrada y sus respectivos tipos de variables. Luego continúa de manera análoga al anterior método con la diferencia que se trabaja con instancia y métodos de la clase DatosSensor y retornando un mensaje que confirma que se detuvo la lectura.

Clase WSDespTablet:

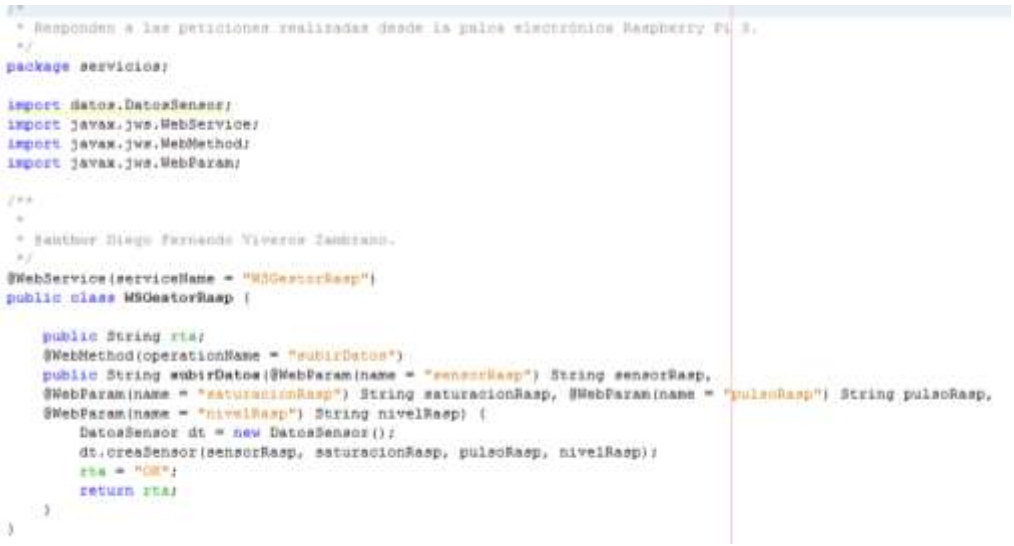
```
package servicios;

import datos.DatosSensor;
import datos.DatosUsuario;
import java.util.ArrayList;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Diego Fernando Viveros Zambrano
 */
@WebService(serviceName = "WSDespTablet")
public class WSDespTablet {

        DatosUsuario datus = new DatosUsuario();
        DatosSensor datsen = new DatosSensor();
        public String nom,sat,pul,niv,est;

    @WebMethod(operationName = "PresentarNom")
    public String PresentarNom(@WebParam(name = "sensorPos") String sensorPos) {

        int i = Integer.parseInt(sensorPos);
        DatosUsuario dat = datus.userPos(i);
         nom = dat.getNombre();
         return nom;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "PresentarSat")
    public String PresentarSat(@WebParam(name = "sensorPos") String sensorPos) {
        int pos2 = Integer.parseInt(sensorPos);
        DatosSensor sen = datsen.sensorPos(pos2);
        sat = sen.getSat_rasp();
        return sat;

    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "PresentarPul")
    public String PresentarPul(@WebParam(name = "sensorPos") String sensorPos) {
        int pos3 = Integer.parseInt(sensorPos);
        DatosSensor sen = datsen.sensorPos(pos3);
        pul = sen.getPulso_rasp();
        return pul;
    }
```

```
    /**
     * Web service operation
     */
    @WebMethod(operationName = "PresentarNiv")
    public String PresentarNiv(@WebParam(name = "sensorPos") String sensorPos) {
        int pos4 = Integer.parseInt(sensorPos);
        DatosSensor sen = datsen.sensorPos(pos4);
        niv = sen.getNivel_triage();
        return niv;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "Estado")
    public String Estado(@WebParam(name = "sensorPos") String sensorPos) {
        int pos5 = Integer.parseInt(sensorPos);
        DatosUsuario dat = datus.userPos(pos5);
        est = dat.getEstado();
        return est;
    }
}
```

Figura B.1.7. Clase WSDespTablet.

El anterior código de la figura B.1.7, se declara a la clase como webservice, la cual se encuentra dentro del paquete *servicios.* Es necesario declarar las clases DatosUsuario y DatosSensor ya que los métodos web que contiene el servicio retornan valores de las variables del usuarioy de los signos.
El primer método que es declarado como método web es *PresentarNom* (Los nombres de los método fueron colocados siempre teniendo en cuenta que sean nombres fáciles de recordar porque se utilizarían en la codificación de la aplicación Android y de la codificación en la Raspberry, el cual recibe la variable *sensorPos,* utilizando su valor como posición del valor solicitado. Se hace uso del método *userPos* y el método *getNombre* para poder finalmente retorna el nombre en una variable *nom* de tipo String.
El siguiente método web declarado es *PresentarSat,* que de manera análoga al método anterior recibe la variable *sensorPos* y por medio de los métodos *sensorPos* y *getSat_rasp,* retorna el valor de la saturación solicitada a través de la variable *sat* de tipo String.
De la misma manera que los métodos mencionados anteriormente, se declaran tres métodos web mas *PresentarPul, PresentarNiv* y *Estado,* los cuales retornan el pulso, el nivel y el estado a través de las variables pul, niv y est respectivamente.

Clase WSGestorRasp:



```
/*
 * Responden a las peticiones realizadas desde la palca electrónica Raspberry Pi 3.
 */
package servicios;

import datos.DatosSensor;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Diego Fernando Viveros Zambrano.
 */
@WebService(serviceName = "WSGestorRasp")
public class WSGestorRasp {

    public String rta;
    @WebMethod(operationName = "subirDatos")
    public String subirDatos(@WebParam(name = "sensorRasp") String sensorRasp,
    @WebParam(name = "saturacionRasp") String saturacionRasp, @WebParam(name = "pulsoRasp") String pulsoRasp,
    @WebParam(name = "nivelRasp") String nivelRasp) {
        DatosSensor dt = new DatosSensor();
        dt.creaSensor(sensorRasp, saturacionRasp, pulsoRasp, nivelRasp);
        rta = "OK";
        return rta;
    }
}
```

Figura B.1.8. Clase WSGestorRasp.

El código visto anteriormente en la figura B.1.8, el último websesrvice que se encuentra almacenado en el paquete *servicios.* Este servicio se comunica con la placa electrónica Raspberry quien le envía los datos procesados recibidos desde los sensores de pulsioximetría inalámbricos, para esto se declara el método llamado *subirDatos* quien se declara como método web y recibe parámetros web como: *sensorRasp, saturacionRasp, pulsoRasp y nivelRasp.* Debido a que todos estos datos representan los signos enviados por los sensores solo es necesario instanciar la clase DatosSensor y por medio de su método *creaSensor* se almacenan los datos en las variables del sistema y se finaliza el método retornando un mensaje de *OK* a través de la variable *rta* de tipo String.

Los códigos presentados anteriormente muestran las clases que representan el controlador y el modelo del servicio web, a continuación se presenta lo correspondiente a la vista que está compuesta por las páginas html y jsp, las cuales son las que se presentan al usuario del sistema:

Página index.html:

Figura B.1.9. index.html

En la figura B.1.9 vista anteriormente, se describe el código que compone la página index.html, primera página que se despliega al administrador del servidor. La página se encuentra en código html, destacando en la cabecera el uso de un estilo para la página escrito entre las etiquetas <style>. El estilo le da un color específico a divisiones realizadas en la página. En el cuerpo de la página que se encuentra entre las etiquetas <body> se encuentra en la división superior para el título principal de la página, luego, otra división que contiene texto informativo entre etiquetas <h3> y por ultimo unos enlaces que llevan a otras páginas web, entre las etiquetas <a>.

