UNIVERSIDAD DEL AZUAY
FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA ELECTRÓNICA

“Diseño e implementación de un sistema de monitoreo remoto de señales biológicas, electrocardiograma, oxímetro de pulso, tensiómetro digital.”

TRABAJO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO.

AUTORES:
FREDDY XAVIER GORDILLO PADILLA.

DIRECTOR:
LCDO. LEOPOLDO CARLOS VÁSQUEZ RODRÍGUEZ.

CUENCA, ECUADOR
2014

DEDICATORIA

A mi madre que siempre me ha apoyado incondicionalmente en todos los proyectos que he emprendido en mi vida y que ha estado a mi lado en los buenos y malos momentos, por esto y por muchas cosas más le dedico este logro muy importante de mi vida.

Xavier Gordillo P.

AGRADECIMIENTO

A mi familia por todo el apoyo anímico y económico que siempre me han entregado, a Erika Córdova por todo el apoyo, paciencia y amor que me ha tenido durante mi vida universitaria, a los profesores de la escuela de Ingeniería Electrónica por su enseñanzas y dedicación que han tenido conmigo y mis compañeros, a Israel Mosquera por la colaboración en la ejecución del presente trabajo y en general a mis compañeros y amigos con los que hemos compartido dificultades y alegrías durante nuestros estudios en la prestigiosa Universidad del Azuay.

Xavier Gordillo P.

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO REMOTO DE SEÑALES BIOLÓGICAS, ELECTROCARDIOGRAMA, OXÍMETRO DE PULSO, TENSIÓMETRO DIGITAL.

RESUMEN

El propósito general del presente proyecto de tesis es unificar, la tecnología de servicios web y software libre, basados en gestión de base de datos con dispositivos de medición de señales biomédicas ambulantes, concretamente, Electrografía, Oximetría y Tensiometría de pulso y dispositivos móviles basados en sistema operativo Android.

Los elementos de control medición, gestión y comunicación que intervienen en el presente tema de tesis son microcontroladores, filtros digitales, amplificadores operacionales de precisión y herramientas de software tales como NetBeans y AppInventor; cuya interacción conforman, en un todo, un equipo biomédico completo capaz de hacer usos de servicios web con distintos tipos de usuarios simultáneamente, tanto en un ordenador convencional como en dispositivos móviles.

PALABRAS CLAVE: Electrografía, Oximetría, Tensiometría, Servicio Web, Arduino, Java, Android.

Lcdo. Leopoldo Carlos Vásquez Rodríguez
DIRECTOR DE TESIS.

Ing. Francisco Eugenio Vásquez Calero
DIRECTOR DE ESCUELA.

Freddy Xavier Gordillo Padilla
AUTOR.

DESIGN AND IMPLEMENTATION OF A REMOTE MONITORING SYSTEM OF BIOLOGICAL SIGNALS, EKG, PULSE OXIMETER, DIGITAL BLOOD PRESSURE MONITOR

ABSTRACT

The general purpose of this paper is to unify web services technology and free software based on database management with devices to measurement mobile biomedical signals such as electrography, pulse oximeter and pulse tensiometer with mobile devices based on Android operating system.

The elements of control, measurement, management, and communication that are featured in this research paper are microcontrollers, digital filters, precision operational amplifiers and software tools such as NetBeans and App Inventor. Their interaction comprises as whole, complete biomedical equipment with the capacity to use simultaneously web services with different users in both conventional computers and mobile devices.

Keywords: Electrographic, Oximetry, Tensiometry, Web Service, Arduino, Java, Android.

Lcdo. Leopoldo Carlos Vásquez Rodríguez
THESIS DIRECTOR

Ing. Francisco Eugenio Vásquez Calero
SCHOOL DIRECTOR

Freddy Xavier Gordillo Padilla
AUTHOR

UNIVERSIDAD DEL AZUAY
DPTO. IDIOMAS

Translated by,
Lic. Lourdes Crespo

ÍNDICE DE CONTENIDO.

ÍNDICE DE ANEXOS

ÍNDICE DE FIGURAS.

Gordillo Padilla Freddy Xavier.
Trabajo de Graduación.
Lcdo. Leopoldo Carlos Vásquez Rodríguez.
Octubre 2014.

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO REMOTO DE SEÑALES BIOLÓGICAS, ELECTROCARDIOGRAMA, OXÍMETRO DE PULSO, TENSIÓMETRO DIGITAL.

INTRODUCCIÓN.

Es innegable el avance cada vez más raudo de la medicina y logros en el campo médico en general, ya sea si se habla de medicamentos, técnicas terapéuticas, tratamiento de enfermedades, terapias de recuperación o verificación del funcionamiento de órganos, fluidos, aparatos, y demás; mismos progresos que derivan, en un todo, en una mejor calidad y expectativa de vida.

Por otro lado, así como es absolutamente indiscutible el progreso de la ciencia en el campo médico, es igualmente innegable el hecho de que los avances médicos y tecnológicos van de la mano; y es que la tecnología ha sido el puente que ha permitido a la ciencia en sus distintas ramas, la evolución como tal.

Partiendo de esta premisa es fácil imaginarse el camino que toma el presente tema de tesis, sin embargo, ya que ambos temas, tanto la medicina como la tecnología, son extremadamente amplios, se ha de puntualizar en que aspecto, tanto en el tema medico como el tecnológico, como se van a unificar los conceptos, para consecuentemente, hacer real un proyecto.

Los temas médicos sobre los cuales se va a hacer un escrutinio son, puntualmente, la electrocardiografía, oximetría y tensiometría, a su vez, nombrando estas tres ramas implícitamente se esclarece también los aspectos tecnológicos sobre los cuales se van aplicar mejoras, lógicamente son, los equipos electromédicos utilizados para medir o cuantificar los valores referentes a cada uno de las ciencias médicas al principio mencionadas.

La tecnología electromédica en cuanto a electrocardiografía, oximetría de pulso, y tensiometría, ya ha sido generada hace algún tiempo, el objetivo del presente tema de

tesis no es simplemente generar este tipo de equipos, sino optimizar, mejorar, enriquecer, y en general, dotar a dichos equipos ya existentes de cualidades altamente útiles, como lo son servicios de consulta web en tiempo de real, registro de usuarios e información de usuarios, almacenamiento de información en base de datos para consulta, y demás servicios que tienen que ver con programación web y bases de datos, obviamente dando por sentado ya contar con un equipo electrocardiográfico, oximétrico, y tensiométrico que generen mediciones con un margen de error mínimo.

Todas estas características ya mencionadas dotan a los equipos electromédicos de cualidades ventajas extremadamente útiles, tales como:

- Ahorro de tiempo tanto en la toma de mediciones como en la difusión de la información a los profesionales encargados de la misma sin importar su ubicación.
- Aumento de eficiencia en la forma de llevar la información de las mediciones de cada uno de los pacientes por parte del profesional.
- Minimización de errores de interpretación y errores en general.

Una vez expuesto lo anterior, es obvia la justificación del presente tema, mismo que se pasa a exponer a continuación.

CAPÍTULO 1

MARCO TEÓRICO.

1.1 Electrocardiógrafo.

Es un equipo electromédico que permite obtener un registro de la actividad eléctrica del corazón para su posterior análisis. Conocido como ECG, el electrocardiograma es uno de los equipos más utilizados en cardiología para estudios electrofisiológicos del corazón, debido a que su uso se ejecuta a través de un método no invasivo que permite registrar la actividad eléctrica del músculo cardiaco desde la superficie del cuerpo del paciente (dermis) (Toloza, http://www.dalcame.com/, 2005).

1.1.1 Fisionomía.

1.1.1.1 Estructura básica del corazón.

El corazón humano, y de los mamíferos en general posee cuatro cavidades que son:

- Aurícula Derecha.
- Aurícula Izquierda.
- Ventrículo Derecho.
- Ventrículo Izquierdo.

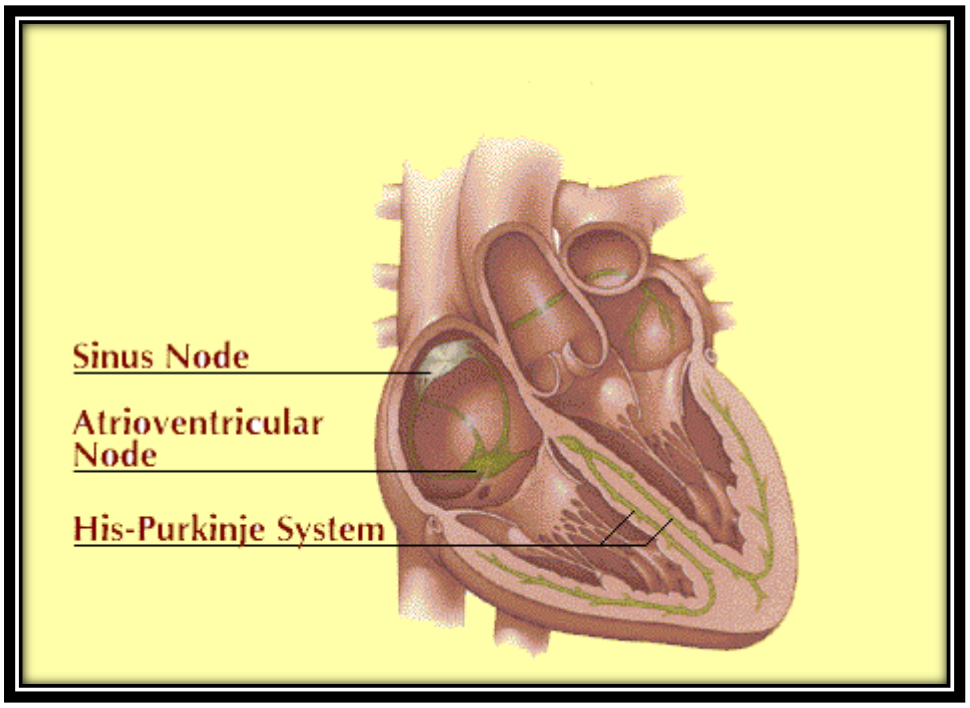
Las paredes de estas cuatro cavidades están formadas por células eléctricamente excitables conocidas como células musculares cardiacas.

Se puede dividir al corazón en dos mitades bien diferenciadas con una cavidad auricular y ventricular en cada mitad, cabe resaltar que existe comunicación entre aurícula y ventrículo dentro de cada mitad pero no existe comunicación entre las dos mitades (FIGURA 1).

La labor que efectúa el corazón es semejante a la de una bomba hidráulica, una en cada mitad. La mitad derecha recibe sangre venosa con predominio de Dióxido de Carbono en ella (CO_2), y la bombea a través de una contracción hacia los pulmones para ser purificada, por otro lado, en la mitad izquierda se recibe la sangre arterial abundante en Oxigeno (O_2) proveniente de los pulmones y la bombea a todo el organismo con

una contracción final a través de la arteria aorta (Toloza, http://www.dalcame.com/, 2005).

FIGURA 1. Fisionomía del corazón.



Fuente: (Toloza, 2005).

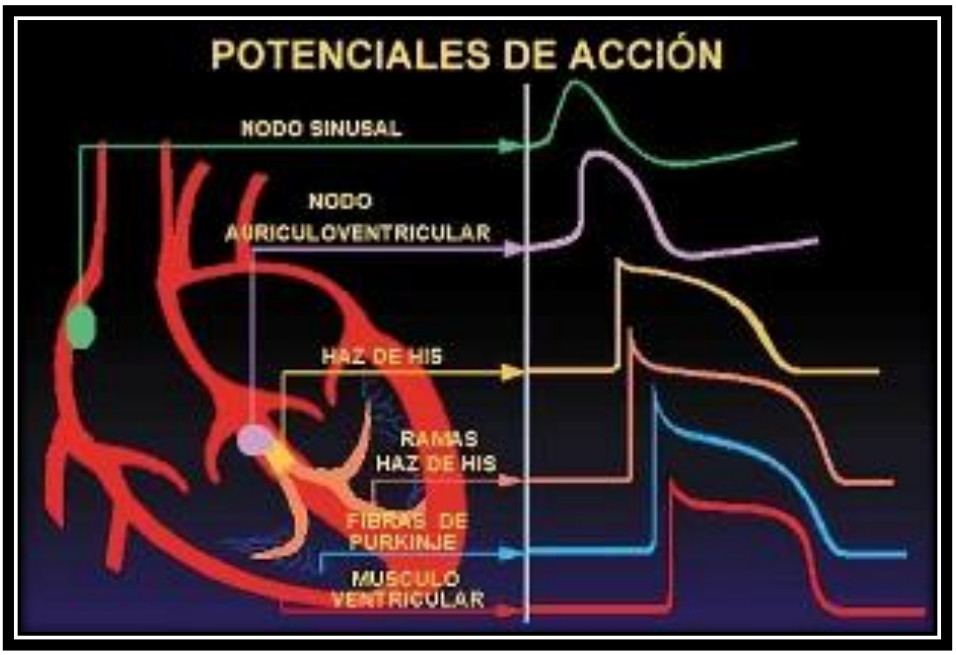
1.1.1.2 El Nodo SA.

El nódulo sinoauricular o de Keith y Flack (nodo SA) es una de las estructuras donde se origina el impulso eléctrico que da origen a un latido cardíaco.

1.1.1.3 Potenciales de acción de las Células Cardíacas.

A raíz de una actividad sincrónica espontánea en el nodo sino auricular se desencadenan los diferentes potenciales de acción en las distintas estructuras celulares del corazón, los registros mostrados en el electrocardiograma corresponden al resultado de la superposición de las ondas generadas por dichos potenciales de acción (FIGURA 2). Las células capaces de generar potenciales de acción espontáneos se llaman “células marcapasos” (Toloza, http://www.dalcame.com/, 2005).

FIGURA 2. Potenciales de acción del corazón.



Fuente: **(Toloza, 2005).**

1.1.2 Nomenclatura de las Ondas del Electrocardiograma.

Onda P: Representa la despolarización de las aurículas. Tiene una morfología redondeada, con una duración máxima de 0.10s (2.5mm) y un voltaje de 0.25 mV (2.5 mm). Es positiva en todas las derivaciones salvo en la VR del plano frontal que es negativa, y en la derivación V1 del plano horizontal (FIGURA 3).

Onda Q: La deflexión negativa inicial resultante de la despolarización ventricular, que precede una onda R (FIGURA 4). La duración de la onda Q es de 0,010 a 0,020 segundos. Normalmente no supera los 0,30 segundos.

Onda R: La primera deflexión positiva (FIGURA 3).

Onda S: La segunda deflexión negativa (FIGURA 3).

Onda T: Es la deflexión lenta producida por la repolarización ventricular (FIGURA 3).

Onda U: Es una onda generalmente positiva que sigue inmediatamente a la onda T (FIGURA 3).

Intervalo R-R: Es la extensión existente entre dos ondas RR sucesivas. (FIGURA 4).

Intervalo P-P: Es la extensión existente entre dos ondas P sucesivas. (FIGURA 4).

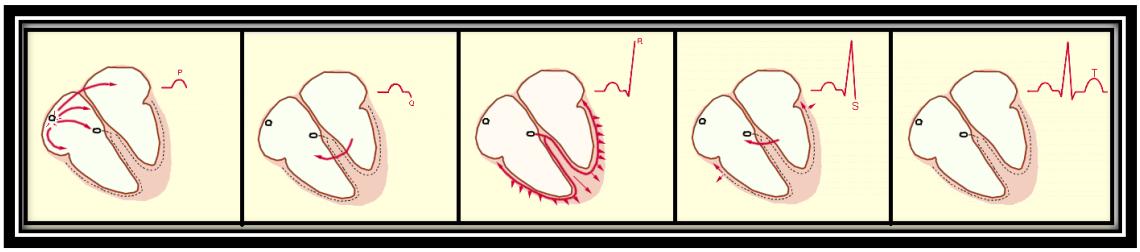
Intervalo P-R: Es la extensión existente desde el inicio de la onda P hasta el inicio de la onda Q o de la onda R. Posee una extensión aproximada entre 0.12 y 0.20 segundos. (FIGURA 4).

Intervalo QRS: Tiempo total de despolarización ventricular. Se mide desde el comienzo de la onda Q o R hasta el final de la onda S. Los valores normales fluctúan entre 0.06 y 0.10segundos. (FIGURA 4).

Intervalo Q-T: Se extiende desde el comienzo del complejo QRS hasta el final de la onda T, representa la sístole eléctrica ventricular. (FIGURA 4).

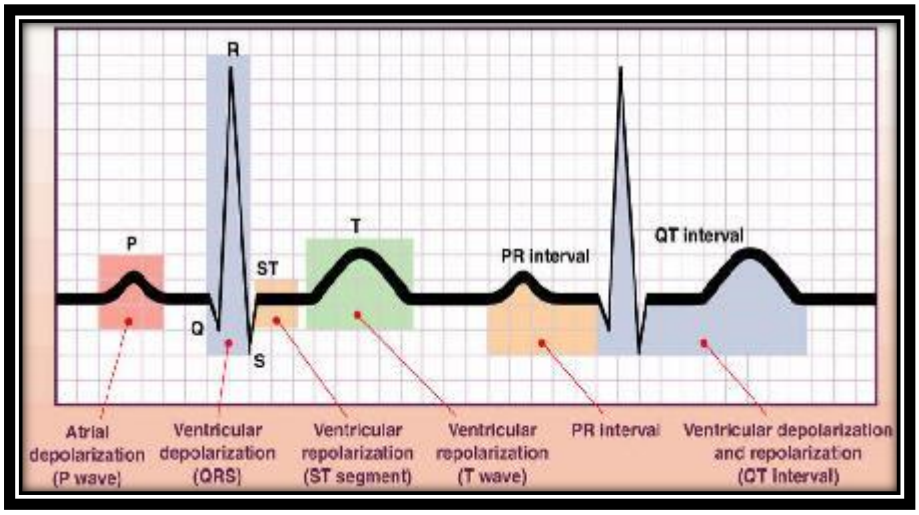
Segmento S-T: Es un periodo de inactividad que va desde el final del complejo QRS hasta el comienzo de la onda T y separa la despolarización ventricular de la repolarización ventricular. (FIGURA 4).

FIGURA 3. Forma de Onda del Electrocardiograma.



Fuente: (Toloza, 2005).

FIGURA 4. Onda del ECG típica.



Fuente: (Toloza, 2005).

1.1.3 Bioelectrodos (Sensor).

Propósitos y Particularidades: Dentro de la electromedicina, los bioelectrodos son utilizados tanto para la medición de señales bioeléctricas, como para la inyección de corriente eléctrica al tejido vivo (Toloza, http://www.dalcame.com/, 2005).

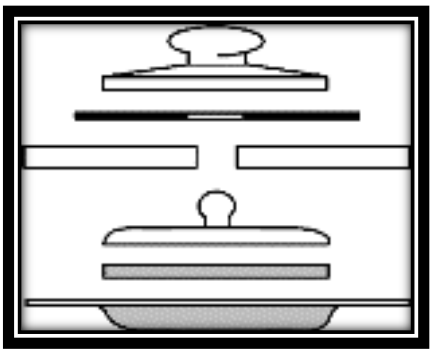
Los electrodos más comúnmente usados, y los que se utilizaran en el desarrollo del presente tema establecen un contacto óhmico con el tejido a través de un electrolito, que al entrar en contacto con el electrodo, se genera un intercambio de iones y electrones, ya que los iones de la substancia electrolítica entran en contacto con el metal, y simultáneamente los electrones del metal con los del electrolito; este intercambio por parte de ambas substancias genera una diferencia de potencial conocido con el nombre de potencial de Media Celda que no es más que el potencial derivado de la interface electrodo-electrolito (Toloza, http://www.dalcame.com/, 2005). Cabe resaltar que la calidad del potencial de media celda depende de los siguientes aspectos a tener en cuenta:

- Pureza del material con el que está fabricado el electrodo
- Interface homogénea electrodo-electrolito.
- Inmovilidad del paciente.
- Homogeneidad del electrolito.

1.1.3.1 Electrodos de Plata o Cloruro de Plata (Ag/AgCl)

Los electrodos que se usaran en el proceso, básicamente consisten en un conductor metálico en contacto con la piel a través de una pasta electrolítica colocada en la dermis del paciente que tiene como propósito establecer y mantener el contacto (FIGURA 5). Tradicionalmente el electrodo se hace de una aleación plata-níquel (Toloza, http://www.dalcame.com/, 2005). El objetivo de los electrodos obviamente consiste en recoger la señal de la superficie cutánea.

FIGURA 5. Electrodos de Ag/AgCl.



Fuente: (Toloza, 2005).

Cabe añadir algunas especificaciones técnicas para su uso, tales como:

- Impedancia inferior a 2 kΩ.
- Voltaje de desplazamiento de corriente debe ser inferior a 100 mV.
- Recuperación de Sobrecarga de desfibrilación debe ser inferior a 100 mV.
- Ruido Interno no debe ser superior a 150 mV.

1.1.4 Cable para Electrocardiografía convencional.

Una normativa internacional para conjuntos de derivaciones y cables diseñados para medir el Electrocardiograma aseguran una correcta aplicación de los electrodos, sus posiciones y código de colores según la norma de la Comisión Electrotécnica Internacional (IEC) y La Asociación Americana para el Desarrollo de la Instrumentación Médica (AAMI) se indican en la junta del cable básico asignado, como se muestra en la Tabla 1.1, a sus distintas derivaciones y extremidades correspondientes (Toloza, http://www.dalcame.com/, 2005).

Posición Electrodos	Color
Mano derecha	
Pie derecho	
Mano izquierda	
Pie izquierdo	
V1	
V2	
V3	
V4	
V5	
V6	

- **Tabla 1.1.** Código de Colores para la posición de los electrodos.

1.1.5 Método de Adquisición de Señales.

1.1.5.1 Derivaciones Electrocardiográficas.

En el electrocardiograma, las derivaciones cardiacas son el registro de la diferencia de potenciales eléctricos entre dos puntos, ya sea entre dos electrodos (derivación bipolar) o entre un punto virtual y un electrodo (derivaciones monopolares) (Toloza, http://www.dalcame.com/, 2005). Dicho de otra forma, las derivaciones electrocardiográficas son disposiciones específicas de los electrodos, llamadas simplemente derivaciones y en la práctica clínica se utilizan un número de doce (Tabla 1.2), clasificadas de la siguiente forma:

- **Derivaciones del plano frontal :** subdivididas a su vez en :

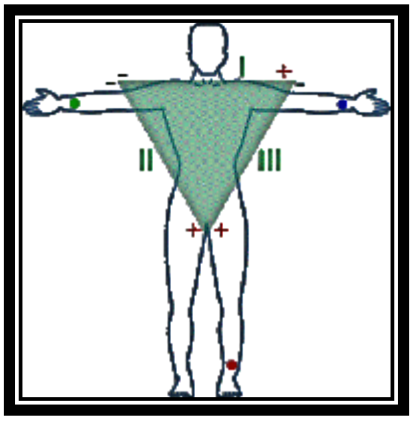
- **Bipolares.** Derivación proveniente de la diferencia de potenciales eléctricos entre dos electrodos.

- **Monopolares.** Derivación proveniente de la diferencia de potenciales eléctricos entre un punto virtual y un electrodo ver FIGURA 6 y FIGURA 7.

Derivación	Tipo	Cálculos
I	Extremidad	LA – RA
II	Extremidad	LL – RA
III	Extremidad	LL – LA
aVR	Aumentada	RA – (LA+LL)/2
aVL	Aumentada	LA – (RA+LL)/2
aVF	Aumentada	LL – (RA + LA)/2
V1	Precordial	V1-(RA+LA+LL)/3
V2	Precordial	V2-(RA+LA+LL)/3
V3	Precordial	V3-(RA+LA+LL)/3
V4	Precordial	V4-(RA+LA+LL)/3
V5	Precordial	V5-(RA+LA+LL)/3
V6	Precordial	V6-(RA+LA+LL)/3

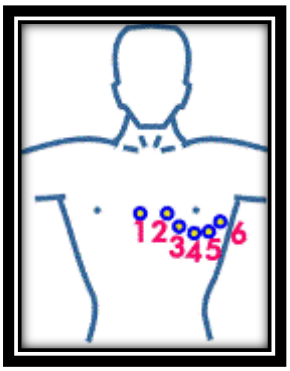
Tabla 1.2. Derivaciones precordiales.

FIGURA 6. Triangulo de Eithoven.



Fuente: (Toloza, 2005).

FIGURA 7. Representaciones de las Derivaciones Precordiales.



Fuente: (Toloza, 2005).

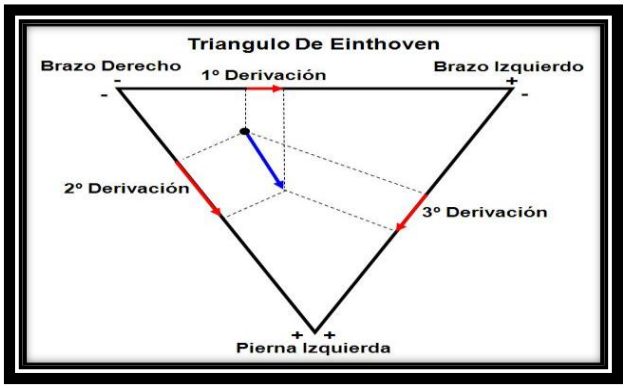
En el desarrollo del presente proyecto, se utilizará 3 derivaciones para la toma de señales y mediciones respectivas en el plano frontal, más concretamente las derivaciones bipolares basadas en los conceptos del método creado por Willen Einthoven (Toloza, http://www.dalcame.com/, 2005).

1.1.5.2 Triángulo de Einthoven.

Según Einthoven, si consideramos al corazón una fuente de corriente y al cuerpo un conductor, se puede imaginar un triángulo alrededor del corazón en cuyos lados se proyectarían los potenciales eléctricos generados por el músculo cardiaco, entonces, de acuerdo al comportamiento eléctrico del corazón y a su proyección en las respectivas extremidades se asignaría polaridad a las extremidades que vienen siendo los vértices del triángulo. Colocando electrodos en los vértices; podían tomarse tres

diferencias de potencial entre los electrodos, y de esta manera, podría describirse acertadamente la actividad eléctrica del musculo cardiaco en un plano paralelo a la superficie frontal del cuerpo. (FIGURA 8.) (Toloza, http://www.dalcame.com/, 2005).

FIGURA 8. Asignación de Polaridad y derivaciones en triangulo de Eithoven de acuerdo al comportamiento eléctrico del corazón.



Fuente (Ramon, 2013).

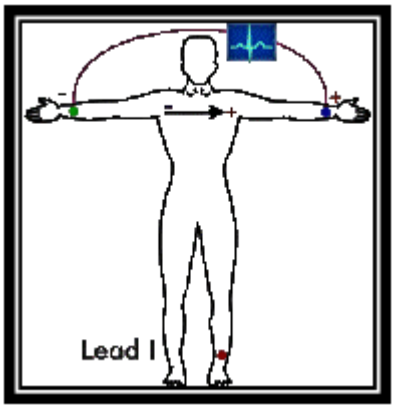
Como vemos, el brazo derecho (RA) presenta una polaridad negativa puesto que la base del corazón se proyecta sobre él. El brazo izquierdo (LA) recibe potenciales altos de la pared lateral del ventrículo izquierdo, que se aproximan a dicho miembro y originan su polaridad positiva.

Por otro lado, la pierna izquierda (LL) recibe los potenciales de la cara diafragmática del corazón, formada por las paredes de ambos ventrículos, por lo que recibe, al igual que el brazo izquierdo, su polaridad positiva (Toloza, http://www.dalcame.com/, 2005).

A continuación se describirán las derivaciones para la toma del electrocardiograma. Las primeras derivaciones fueron D1, D2 y D3, conocidas como derivaciones estándar o de Einthoven. Son derivaciones bipolares en las que cada una de ellas utiliza dos electrodos que registran la diferencia de potencial eléctrico entre dos puntos del triángulo (Toloza, http://www.dalcame.com/, 2005), siendo:

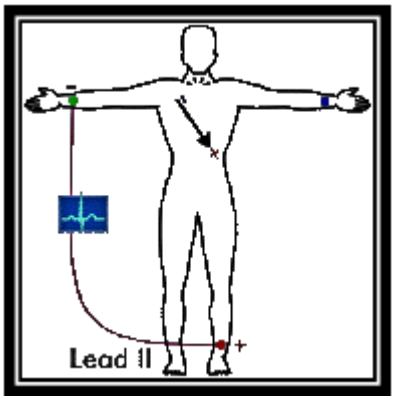
DI = LA – RA (FIGURA 9).
DII = LL - RA (FIGURA 10).
DIII = LL- LA (FIGURA 11).

FIGURA 9. Derivación I.



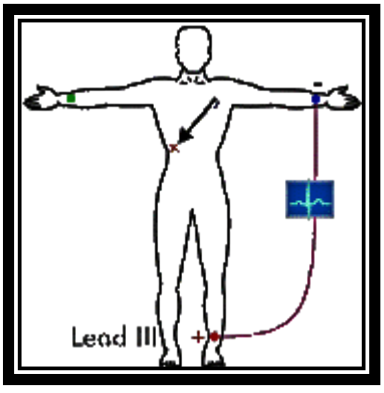
Fuente: (Toloza, 2005).

FIGURA 10. Derivación II.



Fuente: (Toloza, 2005).

FIGURA 11. Derivación III.



Fuente: (Toloza, 2005).

1.2 Oxímetro.

1.2.1 Conceptos Generales

1.2.1.1 Hemoglobina.

Es una proteína soluble (globulina) presente en la sangre constituyendo aproximadamente un 35% de su peso. Es la encargada de transportar el oxígeno desde los órganos respiratorios hasta los tejidos, junto con las cadenas de proteínas que la conforman (grupo hemo), posee un ion de hierro (Fe) lo que permite unir de forma reversible moléculas de oxígeno (Brandan, 2008).

Eventualmente pueden presentarse en la sangre concentraciones anormales y perjudiciales de hemoglobina (Brandan, 2008), lo que caracterizan diversas patologías relacionadas con la sangre tales como

- Carboxihemoglobina (HbCO)
- Metahemoglobina (MetHb)
- Hemoglobina total

1.2.1.2 Oximetría.

En general podemos definir a la oximetría como la medición del oxígeno (O2) en la sangre, concretamente, la cantidad de oxigeno presente en la hemoglobina sanguínea (oxihemoglobina HbO2) que circula a través de los vasos sanguíneos, y, a su vez, comparándola con un valor referencial llamado valor de Saturación parcial de Oxigeno (SpO2), que es inversamente proporcional al valor total de la hemoglobina presente (SYLVIA PALACIOS M., 2010), lo que expresado matemáticamente vendría siendo:

SaO2 = Concentración de oxihemoglobina (HbO2)/Concentración total de hemoglobina.