Página inicializaDatos.html:

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <style type="text/css">
            body{
                background-color: #E6E6FA
            }
            div.superior{
                background-color: #6A5ACD;
                text-decoration-color: white;
                height: 80px
            }
            div.central{
                background-color: #E6E6FA
            }
            button{
                background-color: #778899
            }
        </style>
    </head>
    <body>
        <div class="superior"><h1>Inicializacion de datos</h1></div>
        <div class="central">
            <h3>Los datos con los que se inicializaran las varialbes locales son los siguientes:</h3>
            <table>
                <title>Datos de variables inicializadas:</title>
                <tr>
                    <th>Sensor</th>
                    <th>Usuario</th>
                    <th>Estado</th>
                    <th>Saturación</th>
                    <th>Pulso</th>
                    <th>Nivel</th>
                </tr>
                <tr>
                    <th>0</th>
                    <th>Usuario1</th>
                    <th>0ff</th>
                    <th>00</th>
                    <th>00</th>
                    <th>cero</th>
                </tr>
                <tr>
                    <th>1</th>
                    <th>Usuario2</th>
                    <th>0ff</th>
                    <th>00</th>
                    <th>00</th>
                    <th>cero</th>
                </tr>
                <tr>
                    <th>2</th>
                    <th>Usuario3</th>
                    <th>0ff</th>
                    <th>00</th>
                    <th>00</th>
                    <th>cero</th>
                </tr>
```

Figura B.1.10. inicializaDatos.html

La figura B.1.10 presentada muestra la programación en html de la página inicializaDatos.html, comenzando dándole un estilo a la página asignando colores a las diferentes divisiones. Se continúa en el cuerpo entre las etiquetas <body> con la creación de una tabla con las etiquetas <table> y en esta, se hacen varias filas con la etiqueta <tr> y varias columnas con la etiqueta <th>. En la primera columna en cada fila se colocan primero los títulos de los variables que se van a inicializar, lo cuales son: Sensor, Usuario, Estado, Saturación, Pulso y Nivel, y en las siguientes filas se colocan los valores que serán agregados. Se aclara que el valor cero que se asigna al nivel no tiene nada que ver con el valor cero del nivel de Triage, se usa el valor cero para indicar al sistema que aún no se ha recibido ningún valor de nivel y para que el sistema aun no coloque los colores del Triage. Lego de la tabla, se agrega un botón que se encuentra dentro de la etiqueta <button>, que a su vez se encuentra dentro de un formulario etiqueta <form> que lleva el dato activar con valor *ok,* al servlet *inicializa.* En la parte inferior de la página se escribió textos de información.

Página PrubaTablet.html:

```
    <body>
        <div class="superior">
            <h1>Prueba de envio de datos desde tablet:</h1>
        </div>
        <div class="central">
            <br>
        <form method="post" action="/ServicioWebPaD/tabletServlet">
        nombre: <input name="nombre_tab" type="text">
        estado (on/off): <input name="estado" type="text">
        sensor: <input name="sensor_tab" type="text">
        <p><button>Enviar</button></p>
    </form>
        </div>
    </body>
</html>
```

Figura B.1.11. PruebaTablet.html

Como en las anteriores páginas, esta página de la figura B.1.11, inicia programando su estilo y al pasar al cuerpo se colocó un formulario con las etiquetas <form>, dentro del cual se incluyó varias entradas de texto con las etiquetas <input> y un botón para el envío de los datos al servlet *tabletServlet*.

Página PruebaRasp.html:

```
    <body>
        <div class="superior">
            <h1>Prueba de envio de datos desde raspberry:</h1>
        </div>
        <div class="central">
            <br>
        <form method="post" action="/ServicioWebPaD/raspServlet">
        sensor: <input name="sensor_rasp" type="text">
        saturacion: <input name="sat_rasp" type="text">
        pulso: <input name="pulso_rasp" type="text">
        nivel: <input name="nivel_rasp" type="text">
        <p><button>Enviar</button></p>
        </form>
        </div>
    </body>
</html>
```

Figura B.1.12. PruebaRasp.html

El código que se presenta en la figura B.1.12, se trabaja de la misma manera que la vista en la figura B.11, con la diferencia que se incluyen más entradas de texto y el formulario es enviado al servlet *raspServlet*.

Página index.jsp

```
<%--
    Author     : Diego Fernando Viveros Zambrano
--%>

<%@page import="java.io.FileInputStream"%>
<%@page import="java.net.URL"%>
<%@page import="java.applet.AudioClip"%>
<%@page import="datos.DatosSensor"%>
<%@page import="datos.DatosUsuario"%>
<%@page import="javazoom.jl.player.Player"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Refresh" content="2; url=http://localhost:8080/ServicioWebPaU/index.jsp">
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <style type="text/css">
            body {
               height: 500px
                }
            tr.useroff {background-color: #A9A9A9}
            tr.useronel {background-color: #09FD42 }
            tr.useronel2{background-color: #FF8C00}
            tr.useroone {background-color: red }
            tr.usertwol {background-color: #09FD42}
            tr.usertwo {background-color: red }
            tr.userthreel {background-color: #09FD42}
            tr.userthree {background-color: red }
            tr.userfourl {background-color: #09FD42}
            tr.userfour {background-color:red }
            tr.userfivel {background-color: #09FD42}
            tr.userfive {background-color: red }
            tr.usersixl {background-color: #09FD42}
            tr.usersix {background-color: red }
            tr.usersevenl {background-color: #09FD42}
            tr.usereven {background-color: red }
            tr.usereightl {background-color: #09FD42}
            tr.usereight {background-color: red }

            div.contenedor{
                width: 693px; margin: 0 auto 0 auto
            }
            div.datos {
                 width: 500px; float: left; margin-bottom: 70px
            }
            div.tituloapp{
                height: 80px; margin-bottom: 40px; width: 693px; margin-top: 80px; background-color: #C8BA
            }
            div.titulos{
                width: 180px; float: left; text-transform: uppercase; background-color: #C8BABA
            }
            td{

                border: 5px;
                border-radius: 10px;
                size: 20px

            }
        </style>
```

```
<body background="imagenes/fondo.jpg">
    <%DatosUsuario dat = new DatosUsuario();%>
        <%DatosSensor datsen = new DatosSensor();%>


    <div class="contenedor">
        <div class="tituloapp">
            <h1>Web Service para el manejo de datos de usuario por pulsioximetria</h1>
        </div>
        <div class="datos">

        <%DatosUsuario dp1 = dat.userPos(0);
            if (dp1 == null) {
                dp1 = new DatosUsuario("usuario","na", "99");
        } else {

        }%>
        <%DatosUsuario dp2 = dat.userPos(1);
        if (dp2 == null) {
            dp2 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosUsuario dp3 = dat.userPos(2);
        if (dp3 == null) {
            dp3 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosUsuario dp4 = dat.userPos(3);
        if (dp4 == null) {
            dp4 = new DatosUsuario("usuario","na", "99");