1.2.2 Oximetría de Pulso.

1.2.2.1 Método.

Un oxímetro de pulso es un equipo electromédico capaz de medir la saturación de oxigeno (SpO2) en la sangre a través de un método no invasivo, a diferencia de la oximetría de tipo fibróptica donde se introduce un catéter en el paciente; a su vez, el oxímetro de pulso brinda resultados relativamente instantáneos por lo que se considera una técnica de monitoreo en tiempo real (SYLVIA PALACIOS M., 2010).

1.2.2.2 Fundamento teórico.

Las técnicas de oximetría de pulso se basan principalmente en conceptos relacionados con espectrofotometría, concretamente la ley de Berrlamber (FIGURA 12), cuyo enunciado textualmente dice "La intensidad luminosa que se absorba al pasar por una solución es proporcional a la concentración de la molécula en dicha solución" (Kubota., 2009). En otras palabras, la intensidad de luz que atraviesa un cuerpo es inversamente proporcional a la concentración molecular de dicho cuerpo, que expresado matemáticamente vendría siendo:

$$I_0 = I_1 e^{-\propto cl}$$

En donde:

Io = Intensidad de luz incidente.

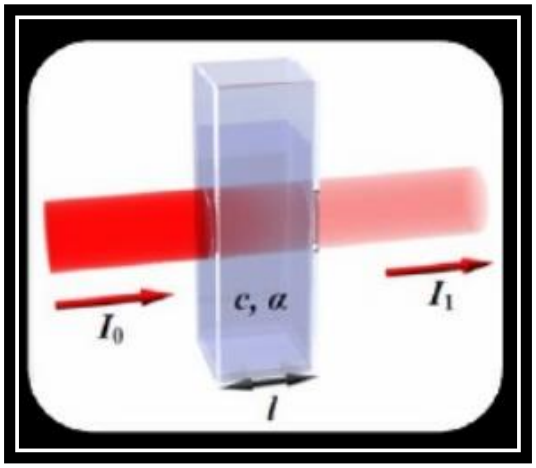
I_1= Intensidad de luz que atraviesa el cuerpo

l =distancia de la luz que atraviesa por el cuerpo

c = concentración

α = coeficiente de absorción

FIGURA 12. Ilustración del teorema de Beer Lambert.



Fuente: (Pérez, 2010).

1.2.2.3 Sensores.

Para la medición de SpO2 el oxímetro de pulso consta de dos sensores luminosos (diodos LED) emisores de luz a diferentes longitudes de onda, el uno rojo (longitud de onda λ= 650 nm), y el otro cerca del valor del infrarrojo (longitud de onda λ= 900 nm), de ahí que los sensores no pueden ser intercambiables entre equipos de marcas o características diferentes.

Dichos sensores van colocados en el dedo de la mano o el lóbulo de la oreja, y en el empeine en caso de los neonatos, al momento que la luz roja incida en el dedo o lóbulo del paciente, cierto flujo lumínico atravesara el mismo, esta intensidad luminosa será recibida por el led infrarrojo cuya potencia lumínica registrará la densidad de la solución que atraviesa la luz, para las lecturas del oxímetro (Cueva, 2008) (FIGURA 13).

FIGURA 13. Pulsioxímetro de dedo.

Fuente: (Multstock, 2012).

La diferenciación óptica de la sangre rica en O_2 (arterial), de la pobre en O_2 (venosa) es efectiva en un Oxímetro gracias a dos circunstancias:

- La trasmisión de la luz a través de la sangre oxigenada (rica en HbO_2) es máxima a una longitud de onda roja (640 a 660 nm) y casi nula para sangre pobre en HbO_2.
- La trasmisión de la luz a través de la sangre oxigenada (rica en HbO_2) y no oxigenada (pobre en O_2) es muy parecida a una longitud de onda infrarroja de 805 nm (Cueva, 2008).

1.3 Tensiómetro.

1.3.1 Presión Arterial

También llamada tensión arterial es la presión que el caudal sanguíneo ejerce contra las paredes de las arterias por donde se transporta la sangre, haciendo posible la circulación sanguínea como tal. La presión arterial es indispensable para que los diferentes órganos del cuerpo reciban nutrientes y oxígeno (Pasquier, 2013).

La tensión arterial está compuesta de dos valores originados en los latidos cardiacos y son los siguientes:

- Sístole Cardiaca : Cuando el corazón empieza un latido, este se contrae, lo que origina el movimiento conocido como sístole, en esta etapa del latido cardiaco, se expulsa cierto volumen sanguíneo a través del ventrículo izquierdo, la medición de este volumen genera el primer valor de tensión arterial (Sardiñas, 2008).

- Diástole Cardiaca: Este movimiento corresponde al espacio entre latidos cardiacos inmediatamente después del Sístole, en donde, las paredes de las arterias quedan distendidas, con un volumen sanguíneo inferior al primero, que constituye el segundo valor de tensión arterial (Sardiñas, 2008).

A su vez, existen otros factores indirectos que inciden en los valores de tensión arterial tales como:

- Resistencia Vascular: Es una fuerza que tiende principalmente a reducir el diámetro de las arterias oponiéndose al flujo sanguíneo y aumentando la presión en las mismas.
- Volemia: Corresponde al volumen total de la sangre en todo el sistema circulatorio, su valor nominal se encuentra entre 5 y 6 litros en un adulto promedio, en donde, una variación anormal tanto superior como inferior puede repercutir directamente en los valores de tensión arterial.

1.3.1.1 Valores de Presión Arterial.

Como ya se mencionó, existen dos valores que conforman la tensión arterial, correspondientes a los valores de tensión arterial en los latidos (tensión arterial sistólica), y los valores de tensión arterial entre latidos (tensión arterial diastólica).
Al momento de escribir un valor de tensión arterial se escriben dos números separados por un slash o guion, correspondientes a los valores de tensión diastólica y sistólica correspondientes y en ese orden. Cabe resaltar que existen agentes externos que dificultan un valor saludable de tensión arterial tales como ejercicio, niveles de stress, energizantes, tabaco, alcohol, etc.
En la práctica, los siguientes valores de presión arterial indican el estado de la misma:

- 119/79 o menos son normales
- 140/90 o más indican hipertensión arterial
- Entre 120 y 139 para el número más elevado, o entre 80 y 89 para el número más bajo es pre-hipertensión.

1.3.1.2 Patologías Relacionadas

1.3.1.2.1 Problemas de la presión arterial.

1.3.1.2.1.1 Hipertensión.

La hipertensión arterial es producida por una elevación anormal de la presión sanguínea en las arterias. Puesto que los problemas relacionados con la presión arterial no presentan síntomas, las consecuencias de la hipertensión suceden al cabo de cierto tiempo ocasionando daños en sus órganos internos e incluso la muerte.

Existen dos tipos principales de hipertensión: Primaria y secundaria

La hipertensión primaria o esencial se origina a raíz de problemas en el organismo y/o desequilibrios electrolíticos, así como también puede ser consecuencia del sobrepeso, elevado consumo de sodio, alcohol o tabaco.

La hipertensión secundaria puede ser el resultado de diversas enfermedades y afecciones, o efectos secundarios de ciertos medicamentos, así como también patologías cardíacas congénitas, enfermedades renales, del sistema endocrino y uso de hormonas (Sardiñas, 2008).

1.3.1.2.1.2 Hipotensión.

La hipotensión presenta valores inferiores a las lecturas normales. Entre las principales causas que la originan tenemos.

Ciertos medicamentos entre los que están incluidos fármacos para quimioterapia, bleomicina o interleuquina, que pueden causar un descenso en la presión arterial.

Por otro lado, conforme envejecemos, el sistema nervioso central tiene más dificultad para regular correctamente los niveles de presión arterial. De ahí que un simple cambio de posición, haría que la presión arterial descienda a un nivel por debajo del normal.

La hipotensión conocida como hipotensión ortostática puede ocurrir como resultado de un conteo de glóbulos rojos por debajo de lo normal (anemia), o deshidratación, ya

que los niveles de líquido en su cuerpo son bajos (Sardiñas, http://www.monografias.com/, 2008).

Otras causas de hipotensión pueden llegar a ser:

- Medicamentos sin regulación que controlan hipertensión.
- Cambios irregulares en el ritmo cardíaco.
- Hemorragia.

1.3.2 Métodos de adquisición.

1.3.2.1 Procedimiento mecánico tradicional.

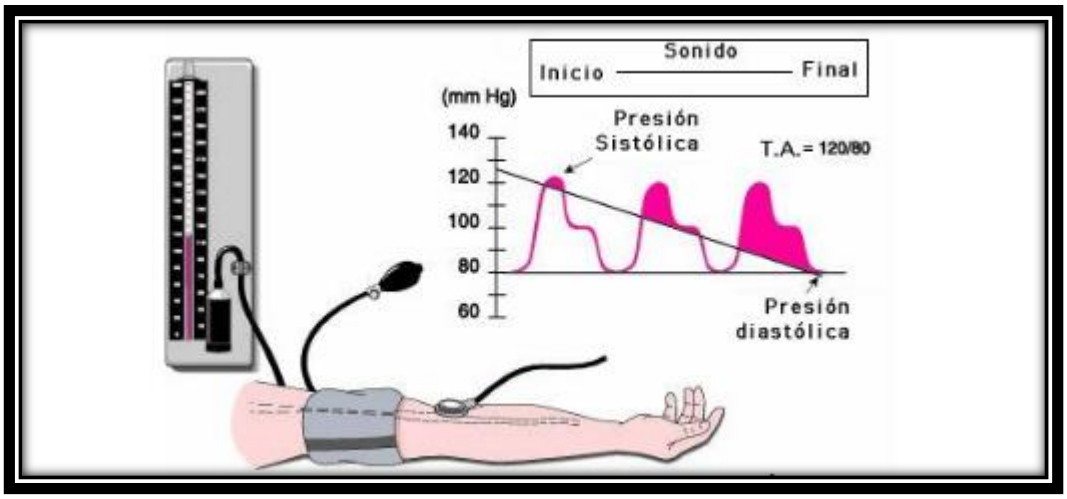
Tradicionalmente con el equipo convencional el procedimiento que se sigue para la adquisición de un valor aproximado de presión arterial a un paciente es el siguiente:

Después de recostar al paciente o ponerlo en una posición de relax, se procede a ubicar el equipo de medición preferiblemente en el brazo desnudo del paciente. Un dispositivo conocido con el nombre de manguito se aplica en la mitad del brazo (el borde inferior queda unos 2 a 3 cm sobre el pliegue cubital) aproximadamente a la altura del corazón. Debe quedar bien aplicado y no suelto (ya que esto último favorecería lecturas falsamente elevadas). Una bolsa de goma debe quedar ubicada de tal forma que justo la mitad de ella quede sobre la arteria braquial.

Conviene palpar el pulso radial durante el procedimiento para saber hasta dónde subir la presión al inflar el manguito, Desde el momento que desaparece el pulso radial, se sigue subiendo la presión unos 30 mm de Hg adicionales. Esto evita comprimir el brazo más de lo necesario. Al desinflar el manguito, se capta el momento en que nuevamente se palpa el pulso y ésta es la presión sistólica palpatoria. Se esperan de 15 a 30 segundos con el manguito desinflado, luego se repite la medición, pero esta vez teniendo la campana del estetoscopio apoyada en el pliegue cubital, sobre la arteria braquial (FIGURA 14). Se infla el manguito unos 30 mm de Hg sobre la presión sistólica palpatoria y se desinfla lentamente mientras se ausculta. La aparición de los primeros ruidos correspondientes a latidos del pulso determina la presión sistólica auscultatoria. Tanto el registro obtenido por la palpación como por la auscultación

deben ser parecidos. De no ser así, se registra como presión sistólica, el valor más elevado (http://www.geosalud.com/, s.f.).

FIGURA 14. Esquema representativo de la toma de la medida de la presión arterial de modo manual.



Fuente: (http://www.geosalud.com/, n.d.).

Después de identificar la presión sistólica auscultatoria, se sigue desinflando el manguito hasta que desaparecen los ruidos. Este momento corresponde a la presión diastólica. En ocasiones, primero los ruidos se atenúan y luego desaparecen. En general se considera como la presión diastólica el momento en que los ruidos desaparecen.

1.3.2.2 Método digital

La forma no invasiva para medir digitalmente la presión arterial se basa en el método oscilométrico que consiste en el monitoreo de las vibraciones o variaciones de una señal de presión a través de una banda inflable colocada a nivel del brazo del paciente. A partir del análisis de dichas oscilaciones en la señal, los respectivos valores sistólicos, diastólicos y medio de presión arterial.

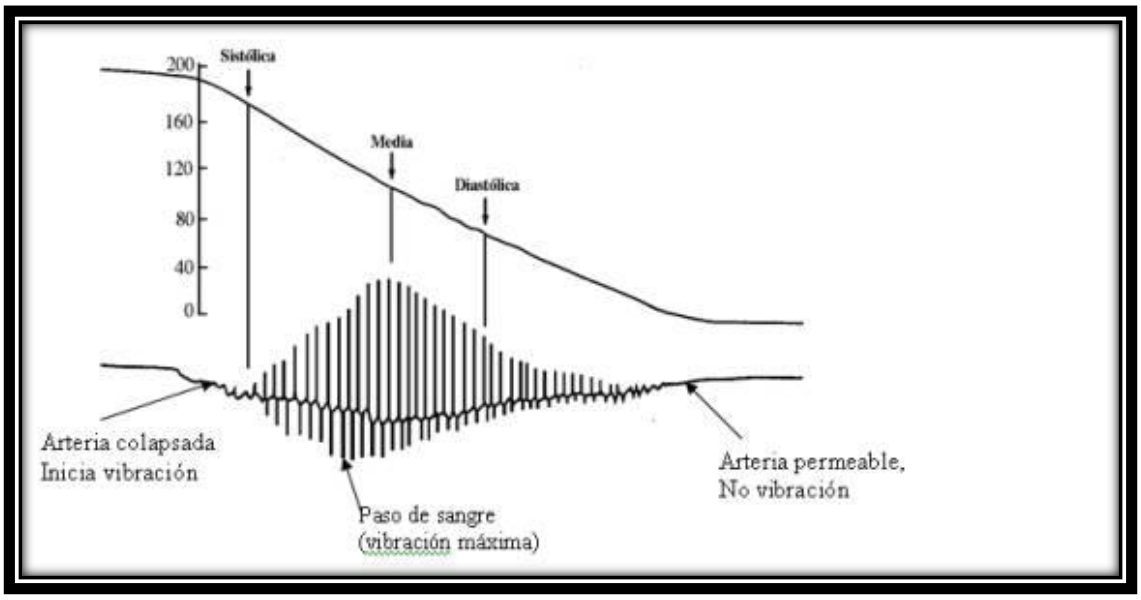
El sistema consta de un motor y un filtro que permite que la banda se infle hasta un determinado punto por encima de la presión sistólica donde las paredes de la arteria comiencen a emitir vibraciones derivadas de la circulación sanguínea a través de la

arteria obstruida por la banda, dichas vibraciones son captadas por el sensor encargado de realizar el monitoreo de la presión a través de la banda, cuando esto ocurre, el motor deja de funcionar y se desfoga el filtro vaciando de aire la banda, consecuentemente la presión en la banda comienza a disminuir vertiginosamente, y el valor de la amplitud de las vibraciones en la arteria comienza a elevarse hasta llegar a un valor máximo después del cual la intensidad de las oscilaciones comienza a disminuir hasta que la banda está completamente vacía y no ejerce ninguna presión y consecuentemente el flujo sanguíneo vuelve a la normalidad.

Después de estos eventos se toman los valores a partir de las oscilaciones provocadas en la arteria para obtener los valores de tensión arterial. El valor de presión de la banda en la cual ocurre el valor máximo de oscilación de la arteria corresponde al valor de presión media.

El valor de presión de la banda correspondiente al momento en que las vibraciones de la arteria comienzan a aumentar rápidamente en amplitud, corresponde a la presión sistólica.
Por último el punto entre oscilaciones donde el valor de estas comienza a disminuir dramáticamente corresponde a la tensión diastólica (FIGURA 15)

FIGURA 15. Curva de presión oscilatoria.



Fuente: (Savón, 2007).

Cabe resaltar que hay varios criterios a la hora de manejar los valores sistólicos y diastólicos de tensión, debido a que el método oscilométrico solo provee como valor relativamente exacto el valor de tensión media, y los otros dos se obtienen empíricamente o mediante algoritmos o criterios matemáticos basados en los valores de tensión media (Savón, 2007).

1.3.3 Sensor de Presión.

Como sabemos, el evento físico que se describe como presión se explica como una fuerza que se ejerce sobre un área determinada, y lógicamente se expresa en unidades de fuerzas por unidades de área, las unidades más conocidas son bares, atmosferas, milímetros de Hg, etc.

Dicha fuerza aplicada se puede ejercer sobre un punto en una superficie o distribuirse sobre esta. Cada vez que se ejerce se produce una deflexión, una distorsión o un cambio de volumen o dimensión en el elemento sobre el cual se aplica la fuerza.

Para la construcción del tensiómetro en base al esquema tradicional de funcionamiento, se utilizara un sensor de presión. Dicho sensor está dotado de un elemento sensible a la presión que emiten una señal eléctrica al variar la presión o que, dependiendo del sensor, provocan acciones de conmutación si el fenómeno supera un determinado valor referencial.
Dentro de la gama de sensores de presión y rangos de medición, nuestra aplicación requiere sensores miniatura de tipo piezoresistivo que conserven características de robustez y resistencia (Brito, 2013).

1.3.3.1 Sensores Piezoresistivos.

Poseen una membrana fabricada de un material semiconductor (generalmente silicio) con estructuras selectivamente distribuidas. El funcionamiento de este tipo de sensores está basado en el efecto piezoresistivo, que no es más que una variación de la resistencia en el semiconductor, causado por su expansión y compresión que influye en la movilidad de los electrodos bajo carga mecánica. En otras palabras, los valores que se proporcionan a través del sensor van cambiando según cambie la geometría del

sensor por el efecto del fenómeno físico que se va a medir, en este caso, la presión sanguínea (Robredo., 1993).

Para la fabricación del tensiómetro se ve necesaria la utilización de un sensor que cuente con las siguientes características:

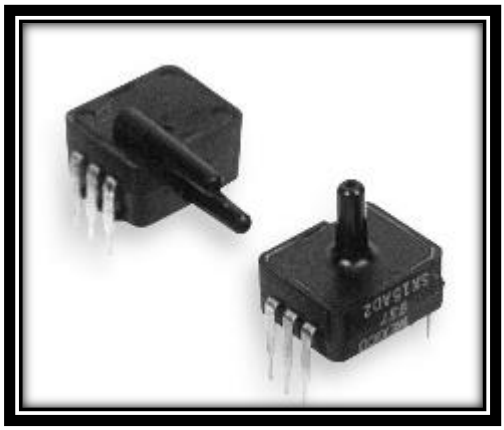
- Relativo bajo costo.
- Compensación de temperatura de precisión.
- Capacidad de Calibración.
- Tamaño relativamente pequeño.
- Bajo Ruido eléctrico.
- Alta impedancia de bajo consumo de energía.

Concretamente utilizaremos sensores de presión de cualquier serie que proporcionen características rentables y nos permitan una aplicación de menor tamaño y mayor rendimiento, proporcionen una salida precisa y estable en un rango de temperatura de 0 ° C a 50 ° C [32 ° F a 122 ° F], que puedan medir la presión absoluta y la galga de 1 psi.

Dichos sensores generalmente operan mediante un circuito integrado (IC) compuesto por sensor y el láser cubiertos por una película gruesa de cerámica dentro de una caja compacta resistente a los disolventes. El DIP se monta en una tarjeta de circuito impreso como un IC estándar con pines a través de hoyos. Los pernos de anclaje del sensor de presión a la placa de circuito impreso y proporcionan una mayor unidad de seguro y estable que otros tipos de paquetes.

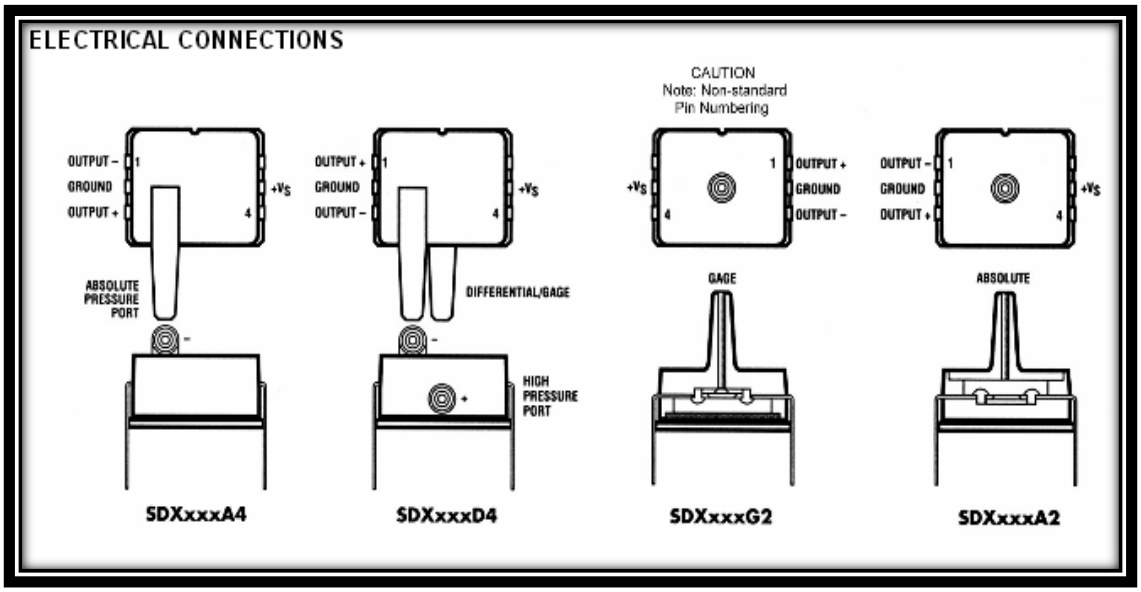
La salida del puente es radiométrica a la tensión de alimentación y la operación de cualquier tensión de alimentación de corriente continua de hasta 20 Vdc es aceptable (Honeywell, 2008) (FIGURA 16 y 17).

FIGURA 16. Sensor de presión ADP 510.



Fuente: (Honeywell, 2008).

FIGURA 17. Disposición de pines según el tipo de encapsulado del sensor ADP 510.



Fuente: (Honeywell, 2008).

El sensor va dentro del equipo tensiómetro, como vimos anteriormente en el procedimiento de toma de tensión arterial se ejerce una presión alrededor de la parte de la extremidad donde se instala el equipo, dicha presión, provoca cierta deformidad en el sensor y se comienza a emitir valores en forma de señales eléctricas que serán decodificadas por el equipo y entregaran los valores finales de tensión arterial.

CAPÍTULO 2.

HARDWARE DEL DISPOSITIVO

2.1 Electrocardiograma

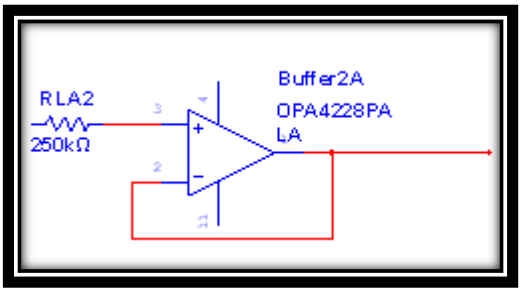
2.1.1 Recolección de Señales.

Con el objetivo de reducir el ruido generado por el ambiente, la recolección de señales se efectúa mediante la utilización de electrodos denominados electrodos 3M desechables conectados a cables apantallados para ECG.

2.1.2 Protecciones y Amplificador Instrumental.

Con el propósito de robustecer la señal, se implementó un Amplificador Operacional, concretamente el INA4228 en configuración Buffer como se muestra en la FIGURA 18 a continuación expuesta.

FIGURA 18. Amplificador Operacional INA 4228 en configuración Buffer.



Fuente: Autor.

Naturalmente, la implementación de dicho buffer se debe repetir para cada una de las señales recogidas por los electrodos. A su vez, la señal robustecida entregada por el buffer debe cumplir con el enunciado del triángulo de eithoven explicado en el capítulo anterior, para que esta condición se lleve a cabo, se implementó

amplificadores diferenciales INA129 o AD620 (FIGURA 19.), cuyo comportamiento cumple con las siguientes ecuaciones:

$$\mathrm{Vo} = (\mathrm{Vin}^{+} - \mathrm{Vin}^{-}) * \mathrm{G}$$

$$\mathrm{G} = 1 + \frac{49.4\mathrm{K}\Omega}{\mathrm{RG}}$$

Donde:

Vo = Voltaje de Salida.

Vin^{+}= Voltaje de entrada por el pin no inversor del INA122.

Vin^{-} = Voltaje de entrada por el pin inversor del INA122.

G = Ganancia del Amplificador instrumental.

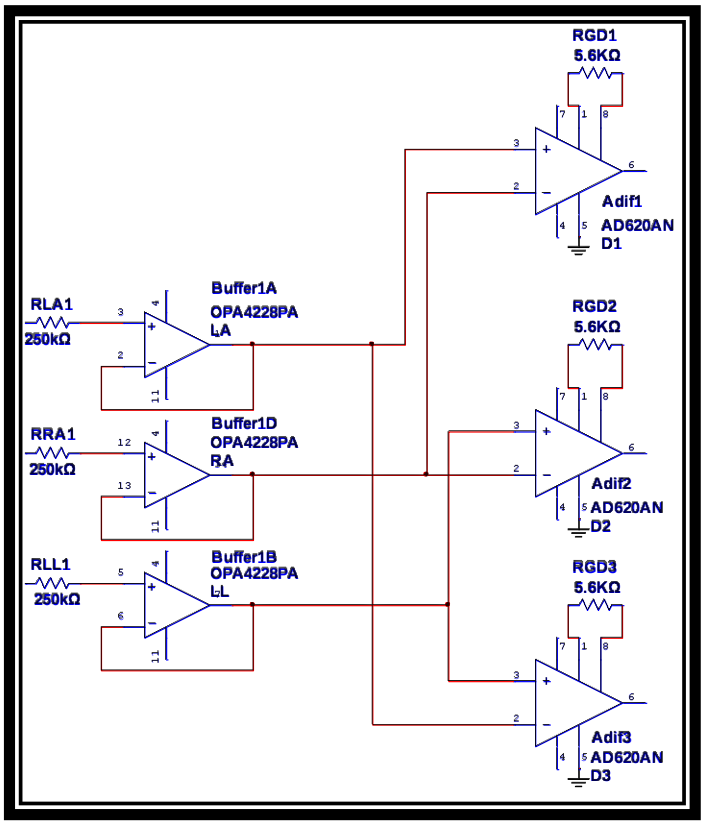
RG= Resistencia de Ganancia.

Conociendo los valores de resistencias obtenemos:

$$\mathrm{G} = 1 + \frac{49.4\mathrm{K}\Omega}{5.6\mathrm{K}\Omega}$$

$$\mathrm{G} = 9.82$$

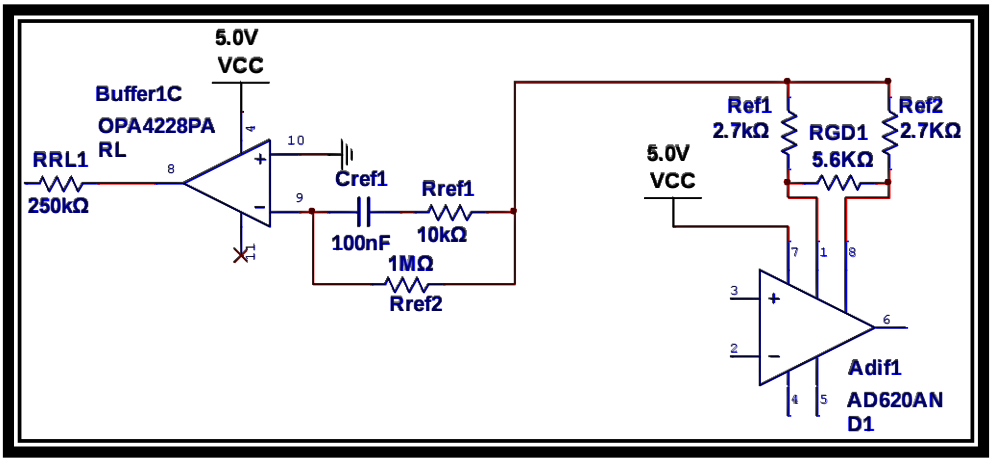
FIGURA 19.Implementación de amplificadores diferenciales INA 129 o AD620 a las salidas del buffer.



Fuente: Autor.

Para que la señal consiga tanto la mayor estabilidad posible como un mínimo margen de ruido, es necesario contar con un electrodo adicional de referencia, dicho electrodo se debe colocar en la extremidad inferior derecha (pierna derecha) del sujeto, cabe mencionar que las casas fabricantes de Amplificadores operacionales recomiendan que dicho electrodo se encuentre interconectado entre las ganancias de los mismos, por resistencias de valor RG/2 y por un filtro, como acontinuación se esquematiza en la FIGURA 20.

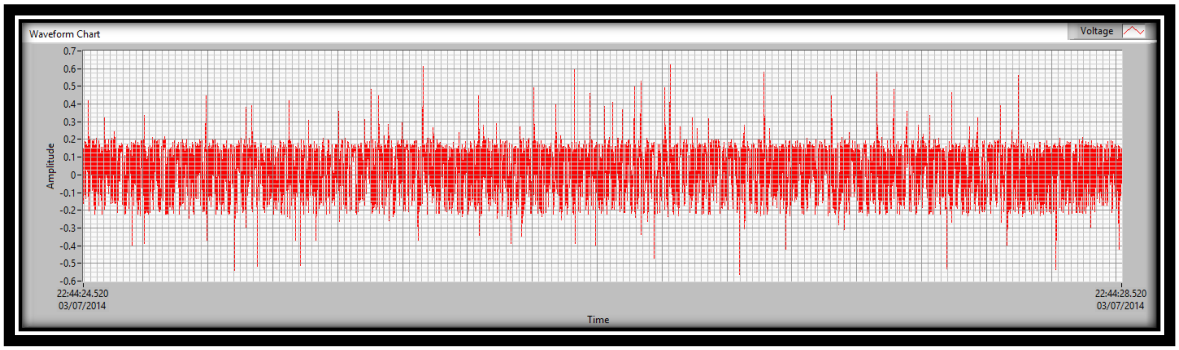
FIGURA 20. Implementación del tercer electrodo.



Fuente: Autor.

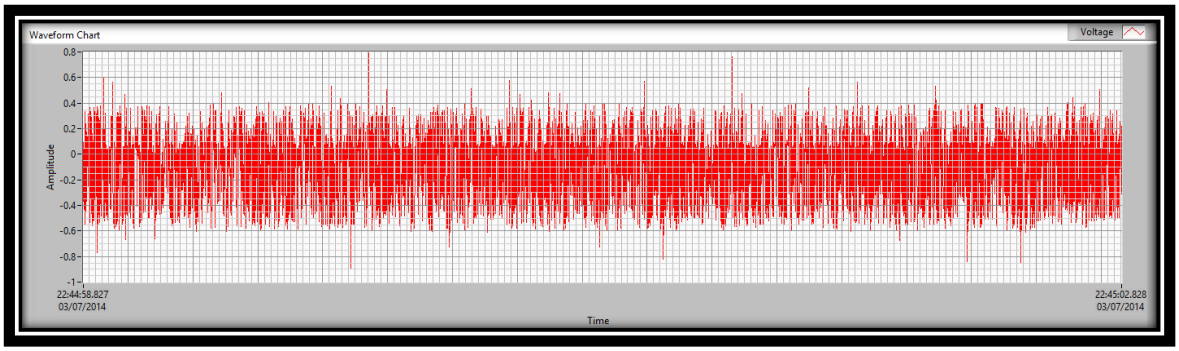
Las gráficas correspondientes de las salidas de los Amplificadores instrumentales (FIGURA 21, FIGURA 22, FIGURA 23) en cada derivacion son las siguientes:

FIGURA 21.Salida del Amplificador Diferencial Primera Derivación.



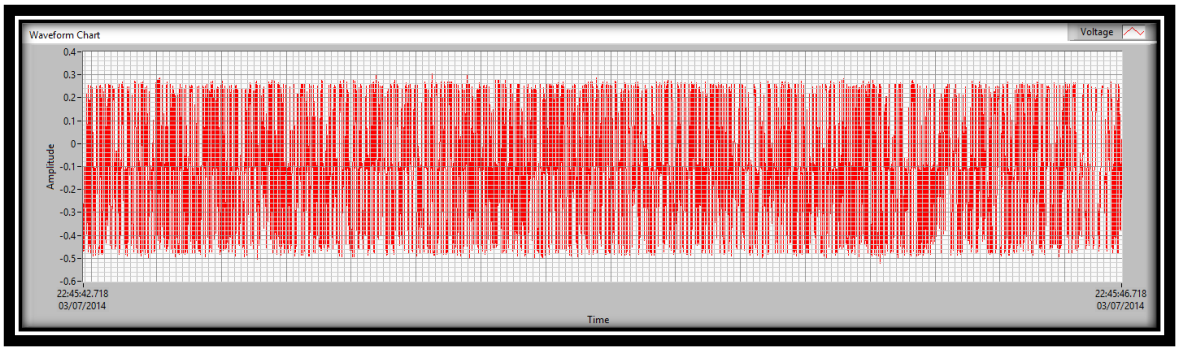
Fuente: Autor.

FIGURA 22. Salida del Amplificador Diferencial Segunda Derivación.



Fuente: Autor.

FIGURA 23. Salida del Amplificador Diferencial Tercera Derivación.



Fuente: Autor.

Como se puede apreciar en las señales anteriores, se puede reconocer diferentes tipos de ruido en ellas, como por ejemplo el ruido proveniente de la señal eléctrica de 60Hz junto con ruidos externos de tipo ambiental desde los 300Hz. Para solucionar este problema, se debe implementar un filtro pasa banda entre 0.05Hz y 100Hz donde se encuentra la señal pura medida (información valida). Por otro lado, si el sistema de medición y generación de señal es alimentado por una fuente derivada de un tomacorriente convencional, es impresindible la utilización de un filtro NOTCH para eliminar el ruido parásito originado por la fuente.

El filtro pasa banda mencionado en el párrafo anterior, está compuesto en primer lugar por un filtro pasa alto con fc = 0.5Hz a continuación del cual, se integra un filtro pasa Bajo de con fc = 10Hz.

2.1.3 Filtro pasa Alto.

Para el cálculo del filtro pasivo se utiliza la siguiente ecuación:

$$fc = \frac{1}{2\pi * R * C}$$

Donde:

fc = Frecuencia de Corte.

R = Resistencia de Filtro.

C = Capacitancia de Filtro.

Puesto que ya se conoce el valor de la frecuencia de corte (0.5Hz), imponemos un condensador de C = 100uF para poder calcular la resistencia R.

$$R = \frac{1}{2\pi * fc * C}$$

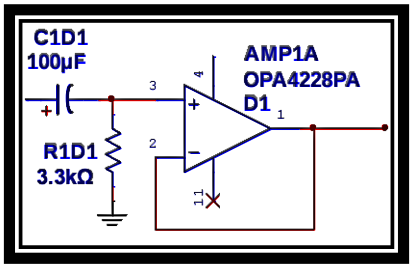
$$R = \frac{1}{2\pi * 0.5Hz * 100uF}$$

$$R = 3.183K\Omega$$

Por razones comerciales la resistencia sera R = 3.3KΩ

Para robusteser su señal, se implementó un buffer a continuación de filtro pasa alto pasivo, su esquema (circuito) se muestra a continuación en la FIGURA 24 Cabe mencionar que este circuito se debe implementar en cada una de las salidas de los amplificadores instrumentales.

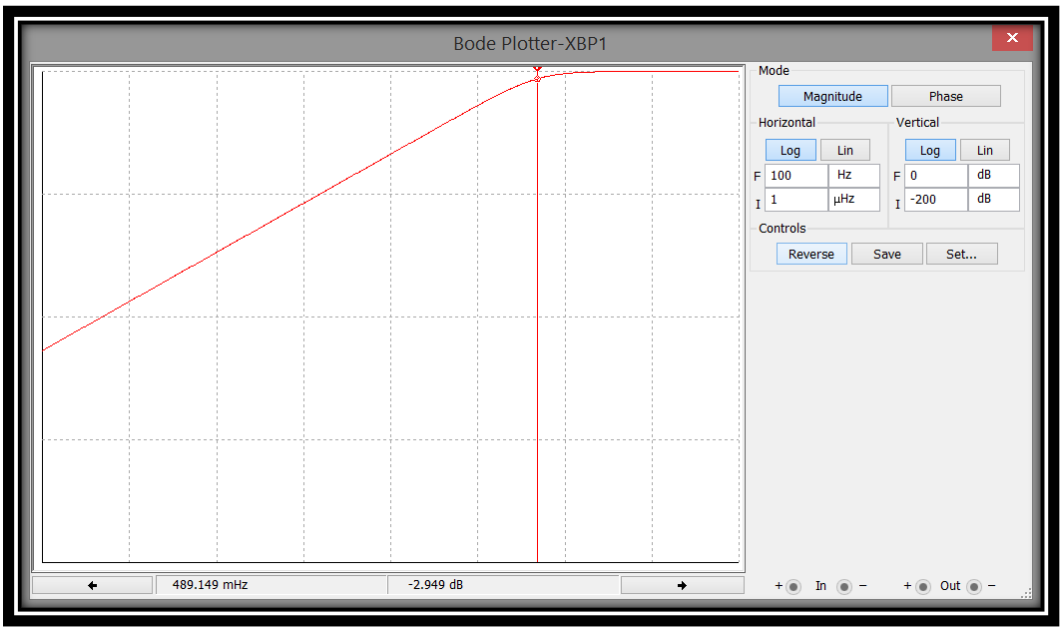
FIGURA 24. Filtro pasa alto con buffer.



Fuente: Autor.

La respuesta del fitro en función se la frecuencia se puede obervar en la FIGURA 25

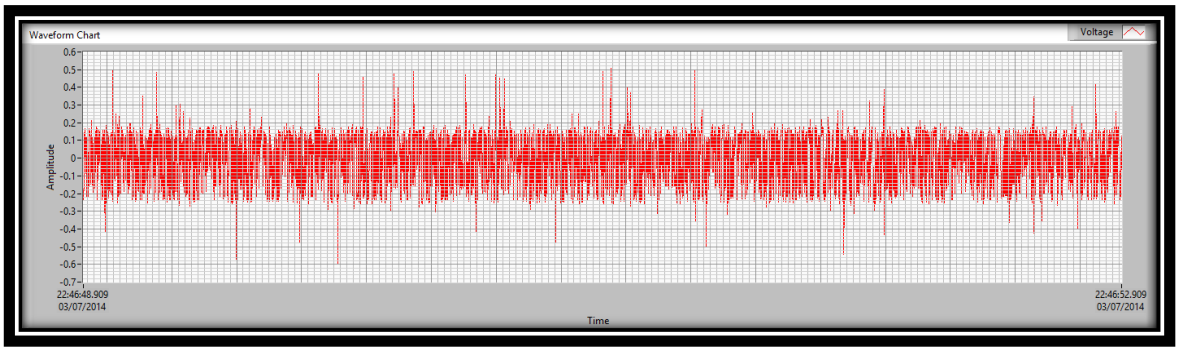
FIGURA 25. Respuesta al filtro.



Fuente: Autor.

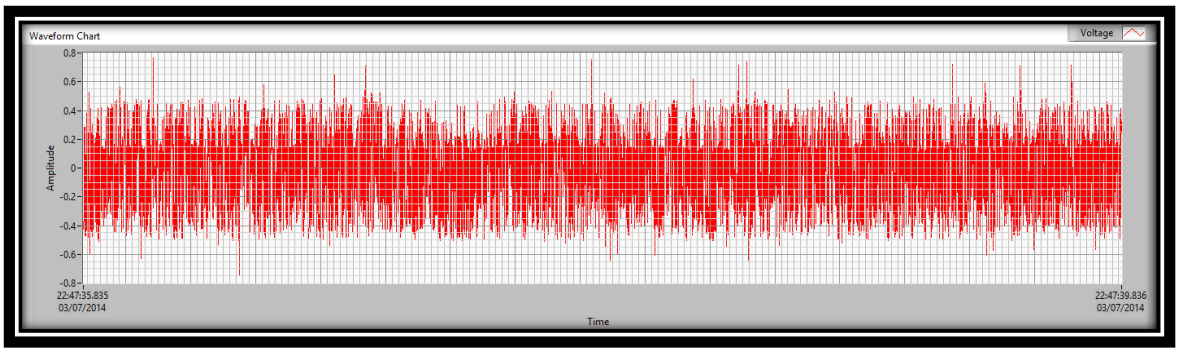
Las siguientes figuras (FIGURA 26, FIGURA 27, FIGURA 28) corresponden a las salidas de los filtros pasa altos de las respectivas derivaciones del electrocardiograma:

FIGURA 26. Salida del filtro Pasa alto Primera Derivación.



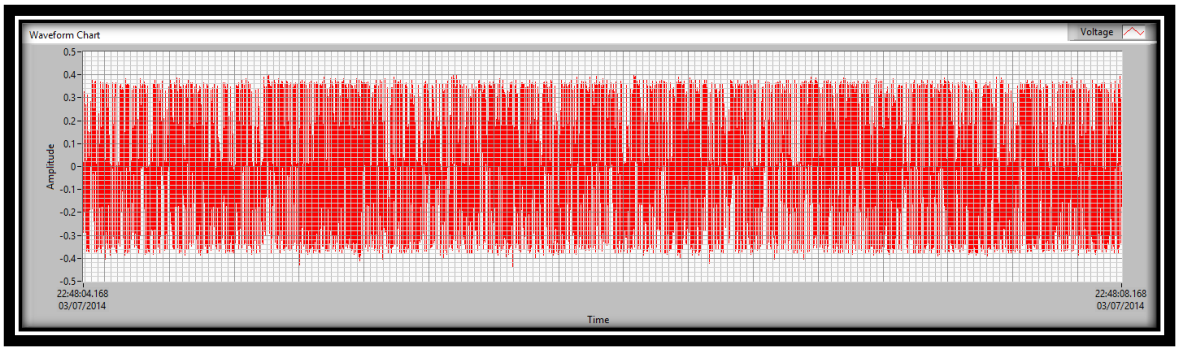
Fuente: Autor.

FIGURA 27. Salida del filtro Pasa alto Segunda Derivación.



Fuente: Autor.

FIGURA 28. Salida del filtro Pasa alto Tercera Derivación.



Fuente: Autor

2.1.4 Filtro pasa Bajo.

Como habiamos mencionado, para completar el filtro pasa banda requerido, a continuación del filtro pasa alto se instala un fltro pasa bajo pasivo, junto con un amplificador no inversor en el mismo para robusteser y amplificar la señal. Para el cálculo del filtro pasivo nos valemos de su ecuación caracteristica:

$$fc = \frac{1}{2\pi * R * C}$$

Donde:

fc = Frecuencia de Corte.

R = Resistencia de Filtro.

C = Capacitancia de Filtro.

Puesto que conocemos que la frecuencia de corte debe ser 50 Hz, nos imponemos un condensador C = 0.47uF para poder calcular la resistencia R.

$$R = \frac{1}{2\pi * fc * C}$$

$$R = \frac{1}{2\pi * 50Hz * 0.47uF}$$

$$R = 6.772K\Omega$$

Por razones comerciales la resistencia sera R = 6.8KΩ

Para el cálculo del amplificador no inversor en el circuito, nos valemos de su ecuación característica:

$$V_{salida} = V_{entrada} * (1 + \frac{R2}{R1})$$

Donde $(1 + \frac{R2}{R1})$ corresponde a la ganancia del amplificador no inversor.

La ganancia que en este caso se requiere es de 100, para ello debemos imponer una de las resistencias, en este caso R1=10KΩ

$$G = 1 + \frac{R2}{R1}$$

$$R2 = (100 - 1) * 10K\Omega$$

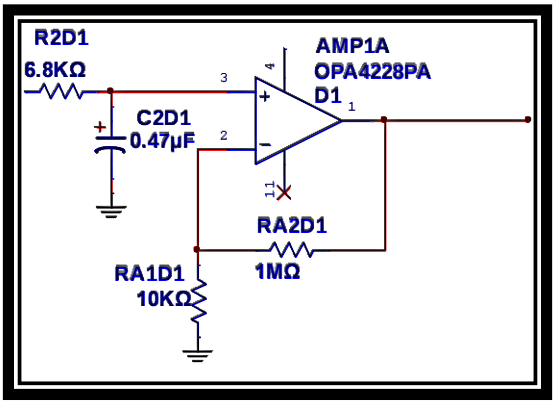
$$R2 = (100 - 1) * 10K\Omega$$

$$R2 = 990K\Omega$$

La resistecia R2 debe tener un valor comercial por lo tanto R2=1MΩ

Se muestra a continuación el circuito del filtro Paso Bajo con amplificación (FIGURA 29.), que, al igual que el anterior, se debe repetir para cada una de las salidas de los filtros pasa alto.

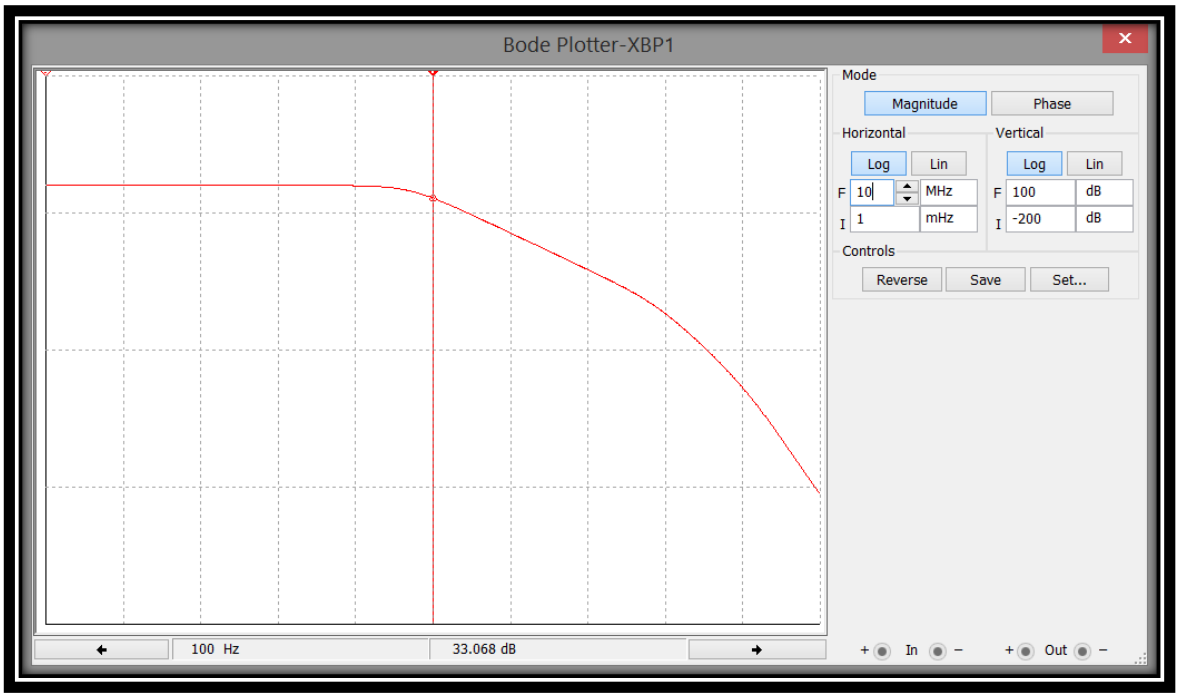
FIGURA 29. Filtro pasa bajo pasivo.



Fuente: Autor.

La respuesta del fitro en función se la frecuencia se puede obervar en la FIGURA 30.

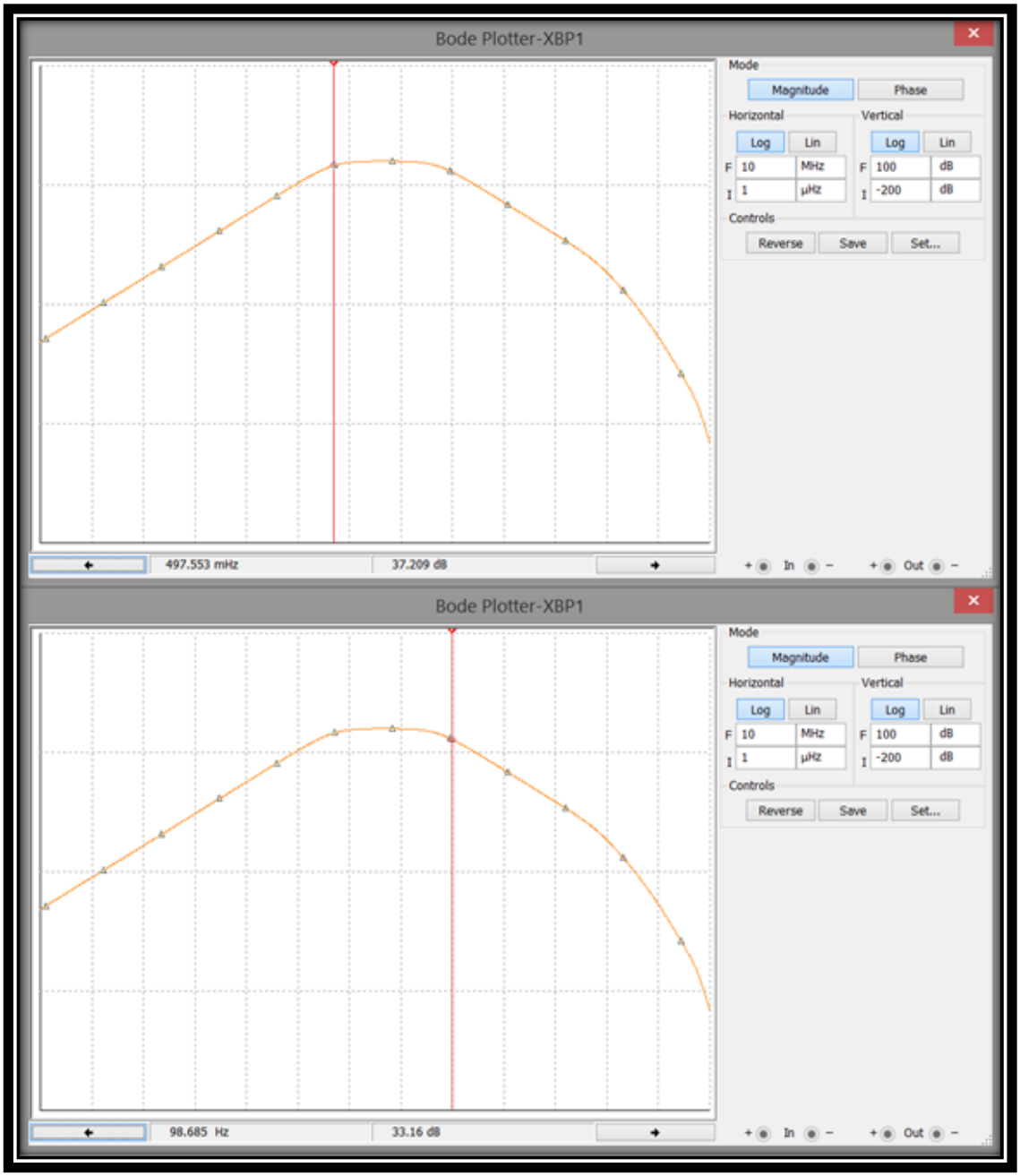
FIGURA 30. Respuesta al filtro pasa bajo.



Fuente: Autor.

La salida de filtro pasa banda total en función de la frecuencia será como el de la FIGURA 31.

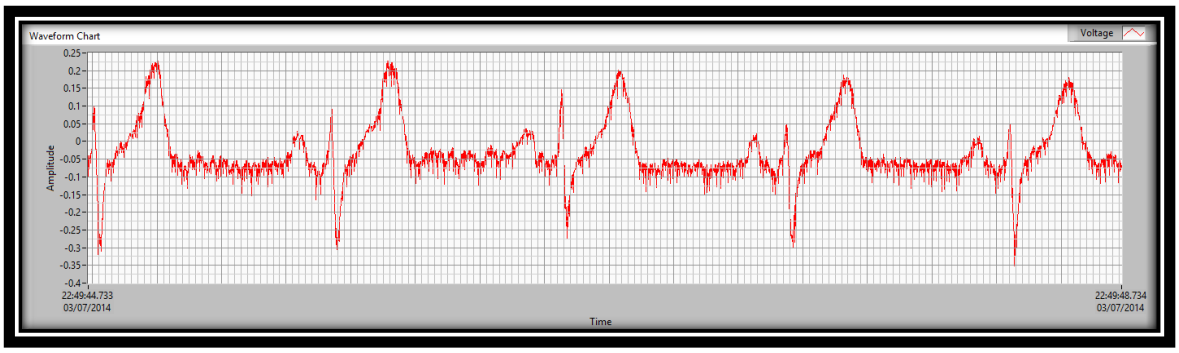
FIGURA 31. Respuesta a través del filtro pasa banda.



Fuente: Autor.

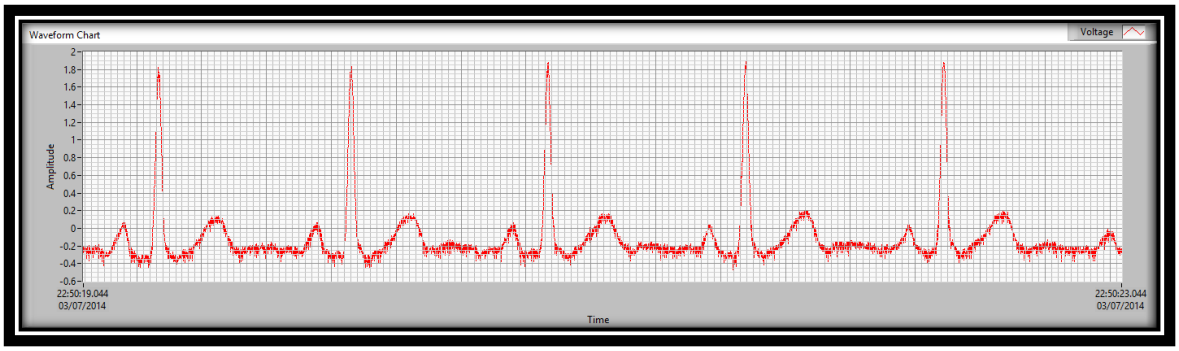
Los siguientes esquemas (FIGURA 32, FIGURA 33, FIGURA 34.) corresponden a las respuestas al sistema después de pasar por los filtros pasa banda.

FIGURA 32. Respuesta al filtro pasa banda de la primera derivación.



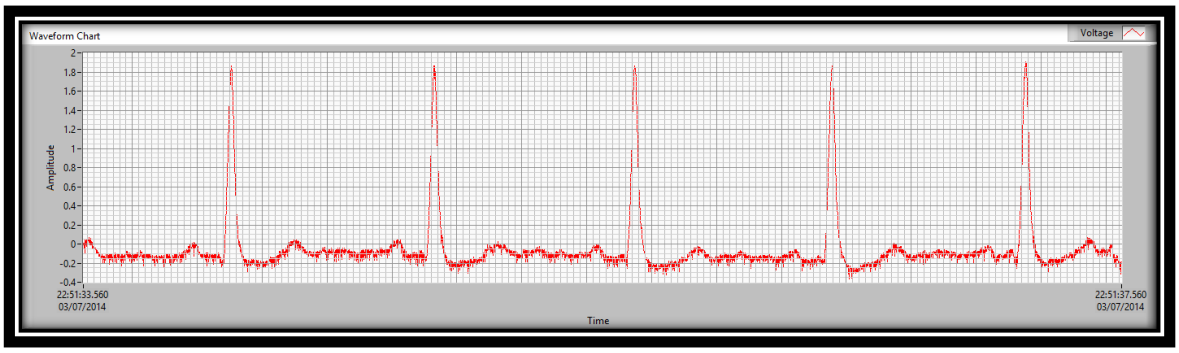
Fuente: Autor.

FIGURA 33. Respuesta al filtro pasa banda de la segunda derivación.



Fuente: Autor.

FIGURA 34. Respuesta al filtro pasa banda de la tercera derivación.



Fuente: Autor.

2.1.5 Acople de Señal.

Como se puede observar en las figuras anteriores, el voltaje de la señal varia entre ± 2V. Debido a que el microcontrolador que se utiliza (ATMega 328) cuenta con un

convertidor analógico-digital de 0 a 5V de 1024 bits de resolución, es necesario acoplar la señal para que este en el rango del convertidor del microcontrolador.

Para este acoplamiento, se necesita simplemente un partidor de tensión que genere el offset deseado, y acoplar la señal por medio de un condensador, los valores de las resistencias para el partidor de tensión y el condensador de acople se calculan tomando en cuenta que no se debe interferir con las frecuencias que forman a su vez un filtro pasa alto.

El partidor de tensión obedece a la siguiente ecuación:

$$V_{salida} = V_{entrada} * \frac{R2}{R1 + R2}$$

El partidor de tensión actúa como offset del acople, por lo es conveniente que dicho offset sea de 2.5V es decir la mitad de la alimentación de 5V del sistema, para esto R2 debe ser igual a R1:

$$V_{salida} = V_{entrada} * \frac{R1}{R1 + R1}$$

$$V_{salida} = 0.5 * V_{entrada}$$

Donde R1 = R2 = 33KΩ

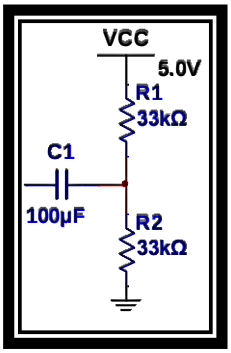
Para que no afecte el sistema con el filtro formado, se utilizó un condensador de 100uF, lo que genera un filtro pasa alto con fc de 0.05Hz.

$$fc = \frac{1}{2\pi * 33K\Omega * 100uF}$$

$$fc = 0.04822Hz$$

El circuito es el mostrado en la FIGURA 35.

FIGURA 35. Partidor de tensión para acoplamiento con el microcontrolador.

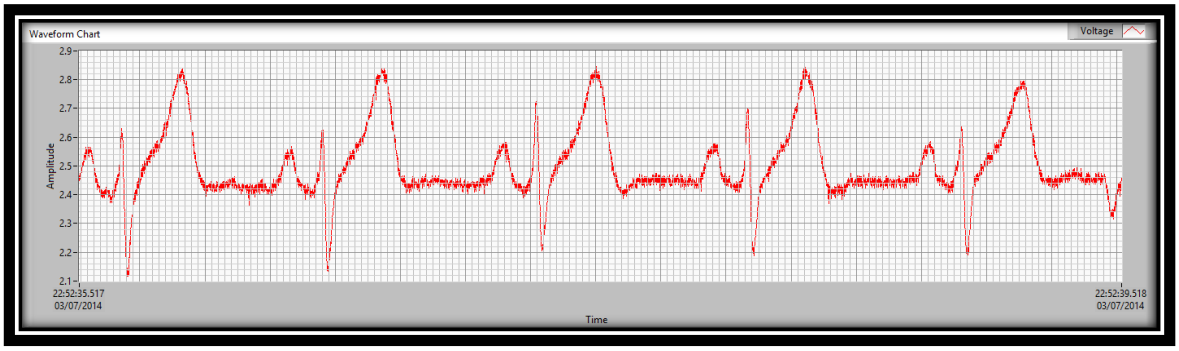


Fuente: Autor.

Naturalmente, este circuito se debe repetir para cada una de las salidas del filtro pasa bajo.

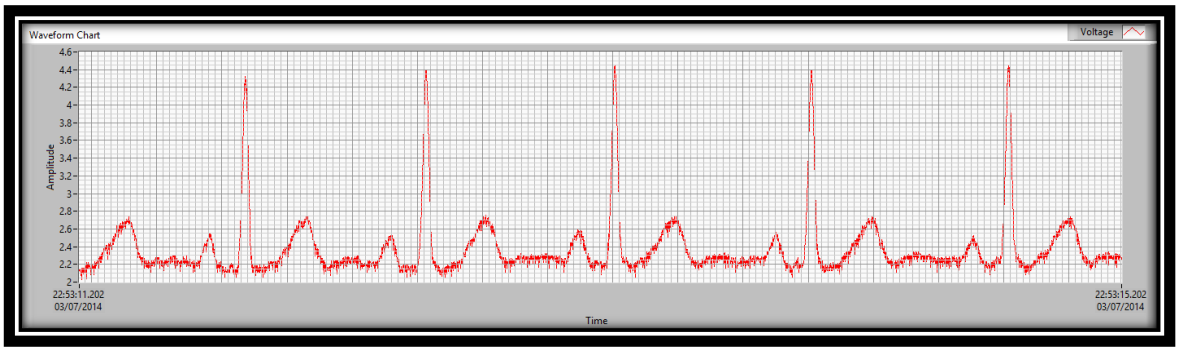
Las siguientes figuras (FIGURA 36, FIGURA 37, FIGURA 38.)corresponden a las señales de las 3 derivaciones después de ser acopladas.