        }%>
        <%DatosUsuario dp5 = dat.userPos(4);
        if (dp5 == null) {
            dp5 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosUsuario dp6 = dat.userPos(5);
        if (dp6 == null) {
            dp6 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosUsuario dp7 = dat.userPos(6);
        if (dp7 == null) {
            dp7 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosUsuario dp8 = dat.userPos(7);
        if (dp8 == null) {
            dp8 = new DatosUsuario("usuario","na", "99");
        }%>
        <%DatosSensor ds1 = datsen.sensorPos(0);
        if (ds1 == null) {
            ds1 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds2 = datsen.sensorPos(1);
        if (ds2 == null) {
            ds2 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds3 = datsen.sensorPos(2);
        if (ds3 == null) {
            ds3 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds4 = datsen.sensorPos(3);
        if (ds4 == null) {
            ds4 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds5 = datsen.sensorPos(4);
        if (ds5 == null) {
            ds5 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds6 = datsen.sensorPos(5);
        if (ds6 == null) {
            ds6 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds7 = datsen.sensorPos(6);
        if (ds7 == null) {
            ds7 = new DatosSensor("99", "00", "00", "99");
        }%>
        <%DatosSensor ds8 = datsen.sensorPos(7);
        if (ds8 == null) {
            ds8 = new DatosSensor("99", "00", "00", "99");
        }%>
```

```
<table>
    <tr>
        <td><strong>Usuario</strong></td>
        <td><strong>Sensor</strong></td>
        <td><strong>Saturación</strong></td>
        <td><strong>Pulso</strong></td>
        <td><strong>Triage</strong></td>

    </tr>
    <%if(dp1.getEstado().equals("on")){if(ds1.getNivel_triage().equals("uno")){%>
    <tr class="userone">
        <td><%=dp1.getNombre()%></td>
        <td><%=ds1.getSensor_rasp()%></td>
        <td><%=ds1.getSat_rasp()%> %</td>
        <td><%=ds1.getPulso_rasp()%> ppm</td>
        <td><%=ds1.getNivel_triage()%></td>
    </tr>
    <audio id="sound" src="Beep.mp3" autoplay="autoplay" preload="preload"></audio>

    <%}else{%>
    <%if(ds1.getNivel_triage().equals("dos")){%>
        <tr class="userone12">
            <td><%=dp1.getNombre()%></td>
            <td><%=ds1.getSensor_rasp()%></td>
            <td><%=ds1.getSat_rasp()%> %</td>
            <td><%=ds1.getPulso_rasp()%> ppm</td>
            <td><%=ds1.getNivel_triage()%></td>
     </tr>
     <%}else{%>
     <tr class="userone1">

            <%--<td><%= new java.util.Date()%></td>--%>
        <td><%=dp1.getNombre()%></td>
        <td><%=ds1.getSensor_rasp()%></td>
        <td><%=ds1.getSat_rasp()%> %</td>
        <td><%=ds1.getPulso_rasp()%> ppm</td>
        <td><%=ds1.getNivel_triage()%></td>
     </tr>
     <%}%>
     <%}else{%>
     <tr class="useroff">

        <%-- <td><%= new java.util.Date()%></td>--%>
        <td>Sin Usuario</td>
        <td>Sensor-OFF</td>
        <td>00 %</td>
        <td>00 ppm</td>
        <td>00 </td>
     </tr>
     <%}%>
```

Figura B.1.13. index.jsp

El código anterior de la figura B.1.13, representa la página dinámica web index.jsp, encargada de desplegar los datos al usuario administrador y en el televisor al personal médico y de enfermería. El código de la página se compone de html y java que es permitido por medio de etiquetas especiales (<% %>), en medio del html.

En el código se muestra como se despliega los datos del primer usuario y no se presentan el código de los demás sensores debido a que se hace de forma análoga al primero y con el fin de no colocar demasiadas imágenes en este documento.

La página inicia por importaciones necesarias para el funcionamiento del código, luego inicia como se mostró en las páginas html, con el estilo que tendrá la página en su despliegue para lo cual se asignó clases a las filas de la tabla donde se despliegan los signos vitales, con lo cual se puede asignar colores de acuerdo a los signos almacenados en las variables del servicio web. Ya en el cuerpo del html, por medio de código Java se comprueba si los datos de las variables son nulos, si esto es asi, quiere decir que no se ha almacenado nada y no se ha inicializado las variables, para lo cual se muestran unos datos por defecto al usuario quien observa los signos. Luego de esto se crearon tablas con filas y columnas donde se colocaran cada uno de los datos de los usuarios y sus

signos. Dentro del código se instancio a las clases DatosUsuario y DatosSensor y en las filas de la tabla por medio de lenguaje Java se traen los valores con los métodos get de cada una de las clases. Por medio de comparaciones de los datos en los estados y niveles se elige el color que se colocara en el fondo de la fila indicando las alarmas cuando sea necesario. Para el caso en el que se cumple la condición de estado on y nivel uno, además de colocarse en rojo la fila, también se inicia una alarma sonora, mediante la reproducción de un sonido haciendo uso de la etiqueta <audio> de html5. Este audio solo se reproducirá mientras ,a etiqueta de la fila esta activa así como su condición.

El uso de este tipo de etiquetas pertenecientes a html5, implica que el dispositivo de despliegue que accede al servicio web, debe contar un un navegador que soporte html5.

Al final del código se agrega una división más con las etiquetas , para agregar 10 espacios más con el fin de agregar los 10 derechos de los pacientes.

Con este código se finaliza el ANEXO B.1 correspondiente al código del servicio web.

B.2. CÓDIGO APLICACIÓN MOVIL POaD.

Se presentara el código de la aplicación Android POaD (POaD, PulsiOximetría a Distancia) en dos etapas, primero se mostrara el código de las interfaces de usuario o layout y luego las clases que controlan estas interfaces y realizan los procesos de la aplicación.

Estas aplicación se compone de 4 layout, presentados en orden a continuación.

main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#0B4B8A"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".InicioActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="POaD"
        android:textColor="#FFFFFF"
        android:layout_centerHorizontal="true"
        android:textSize="30sp" />

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="120dp"
        android:onClick="IniciarPrograma"
        android:text="@string/conectar" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/pruebacosas"
        android:textColor="#060606"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="100dp" />

</RelativeLayout>
```

Figura B.2.1. main.xml

El anterior código, figura.2.2.1, muestra el xml utilizado para la primera interfaz que se presenta al usuario una vez abre la aplicación Android, de arriba hacia abajo se compone primero de un textview donde se coloca el nombre de la aplicación POaD mediante la etiqueta <TextView>. Luego un botón mediante la etiqueta <Button>, del cual se destaca el evento onClick con el texto IniciarPrograma, que hace el llamado a una función del código que se verá más adelante y más abajo otro textview oculto que se activa cuando no hay conexión. En las configuraciones del botón y los textview se ve el texto pero es solo porque el Android Studio lo coloca, en realidad se programa como se observa en el text del botón donde se llama a una variable String que contiene el texto, es así como se hará en toda la aplicación con el fin de que el código no genere alertas ni problemas.

control.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".ControlActivity"
    android:background="#004B8A">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="POaD" />
    <Button
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:text="Agregar Usuario"
        android:id="@+id/btnagregar"
        android:layout_below="@+id/textView1"
        android:onClick="addUser"
        android:layout_weight="0.90" />
```

```
<TableLayout
    android:layout_width="wrap_content"
    android:layout_height="800dp"
    android:layout_below="@+id/btnagregar"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:id="@+id/tableLayout">