FIGURA 36. Respuesta del Acople de la Primera Derivación.



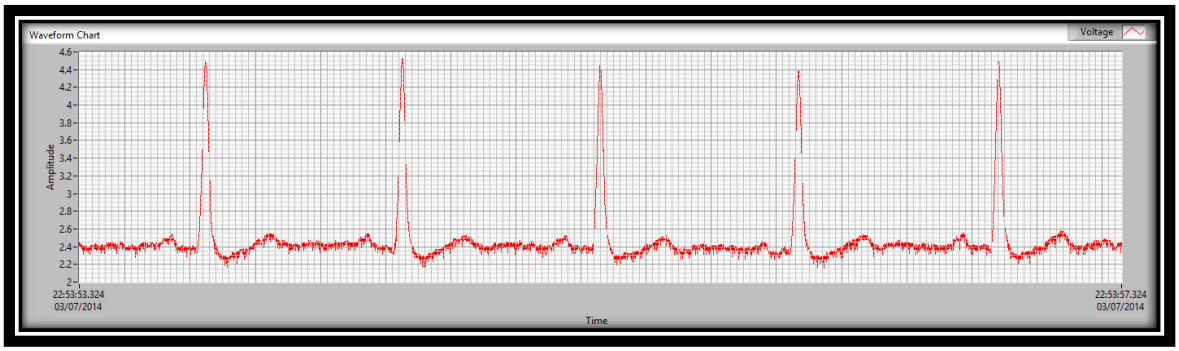
Fuente: Autor.

FIGURA 37. Respuesta del Acople de la Segunda Derivación.



Fuente: Autor.

FIGURA 38. Respuesta del Acople de la Tercera Derivación.



Fuente: Autor.

2.1.6 Filtros Digitales.

Al ser los filtros analógicos pasivos, la frecuencia de trabajo no es tan selectiva, por lo que se puede utilzar filtros digitales, para que la señal de las derivaciones del electrocardiograma sean mas puras. Antes de elaborar los filtros, se debe tener en cuenta que el microcontrolador tiene una frecuencia de muestreo de 1Khz para cada una de las derivaciones, dato que sirve para calcular las frecuencias digitales de corte para el filtro.

Las frecuecias de corte seran de 0.5Hz y 50Hz, información a tomar en cuenta, pues dentro de estas frecuencias se encuantran datos importantes. Como resultado aplicamos dicha información para calcular frecuencias de muestreo con sus ecuaciones características como se muestra a continuación:

$$\Omega = \omega * T_s$$

$$T_s = \frac{1}{f_s}$$

$$\omega = 2\pi f$$

$$\Omega = \frac{2\pi f}{f_s}$$

Donde:

Ω = Frecuencia Digital normalizada.

T_s= Período de Muestreo.

f_s=Frecuencia de Muestreo.

ω=Frecuencia angular analógica.

f=Frecuencia analógica.

La frecuencia digital esta entre $0 < \Omega < \pi$

$$f_s = 1\text{KHz}$$

$$\Omega_{c1} = \frac{1}{1000}\pi = 0.001\pi$$

$$\Omega_{c2} = \frac{1}{10}\pi = 0.1\pi$$

Con las frecuencias calculadas sumado la ayuda de la herramienta de software Labview con su toolkit de diseño de filtro digital, se pueden obtener las funciones de transeferencia y con esto las ecuaciones diferenciales y sus respectivas constantes.

El filtro elegido fue de tipo ventana Hamming, cuya función de transferencia es la siguiente;

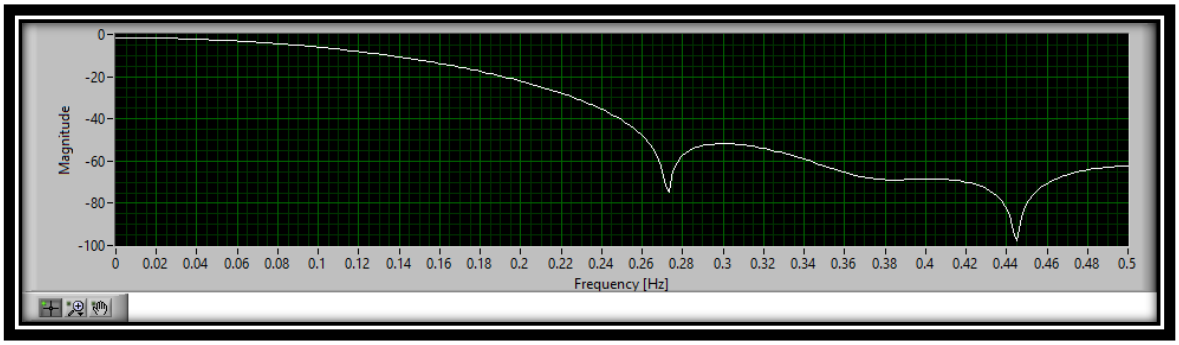
$$H(z) = -0.000159974 + 0.00751553z^{-1} + 0.0393517z^{-2} + 0.101889z^{-3} + 0.168837z^{-4} + 0.198z^{-5} + 0.168837z^{-6} + 0.101889z^{-7} + 0.0393517z^{-8} + 0.00751553z^{-9} - 0.000159974z^{-10}$$

De lo que se puede obtener la siguiente ecuacion diferencial:

$$y[n] = -0.000159974x[n] + 0.00751553x[n-1] + 0.0393517x[n-2] + 0.101889x[n-3] + 0.168837x[n-4] + 0.198x[n-5] + 0.168837x[n-6] + 0.101889x[n-7] + 0.0393517x[n-8] + 0.00751553x[n-9] - 0.000159974x[n-10]$$

La FIGURA 39. a continuación muestra el filtro en funcion de la frecuencia digital:

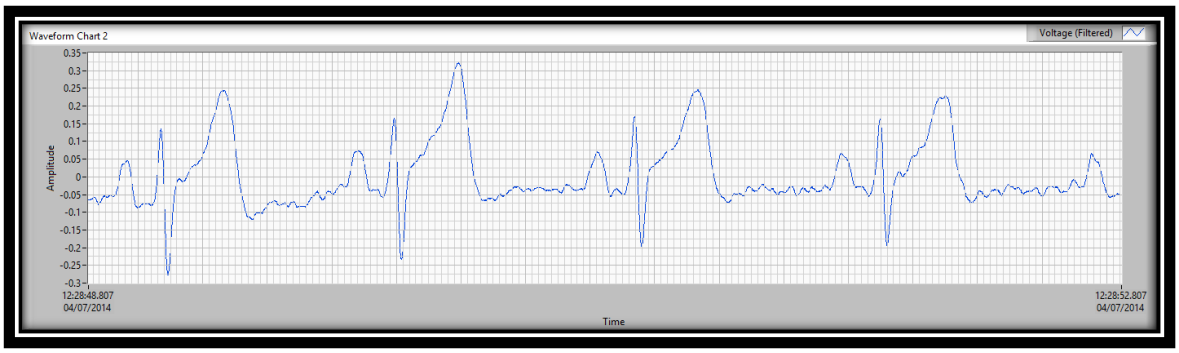
FIGURA 39. Filtro calculado en función de la frecuencia digital.



Fuente: Autor.

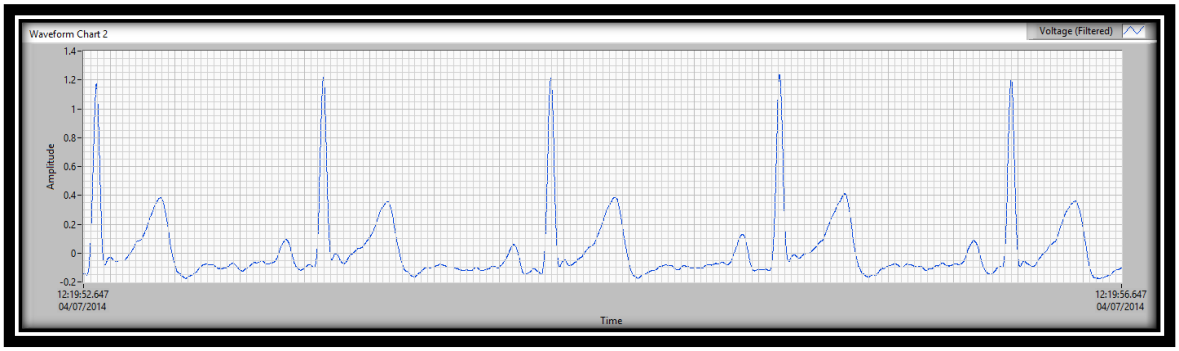
La respuesta al filtro digital se puede observar en la FIGURA 40, FIGURA 41, FIGURA 42 a continuación:

FIGURA 40. Respuesta del Sistema de filtro pasa banda digital, primera derivación.



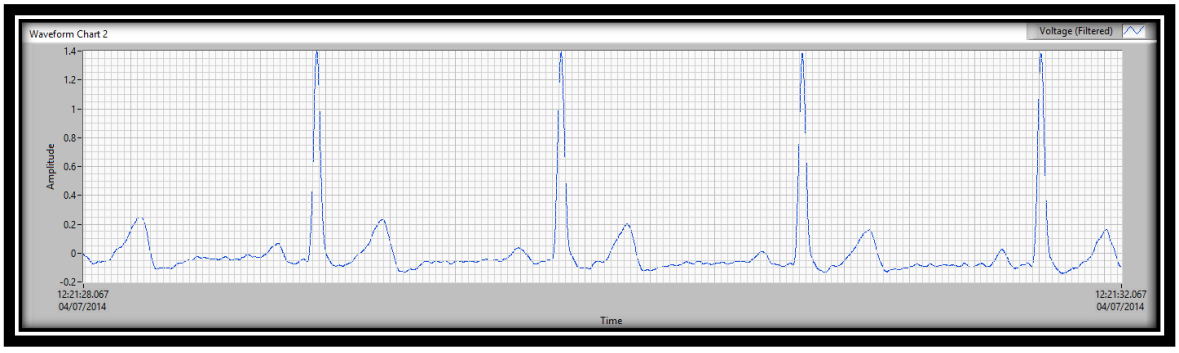
Fuente: Autor.

FIGURA 41. Respuesta del Sistema de filtro pasa banda digital, segunda derivación.



Fuente: Autor.

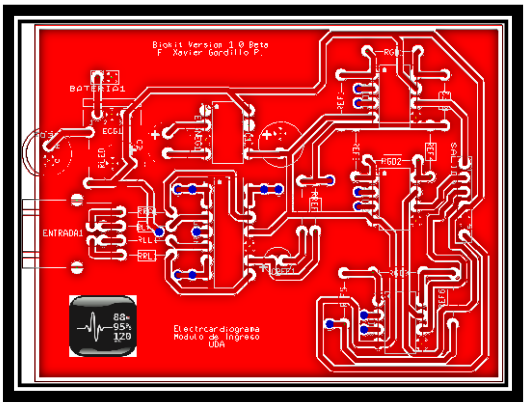
FIGURA 42. Respuesta del Sistema de filtro pasa banda digital, tercera derivación.



Fuente: Autor.

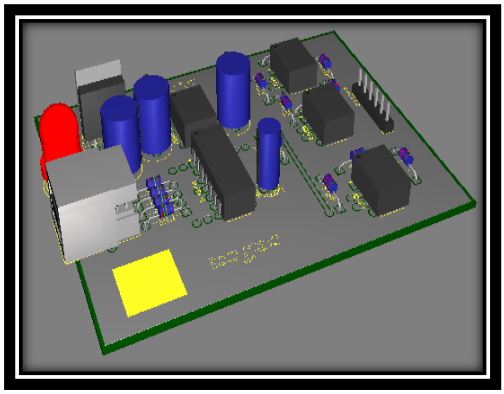
Se realizaron módulos separados para cada una de las etapas citadas, de manera que las siguientes figuras (desde FIGURA 43 hasta la FIGURA 54) corresponden a sus respectivos circuitos diseñados, simulación y finalmente impresos.

FIGURA 43. Diseño del circuito de adquisición de señales.



Fuente: Autor.

FIGURA 44. Simulación del circuito real de adquisición de señales.

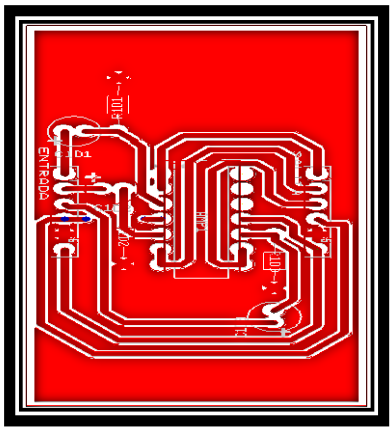


Fuente: Autor.

FIGURA 45. Circuito impreso de la adquisición de señales.

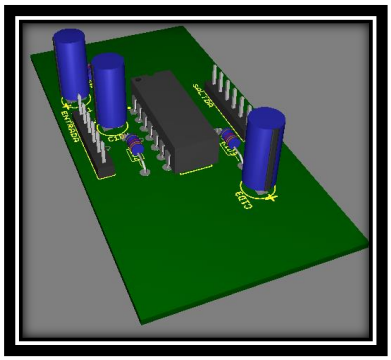
Fuente: Autor.

FIGURA 46. Diseño del circuito del filtro pasa alto.



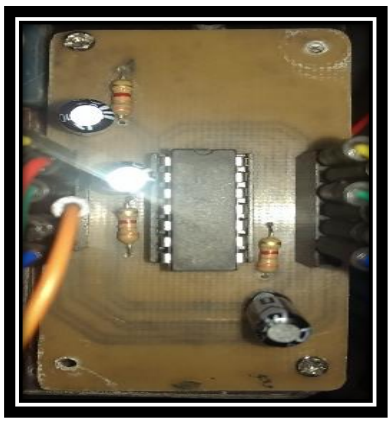
Fuente: Autor.

FIGURA 47. Simulación del circuito real del filtro pasa alto.



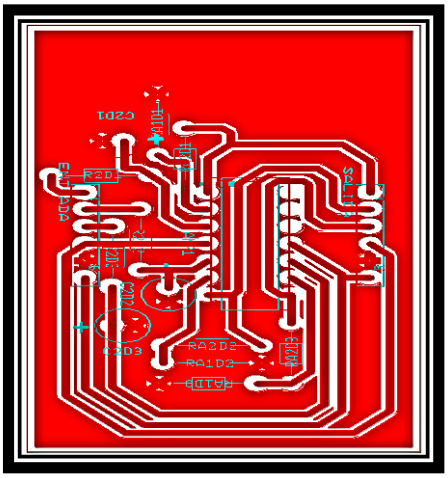
Fuente: Autor.

FIGURA 48. Circuito impreso del filtro pasa alto.



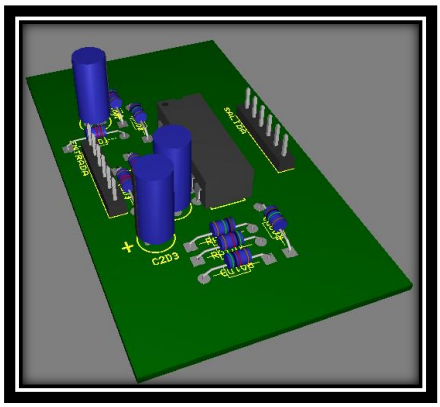
Fuente: Autor.

FIGURA 49. Diseño del circuito del filtro pasa bajo.



Fuente: Autor.

FIGURA 50. Simulación del circuito real del filtro pasa bajo.

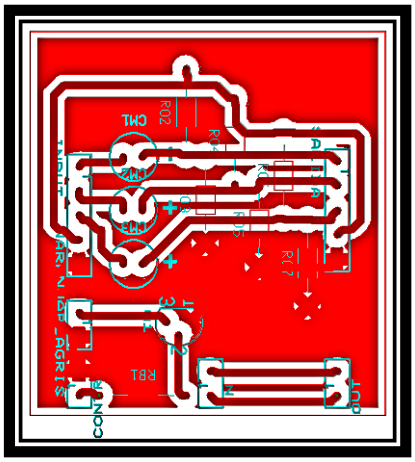


Fuente: Autor.

FIGURA 51. Circuito impreso de filtro pasa bajo.

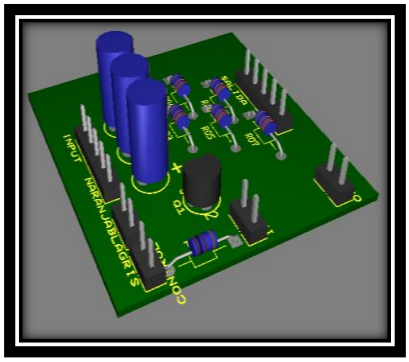
Fuente: Autor.

FIGURA 52. Diseño del circuito de acople de señales.



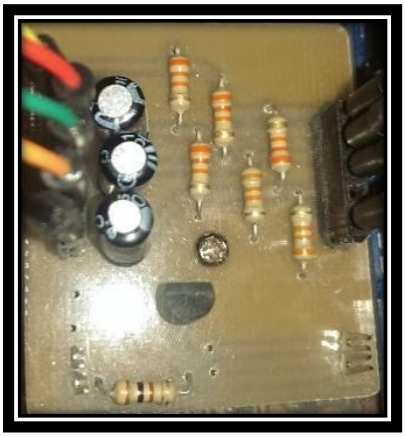
Fuente: Autor.

FIGURA 53. Simulación del circuito real de acople de señales.



Fuente: Autor.

FIGURA 54. Circuito impreso de acople de señales.



Fuente: Autor.

2.2 Tensiómetro.

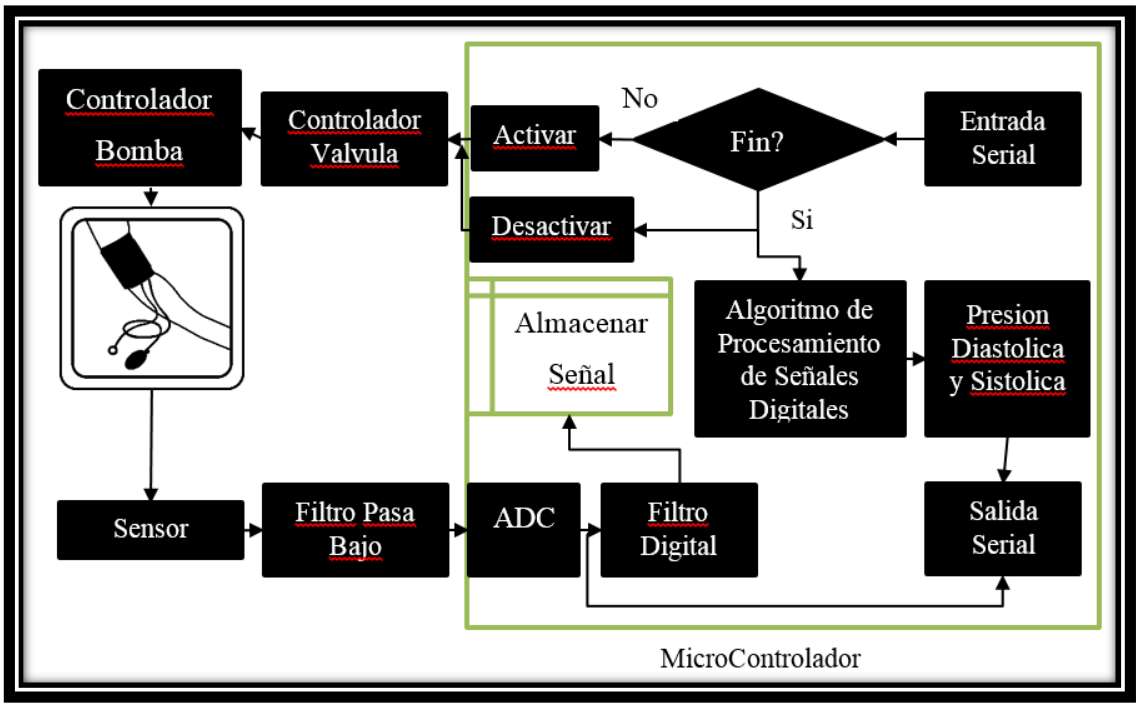
2.2.1 Principio de Funcionamiento.

Como se había visto, se denomina tensiómetro a aquel equipo que, mediante una compresión en la extremidad superior, genera un metodo oscilométrico de detección no invasiva de la presión arterial. Naturalmente dicho proceso se efectúa mediante diferentes medios electrónicos, mismos que se utiliza desde la compresión de la extremidad hasta la adquisición e interpretación de valores adquiridos para generar los datos finales de tensión arterial.

2.2.2 Diagrama de Bloques.

El proceso que se describió anteriormente se esquematiza mediante bloques en la FIGURA 55 a continuación expuesta.

FIGURA 55. Diagrama de bloques del proceso de adquisición y generación de datos en el equipo Tensiométrico.

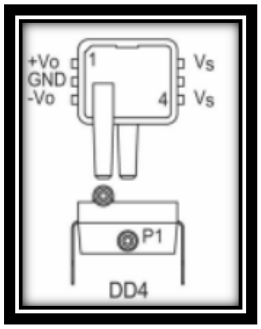


Fuente: Autor.

2.2.3 Sensor.

Como en todo proceso de adquisicion de señales, el medio electrónico por excelencia en contacto con el fenomeno físico a medir, es el sensor , lógicamente en el caso del tensiómetro, se utilizo un sensor de presión, específicamente el sensor SCC05DD4 (FIGURA 56) el cual nos permite medir una presión de 0 hasta 5 P.S.I. (258.5746205121946 mm Hg). Como la mayoría de sensores, el sensor SCC05DD4 devuelve un valor de voltaje por cada rango de valor del fenómeno físico a medir, en este caso la presión; por cada PSI el sensor SCC05DD4 entrega un valor de 1.2 milivoltios.

FIGURA 56. Esquema gráfico del sensor SCC05DD4.



Fuente: (Honeywell, 2008).

2.2.4 Desarrollo.

A su vez, para el funcionamiento esperado del sensor, es necesario la implementación de un amplificador instrumental a continuación del mismo, ya que los datos que devuelve son la presión atmosférica a través del pin negativo (consultar catálogo del sensor) y la presión ingresada por el sistema, mas la presión atmosférica en el pin positivo (consultar catalogo del sensor); por otro lado, como se mencionó anteriormente, los valores de voltaje perceptibles por el microcontrolador, en donde se ingresa y hace el tratamiento de las señales, oscila entre 0 y 5 voltios; se requiere por lo tanto, un sistema que amplifique proporcionalmente los valores adquiridos directamente del sensor junto con cualidades de eliminación de ruido para una medición mas precisa.

El amplificador instrumental incorporado es el amplificador INA122, cuya característica predominante es su alimentación, concretamente una fuente asimétrica desde 2.2V a 36V y cuya salida esta dado por las siguientes ecuaciones :

$$\mathrm{Vo} = (\mathrm{Vin}^{+} - \mathrm{Vin}^{-}) * \mathrm{G}$$

$$\mathrm{G} = 5 + \frac{200\mathrm{K}\Omega}{\mathrm{RG}}$$

Donde:

Vo = Voltaje de Salida.

Vin^{+}= Voltaje de entrada por el pin no inversor del INA122.

Vin^{-} = Voltaje de entrada por el pin inversor del INA122.

G = Ganancia del Amplificador instrumental.

RG= Resistencia de Ganancia.

Como habíamos mencionado anteriormente, para conseguir que la presión sensada este en el rango de 0 a 5V, que es un rango fácilmente detectable por el microcontrolador sin necesidad de introducir un voltaje de referencia, es necesario amplificar la señal, para conseguirlo, la ganancia del amplificador instrumental requerida debe ser cercana a 100, por lo tanto los valores de resistencia de ganancia a implementar en el amplificador (RG) serán calculados en base a la siguiente ecuación con su consecuente valor:

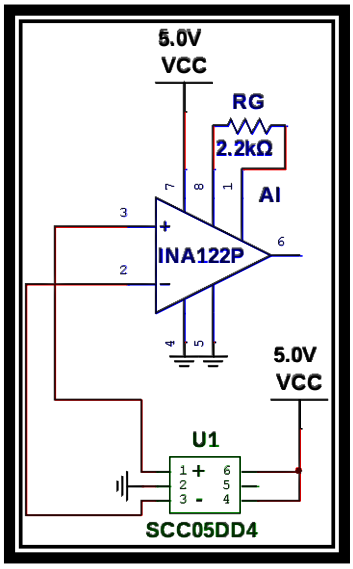
$$\mathrm{RG} = \frac{200\mathrm{K}\Omega}{\mathrm{G} - 5}$$

$$\mathrm{RG} = \frac{200\mathrm{K}\Omega}{100 - 5}$$

$$\mathrm{RG} = 2.105\ \mathrm{K}\Omega$$

Por razones comerciales el valor de la resistencia mas cercana es de 2.2KΩ, por lo que la ganancia será, consecuentemente,proxima a 95.91. El diagrama del amplificador con su calculado valor de resistecia que proporciona la ganancia requerida se muestra a continuacion en la FIGURA 57

FIGURA 57. Amplificador instrumental de ganancia aproximada a 100.



Fuente: Autor.

Por otro lado, un factor vital a tomar en cuenta es que las unidades de medición de presión arterial no son P.S.I sino mmHg (Milímetros de Mercurio) por lo que es necesario efectuar los cálculos con la siguiente relación de conversión:

$$1\ \mathrm{P.S.I} = 51.718\mathrm{mmHg}$$

La salida del amplificador instrumental tendra las siguientes referencias:

$$1\mathrm{P.S.I} = 12\mathrm{mV} * \mathrm{G}$$

$$1\ \mathrm{P.S.I} = 12\mathrm{mV} * 95.91$$

$$1\ \mathrm{P.S.I} = 1.15091\mathrm{V}$$

$$1\mathrm{V} = 44.937\ \mathrm{mmHg}$$

Se debe recordar también que la resolución del microcontrolador es de 1024 valores distribuídos en 5V, en otras palabras, la resolución del microcontrolador es de 1024 bits, por lo tanto sus valores de cálculo en resolución con respecto a los valores en mm Hg se obtienen de la siguiente manera:

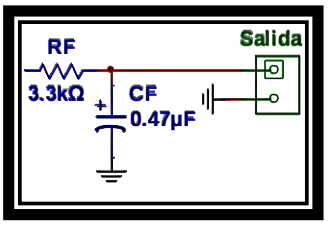
$$5\mathrm{V} = 224.685\mathrm{mmHg}$$

$$5\mathrm{V} = 1024\mathrm{b}$$

$$1\mathrm{b} = 0.219041\mathrm{mmHg}$$

Para reducir considerablemente el ruido ambiental, agente generador de error, se introdujo un filtro Pasivo pasa bajo, con frecuenca de corte aproximado a 100Hz (FIGURA 58), útil para atenuar dicho ruido ambiental.

FIGURA 58. Filtro pasa bajo de 100 Hz aproximadamente.



Fuente: Autor.

Para conseguir los valores de resistencia a implementar en el filtro mencionado, utilizamos su ecuación correspondiente utilizando un condensador de 0.47uF lo cual nos arrojara el dato de resistencia requerida Rf:

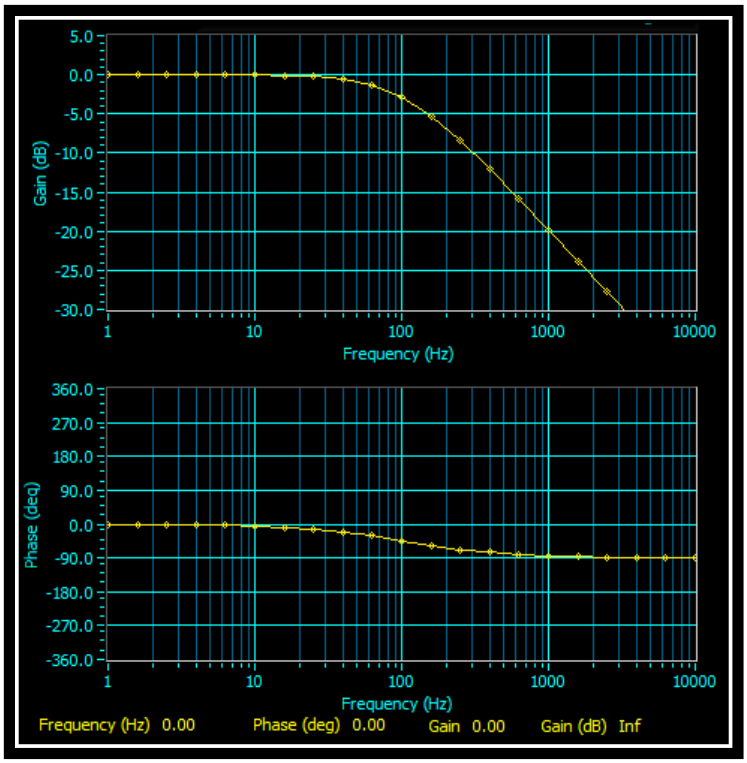
$$\mathrm{Rf} = \frac{1}{2\pi * \mathrm{fc} * \mathrm{Cf}}$$

$$Rf = \frac{1}{2\pi * 100Hz * 0.47uF}$$

$$Rf = 3386.27\Omega \approx 3.3K\Omega$$

En la FIGURA 59. podemos ver mediante un gráfico el comportamiento de la señal antes y después de ser atenuada lo cual nos da una idea de la importancia del filtro.

FIGURA 59. Gráfico del comportamiento de las señales antes y después de la atenuación.



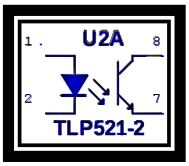
Fuente: Autor.

Una parte fundamental en el equipo tensiométrico, es el fragmento generador de presión en la extremidad, es decir la bomba y electroválvula , para la activación de las mismas, es altamente recomendable aislar la señal de control de la señal de potencia que activa la bomba, por lo que se ve necesario la utilización de optotransistores, que, diciéndolo de forma siempre, permiten controlar diversos mecanismos sin que estén conectados físicamente.

El optotransistor usado se encuentra en modo de saturación y corte, es decir, funciona como un interruptor, de esta manera evitamos corrientes de fuga que se podrían generar al ser activada la bomba o electroválvula

En el presente proyecto se utilizó el optotransistor TLP521-2. (FIGURA 60)

FIGURA 60. Optotransistor TLP521.



Fuente: Autor.

De forma parecida a la bomba, la electroválvula es activada por un transistor BJT 3904 que, al igual que la bomba, funciona como interruptor, es decir, se encuentra en modo de Corte y saturación, para que el transistor funcione adecuadamente según lo requerido, se debe calcular adecuadamente la resistencia que va antes del pin de base del transistor (resistencia de base), y se hace mediante la siguiente ecuación obteniendo el valor de resistencia de base correspondiente:

$$\mathrm{Ic} = \frac{\mathrm{Vcc}}{\mathrm{Rc}} = \frac{5\mathrm{V}}{100\Omega} = 0.05\mathrm{A}$$

$$\mathrm{Ib} = \frac{\mathrm{Ic}}{\mathrm{Hfe}} = \frac{0.05\mathrm{A}}{200} = 0.25\mathrm{mA}$$

$$\mathrm{Ib_{sat}} = \mathrm{Ib} * \mathrm{G_{ar}} = 0.25\mathrm{mA} * 4 = 1\mathrm{mA}$$

$$\mathrm{Rb} = \frac{\mathrm{Vcc} - 0.6\mathrm{V}}{\mathrm{Ibsat}} = \frac{5\mathrm{V} - 0.6\mathrm{V}}{1\mathrm{mA}} = 4.4\mathrm{K}\Omega$$

$$\mathrm{Rb} = \mathrm{Rb_1} + \mathrm{Rb_2} = 2.2\mathrm{K}\Omega + 2.2\mathrm{K}\Omega$$

$$\mathrm{Rb_1} = 2.2\mathrm{K}\Omega$$

$$\mathrm{Rb_2} = 2.2\mathrm{K}\Omega$$

Donde:

Vcc = Voltaje de Fuente.

Ic = Corriente de Colector.

Ib = Corriente de Base.

Hfe = Ganancia de Transistor.

$\mathrm{Ib_{sat}}$ = Corriente de base en saturación.

$\mathrm{G_{ar}}$ = Valor para garantizar la saturación en el transistor normalmente entre 2 a 10.

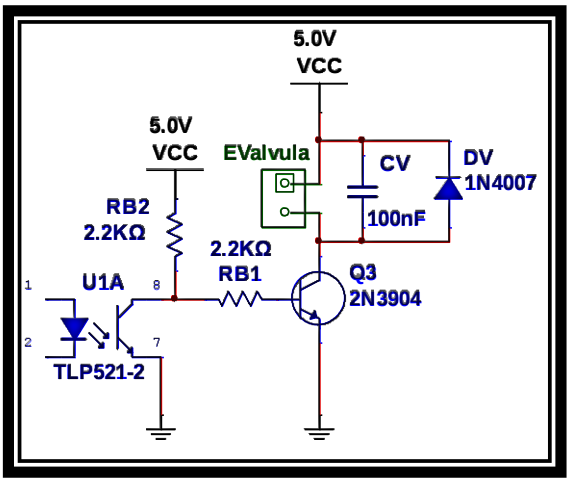
Rb = Resistencia de base.

Rb_1= Resistencia de Base 1.

Rb_2 = Resisencia de Base 2.

Como habíamos visto en un caso anterior, ya que no existe un valor de resistencia comercial de 4.4KΩ se pueden usar en serie dos resistencias de 2.2KΩ, con la ventaja de que una de las resistencias de base se pueden usar para acoplar la señal del optotransistor como se indica en la FIGURA 61.

FIGURA 61. Implementación del transistor y optoacoplador con los parámetros requeridos.



Fuente: Autor.

Para completar el circuito, concretamente para atenuar el ruido generado por los elementos que generan la presión en la extremidad, se implementa un condensador de 100nF para eliminar el ruido inductivo generado por la electroválvula y un diodo de protección de corrientes de desahogo, ya que al ser la electroválvula un elemento inductivo, al momento de ser desactivada, mantiene corrientes menores circulando a través de la misma, mismas corrientes que pueden tender a fugarse hacia la fuente o dañar otros componentes.

A diferencia de la electroválvula el motor se controla por medio de un transistor MOSFET (Metal-oxide-semiconductor Field-effect transistor), concretamente el transistor IRG640, de un tipo N de enriquecimiento que, al igual que los componentes

vistos anteriormente, se encuentra en zona activa y de corte, para que su funcionamiento sea como un interruptor.

Para futuras referencias, el funcionamiento de los transistores MOSFET de enriquecimiento esta basado en el control de canales entre las compuertas Drain (drenaje) y Source(fuente), este control esta ligado a la tensión aplicada en la compuerta gate(compuerta), siempre y cuando esta supere una tensión de Umbral (V_TH).

El Voltaje en Gate (Vg) del transistor esta controlado directamente por el optotransistor, si esta polarizado directamente, existirá una corriente de Colector, haciendo que la tensión en Vg sea 0V, caso contrario Vg será 5V siendo estos los únicos valores de Voltajes en el Gate garantizando el funcionamiento de la bomba y motor mediante la siguiente demostración basada en las ecuaciones de funcionamiento de los transistores de efecto de campo (MOSFET):

Cuando Vg=0

$$V_{GS} = V_G - I_D * R_S$$

$I_D = 0$ ya que V_{GS} seria Negativo y no cumple con $V_{GS} \geq V_{TH}$

Cuando Vg = 5V

$$V_{GS} = 5v - I_D * 10\Omega$$

$$k = 0.014 \frac{A}{V^2}$$

$$V_{TH} = 2V$$

$$I_D = k * (V_{GS} - V_{TH})^2$$

$$0.126 - 1.84I_D + 1.4{I_D}^2 = 0$$

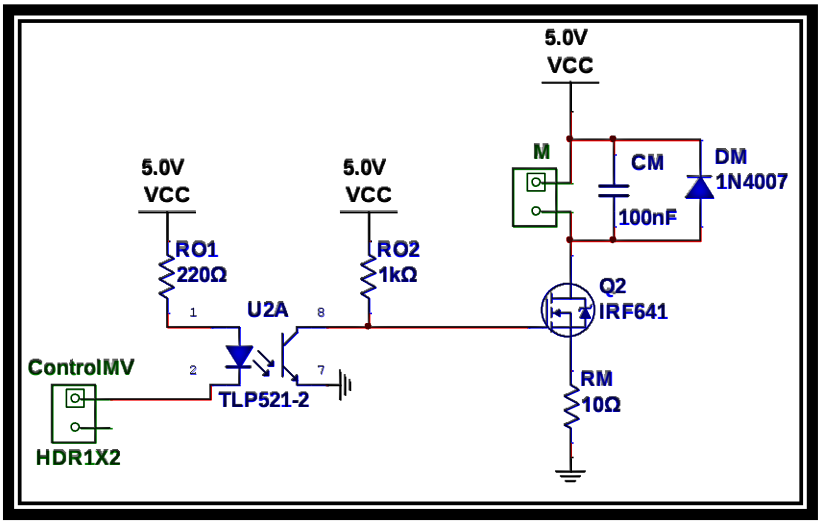
$$I_D = 0.07243055\ A$$

$$V_{GS} = 5v - 0.07243055\ A * 10\Omega$$

$$V_{GS} = 4.2756945V$$

Con lo que cumple $V_{GS} \geq V_{TH}$ y también cumple con la exigencia de corriente de la bomba ya que solamente se necesita 50mA (FIGURA 62.).

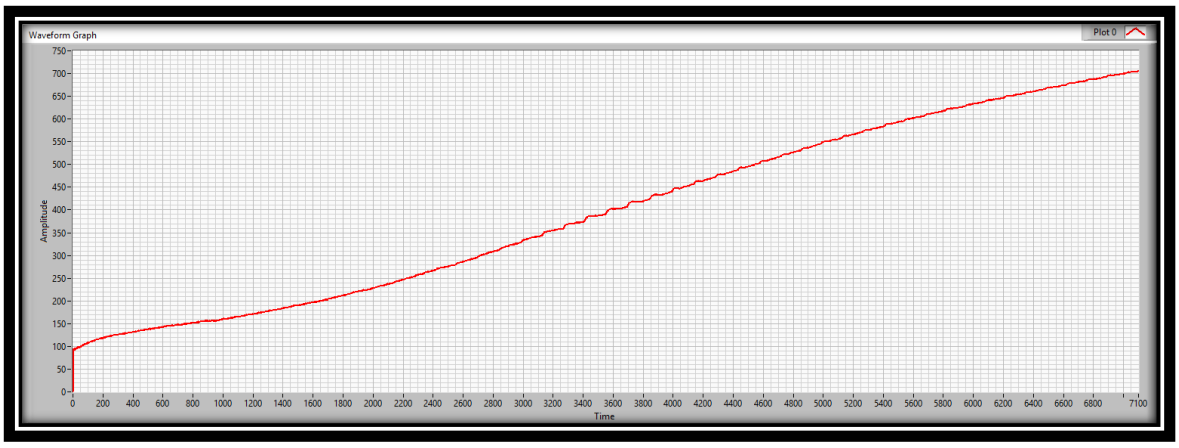
FIGURA 62. Esquema de control de bomba y electroválvula.



Fuente: Autor.

Al iniciar el proceso se activan al mismo tiempo los optotransistores que controlan a la bomba y a la electroválvula asegurando así que el aire circule por el mango del tensiómetro sin escaparse, en inflandolo cada vez más. El sensor captará la presión que se genere por la fuerza el mango en la muñeca del paciente, cuya señal recibida se visualiza en la FIGURA 63:

FIGURA 63. Visualización de la señal captada.

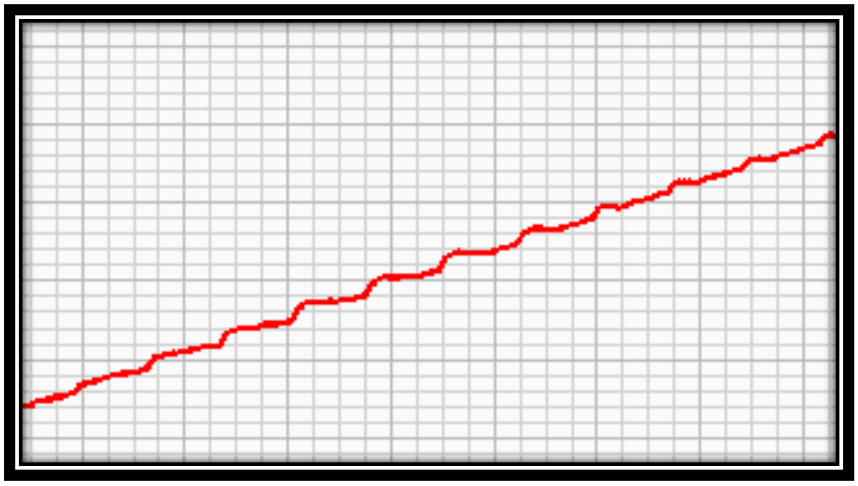


Fuente: Autor.

En el gráfico expuesto se puede observar que la presión va aumentando de forma casi lineal, sin embargo, el fragmento de señal que nos interesa , es una pequeña variación

de presión que se generan casi en medio de la señal completa vista en el gráfico anterior como se observa en la FIGURA 64, que es una ampliación de la anterior:

FIGURA 64. Fragmento de interés de la señal completa generada por el tensiómetro.



Fuente: Autor.

Estas variaciones, corresponden expansión y contracción de la arteria braquial, y basándonos en el análisis de las oscilaciones cíclicas, se puede obtener los valores propios del tensiómetro, es decir, los valores de presión arterial diastólica y sistólica.
Para obtener la onda lo más limpia posible, se necesita filtrar la señal adquirida, por lo que es necesaria la utilización de un filtro pasa banda digital IIR butterworth de modo 2, cuyas frecuencias de corte son 1Hz y 10 Hz.
Para obtener la ecuación diferencial y los coeficientes para el filtro, hay que normalizar las frecuencias dentro de las ecuaciones de filtros, de la siguiente manera:

$$\Omega = \omega * T_s$$

$$T_s = \frac{1}{f_s}$$

$$\omega = 2\pi f$$

$$\Omega = \frac{2\pi f}{f_s}$$

Donde:

Ω = Frecuencia Digital normalizada.

T_s= Periodo de Muestreo.

f_s=Frecuencia de Muestreo.

ω=Frecuencia angular analógica.

f=Frecuencia analógica.

La frecuencia digital esta entre $0 < \Omega < \pi$

$$f_s = 1\text{KHz}$$

$$\Omega_{c1} = \frac{2}{1000}\pi = 0.002\pi$$

$$\Omega_{c2} = \frac{2}{100}\pi = 0.02\pi$$

Con los valores de frecuencia adquiridos junto con la utilización de la herramienta de software Labview y su toolkit de Filtros digitales, se puede obtener fácilmente la siguiente función de transferencia:

$$H(z) = \frac{0.101997 - 0.101997z^{-2}}{1 - 1.79031z^{-1} + 0.796006z^{-2}}$$

De la cual se obtiene la ecuación diferencial:

$$y[n] - 1.79031y[n-1] + 0.796006y[n-2] = 0.101997x[n] - 0.101997x[n-2]$$

Donde:

y[n] = Salida actual.

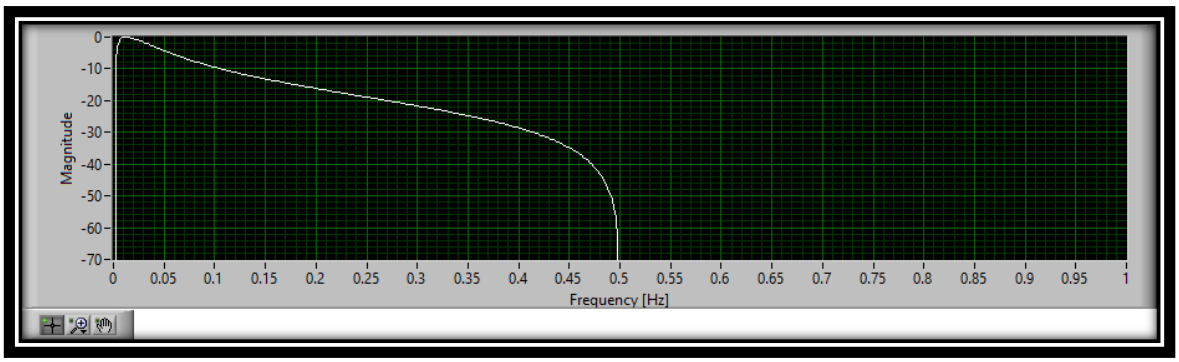
y[n-k] = Salida de k muestras anteriores.

x[n] = Muestra actual de entrada.

x[n-k] = k muestra anterior.

La FIGURA 65 muestra la respuesta del filtro digital en función de la Potencia vs Frecuencia.

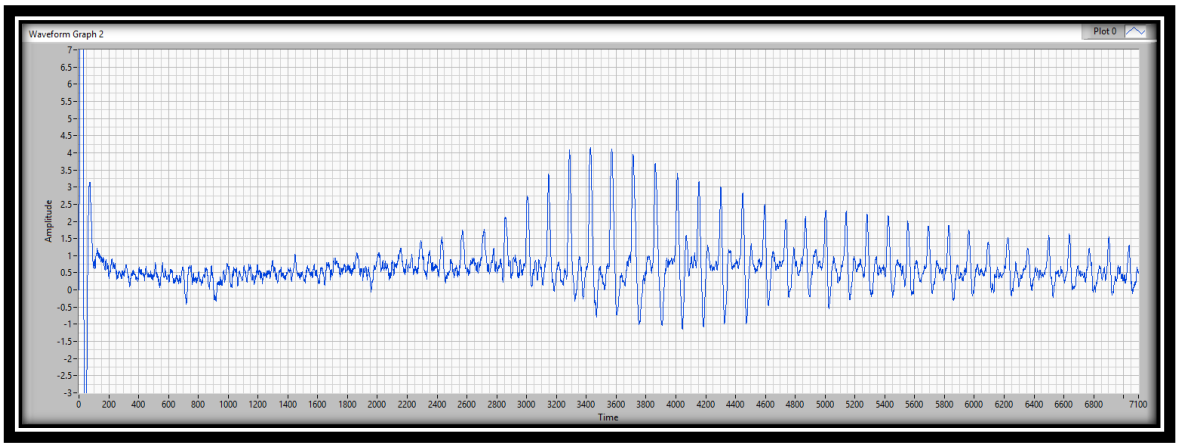
FIGURA 65. Respuesta del filtro digital en función de la Potencia vs Frecuencia.



Fuente: Autor.

Al aplicarse el filtro digital a la señal adquirida se obtendrá a la salida la forma de onda característica al método oscilatorio de medición de la presión arterial, la forma de onda resultante se muestra en la FIGURA 66:

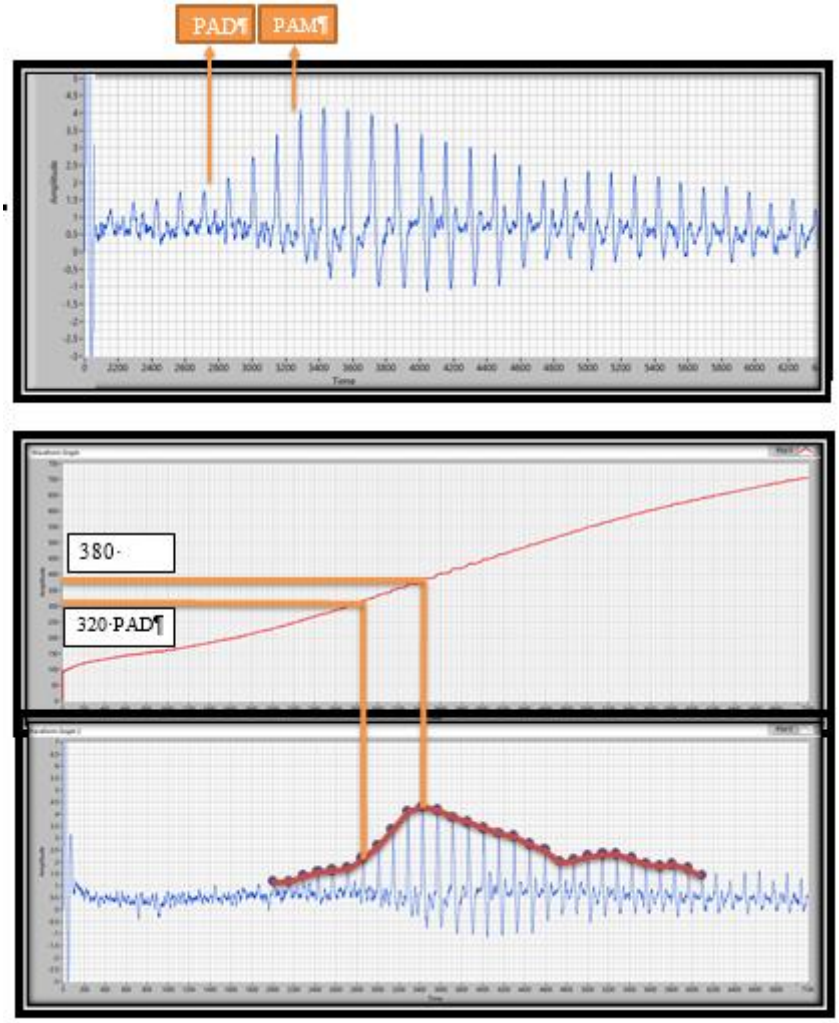
FIGURA 66. Forma de onda derivada de la aplicación del filtro a la señal adquirida.



Fuente: Autor.

En esta onda se puede distinguir que el pico más alto corresponde a la presión arterial media (PAM), y el pico inicial más cercano al 1/3 del pico más alto es la presión arterial diastólica (PAD) como se muestra en la FIGURA 67. Este procedimiento es indispensable para poder calcular la presión sistólica, pues primero se deben reconocer los puntos de los picos y sus correspondientes valores en la presión ingresada originalmente de la siguiente forma:

FIGURA 67. Disminución de picos de señal característicos PAD y PAM entre la señal propia del tensiómetro.



Fuente: Autor.

Una vez identificados los picos requeridos, pasamos aplicar las fórmulas propias del tensiómetro que nos permiten encontrar los valores propios de tensión arterial como se muestra a continuación:

$$\mathrm{PAM} = 380 * 1\mathrm{b}$$

$$\mathrm{PAM} = 380 * 0.219041\mathrm{mmHg}$$

$$\mathrm{PAM} = 83.2355\mathrm{mmHg}$$

$$\mathrm{PAD} = 320 * 0.219041\mathrm{mmHg}$$

$$\mathrm{PAD} = 70.09312\mathrm{mmHg}$$

Además de esto, se debe considerar lo siguiente:

$$PP = PAS - PAD$$

$$PM = PAD + \frac{PP}{3}$$

$$PAM = \frac{2PAD + PAS}{3}$$

$$PAS = 3PAM - 2PAD$$

Donde:

PP = Presión de pulso.

PAM = Presión Arterial Media.

PAD = Presión Arterial Diastólica.

PAS = Presión Arterial Sistólica.

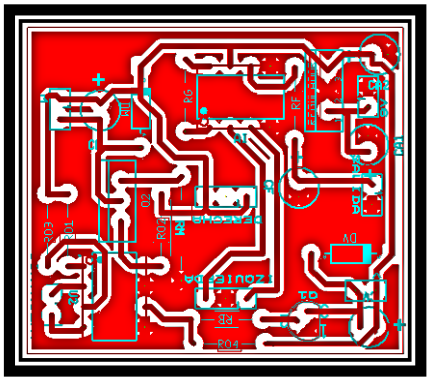
Por lo que:

$$PAS = 3 * 83.2355\text{mmHg} - 2 * 70.09312$$

$$PAS = 109.52\ \text{mmHg}$$

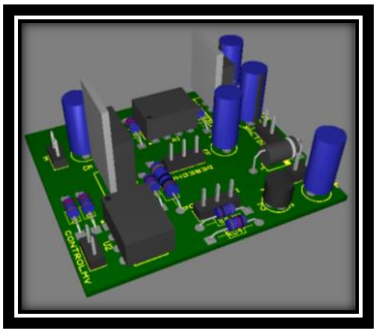
El circuito impreso correspondiente al control y adquisición de datos del tensiómetro digital se muestra en las FIGURA 68, 69 y 70:

FIGURA 68. Diseño del circuito de control y adquisición de datos del tensiómetro digital.



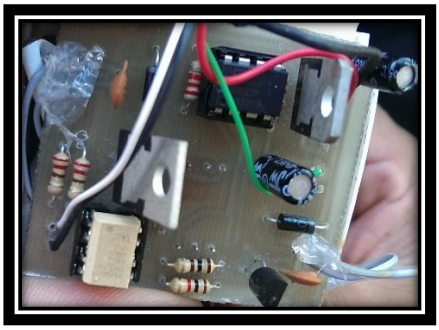
Fuente: Autor.

FIGURA 69. Simulación del circuito impreso real del control y adquisición de datos del tensiómetro digital.



Fuente: Autor.

FIGURA 70. Circuito impreso correspondiente al control y adquisición de datos del tensiómetro digital.



Fuente: Autor.

2.3 Oxímetro.

El marco teórico del oxímetro de pulso está citado en el capítulo anterior, pero a manera de resumen, se trabajó con un módulo de SpO2 cms50dl, el cual maneja un diodo LED rojo de longitud de onda 660nm, y un diodo LED infrarrojo de 880nm, su fuente de alimentación puede ser de 2.6V a 3.6V, la medición del SpO2 es de 0% a 100% y tiene un rango de pulsaciones por minuto de 30 ppm a 250ppm, además tiene un rango de error de medición de $\pm$ 2%.

2.3.1 Principio de Funcionamiento:

El principio de Funcionamiento del oxímetro de pulso se basa en la ley de Lambert Beer que relaciona las características de intensidad de absorción de la longitud de onda en la hemoglobina desoxigenada (Hb) y la hemoglobina oxigenada (HbO2) por un fotodiodo. La relación depende directamente de longitud que atraviesa la luz, la concentración del absorbente en el medio y el coeficiente de absorción como se muestra en la siguiente ecuación:

$$I_{roja,Ir} = I_{salida} * 10^{-l*\alpha}$$

$$\alpha = \frac{4\pi k_{\lambda}}{\lambda}$$

Donde:

$I_{roja,Ir}$=Intensidad de Luz transmitida por el LED rojo e infrarrojo.

I_{salida}=Intensidad recibida por el Fotodiodo.

l= Longitud que atraviesa la luz.

α= Coeficiente de absorción.

k_{λ}= Coeficiente de extinción.

λ=Longitud de onda de luz aplicada.

El coeficiente de extinción del cuerpo por donde atraviesa el haz, se refiere a la cantidad de luz absorbida por el mismo según la longitud de onda, por unidad de masa o la parte imaginaria del índice de refracción.

El oxímetro de pulso brinda una medición relativa no absoluta entre la cantidad de HbO_2 y Hb lo que se conoce como saturación funcional SpO_2.

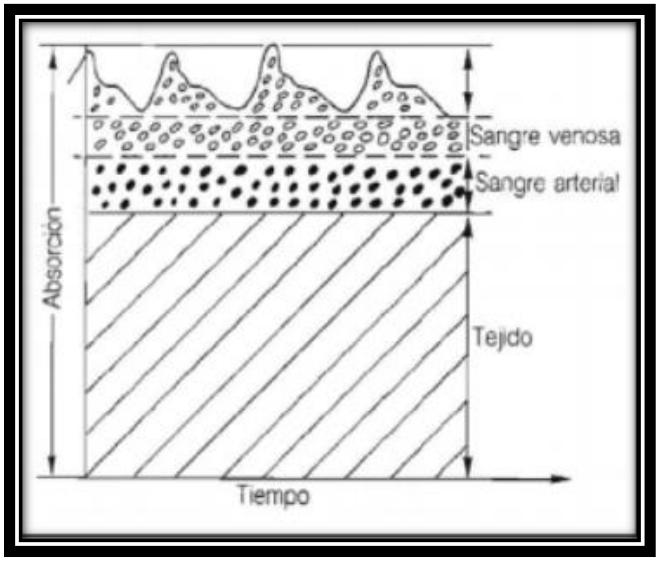
$$SpO_2 = 100\ x \frac{HbO_2}{Hb + HbO_2}$$

El SpO_2 se determina por la intensidad de luz roja que llega al fotodiodo, después de multiplexar el diodo LED rojo y el diodo LED infrarrojo funcionando uno a la vez.

Esta técnica se basa en el hecho de que el flujo de sangre arterial es pulsátil a diferencia del resto de tejidos y fluidos (FIGURA 71), haciendo que la luz se module dependiendo del flujo de sangre, se mide entonces la intensidad cuando existe pulso,

y luego la intensidad cuando no existe pulso para cada longitud de onda, la relación entre estas mediciones es el parámetro de absorbancia normalizado R, catalizados en el diagrama de bloques de la FIGURA 72 (Universidad CES, 2007).

FIGURA 71. Esquema del flujo de sangre arterial.



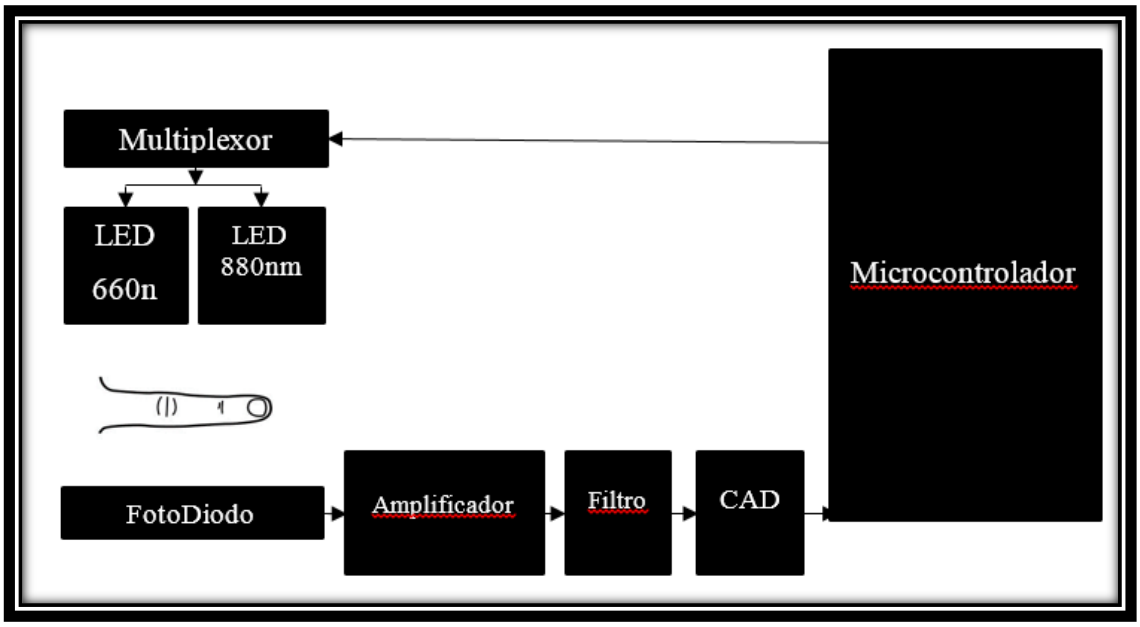
Fuente: (Universidad CES, 2007).

$$R = \frac{\frac{\text{Intensidad Pulsante}}{\text{Intensidad no Pulsante}}(\lambda 660)}{\frac{\text{Intensidad Pulsante}}{\text{Intensidad no Pulsante}}(\lambda 880)}$$

$$SpO_2 = 100 \text{ x } R$$

2.3.2 Diagrama de bloques:

FIGURA 72. Diagrama de bloques del funcionamiento del Oxímetro de pulso.