    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="100dp"
        android:background="#A9E2F3" >

        <TextView
            android:id="@+id/nomuseruno"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="25sp"
            android:text="Usuario1" />


        <TextView
            android:id="@+id/satuseruno"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="0"
            android:textSize="25sp"
            android:layout_column="2" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="%"
            android:textSize="25sp"
            android:layout_column="3" />

        <TextView
            android:id="@+id/pulseuno"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="0"
            android:textSize="25sp"
            android:layout_column="4" />

        <TextView
            android:id="@+id/textView14"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="PPM"
            android:textSize="25sp"
            android:layout_column="5" />


    </TableRow>
```

Figura B.2.2. control.xml

En la figura anterior, figura B.2.2, inicia con un textview para el título, luego un botón para agregar el usuario. Más adelante se crea una tabla con la etiqueta <TableLayout> en la que se desplegaran todos los usuarios conectados al sistema y sus signos, para esto se crean filas con la etiqueta <TableRow>. En la fila se agregan los textview para agregar en orden el nombre, saturación de oxígeno, símbolo de porcentaje, pulso cardiaco y símbolo de pulsaciones por minuto. Cada etiqueta tiene sus respectivas configuraciones en cuanto a altura, ancho, tamaño de texto, entre otras.

En el anterior código solo se presenta la primera fila que representa al primer usuario y no se presentan los demás usuarios debido a que se programaron de la misma forma, análogo al primero.

usuario_uno.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#084B8A"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".UsuarioUnoActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ingrese el nombre del usuario:"
        android:textSize="20sp"
        android:textColor="#070707" />

    <Button
        android:id="@+id/button2"
        android:layout_width="250dp"
        android:layout_height="200dp"
        android:onClick="iniciarLectura"
        android:text="Iniciar"
        android:background="#37df19"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

    <Spinner
        android:id="@+id/spn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/numsensor"
        android:layout_below="@+id/textView4"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignRight="@+id/textView4"
        android:layout_alignEnd="@+id/textView4" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sensor Asociado:"
        android:layout_below="@+id/inguseruno"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="20sp"
        android:layout_marginTop="44dp"
        android:textColor="#040404" />


    <EditText
        android:id="@+id/inguseruno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:layout_below="@+id/textView1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="46dp" />

</RelativeLayout>
```

Figura B.2.3. usuario_uno.xml

La figura anterior, B.2.3, representa la interfaz donde se agrega un nuevo usuario para su monitoreo. Su nombre usuario_uno hace alusión de ser el primer paso que se requiera cuando inicia el sistema, este comienza con un textview para pedir el nombre del usuario, luego continua con un EditText que permite ingresar en un texto el nombre del usuario. Más abajo, otro textview para pedir el sensor a iniciar a monitorear junto con un Spinner que despliega todos los sensores y donde se puede elegir uno de ellos. Esta interfaz finaliza con un botón por medio de la etiqueta <Button>.

usuario_dos.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="#004D0A"
    tools:context=".UsuarioDosActivity" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="34dp"
        android:textSize="20sp"
        android:text="Sensor Asociado:"
        android:textColor="#060606" />

    <Spinner
        android:id="@+id/spn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="44dp"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:entries="@array/numsensor"
        android:layout_alignRight="@+id/textView2"
        android:layout_alignEnd="@+id/textView2" />

    <Button
        android:id="@+id/button1"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:onClick="finMonitoreo"
        android:text="Detener"
        android:background="#942705"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Figura B.2.4. usuario_dos.xml

Ahora, se presentan el código que corresponden a las clases que controlan las interfaces y sus datos, además de las conexiones con el servicio web.

El lenguaje en Android se maneja de manera similar al lenguaje Java. Para este proyecto se separaron los códigos en dos paquetes: el primer paquete se encuentra en el código fuente como co.edu.unicauca.telepulsioximetro y el segundo paquete se encuentra como VerificaCambioDatos. El primer paquete contienen los códigos que se encargan de las

interfaces, llamados Activity, manejan el cambio de una a otra y los datos que aparecen en ellas y los códigos del segundo paquete se encargan de las conexiones con el servidor.

InicioActivity.java:

```
package co.edu.unicauca.telepulsicasimetro;

import java.io.IOException;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;


public class InicioActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(getWindow().FEATURE_NO_TITLE);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.inicio, menu);
        return true;
    }

    public void IniciarPrograma(View v) throws IOException{

        ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {

            Intent siguiente = new Intent(this, ControlActivity.class);
            startActivity(siguiente);
        } else {
            TextView tx = (TextView)findViewById(R.id.pruebaconex);
            String pc = "No hay conexion";
            tx.setText(pc);
        }
    }
}
```

Figura B.2.5. InicioActivity.java

El código anterior de la figura B.2.5, inicia con la declaración del paquete que lo contiene, luego con las importaciones a los paquetes y clases necesarias para el funcionamiento del código. Se continúa con la declaración de la clase InicioActivity y su extensión de la clase Activity para el uso de sus métodos que permiten controlar los layout.

Luego se sobrescribe el método onCreate donde se crea la actividad y se elige la vista o layout que se va a usar, en este caso el layout *main.* Más adelante se crea el método IniciarPrograma que recibe como parámetro una vista o view, este método es llamado desde el botón de la interfaz main.xml. Una vez se presiona el botón este método comprueba si existe alguna conexión a una red de área local, si la encuentra hace el llamado a la siguiente actividad ControActivity.java por medio de la clase Intent, si no

encuentra conexión despliega un mensaje en la interfaz actual informando que no hay conexión.

ControlActivity.java:

El siguiente código no será presentado tal cual se encuentra en el código fuente, ya que es demasiado extenso y sería necesario colocar demasiadas imágenes, además teniendo en cuenta que el código que se realiza para desplegar la información de un usuario y sus signos es el mismo para los otros 7 usuarios y sus signos, y por esto no se hace necesario colocarlo. Por tanto se colocaran segmentos de código y se lo explicara uno a uno, como sigue a continuación:

```
package co.edu.unicauca.telepulsioximetro;
import android.graphics.Color;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.TableRow;
import android.widget.TextView;

import VerificaCambioDatos.CambioNivel.CamNiv;
import VerificaCambioDatos.CambioNivel.CamNiv2;
import VerificaCambioDatos.CambioNivel.CamNiv3;
import VerificaCambioDatos.CambioNivel.CamNiv4;
import VerificaCambioDatos.CambioNivel.CamNiv5;
import VerificaCambioDatos.CambioNivel.CamNiv6;
import VerificaCambioDatos.CambioNivel.CamNiv7;
import VerificaCambioDatos.CambioNivel.CamNiv8;
import VerificaCambioDatos.CambioPulso.CamPulso;
import VerificaCambioDatos.CambioPulso.CamPulso2;
import VerificaCambioDatos.CambioPulso.CamPulso3;
import VerificaCambioDatos.CambioPulso.CamPulso4;
import VerificaCambioDatos.CambioPulso.CamPulso5;
import VerificaCambioDatos.CambioPulso.CamPulso6;
import VerificaCambioDatos.CambioPulso.CamPulso7;
import VerificaCambioDatos.CambioPulso.CamPulso8;
import VerificaCambioDatos.CambioSaturacion.CamSat;
import VerificaCambioDatos.CambioSaturacion.CamSat2;
import VerificaCambioDatos.CambioSaturacion.CamSat3;
```

```
import VerificaCambioDatos.CambioSaturacion.CamSat4;
import VerificaCambioDatos.CambioSaturacion.CamSat5;
import VerificaCambioDatos.CambioSaturacion.CamSat6;
import VerificaCambioDatos.CambioSaturacion.CamSat7;
import VerificaCambioDatos.CambioSaturacion.CamSat8;
import VerificaCambioDatos.TraeNombre.TraeNom;
import VerificaCambioDatos.TraeNombre.TraeNom2;
import VerificaCambioDatos.TraeNombre.TraeNom3;
import VerificaCambioDatos.TraeNombre.TraeNom4;
import VerificaCambioDatos.TraeNombre.TraeNom5;
import VerificaCambioDatos.TraeNombre.TraeNom6;
import VerificaCambioDatos.TraeNombre.TraeNom7;
import VerificaCambioDatos.TraeNombre.TraeNom8;


public class ControlActivity extends Activity {


    public static final String DIR_IP = "192.168.1.11";
    String nomsv,satsv,pulsv,nivsv,estsv;
    String nomsv2,satsv2,pulsv2,nivsv2,estsv2;
    String nomsv3,satsv3,pulsv3,nivsv3,estsv3;
    String nomsv4,satsv4,pulsv4,nivsv4,estsv4;
    String nomsv5,satsv5,pulsv5,nivsv5,estsv5;
    String nomsv6,satsv6,pulsv6,nivsv6,estsv6;
    String nomsv7,satsv7,pulsv7,nivsv7,estsv7;
    String nomsv8,satsv8,pulsv8,nivsv8,estsv8;
    TextView tv1,tv2,tv3,tv4,tv5,tv6,tv7,tv8,tv9,tv10;
    TextView tv11,tv12,tv13,tv14,tv15,tv16,tv17,tv18,tv19,tv20;
    TextView tv21,tv22,tv23,tv24;
    TableRow tr,tr1,tr2,tr3,tr4,tr5,tr6,tr7,tr8;
    private MediaPlayer alarma;
```

```
    Handler handler = new Handler();
    ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(getWindow().FEATURE_NO_TITLE);
        setContentView(R.layout.control);
        tv1 = (TextView) findViewById(R.id.nomuseruno);
        tv2 = (TextView) findViewById(R.id.satuseruno);
        tv3 = (TextView) findViewById(R.id.pulseruno);
        tv4 = (TextView) findViewById(R.id.nomuserdos);
        tv5 = (TextView) findViewById(R.id.satuserdos);
        tv6 = (TextView) findViewById(R.id.pulserdos);
        tv7 = (TextView) findViewById(R.id.nomusertres);
        tv8 = (TextView) findViewById(R.id.satusertres);
        tv9 = (TextView) findViewById(R.id.pulsertres);
        tv10 = (TextView)findViewById(R.id.nomusercuatro);
        tv11 = (TextView)findViewById(R.id.satusercuatro);
        tv12 = (TextView)findViewById(R.id.pulsercuatro);
        tv13 = (TextView)findViewById(R.id.nomusercinco);
        tv14 = (TextView)findViewById(R.id.satusercinco);
        tv15 = (TextView)findViewById(R.id.pulsercinco);
        tv16 = (TextView)findViewById(R.id.nomuserseis);
        tv17 = (TextView)findViewById(R.id.satuserseis);
        tv18 = (TextView)findViewById(R.id.pulserseis);
        tv19 = (TextView)findViewById(R.id.nomusersiete);
        tv20 = (TextView)findViewById(R.id.satusersiete);
        tv21 = (TextView)findViewById(R.id.pulsersiete);
        tv22 = (TextView)findViewById(R.id.nomuserocho);
        tv23 = (TextView)findViewById(R.id.satuserocho);
        tv24 = (TextView)findViewById(R.id.pulserocho);
        tr = (TableRow)findViewById(R.id.tableRow1);
```

```
tr2 = (TableRow)findViewById(R.id.tableRow2);
tr3 = (TableRow)findViewById(R.id.tableRow3);
tr4 = (TableRow)findViewById(R.id.tableRow4);
tr5 = (TableRow)findViewById(R.id.tableRow5);
tr6 = (TableRow)findViewById(R.id.tableRow6);
tr7 = (TableRow)findViewById(R.id.tableRow7);
tr8 = (TableRow)findViewById(R.id.tableRow8);

TraeNom traeNom = new TraeNom("0");
CamNiv camNiv = new CamNiv("0");
CamSat camSat = new CamSat("0");
CamPulso camPulso = new CamPulso("0");
CamNiv2 camNiv2 = new CamNiv2("1");
TraeNom2 traeNom2 = new TraeNom2("1");
CamSat2 compsat2 = new CamSat2("1");
CamPulso2 camPulso2 = new CamPulso2("1");
CamNiv3 camNiv3 = new CamNiv3("2");
TraeNom3 traeNom3 = new TraeNom3("2");
CamSat3 compsat3 = new CamSat3("2");
CamPulso3 camPulso3 = new CamPulso3("2");
CamNiv4 camNiv4 = new CamNiv4("3");
TraeNom4 traeNom4 = new TraeNom4("3");
CamSat4 compsat4 = new CamSat4("3");
CamPulso4 camPulso4 = new CamPulso4("3");
CamNiv5 camNiv5 = new CamNiv5("4");
TraeNom5 traeNom5 = new TraeNom5("4");
CamSat5 compsat5 = new CamSat5("4");
CamPulso5 camPulso5 = new CamPulso5("4");
CamNiv6 camNiv6 = new CamNiv6("5");
TraeNom6 traeNom6 = new TraeNom6("5");
CamSat6 compsat6 = new CamSat6("5");
CamPulso6 camPulso6 = new CamPulso6("5");
CamNiv7 camNiv7 = new CamNiv7("6");
TraeNom7 traeNom7 = new TraeNom7("6");
CamSat7 compsat7 = new CamSat7("6");
CamPulso7 camPulso7 = new CamPulso7("6");
CamNiv8 camNiv8 = new CamNiv8("7");
TraeNom8 traeNom8 = new TraeNom8("7");
CamSat8 compsat8 = new CamSat8("7");
CamPulso8 camPulso8 = new CamPulso8("7");
camNiv.execute();
traeNom.execute();
camSat.execute();
camPulso.execute();
camNiv2.execute();
traeNom2.execute();
compsat2.execute();
camPulso2.execute();
camNiv3.execute();
traeNom3.execute();
compsat3.execute();
camPulso3.execute();
camNiv4.execute();
traeNom4.execute();
compsat4.execute();
camPulso4.execute();
camNiv5.execute();
traeNom5.execute();
compsat5.execute();
camPulso5.execute();
camNiv6.execute();
traeNom6.execute();
compsat6.execute();
camPulso6.execute();
camNiv7.execute();
traeNom7.execute();
compsat7.execute();
camPulso7.execute();
camNiv8.execute();
traeNom8.execute();
compsat8.execute();
camPulso8.execute();
```

```
nomsw = traeNom.getNuevonom();
nivsw = camNiv.getNivcam();
satsw = camSat.getSatcam();
pulsw = camPulso.getPulcam();
nomsw2 = traeNom2.getNuevonom2();
nivsw2 = camNiv2.getNivcam2();
satsw2 = compsat2.getSatcam2();
pulsw2 = camPulso2.getPulcam2();
nomsw3 = traeNom3.getNuevonom3();
nivsw3 = camNiv3.getNivcam3();
satsw3 = compsat3.getSatcam3();
pulsw3 = camPulso3.getPulcam3();
nomsw4 = traeNom4.getNuevonom4();
nivsw4 = camNiv4.getNivcam4();
satsw4 = compsat4.getSatcam4();
pulsw4 = camPulso4.getPulcam4();
nomsw5 = traeNom5.getNuevonom5();
nivsw5 = camNiv5.getNivcam5();
satsw5 = compsat5.getSatcam5();
pulsw5 = camPulso5.getPulcam5();
nomsw6 = traeNom6.getNuevonom6();
nivsw6 = camNiv6.getNivcam6();
satsw6 = compsat6.getSatcam6();
pulsw6 = camPulso6.getPulcam6();
nomsw7 = traeNom7.getNuevonom7();
nivsw7 = camNiv7.getNivcam7();
satsw7 = compsat7.getSatcam7();
pulsw7 = camPulso7.getPulcam7();
nomsw8 = traeNom8.getNuevonom8();
nivsw8 = camNiv8.getNivcam8();
satsw8 = compsat8.getSatcam8();
pulsw8 = camPulso8.getPulcam8();
```

```
tv1.setText(nomsw);
tv2.setText(satsw);
tv3.setText(pulsw);
tv4.setText(nomsw2);
tv5.setText(satsw2);
tv6.setText(pulsw2);
tv7.setText(nomsw3);
tv8.setText(satsw3);
tv9.setText(pulsw3);
tv10.setText(nomsw4);
tv11.setText(satsw4);
tv12.setText(pulsw4);
tv13.setText(nomsw5);
tv14.setText(satsw5);
tv15.setText(pulsw5);
tv16.setText(nomsw6);
tv17.setText(satsw6);
tv18.setText(pulsw6);
tv19.setText(nomsw7);
tv20.setText(satsw7);
tv21.setText(pulsw7);
tv22.setText(nomsw8);
tv23.setText(satsw8);
tv24.setText(pulsw8);
```

```
handler.post(runnable);
```

```
public void addUser(View v){

    Intent datos = new Intent(this, UsuarioUnoActivity.class);
    startActivity(datos);
    handler.removeCallbacks(runnable);
}
public void stopMon(View v){

    Intent datos2 = new Intent(this, UsuarioDosActivity.class);
    startActivity(datos2);
    handler.removeCallbacks(runnable);


}
Runnable runnable = () -> {

        String txts="",txts2="",txts3="",txts4="";
        String txts5="",txts6="",txts7="",txts8="";
        String txtp="",txtp2="",txtp3="",txtp4="";
        String txtp5="",txtp6="",txtp7="",txtp8="";
        String ingNom="",ingNom2="",ingNom3="",ingNom4="";
        String ingNom5="",ingNom6="",ingNom7="",ingNom8="";
        String comp="",comp2="",comp3="",comp4="";
        String comp5="",comp6="",comp7="",comp8="";
        String comppul="",comppul2="",comppul3="",comppul4="";
        String comppul5="",comppul6="",comppul7="",comppul8="";
        String compest="",compest2="",compest3="",compest4="";
        String compest5="",compest6="",compest7="",compest8="";
        String compniv="",compniv2="",compniv3="",compniv4="";
        String compniv5="",compniv6="",compniv7="",compniv8="";

        handler.removeCallbacks(runnable);
        handler = null;
        TraeNom traeNom = new TraeNom("0");

        CamNiv camNiv = new CamNiv("0");

        CamSat camSat = new CamSat("0");

        CamPulso camPulso = new CamPulso("0");
    camNiv.execute();
    nivsw = camNiv.getNivcam();
    if (nivsw != null) {
        if (nivsw.equals("uno")) {
            tr.setBackgroundColor(Color.RED);
            alarma = MediaPlayer.create(ControlActivity.this,R.raw.beep);
            alarma.start();
        }
        if (nivsw.equals("dos")) {
            tr.setBackgroundColor(Color.rgb(245, 165, 15));
        }
        if (nivsw.equals("tres")) {
            tr.setBackgroundColor(Color.GREEN);
        }
        if (nivsw.equals("cero")) {
            tr.setBackgroundColor(Color.GRAY);
            alarma = MediaPlayer.create(ControlActivity.this,R.raw.beep);
            alarma.stop();
        }
    }
        camNiv = null;
        traeNom.execute();
        nomsw = traeNom.getNuevocom();
        tv1.setText(nomsw);
        traeNom = null;
        camSat.execute();
        satsw = camSat.getSatcam();
        tv2.setText(satsw);
        camSat = null;
        camPulso.execute();
        pulsw = camPulso.getPulcam();
        tv3.setText(pulsw);
        camPulso = null;
```

```
        handler = null;
        handler = new Handler();
        handler.postDelayed(runnable,1500);


};
```