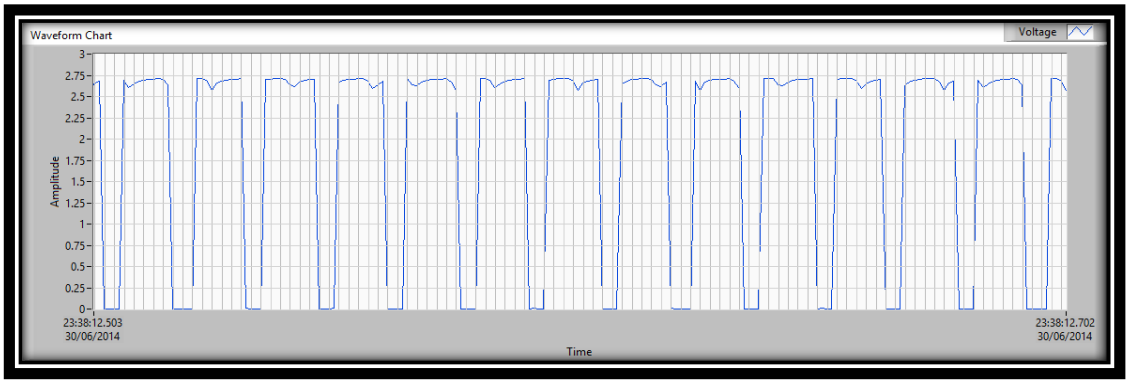


Fuente: Autor.

2.3.3 Desarrollo.

El módulo presentado tiene la ventaja de exponer los resultados en forma de un PWM (Modulación por ancho de pulsos) donde el porcentaje de SpO2 es el duty Cicle (ciclo de trabajo) de la señal como se muestra en la FIGURA 73.

FIGURA 73. Ciclo de trabajo del oxímetro en forma de PWM.



Fuente: Autor.

Se puede calcular de diferentes maneras el SpO_2 de la señal adquirida, pero en este caso se calculara la señal con un filtro digital butterworth modo 2, con frecuencia de corte de 10Hz (FIGURA 74). El cálculo del filtro paso bajo se la realiza de manera similar que para el filtro pasa banda del tensiómetro como se muestra a continuación.

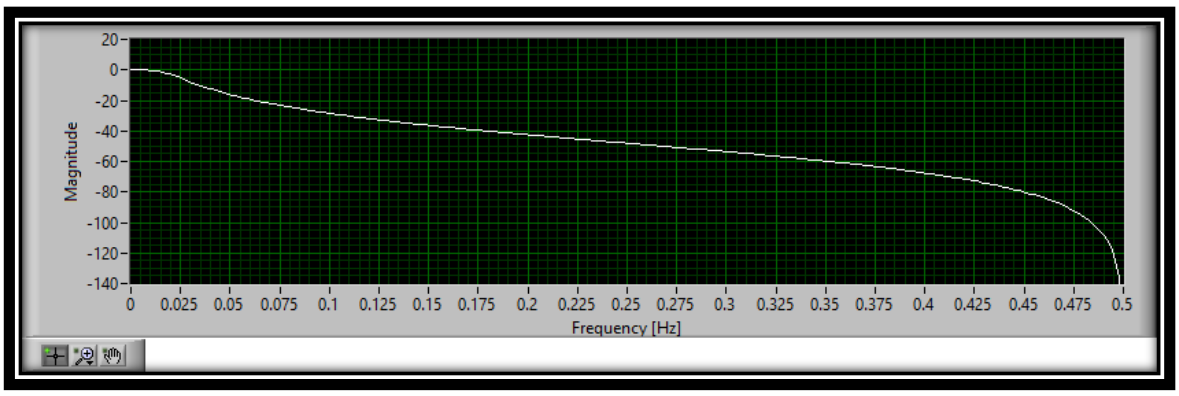
$$f_s = 1\text{KHz}$$

$$\Omega_c = \frac{2}{100}\pi = 0.02\pi$$

$$H(z) = \frac{0.00362168 + 0.00724336z^{-1} + 0.00362168z^{-2}}{1 - 1.82269z^{-1} + 0.837182z^{-2}}$$

$$y[n] - 1.82269y[n-1] + 0.837182y[n-2] = 0.00362168x[n] + 0.00724336x[n-1] + 0.00362168x[n-2]$$

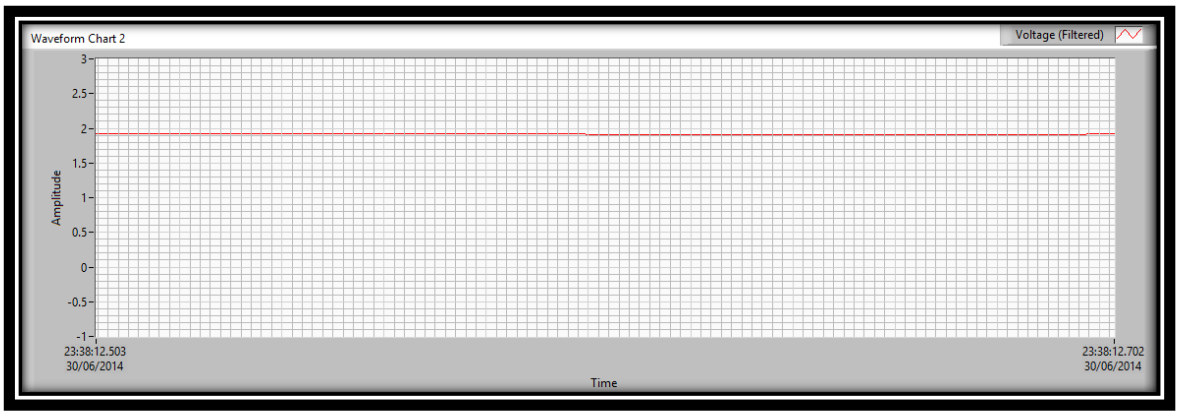
FIGURA 74. Respuesta del filtro digital en función de la Potencia vs Frecuencia.



Fuente: Autor.

El resultado de la señal filtrada es su valor equivalente a una señal analógica, como se muestra en la FIGURA 75.

FIGURA 75. Resultado de la señal filtrada.



Fuente: Autor.

Para obtener el SpO_2 se debe calcula de la siguiente manera:

$$SpO_2 = 100 \text{ x } \frac{V_{filtrado}}{V_{referencial}}$$

Donde:

$V_{filtrado}$ = Voltaje obtenido del filtro.

$V_{referencial}$ = Voltaje referencial equivalente al 100% de la señal PWM.

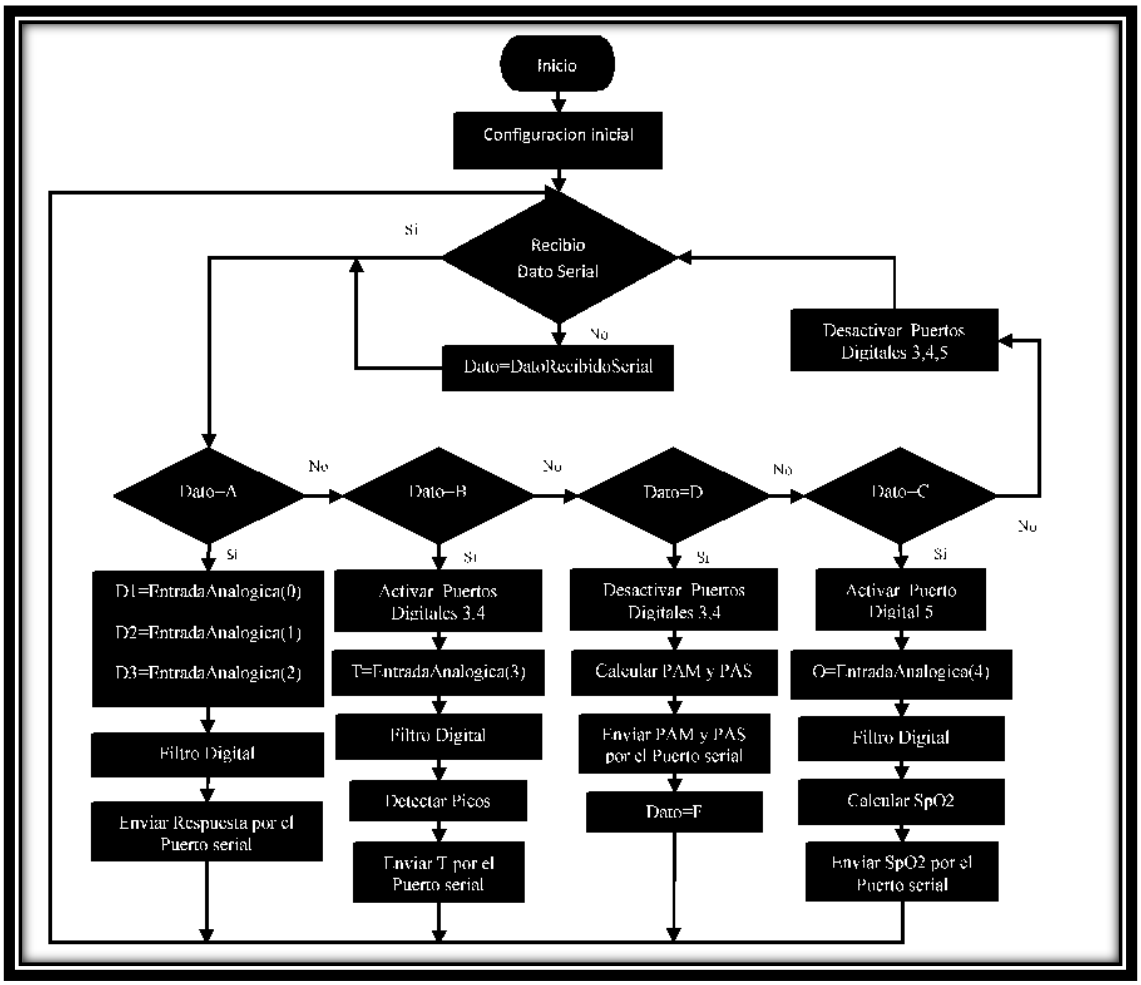
2.4 El Microcontrolador.

Una vez cumplidos con los objetivos de obtener la señal deseada de los respectivos sistemas, el microcontrolador debe ser el encargado de interactuar con el dispositivo móvil por medio del módulo bluetooth, esta interacción debe cumplir con los siguientes objetivos:

- Recibir los Datos del Módulo Bluetooth.
- Convertir las señales ingresadas por los puertos analógicos en digitales.
- Realizar procesamiento digital de señales, en especial filtros digitales.
- Enviar las respuestas de los procesamientos digitales por el puerto serial hacia el módulo Bluetooth.
- Activar y desactivar los puertos digitales.

Para cumplir con el propósito de controlar los dispositivos bioelectrónicos previamente mencionados, se debe obedecer al diagrama de flujo de la FIGURA 76

FIGURA 76. Diagrama de bloques del funcionamiento del microcontrolador.



Fuente: Autor.

CAPÍTULO 3.

SOFTWARE DEL DISPOSITIVO MÓVIL.

3.1 ANDROID.

El sistema operativo Android es una plataforma de Software basada en Linux. Diseñada inicialmente para dispositivos móviles, entre otras cosas, Android permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptados por Google a través del lenguaje de programación Java. Inicialmente desarrollada por Google Inc., al que después se unieron, un consorcio de 48 compañías de Hardware, Software y telecomunicaciones, que acordaron promocionar los estándares de códigos abiertos para dispositivos móviles. Google, sin embargo, fue la encargada de desarrollar mayoría del código fuente de Android bajo la licencia de Software Apache, una licencia de software libre y de código abierto a cualquier desarrollador.

Como ya se mencionó Android es una plataforma de código abierto. Esta característica implica que cualquier programador independiente puede crear y desarrollar aplicaciones escritas con lenguaje C u otros lenguajes y compilarlas a código nativo de ARM (Interfaz de Aplicación de Android) (Nuñez, 2013).

3.1.1 Características.

- Navegador integrado: basado en el motor open Source Web kit.
- Framework de aplicaciones: permite el reemplazo y la reutilización de los componentes.
- SQlite: base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.
- Máquina virtual Dalvik: Base de llamadas de instancias muy similar a Java.
- Telefonía GSM: dependiente del terminal.
- Cámara, GPS, brújula y acelerómetro: Dependiente del terminal.
- Bluetooth, EDGE, 3g y WiFi: dependiente del terminal.
- Multimedia: Soporte para medios con formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).

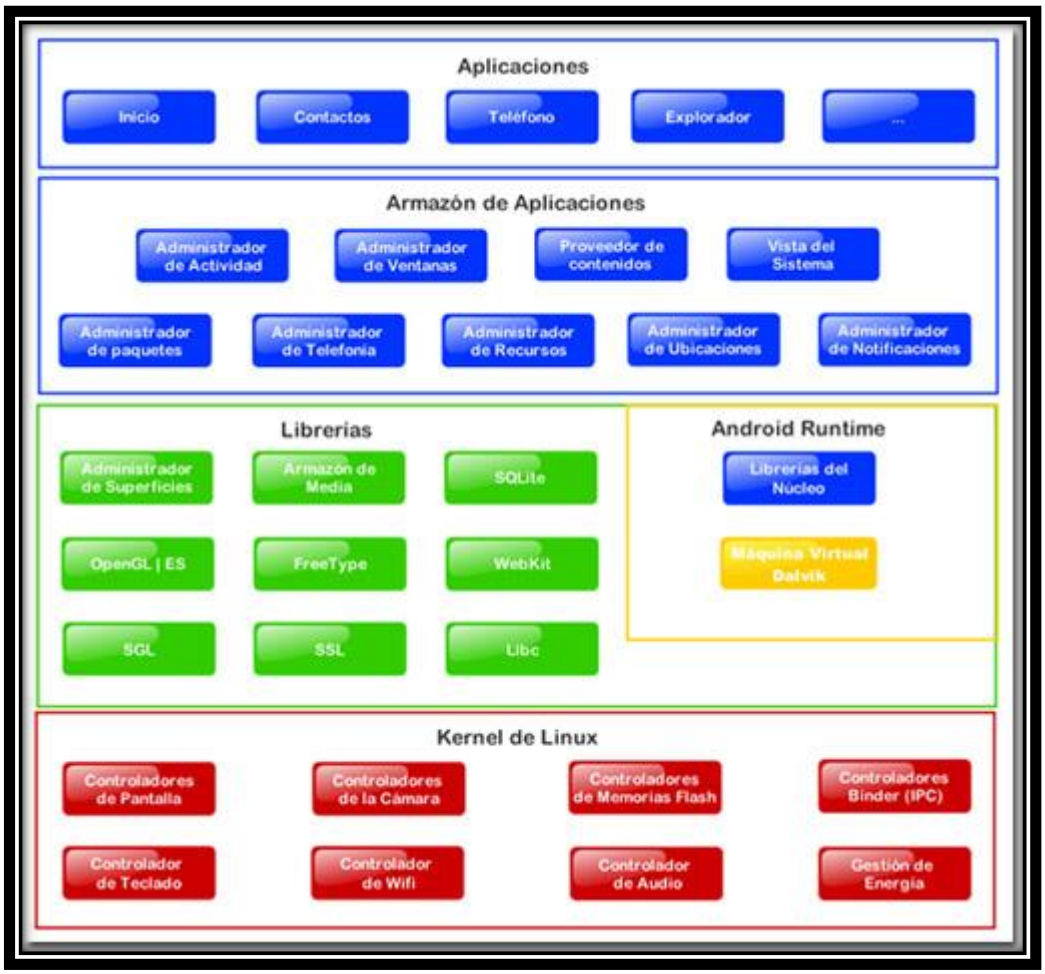
- Pantalla táctil.

-

3.1.2 Arquitectura de Android.

La arquitectura interna de la plataforma Android, está básicamente formada por 4 componentes (FIGURA 77):

FIGURA 77. Componentes de la arquitectura Android.



Fuente: (aplicaciones-android.org, 2011).

3.1.3 Aplicaciones.

Todas las aplicaciones creadas a través de la plataforma Android, incluirán como base un cliente de email (correo electrónico), calendario, programa de SMS, mapas, navegador, contactos, y algunos otros servicios adicionales. Todas escritas en el lenguaje de programación Java (aplicaciones-android.org, 2011).

3.1.4 Framework de Aplicaciones.

Todo desarrollador de aplicaciones Android, tienen acceso total al código fuente usado en las aplicaciones base. Con esto se evita que se generen componentes de aplicaciones distintas, que respondan a la misma acción, dando la posibilidad de que los programas sean modificados o reemplazados por cualquier usuario sin tener que empezar a programar sus aplicaciones desde el principio.

3.1.4.1 Librerías.

Android incluye en su base de datos un set de librerías C/C++, que están al alcance de todos los desarrolladores a través del framework de las aplicaciones Android System C library, librerías de medios, librerías de gráficos, 3D, SQlite, etc.

Android incorpora un set de librerías que aportan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. La Máquina Virtual está basada en registros, y corre clases compiladas por el compilador de Java que anteriormente han sido transformadas al formato .de (Dalvik Executable) por la herramienta "dx".

Android viene siendo competencia directa a los sistemas operativos móviles como Windows Mobile, Symbian, iPhone OS 3.0, etc. aunque también podría aminorar o reducir el impacto actual de Microsoft y sus Sistemas Operativos Windows, pues muchos grandes fabricantes de hardware y ordenadores en general contempla la posibilidad de implantar el sistema operativo Android en Portátiles, Notebooks, y PC; como es el caso de la compañía HP.

3.1.5 App Inventor.

En el año 2010, la empresa multinacional Google, especializada en productos que tienen que ver con internet y software relacionado; consiente de la complejidad que hasta entonces suponía crear aplicaciones para el sistema operativo ANDROID con el que cuentan gran cantidad de dispositivos móviles, comenzó a desarrollar un programa que facilitaría de gran manera la creación de dichas aplicaciones por parte de las personas que no poseían conocimientos superiores en el área de programación informática. A este programa se lo conoce con el nombre de **App Inventor.**
Un año después, aunque el proyecto era muy prometedor, Google decidió dejar de trabajar en él, no sin antes dejar libre el código para que cualquiera pueda retomar su labor donde la dejo, y es ahí donde entra en escena el Instituto tecnológico de Massachusetts (MIT) donde se creó el centro MIT para aprendizaje móvil y, con la ayuda económica de parte del mismo Google se daba forma al App Inventor como lo conocemos (Osuna, 2012).

3.1.5.1 Características y Ventajas.

En nuestros días el App Inventor se volvió toda una realidad. Este programa no requiere ser instalado pues se ejecuta desde el propio navegador. Posee un lenguaje gráfico y relativamente sencillo de aprender y dominar, permitiendo que cualquier usuario "novato" en asuntos de programación pueda desarrollar su propia aplicación.
Sumado a esto, la página principal de App Inventor incluye un completo soporte con anexos tales como:

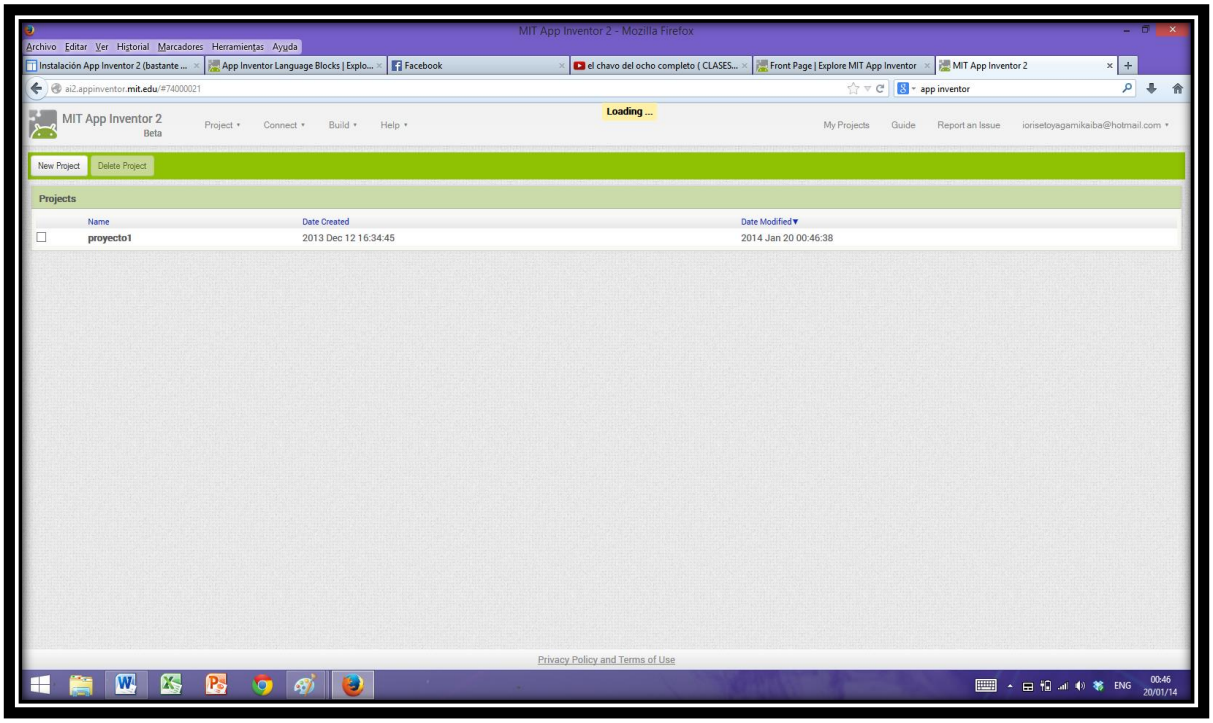
- Tutoriales
- Información relacionada con el programa y sus componentes
- Pautas para el desarrollo de aplicaciones en general.

Cabe resaltar que desde su primera versión (fase beta), el programa era absolutamente funcional y se encontraba al alcance de todos. Actualmente se está poniendo en marcha una segunda versión del App Inventor que incluye algunos cambios y mejoras que lo vuelven aún más práctico y fácil de usar sin dejar de lado su amigable esquema principal de funcionamiento (Osuna, 2012).

3.1.5.2 Esquema de Funcionamiento.

Para empezar a utilizar el App Inventor basta con crear una cuenta de usuario en google y acceder a ella a través de la página (ai2.appinventor.mit.edu), a partir de la misma, se puede escoger entre crear una nueva aplicación (New Project) o trabajar con las aplicaciones ya previamente iniciadas y almacenadas en nuestra cuenta.(FIGURA 78)

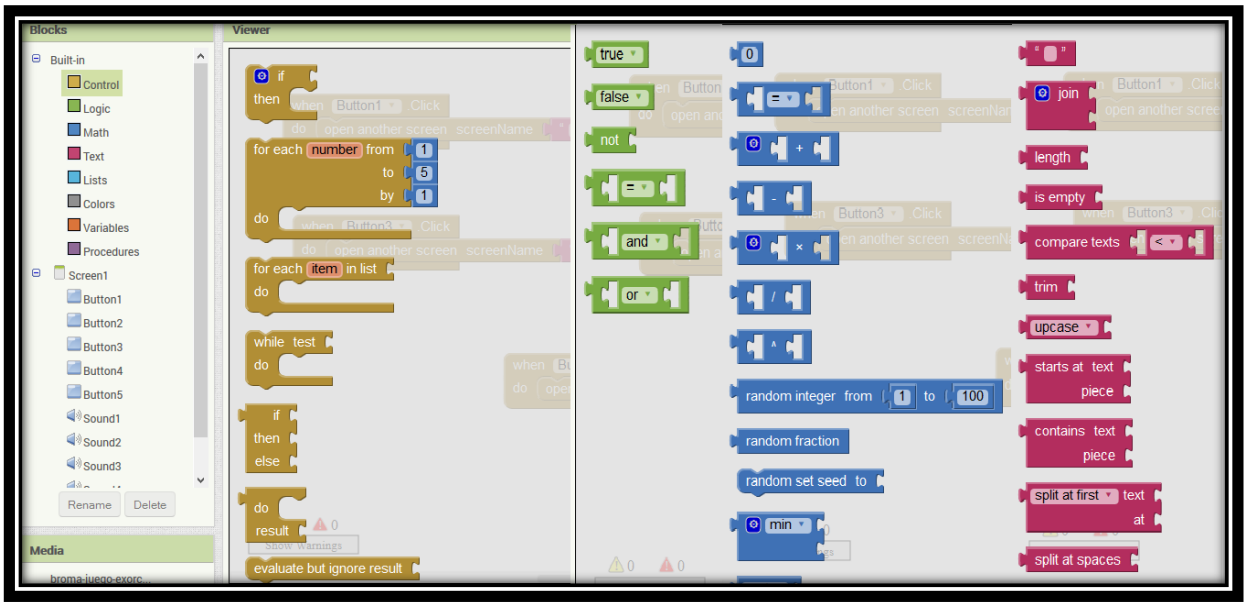
FIGURA 78. Formato de inicio de aplicación.



Fuente: Autor.

Una vez ingresada en la aplicación requerida, sea nueva o anterior, podemos apreciar que el patrón de programación realiza las sentencias en forma gráfica a través de bloques, los cuales se pueden ir tomando del banco de bloques ubicado a la derecha del lienzo, dependiendo si las sentencias son de carácter lógico, control, matemáticas, texto, listas, colores, variables, etc. (FIGURA 79.), de tal manera que el cuerpo del programa se desarrolle encajando piezas de forma lógica que cumplan una función en conjunto, y que a su vez, estén en concordancia con otro conjunto de piezas armadas equivalentes a las sentencias de programación (Riego, https://sites.google.com/site/appinventormegusta, 2013).

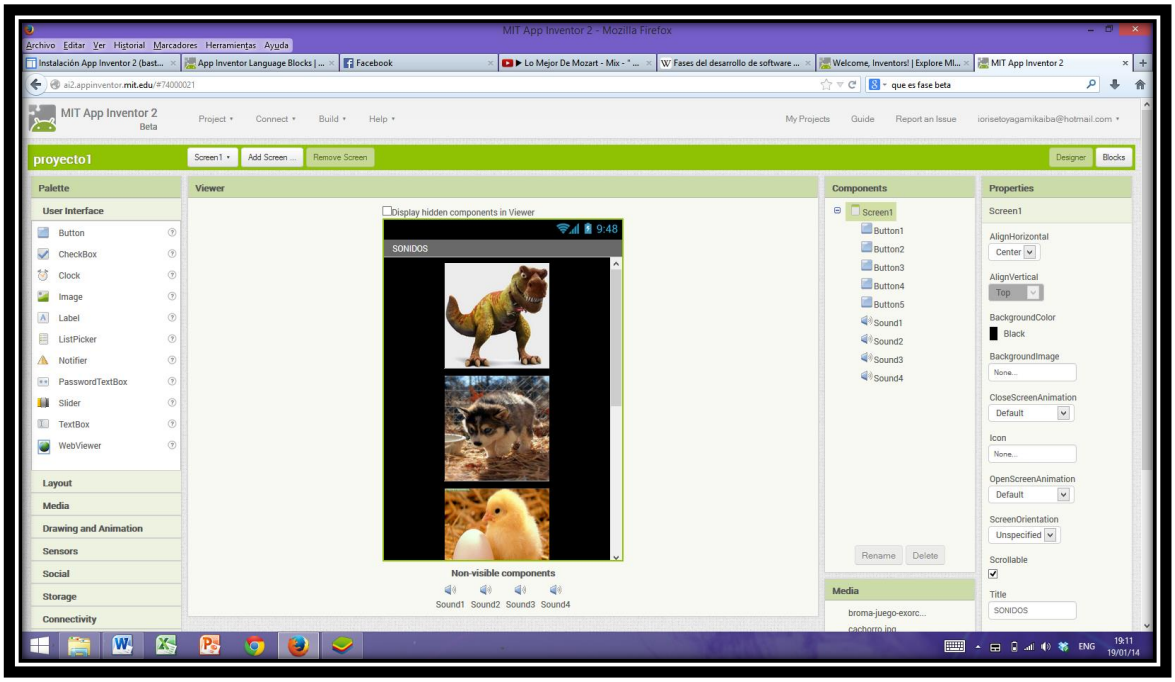
FIGURA 79. Banco de Bloques.



Fuente: Autor.

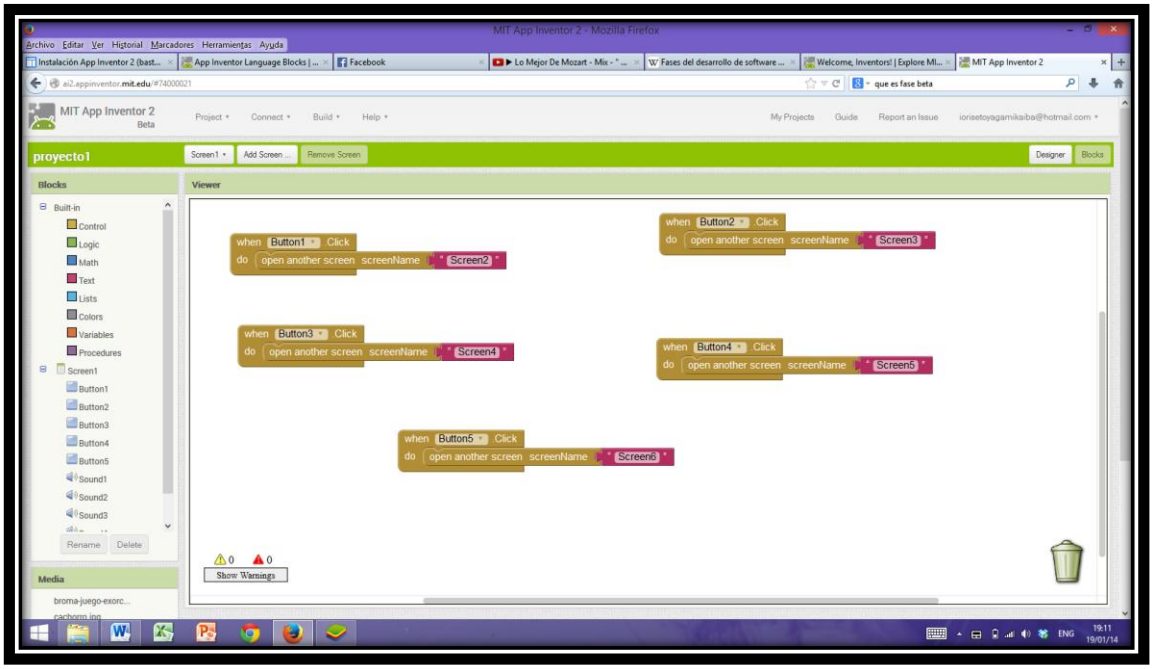
Por otro lado, el sistema cuenta con dos tipos de visualizadores, el de bloques (Blocks) que es donde se desarrolla el programa, y la pantalla de diseño (Designer) que muestra el diseño o los componentes como tales que se visualizaran en el display del móvil al momento de ejecutar el programa. (Riego, https://sites.google.com/site/appinventormegusta, 2013). (FIGURA 80 y FIGURA 81.)

FIGURA 80. Visualizador de diseño.



Fuente: Autor.

FIGURA 81.Esquema de bloques.



Fuente: Autor.