Figura B.2.6. ControlActivity.java

El anterior código muestra la actividad principal de la aplicación, donde después de las importaciones necesarias, se inicializan las variables principales a ser usadas más adelante, asi como los editores de texto donde se ubicaran los valores recibidos desde el servidor web. Luego se instancian varios objetos de clases que extienden de AsyncTask, que crea tareas asíncronas con las cuales se realizan las conexiones con el servicio web y se extraen cada uno de los datos sincronizados. Una vez instanciados se ejecutan y se reciben los resultados para luego ser puestos en los editores de texto de la interfaz y mostrarse al usuario de la aplicación. Todo esto se hace dentro de un método Runnable con el fin de repetir el código cada 1.5 segundos, para asi mantener una muestra en tiempo real de los datos de signos vitales.

Existen dos clases más que se llaman UsuarioUnoActivity.java y UsuarioDosActivity.java que se encargan de iniciar una nueva lectura y detener una lectura respectivamente. No se presenta el código porque es similar a la clase anteriormente presentada. Estas dos clases y las clases asíncronas que representan la conexión con el servidor (Las cuales se encuentran en el paquete VerificaCambioDatos) se encuentran en el código fuente entregado con este documento.

B.3. CÓDIGO APLICACIÓN DE PRUEBA.

La aplicación de prueba, es realizada en Android Studio como se hizo con la aplicación principal. La aplicación cuenta con la interfaz presentada en el capítulo anterior y a continuación se presenta el código de la clase que la controla:

MainActivity.java

```
package co.edu.unicauca.sensor.pruebas.simuladorpo;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.media.MediaPlayer;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

/**
 * Created by Diego Fernando Viveros Zambrano
 */
public class MainActivity extends AppCompatActivity {
    public static final String DIR_IP = "192.168.0.104";
    public static String tipo_variacion;
    TextView ets,etp;
    private MediaPlayer mp;
    String val_niv,val_sat,val_pul;
    Boolean salir,leer;
    GeneraSignos gs = new GeneraSignos();
    Handler handler = new Handler();

    Conectar con = new Conectar();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ets = (TextView)findViewById(R.id.setura);
        etp = (TextView)findViewById(R.id.pulsa);
        ets.setText("00");
        etp.setText("00");

    }
    public void inicio(View view){
        ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            //tipo_variacion = "normal";
            con = new Conectar("0","inicia");
            con.execute();
            salir = true;
            leer = true;
            handler.post(runnable);


        } else {
            TextView ta = (TextView)findViewById(R.id.pruebaconexi);
            String pc = "No hay conexion";
            ta.setText(pc);
        }
```

```
    public void para(View view){
        con = null;
        con = null;
        salir = false;
        leer = false;
        handler = null;
        con = new Conectar("0","parar");
        con.execute();
        reiniciarActivity(MainActivity.this);
    }
    public void alarmal(View view){
        con = null;
        con = null;
        con = new Conectar("0","alarmaI");
        con.execute();
    }
    public void alarmaII(View view){
        con = null;
        con = null;
        con = new Conectar("0","alarmaII");
        con.execute();
    }
    public static void reiniciarActivity(Activity actividad){

        Intent intent=new Intent();
        intent.setClass(actividad, actividad.getClass());
        //llamamos a la actividad
        actividad.startActivity(intent);
        //finalizamos la actividad actual
        actividad.finish();
    }
Runnable runable = () -> {
       if ((leer!=null) && leer){
           Nivel niv = new Nivel("0");
           Saturacion satu = new Saturacion("0");
           Pulso puls = new Pulso("0");
           niv.execute();
           satu.execute();
           puls.execute();
           val_niv = niv.getNivcam();
           if (val_niv.equals("tres")){
               ets.setBackgroundColor(Color.GREEN);
               etp.setBackgroundColor(Color.GREEN);
           }
           if (val_niv.equals("dos")){
               ets.setBackgroundColor(Color.rgb(245, 165, 15));
               etp.setBackgroundColor(Color.rgb(245, 165, 15));
           }
           if (val_niv.equals("uno")){
               ets.setBackgroundColor(Color.RED);
               etp.setBackgroundColor(Color.RED);
           }

           val_sat = satu.getSatcam();
           if (val_sat.equals(null)){
               ets.setText("00");
           }else{
           ets.setText(val_sat);}
           Log.i("HOM", "ESTA ES SATURACION NORMAL: " + val_sat);
           val_pul = puls.getPulcam();
           if (val_pul.equals(null)){
               etp.setText("0");
           }else{
           etp.setText(val_pul);}
           Log.i("HOM", "ESTE ES PULSO NORMAL: " + val_pul);
           niv = null;
           satu = null;
           puls = null;

       }
       if (salir) {
          handler.postDelayed(runable, 1000);
       }
    };
}
```

Figura B.3.1. MainActivity.java

En la clase anterior se puede apreciar el funcionamiento completo de la aplicación. Se trabaja de forma similar a la aplicación principal presentada en el ítem anterior B.2, destacando que se presentan 4 metodos para los cuatro botones: inicio, para, alarmaI y alarmaII. En cada método se realiza instancias a objetos de clases asíncronas para la conexión con el servidor, con las cuales se puede activar la transmisión desde la Raspberry. El código de estas clases es entregado en el código fuente junto con este documento. Más abajo en el código se encuentra un método Runnable, que contiene el llamado a las variables que contienen los signos de saturación de oxígeno y pulso cardiaco y luego son presentados en los cuadros de texto de la interfaz de la aplicación. Esto se repite cada segundo, hasta que el botón de Parar la transmisión sea presionado.

ANEXO C. DATOS ESTADISTICOS DE INGRESOS A URGENCIAS.

HOSPITAL UNIVERSITARIO SAN JOSE POPAYAN ESE
INDICADORES DE HOSPITALIZACIÓN POR MES

URGENCIAS ADULTOS 2011

HOSPITAL UNIVERSITARIO SAN JOSE POPAYAN ESE
INDICADORES DE HOSPITALIZACIÓN POR MES

URGENCIAS ADULTOS 2012

HOSPITAL UNIVERSITARIO SAN JOSE POPAYAN ESE

INDICADORES DE HOSPITALIZACIÓN POR MES

URGENCIAS ADULTOS 2013

HOSPITAL UNIVERSITARIO SAN JOSE POPAYAN ESE

INDICADORES DE HOSPITALIZACIÓN POR MES

URGENCIAS ADULTOS 2014

ANEXO D. CONSENTIMIENTO INFORMADO.

INFORMACIÓN PARA EL PACIENTE Y CONSENTIMIENTO INFORMADO

Usted ha consultado al servicio de urgencias por presentar una enfermedad, usted ha sido atendido previamente en el consultorio de Triage, quienes lo clasificaron de acuerdo a la gravedad de su enfermedad y le dijeron que aguardara en sala de espera mientras lo llaman para ser atendido por el médico o aun no ha sido llamado a la sala de Triage. En ocasionas los pacientes mientras esperan pueden agravarse de un momento a otro. Esto es posible evitarlo si a usted se le monitorea constantemente su pulso y su capacidad de oxigenar su organismo. Para saber cómo es su pulso y su oxigenación, se le colocará en uno de sus dedos de la mano, un sensor electrónico que le medirá el pulso y la oxigenación de su organismo. Junto con usted participarán 30 personas. Con este estudio esperamos poder determinar el pulso y la oxigenación del organismo de los pacientes que están en la sala de espera aguardando ser llamados para recibir la atención médica, y así evitar posibles complicaciones en su estado de salud. Su participación en esta investigación es absolutamente voluntaria y no afectará su atención médica posterior. Para este estudio, es independiente de la posterior atención que usted recibirá por parte del personal médico del servicio.

Criterios de Selección

Criterios de inclusión: Se incluirán pacientes de ambos sexos mayores de 18 años de edad, conscientes, con signos vitales estables al momento de la evaluación del triage.

Signos de Exclusión: Pacientes inconscientes, pacientes con signos vitales inestables desde el ingreso al servicio, pacientes politraumatizados con hemorragias.

Procedimientos del estudio

Si usted acepta participar, se le colocará un sensor que le tomara los datos de pulso cardiaco y nivel de oxígeno en la sangre, por medio de la luz absorbida por la sangre, que no le causará dolor ni ningún otro efecto secundario indeseado. Los datos obtenidos serán llevados a un dispositivo de procesamiento para su estudio.

Beneficios

Si usted acepta participar, no recibirá beneficio directo de los resultados de este estudio. La información obtenida de este estudio, podría ayudarnos en el futuro a mejorar la atención del paciente que consulta al servicio de urgencias.

Riesgos

Al colocársele los electrodos usted no tendrá ningún riesgo para su salud, debido a que el sensor solo tiene contacto con el dedo índice de la mano izquierda o derecha, a través diodos emisores y receptores de luz.

Responsabilidades del paciente

Usted debe permitir la realización de las pruebas. Los resultados de este procedimiento podrán ser conocidos por usted o por su representante legal.

Alternativas

No existe un método más seguro para obtener la información que estamos recopilando. Si usted escoge no participar en el estudio, su decisión será respetada y no cambiara en nada su tratamiento.

Confidencialidad

Sólo los miembros del equipo de investigación sabrán que usted está participando en el estudio. Le será colocado un sensor electrónico en sus dedos, tal como lo ilustra la figura 1. Si los resultados de este estudio son publicados, usted no será identificado por el nombre. Los registros que se hagan se harán identificándolo solo con un código, por ejemplo, POX-001, y no con su nombre; sin embargo representantes autorizados de las autoridades reguladoras podrán revisar sus registros como parte de su actividad de supervisión del estudio.

Figura 1.

Compensación

Usted no tendrá que incurrir en ningún gasto por participar en este estudio.

Personas a contactar

Si tiene cualquier pregunta acerca de este estudio o acerca de lo que debe hacer en caso de que sienta alguna molestia durante el estudio, puede comunicarse con el Dr. Jairo Alfonso Vásquez, al teléfono 3007800060.