A su vez, una vez terminada nuestra aplicación, la podemos probar a través de las siguientes tres opciones (FIGURA 82):

1.- **Si se está usando un dispositivo Android y se posee conexión inalámbrica a Internet (WiFi) (recomendado)**. Es necesario instalar la aplicación Companion App Inventor en tu dispositivo.

2.- **Si no se cuenta con un dispositivo Android**, hay que instalar el software en la pc para poder utilizar el emulador de Android en la pantalla del mismo.

3.- **Si no se posee una conexión inalámbrica a Internet (WiFi)**, Se deberá instalar el software en la pc de modo que se pueda conectar al dispositivo Android a través de USB. La opción de conexión USB puede ser complicada, especialmente en Windows. Se recomienda utilizar este como un último recurso (Riego, https://sites.google.com/site/appinventormegusta, 2013).

FIGURA 82. Opciones de conexión.

Fuente (Riego, https://sites.google.com/site/appinventormegusta, 2013).

3.1.5.3 Desventajas.

Al ser AppInventor una aplicación en desarrollo, no es una herramienta lo suficientemente potente para cumplir todas las necesidades del desarrollador, por lo que se han superado sus limitantes por medio de APIs que se encuentran en un servidor al que se puede ser accedido por medio de cualquier red móvil.

3.2 Objetivos de la Aplicación.

Para poder cumplir con los requerimientos de interfaz gráfica del dispositivo móvil necesitamos establecer los siguientes objetivos:

- Una pantalla donde se visualiza el ingreso de sesión (login); permitiendo el acceso solo a usuarios registrados junto con una opción que permite registrar nuevos usuarios.
- Un Registro en la base de datos la información personal del usuario incluyendo fotografía de perfil.
- Un menú en el que se pueda elegir la señal que se desea adquirir, junto con opciones alternativas que permitan modificar datos del usuario, a su vez, al momento de elegir la señal (ECG, Oxímetro, Tensiómetro), se desplegara un submenú de opciones correspondientes con la señal seleccionada, mismos que detallamos a continuación:

- **Electrocardiograma**
    - Conectar el dispositivo electrónico vía bluetooth, caso contrario, denegar la opción.
    - Adquirir datos en tiempo real y graficar simultáneamente las tres derivaciones del ECG.
    - Borrar el registro actual en cualquier momento en caso de error.
    - Grabar datos adquiridos en la base de datos.
    - Mostrar el resultado final de la adquisición.
    - Revisar historiales de registros anteriores

- **Oxímetro de Pulso**
    - Conectar el dispositivo electrónico vía bluetooth, caso contrario, denegar la opción.
    - Adquirir datos en tiempo real y mostrar al usuario el porcentaje de oxígeno en la sangre.
    - Borrar el registro actual en cualquier momento en caso de error.
    - Grabar datos adquiridos en la base de datos.
    - Revisar historiales de registros anteriores.

- **Tensiómetro**
    - Conectar el dispositivo electrónico vía bluetooth, caso contrario, denegar la opción.
    - Adquirir datos en tiempo real.
    - Calcular los valores de presión diastólica y sistólica basada en los datos adquiridos.
    - Exhibir los valores de presión diastólica y sistólica.
    - Borrar el registro actual en cualquier momento en caso de error.
    - Grabar datos adquiridos en la base de datos.
    - Revisar historiales de registros anteriores.

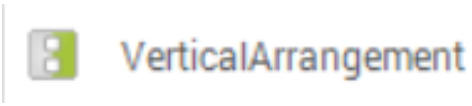
Nota: Para poder cumplir con los objetivos de interfaz gráfica en el dispositivo móvil es necesario recurrir a aplicaciones web debido a las limitaciones del app inventor.

3.3 Desarrollo de Interfaz Gráfica en el Dispositivo Móvil.

3.3.1 Pantalla de Bienvenida y Sesión.

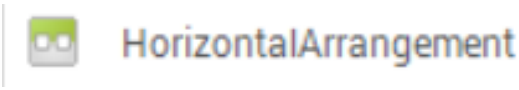
Dentro del software de programación de móviles **Android App Inventor**, para poder cumplir con una interfaz atractiva y amigable se requieren los siguientes elementos:

3.3.1.1 VerticalArrangement.



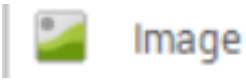
Se encuentra en la sección **Layout** dentro del diseño de App inventor, y sirve para organizar los elementos uno debajo del otro arrastrados sobre éste.

3.3.1.2 HorizontalArrangement.



Se encuentra en la sección **Layout** dentro del diseño de App inventor, y sirve para organizar los elementos uno a lado del otro arrastrados sobre éste.

3.3.1.3 Image.



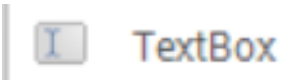
Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente sirve para mostrar una imagen que sea precargada en su propiedad Picture, o para mostrar cualquier imagen que sea modificada dentro de la sección de programación por bloques del App inventor.

3.3.1.4 Label.

Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente sirve para mostrar una porción de texto que sea precargada en su

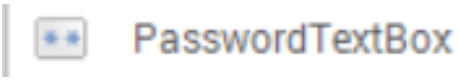
propiedad Text o sea modificada dentro de la sección de programación por bloques del App inventor.

3.3.1.5 TextBox.



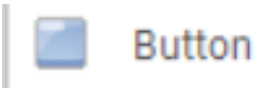
Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente coloca un texto dentro de una caja que sea precargada en la propiedad Text, a su vez, puede ser modificado dentro de la sección de programación por bloques de App inventor o ingresado por el usuario.

3.3.1.6 PasswordTextBox.



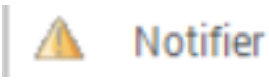
Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente coloca el texto oculto tipo contraseña dentro de una caja que sea precargada en su propiedad Text, es decir, modificado dentro de la sección de programación por bloques del app inventor o ingresado por el usuario.

3.3.1.7 Button.



Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente puede detectar los clicks y su aspecto puede ser modificado en sus propiedades o dentro de la sección de programación por bloques de App inventor.

3.3.1.8 Notifier.



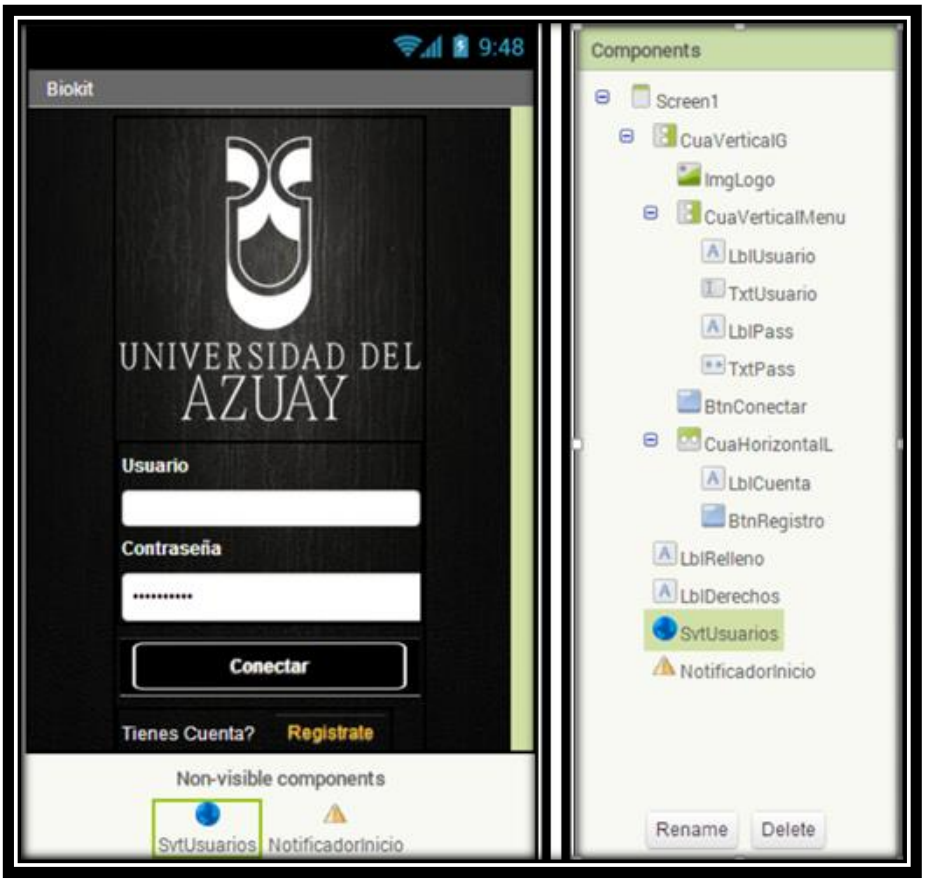
Se encuentra en la sección **User Interface** dentro del diseño de App inventor, este componente puede generar alertar y reportes Android.

3.3.1.9 Web.

Se encuentra en la sección **Connectivity** dentro del diseño de App inventor, este componente no es visible y es capaz de generar funciones web tales como http, post, get, put, etc.

Una vez conocida la función de los componentes a usar, se procede a colocar dichos componentes como se muestra en la FIGURA 83, cambiando sus nombres por defecto por nombres fácilmente reconocibles, y precargando las imágenes para una mejor sensación visual.

FIGURA 83. Aplicación de los componentes de interfaz gráfica de AppInventor.

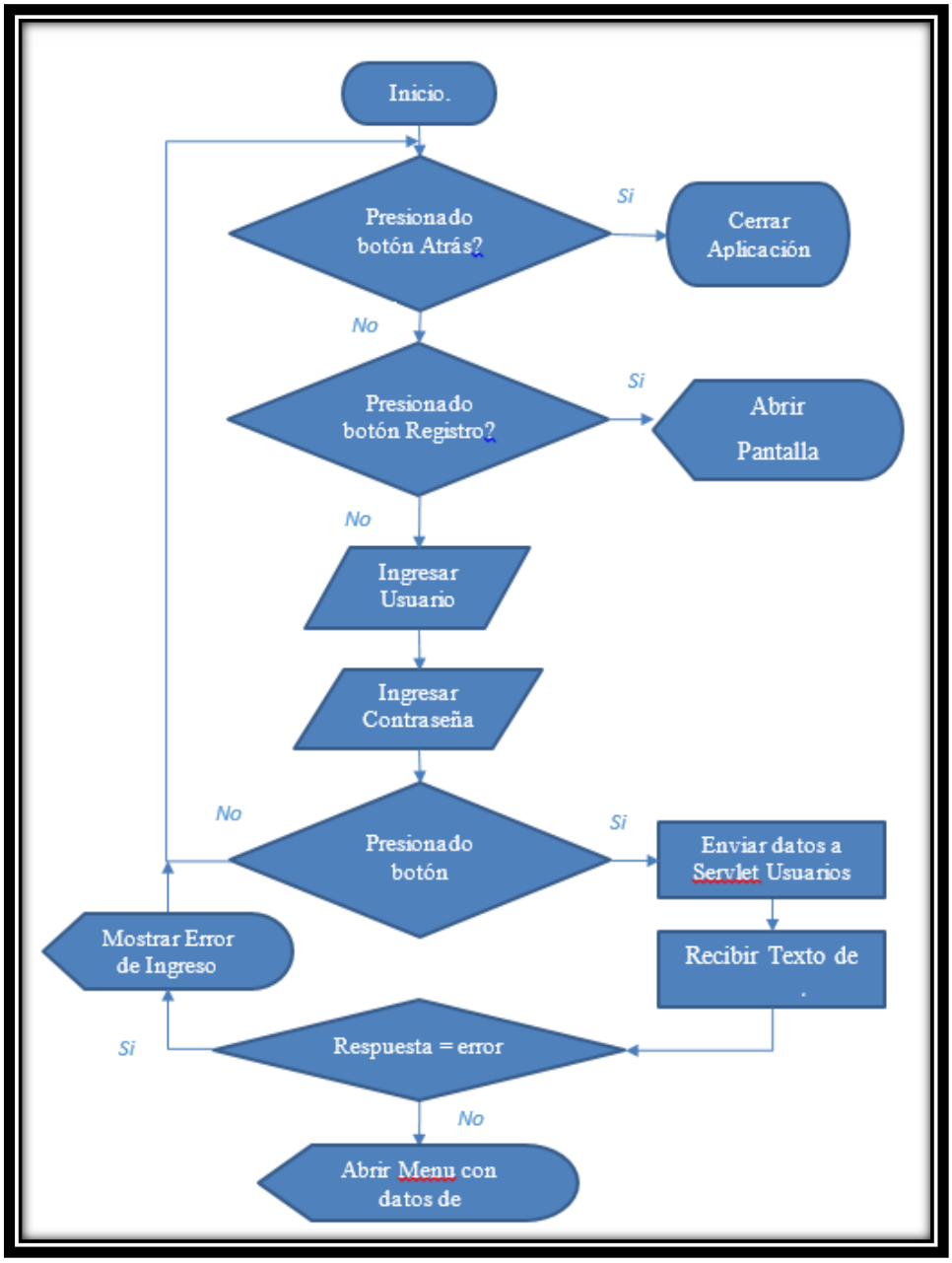


Fuente: Autor.

La programación correspondiente obedece al diagrama de flujo de la FIGURA 84 para cumplir con la función de Login o Inicio de sesión de los diferentes usuarios los datos ingresados se verifican en la base de datos por medio de un servlet donde se acepta o

rechaza el acceso a la aplicación. Además, en caso de no ser un usuario registrado, existe la posibilidad de llamar a otra pantalla para el registro.

FIGURA 84. Diagrama de flujo de la programación de la aplicación.

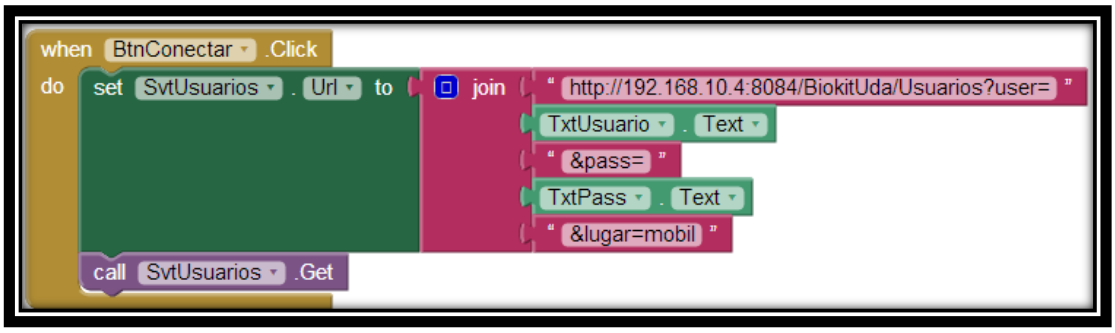


Fuente: Autor.

En el diagrama de bloques de App inventor se puede generar un evento de click en el botón “Conectar” (FIGURA 3 9), en este caso, se toman los valores ingresados en los campos de texto **TxtUsuario** y **TxtPass** para invocar al servlet Usuarios. El servlet

Usuarios verifica la coincidencia de los datos que ingresa el usuario con los que existen en la base de datos, devolviendo un texto con su nombre y apellido en el caso de estar registrados, y un texto de error en el caso de no estarlo. El diagrama de bloques se ilustra en la FIGURA 85.

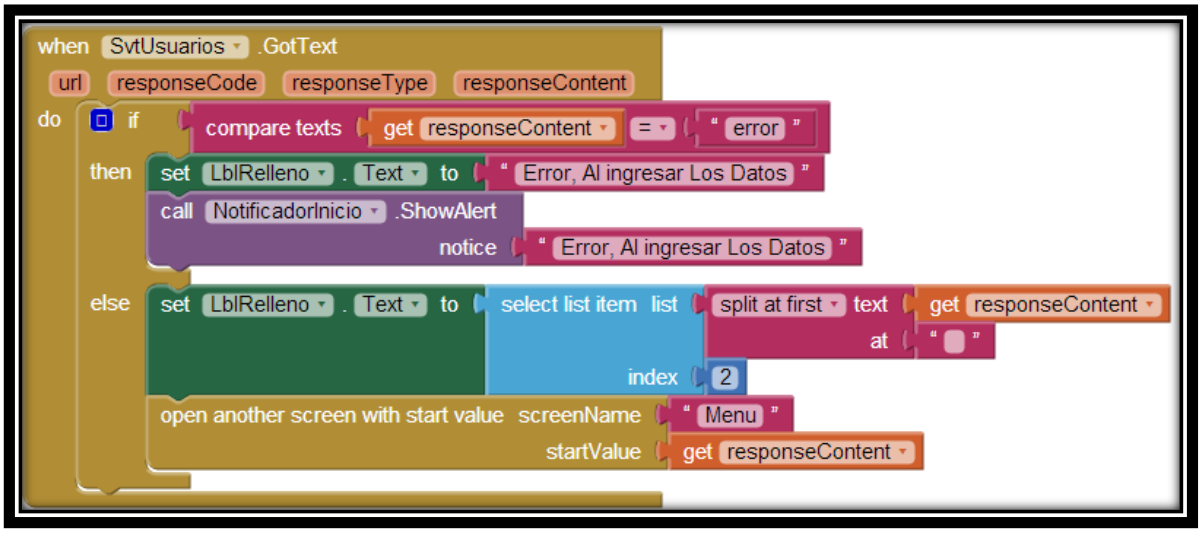
FIGURA 85. Bloques de evento Click del Botón Conectar.



Fuente: Autor.

Cuando el componente web recibe un texto como el que devuelve el servlet se genera un evento GotText, en este, comprobamos el texto recibido, si el texto es un error, por medio del NotificadorInicio mostramos un mensaje al usuario “Error al ingresar los datos”, en caso contrario se abrirá la pantalla Menú pasándole el texto recibido. El bloque de GotText se muestra en la FIGURA 86.

FIGURA 86. Bloques de evento GotText del componente Web.



Fuente: Autor.

Si se presiona el botón Registro se dará paso a la pantalla Registro el bloque se muestra en la FIGURA 87.

FIGURA 87. Bloque del evento click en el botón Registro.

Fuente: Autor.

Al presionar el botón de atrás se cerrará la aplicación. El bloque se muestra en la FIGURA 88.

FIGURA 88. Bloque del evento click en el botón atrás.

Fuente: Autor.

3.3.2 Pantalla de Registro.

La pantalla de registro tiene la capacidad de ingresar la información básica del usuario para luego ser grabada en la base de datos. Además advierte de datos ingresados por usuarios anteriores como nicknames o emails y la no correspondencia de la verificación de las contraseñas. Como campos se colocaron:

Nickname: Apodo que el usuario desee tener para ingresar a su sesión.

Nombre: Nombre del usuario.

Apellido: Apellido del usuario.

Email: Correo electrónico donde se puede comunicarse con el usuario.

Teléfono: Número de teléfono donde se puede contactar con el usuario.

Contraseña: Contraseña con la cual el usuario desee conectarse a su sesión.

Comprobar Contraseña: Verificación de la contraseña anteriormente ingresada.

Dirección: Dirección de domicilio o trabajo del usuario.

Ciudad: Ciudad de domicilio o trabajo del usuario.

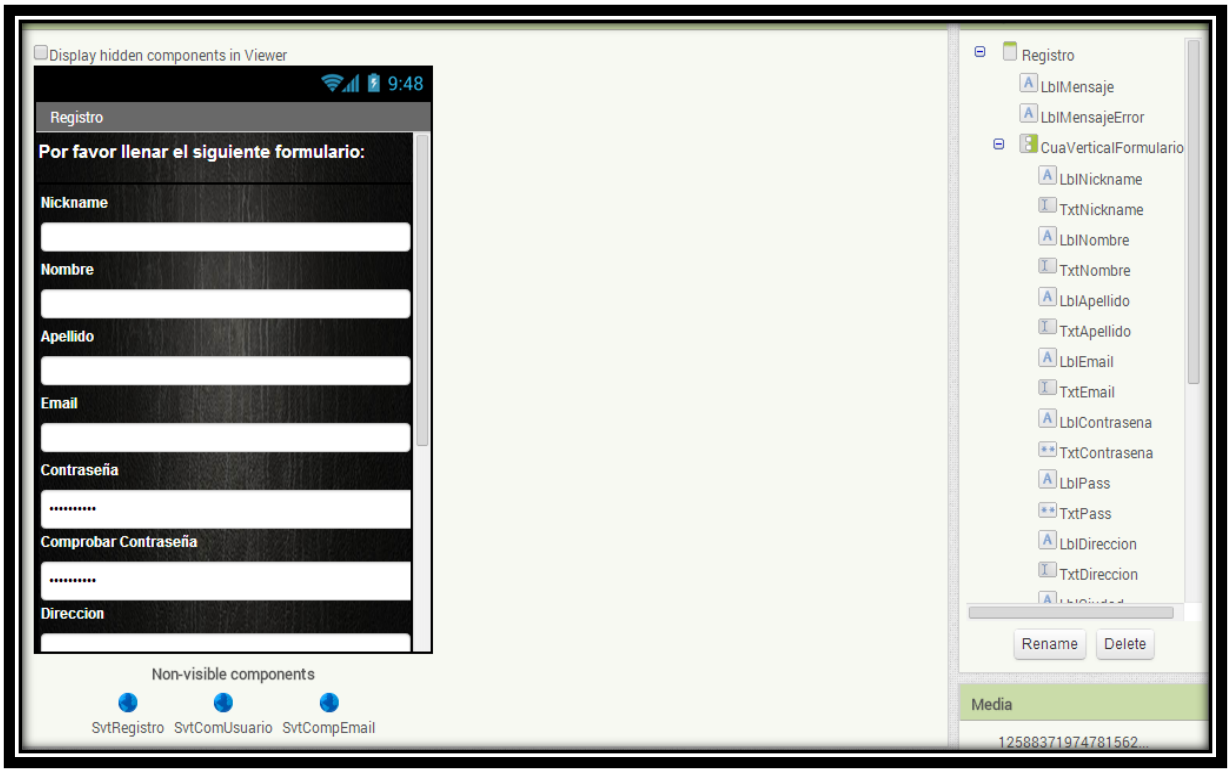
Provincia/Estado: Provincia o Estado correspondiente al domicilio o trabajo del usuario.

País: País de residencia o trabajo del usuario.

Zip Code: Código Postal del domicilio o trabajo del usuario.

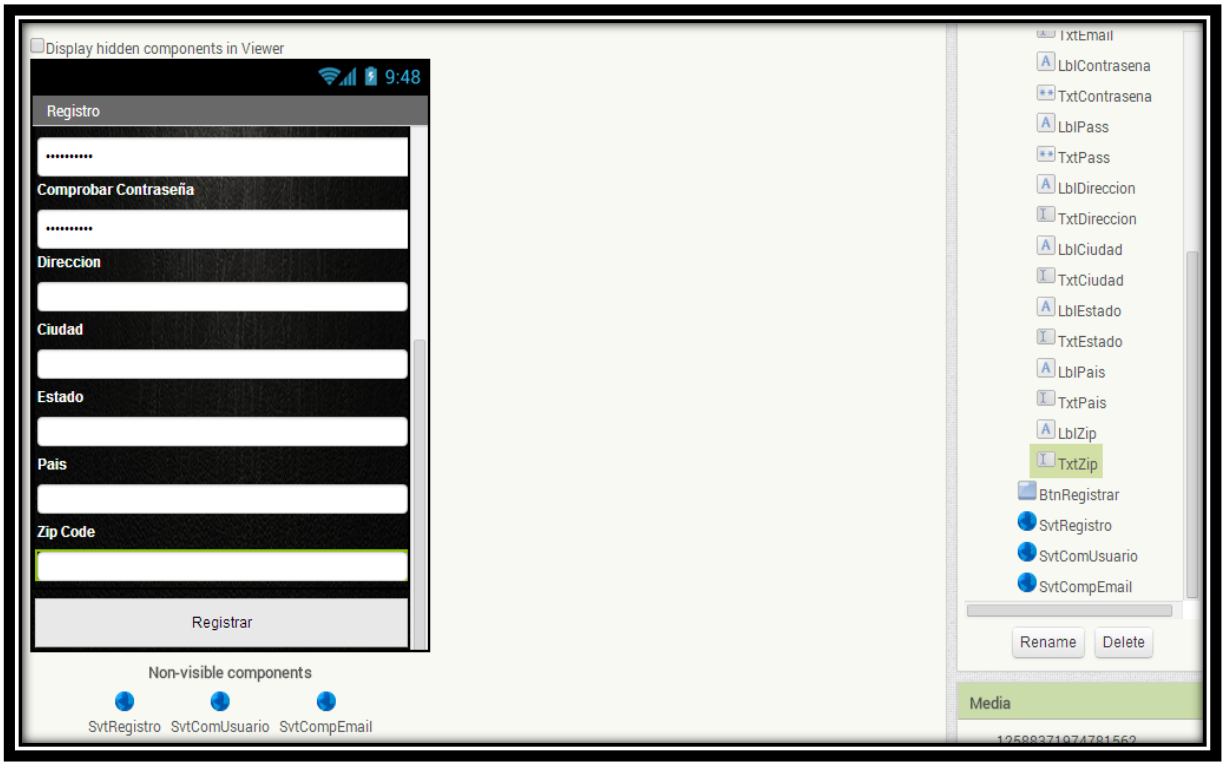
Para el diseño de la pantalla del registro se utilizaron componentes previamente explicados como se lo puede ver en la FIGURA 89 y FIGURA 90.

FIGURA 89. Diseño de la pantalla de Registro en la sección de App inventor parte1.



Fuente: Autor.

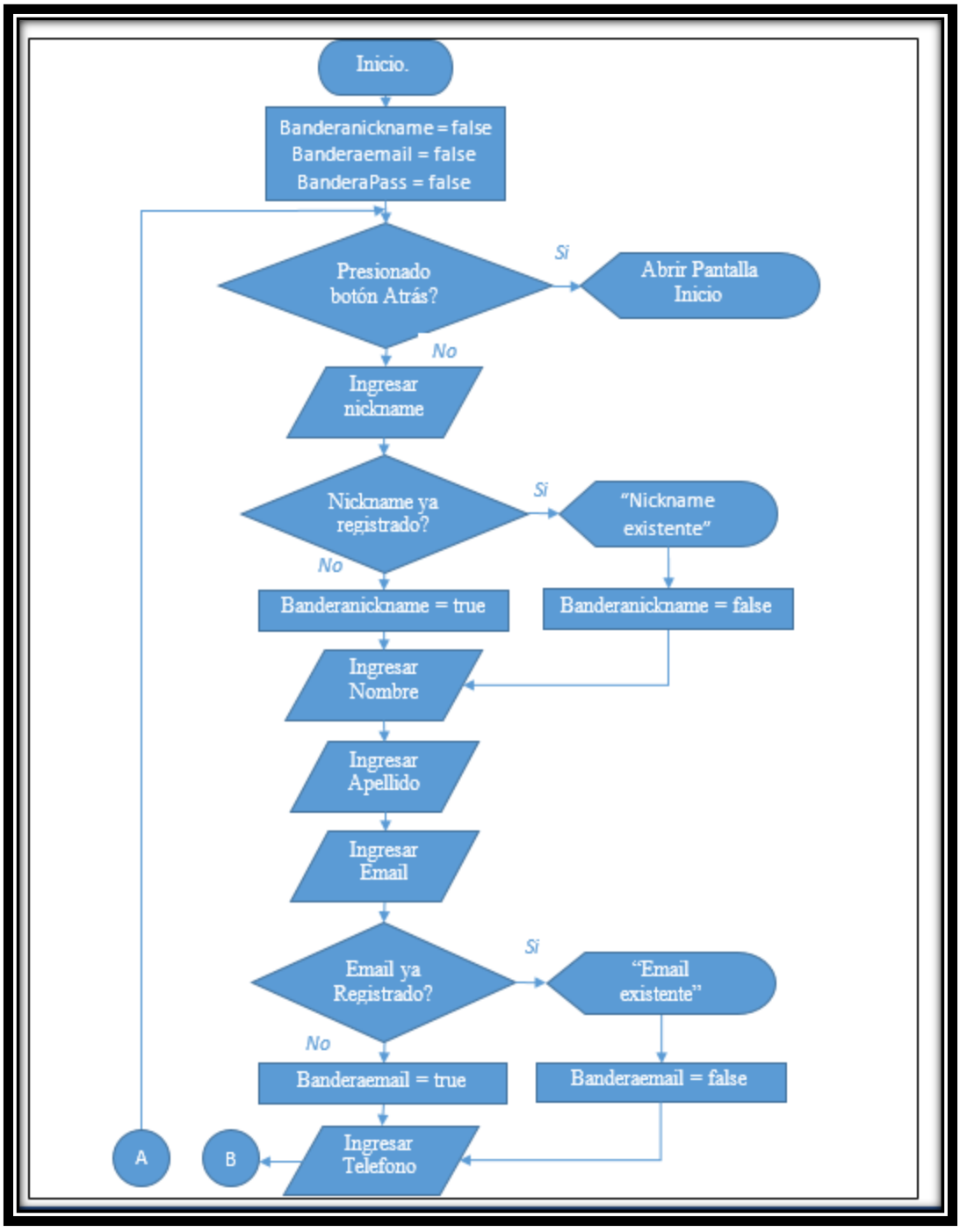
FIGURA 90. Diseño de la pantalla de Registro en la sección de App inventor parte2.



Fuente: Autor.