Si tiene dudas con respecto a los derechos y deberes que tiene por su participación en este estudio, puede comunicarse con el Comité de Ética del Hospital Universitario San José, teléfono: 8234508 ext: 157

El médico responsable de esta investigación, estará disponible para responder cualquier pregunta adicional.

Dr. JAIRO ALFONSO VÁSQUEZ

Oficina 145, Departamento de Morfología

Facultad de Ciencias de la Salud
Universidad del Cauca
Teléfono: 8234118 Ext.: 229-121

Terminación del estudio

Usted entiende que su participación en el estudio es **VOLUNTARIA**. En cualquier momento usted puede retirar su consentimiento a participar en el estudio, sin que su tratamiento médico posterior se vea afectado. Su médico también podrá detener el estudio por razones médicas u otras razones.

Autorización para uso de los datos obtenidos en este estudio

Se le solicita la autorización al participante para que los datos obtenidos en este estudio, puedan ser utilizados para realizar otros estudios, previa Aprobación del Comité de Ética del Hospital Universitario San José.

Aceptación

La resolución 008430 del Ministerio de Salud Nacional exige consignar el nombre del paciente o participante, su firma o huella digital, su identificación personal. Exige también la firma de dos testigos con su nombre, dirección y fecha de la firma, y que indique su parentesco con el paciente. El investigador o responsable de obtener el consentimiento informado, debe firmar y consignar sus datos de identificación personal, lugar y fecha de obtención del consentimiento.

SU FIRMA (O HUELLA DIGITAL) INDICA QUE USTED HA DECIDIDO PARTICIPAR VOLUNTARIAMENTE EN ESTE ESTUDIO HABIENDO LEIDO (O ESCUCHADO) LA INFORMACIÓN ANTERIOR.

	Nombre completo (Letra imprenta)	**Lugar y fecha (dd/mm/aa)**	**Firma o huella digital**
Paciente o participante C.C.			

ANEXO E. CERTIFICACION DE LAS PRUEBAS REALIZADAS.

Señores

FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES

Referencia: Certificación de pruebas de trabajo de grado

Cordial saludo

Por medio de la presente certifico que el estudiante DIEGO FERNANDO VIVEROS ZAMBRANO, identificado con código 06071072, estudiante activo del programa de Ingeniería Electrónica y Telecomunicaciones, quien está realizando el trabajo de grado titulado SISTEMA PARA MULTIPLES MEDICIONES DE PULSIOXIMETRÍA ORIENTADO AL MONITOREO DE PACIENTES PARA APOYO EN ZONA DE TRIAGE EN LA SALA DE ESPERA DE URGENCIAS. , realizó las pruebas de funcionamiento de su sistema en la sala de espera de urgencias del Hospital Universitario San José bajo mi supervisión entre los días 17 y 19 de diciembre del presente año, con el fin de cumplir con el objetivo específico número 3 "Realizar pruebas de simulación del sistema desarrollado y en un ambiente real con la supervisión de un médico".

La metodología para realizar las pruebas consistió básicamente en tomar los signos vitales de saturación de oxígeno y pulso cardiaco con un sensor de pulsioximetría a personas que esperan por atención en la sala de espera de urgencias y luego de esto, se mostraban los signos en dispositivos móviles y una pantalla junto con posibles alarmas. Todo esto fue posible de realizar con el consentimiento informado firmado por los pacientes.

Atentamente:

Nombre: Arnulfo Orobio

Cédula: 10546896

Cargo: Urgentólogo

Popayán- Cauca. Ciudad Universitaria. 17 de Diciembre de 2016

Señores

FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES

Referencia: Certificación de pruebas de trabajo de grado

Cordial saludo

Por medio de la presente certifico que el estudiante DIEGO FERNANDO VIVEROS ZAMBRANO, identificado con código 06071072, estudiante activo del programa de Ingeniería Electrónica y Telecomunicaciones, quien está realizando el trabajo de grado titulado SISTEMA PARA MÚLTIPLES MEDICIONES DE PULSIOXIMETRÍA ORIENTADO AL MONITOREO DE PACIENTES PARA APOYO EN ZONA DE TRIAGE EN LA SALA DE ESPERA DE URGENCIAS. , realizó las pruebas de funcionamiento de su sistema en ambiente simulado, en la sala de televisión digital de la facultad de Ingenierías el día 17 de diciembre del presente año, con el fin de cumplir con el objetivo especifico número 3 "Realizar pruebas de simulación del sistema desarrollado y en un ambiente real con la supervisión de un médico".

La metodología para realizar las pruebas consistió básicamente en tomar los signos vitales de saturación de oxigeno y pulso cardiaco con un sensor de pulsioximetría a un asistente a la prueba y luego de esto, se mostraban los signos en dispositivos móviles y una pantalla junto con posibles alarmas. Además se realizó introducción de datos por medio de sensores simulados por una aplicación móvile.

Atentamente:

Andres F. Benavides R.

Nombre: Andres Fernando Benavides Ruiz

Cédula: 1061727521

Cargo: Contratista Universidad del cauca (Ingeniero civil)

ANEXO F: ENCUESTA A LOS PACIENTES.

Encuesta de apreciación del sistema de monitoreo por pulsioximetría a distancia.

1. Durante el tiempo que permanece en la sala de espera de urgencias, ¿Usted prefiere ser monitoreado?

   SI NO

2. Califique del 1 al 10 la importancia de utilizar este sistema de monitoreo en la sala de urgencias durante el tiempo que está usted en la sala de urgencias. Donde 10 significa muy importante y 1 poco importante.

Importancia	

3. Califique del 1 al 10, cuan cómodo se siente al usar el sensor de pulsioximetría inalámbrico. 10 significa muy cómodo y 1 muy incomodo

Comodidad	

4. ¿Piensa usted que es importante aplicar tecnología a la salud con el fin de solucionar problemas de atención?

   SI NO

ANEXO G. AVAL DEL HOSPITAL UNIVERSITARIO SAN JOSE DE POPAYAN

El nombre en el aval es diferente debido a que desde que se presentó el proyecto al hospital y luego cuando se presentó el proyecto a la universidad. En el proceso de aprobación del anteproyecto se realizaron cambios solicitados por la universidad.

ANEXO H: CERTIFICADO

Se realizó el curso de protección de los participantes humanos en investigación, teniendo en cuenta, que en este proyecto se realizaron pruebas con participantes humanos, por lo cual se anexa el siguiente certificado.

Certificado de finalización

La Oficina para Investigaciones Extrainstitucionales de los Institutos Nacionales de Salud (NIH) certifica que **Diego Fernando Viveros Zambrano** ha finalizado con éxito el curso de capacitación de NIH a través de Internet "Protección de los participantes humanos de la investigación".

Fecha de finalización: 06/16/2014

Número de certificación: 338322

DEDICATORIAS

Dios

Mi Familia

Mi esposa y mi hijo

Director: MSc. Delio Eduardo Henríquez Cabrera

Codirectores: MSc. Jairo Alfonso Vásquez López

PhD. Rubiel Vargas Cañas

Miembros del Comité de Ética del H.U.S.J.

Coordinador de Urgencias del H.U.S.J.

Directora del H.U.S.J.

Personal médico y de enfermería del servicio de Urgencias del H.U.S.J.

Pacientes participantes

Jurados: MSc. Judy Cristina Realpe Chamorro

MSc. Diego Gerardo Gómez Orozco