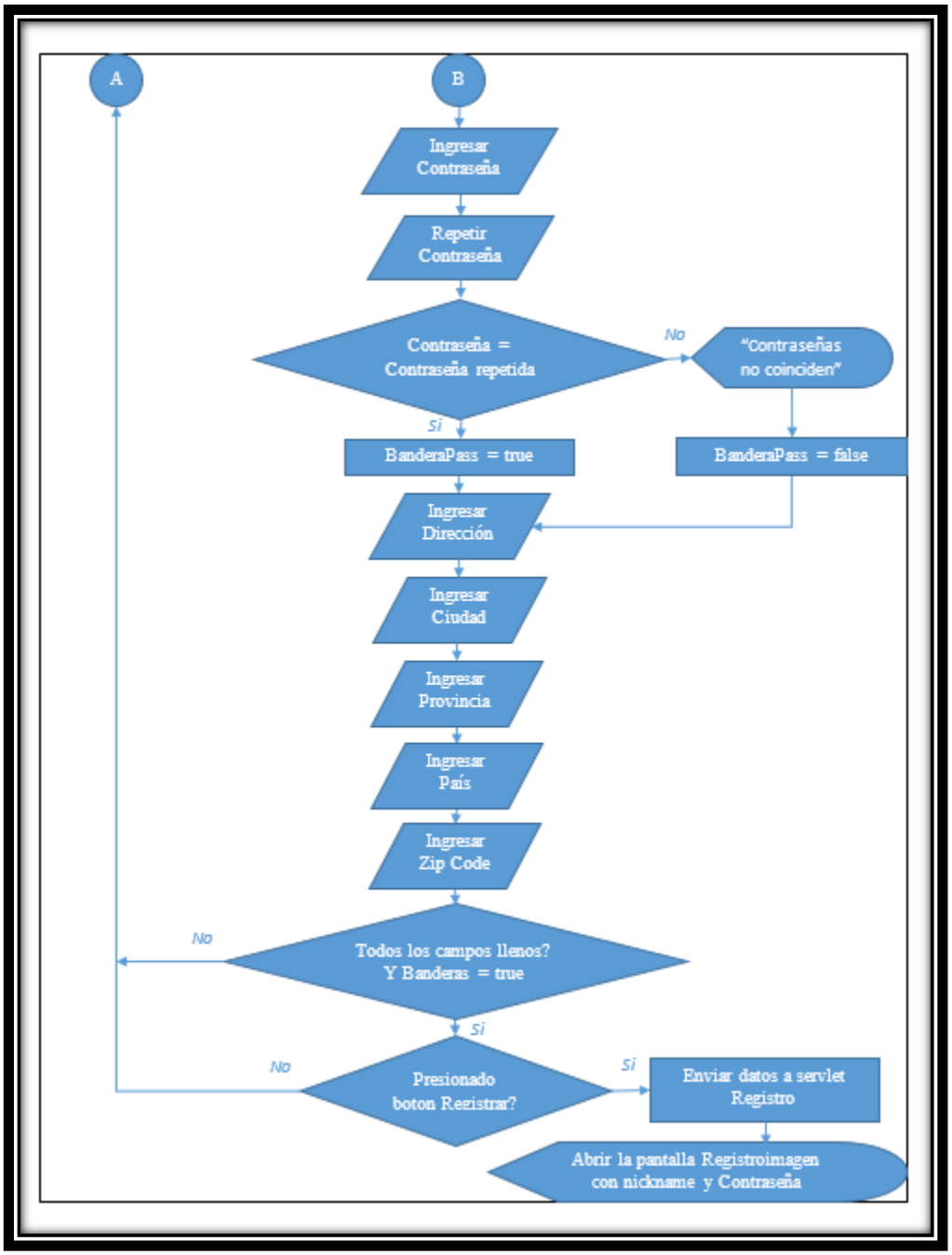
Nota: Si los datos ingresados son correctos se da paso al registro, que incluye una fotografía del usuario para su fácil reconocimiento. El diagrama de bloques de la pantalla de Registro obedece al diagrama de flujo de la FIGURA 91 y FIGURA 92

FIGURA 91. Diagrama de flujo del funcionamiento de la pantalla de Registro.



Fuente: Autor.

FIGURA 92. Diagrama de flujo del funcionamiento de la pantalla de Registro.



Fuente: Autor.

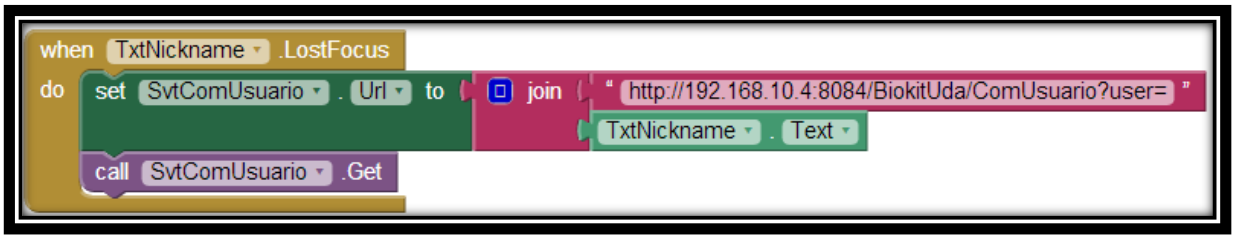
Para que se cumpla con el diagrama de flujo inicializamos las variables de las banderas con valores "false" como se muestra en la FIGURA 93.

FIGURA 93. Bloques de inicialización de variables.

Fuente: Autor.

Inicializadas las variables, se ingresa los datos del usuario, si se llega a perder el enfoque en el área de texto **TxtNickname,** se dispara un evento llamado LostFocus, que se encarga de invocar al servlet ComUsuario con el dato ingresado en dicha área como se muestra en la FIGURA 94.

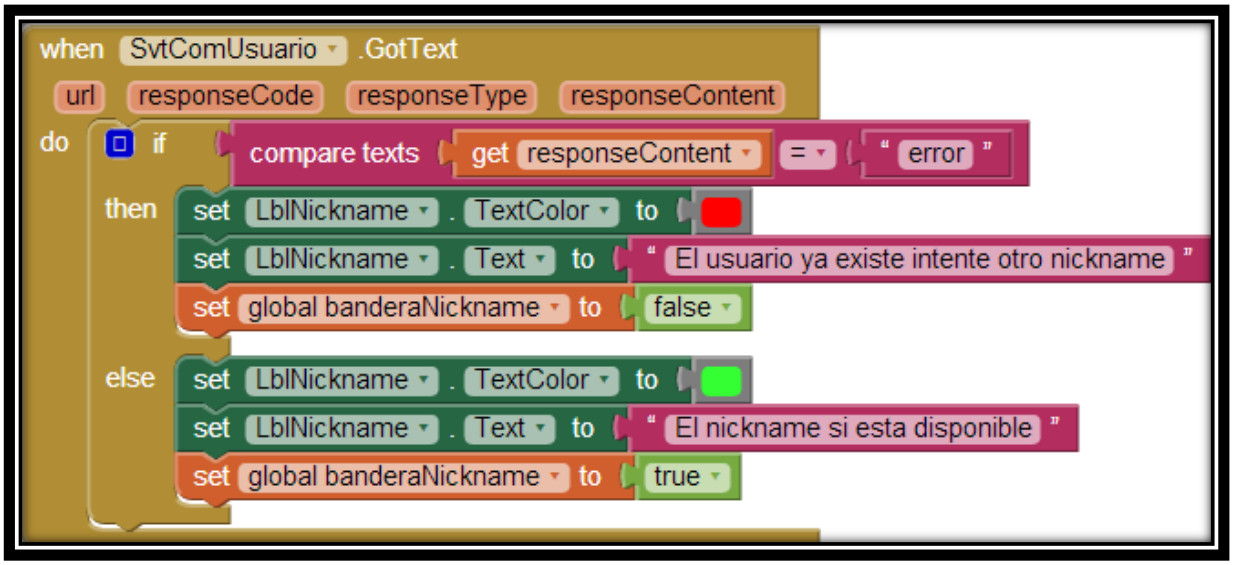
FIGURA 94. Bloques del evento LostFocus del componente TxtNickname.



Fuente: Autor.

El servlet ComUsuario verifica si el Nickname recibido ya ha sido previamente registrado, en cuyo caso devuelve un texto de error, con el que se mostrará en la etiqueta **LblNickname** el texto: "El usuario ya existe intente otro Nickname" con color rojo, con lo que la bandera Nickname será false, caso contrario la etiqueta dirá "El Nickname si está disponible" de color verde, y la variable bandera Nickname pasara a ser true. El bloque correspondiente se muestra en la FIGURA 95;

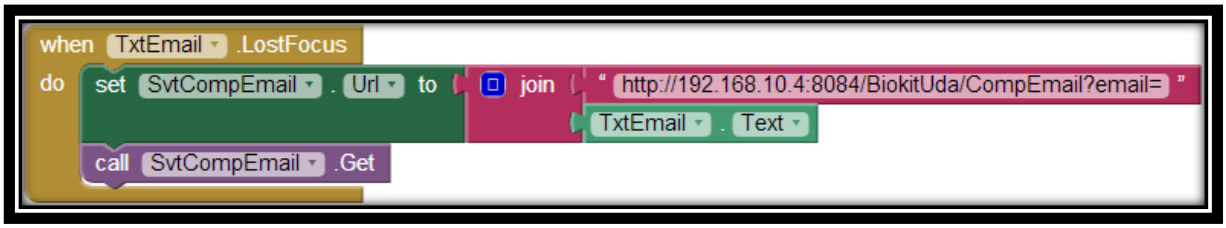
FIGURA 95. Bloque del evento GotText del componente web SvtComUsuario.



Fuente: Autor.

Similar situación ocurre cuando después de llenar el campo de TxtEmail se pierde el enfoque, correspondientemente se invocará al servlet CompEmail con el dato ingresado como se muestra en la FIGURA 96

FIGURA 96. Bloque del evento LostFocus del componente TxtEmail.

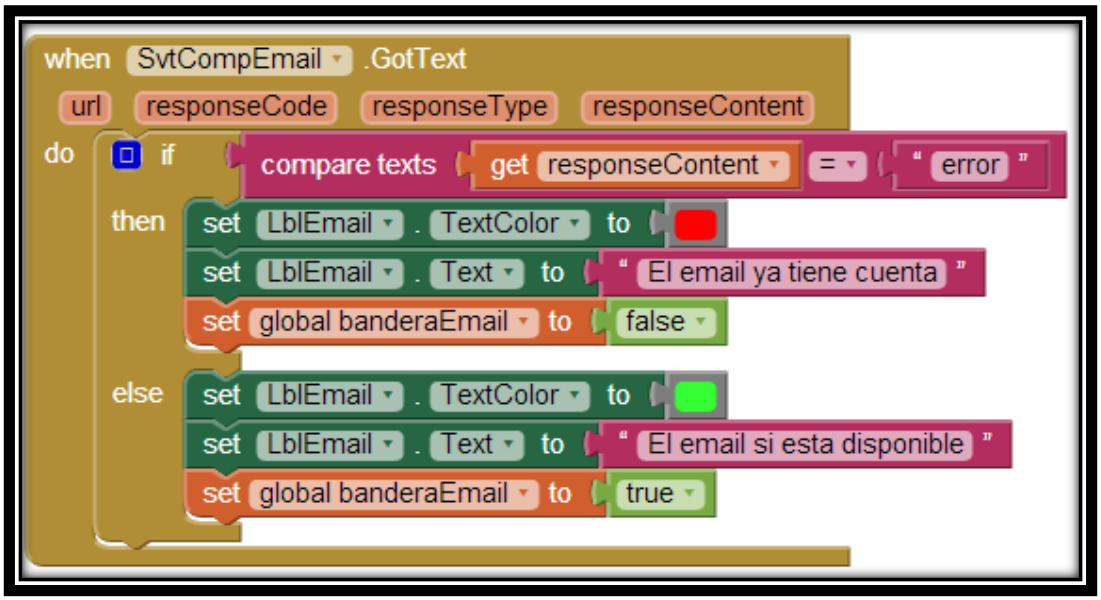


Fuente: Autor.

El servlet CompEmail se comporta de manera similar al ComUsuario devolviendo un error en el caso de que el email ya haya sido registrado previamente, con lo que en el

LblEmail se mostrará de color rojo “El email ya corresponde a una cuenta” la variable banderaEmail cambia a false, y de color verde si no había sido registrado, con la leyenda “El email si está disponible” y la variable banderaEmail sería true. Los bloques se ilustran en la FIGURA 97.

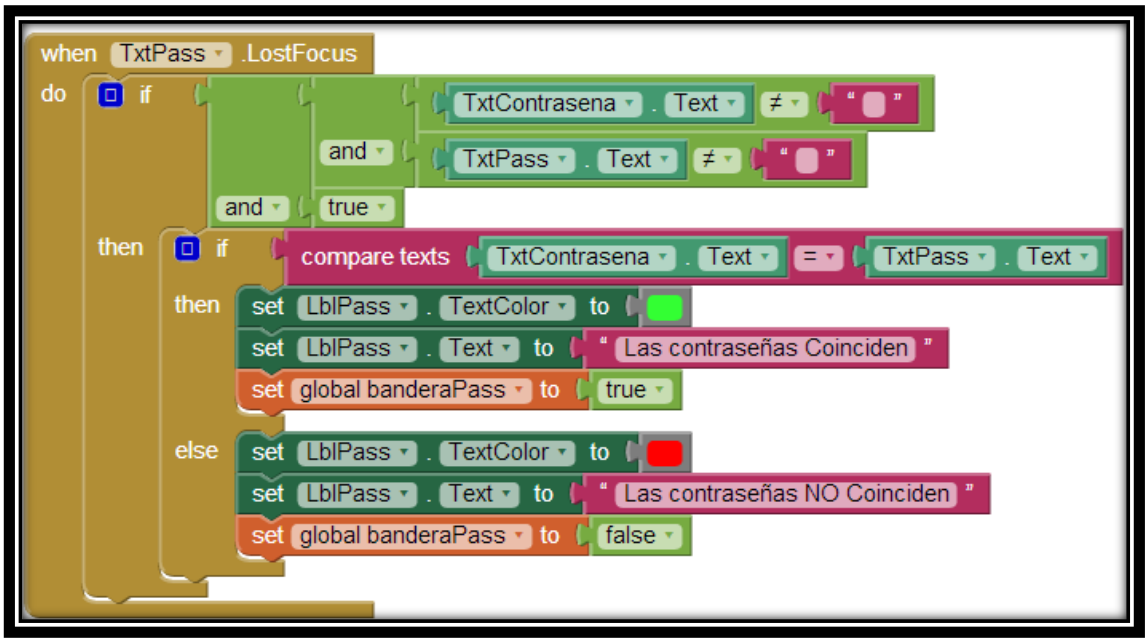
FIGURA 97. Bloque del evento GotText del componente web SvtCompEmail.



Fuente: Autor.

Una vez llenado el campo de TxtPass que corresponde a la comprobación de la contraseña se dispara el evento LostFocus, que comprueba en primera instancia que los campos no estén vacíos y que los datos ingresados en el campo TxtContrasena y TxtPass sean iguales, en cuyo caso la variable banderaPass pasa a ser true y caso contrario pasaría a ser false. Los bloques del evento se ilustran el la FIGURA 98.

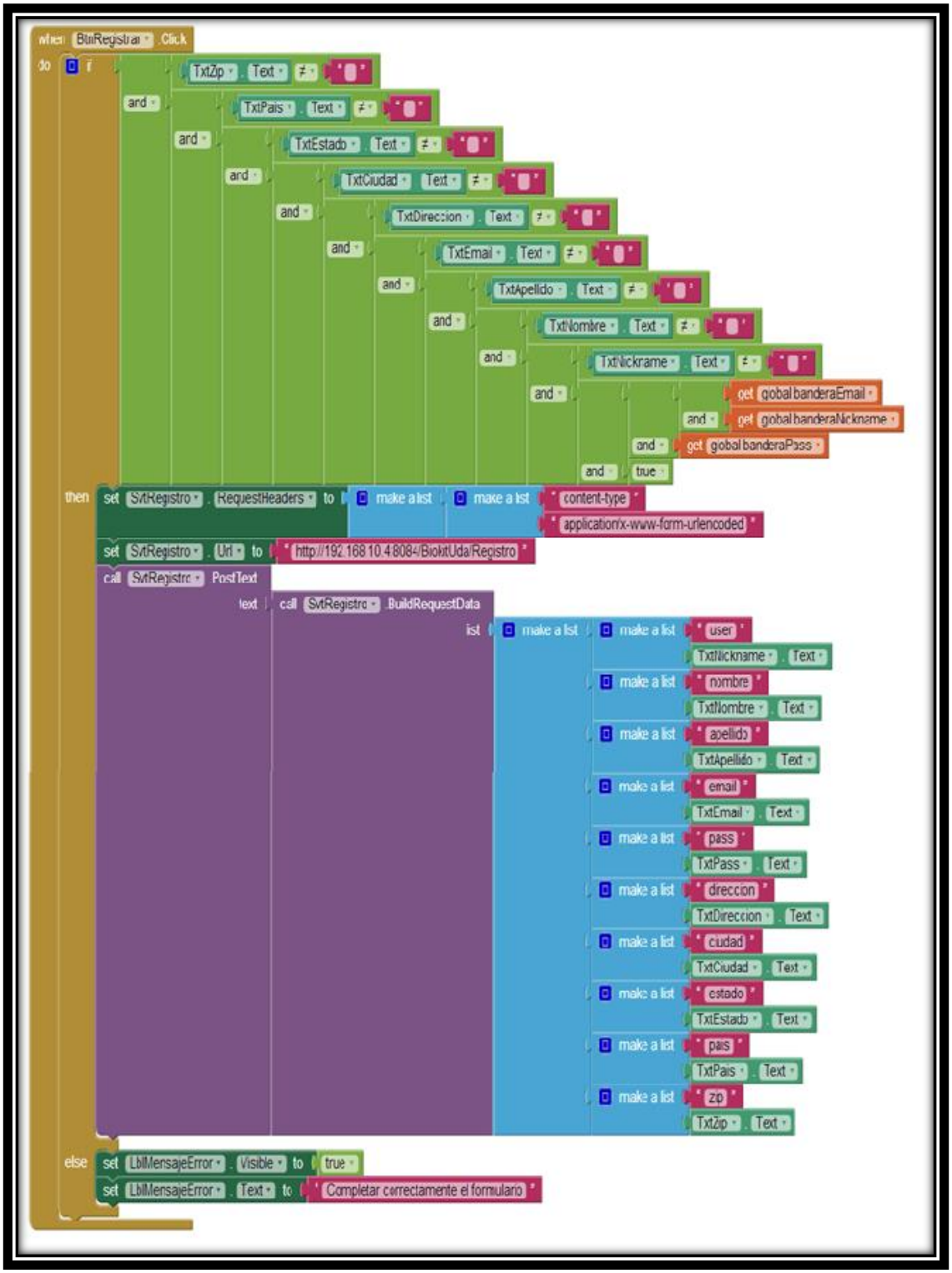
FIGURA 98. Bloque del evento LostFocus del componente TxtPass.



Fuente: Autor.

Al presionar el botón Registrar se comprueba que todos los campos estén llenos y todas las banderas son verdaderas (true), si estas condiciones se cumplen, se procede a invocar el servlet de Registro pasándole todos los datos ingresados para ser posteriormente grabados en la base de datos, caso contrario se visualizará una leyenda en el LblMensajeError con el texto “Completar correctamente el formulario” como se ilustra el bloque en la FIGURA 99.

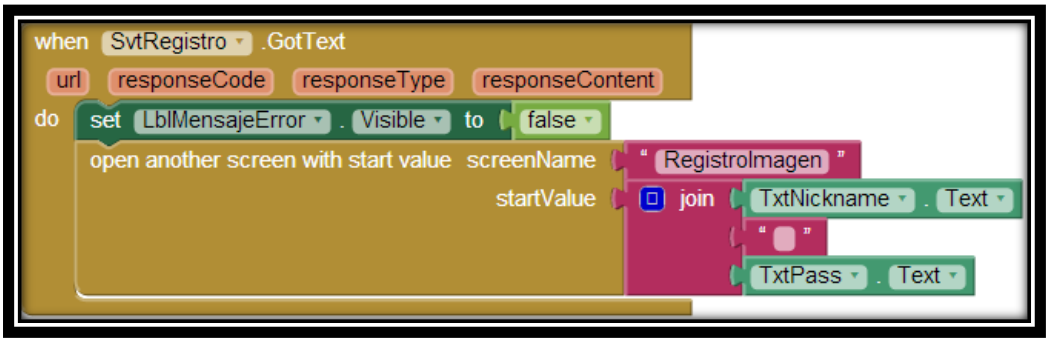
FIGURA 99. Bloque del evento Click del botón Registrar.



Fuente: Autor.

Si el registro fue exitosamente guardado se abrirá la pantalla de RegistroImagen con el Nickname y contraseña obtenidos como se muestra en la FIGURA 100.

FIGURA 100. Bloque del evento GotText del componente web SvtRegistro.



Fuente: Autor.

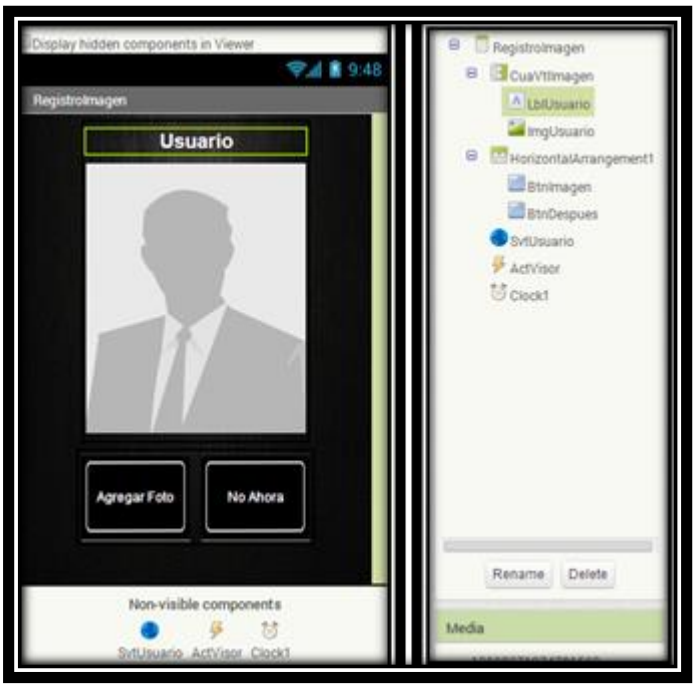
3.3.3 Pantalla RegistroImagen.

La pantalla de RegistroImagen estará encargada de abrir una página web para el registro de la fotografía de identificación de usuario, ya que por las limitaciones de App inventor no se la pude realizar directamente desde la aplicación.
Una vez registrada la fotografía se abrirá la pantalla de Menú.
Para efectuar la apertura de una página web se necesita el siguiente componente:

3.3.3.1 ActivityStarter.

Se encuentra en la sección de **Connectivity** dentro del diseño de App inventor, este componente es capaz de iniciar diferentes actividades, entre alguna de ellas la apertura de otras aplicaciones, la inicialización de la cámara, mostrar la ubicación en un mapa, etc. Existen algunas aplicaciones que pueden devolver ciertos valores de resultados.
Conociendo la función del ActivityStarter y de los componentes previamente citados, el diseño quedará como muestra en la FIGURA 101.

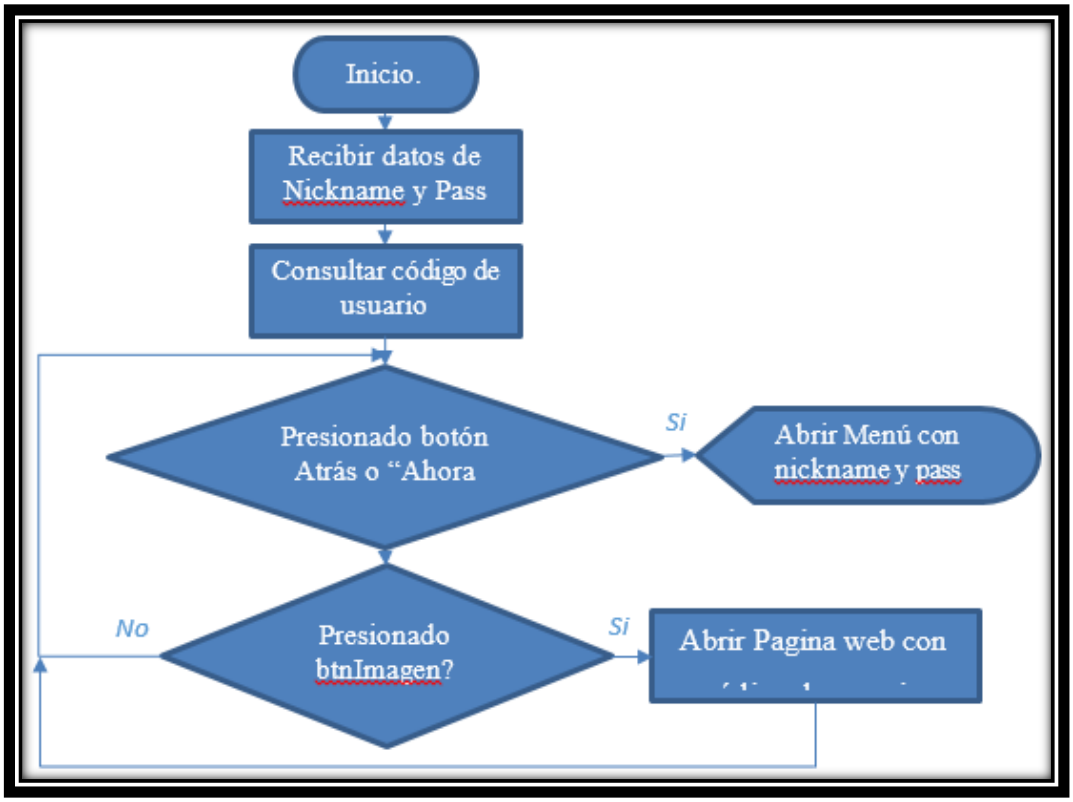
FIGURA 101. Diseño de la pantalla RegistroImagen.



Fuente: Autor.

En la parte de la programación se trata de registrar una imagen existente en el celular para identificar al usuario al presionar “Agregar Foto”, y de saltar el registro al presionar “No ahora” o el botón Atrás. Lo que obedecerá al diagrama de flujo de la FIGURA 102.

FIGURA 102. Diagrama de flujo de la pantalla RegistroImagen.

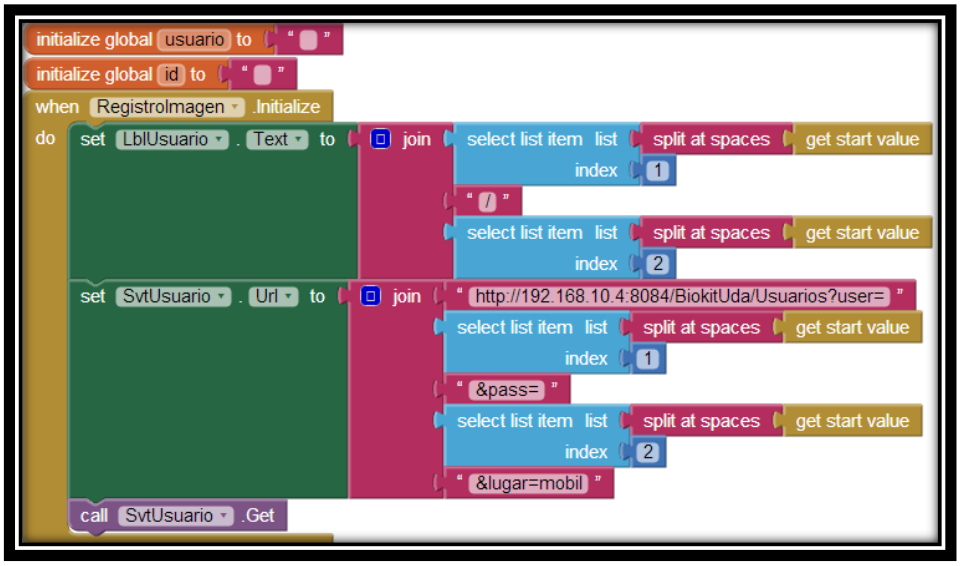


Fuente: Autor.

Una vez clara la función que debe tener la pantalla Registro imagen se procede a realizar la programación en bloques en el App inventor.

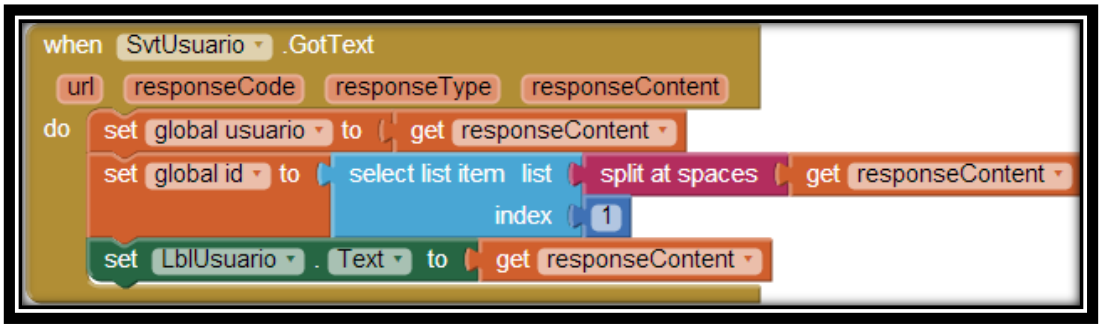
En primer lugar se debe inicializar las variables que recibimos de la pantalla anterior, y consultamos por medio del servlet Usuarios el código con el que fue registrado el usuario como se muestra en la FIGURA 103, este código servirá como nombre para el registro de la imagen. Los bloques se pueden observar en la FIGURA 104.

FIGURA 103. Bloque de inicialización de variables y pantalla.



Fuente: Autor.

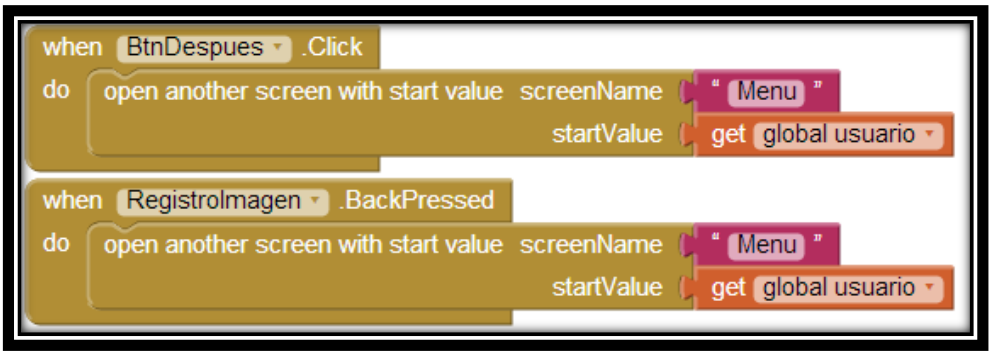
FIGURA 104. Bloque de evento GotText del componente web SvtUsuario.



Fuente: Autor.

Si se presionan los botones “atrás” o “No ahora” se procederá abrir la pantalla de Menú pasando los datos de Nickname y contraseña; como se ilustra en la FIGURA 105.

FIGURA 105. Bloques de los eventos click en los botes BtnDespues “Ahora No” y “Atrás”.



Fuente: Autor.

Finalmente si se presiona el botón “Agregar Foto” (BtnImagen) se inicializa la aplicación de explorador web, pasándole los atributos de la página web que queremos que sea visualizada y sus parámetros como se puede observar en la FIGURA 106.

FIGURA 106. Bloque de evento Click del BtnImagen.

Fuente: Autor.

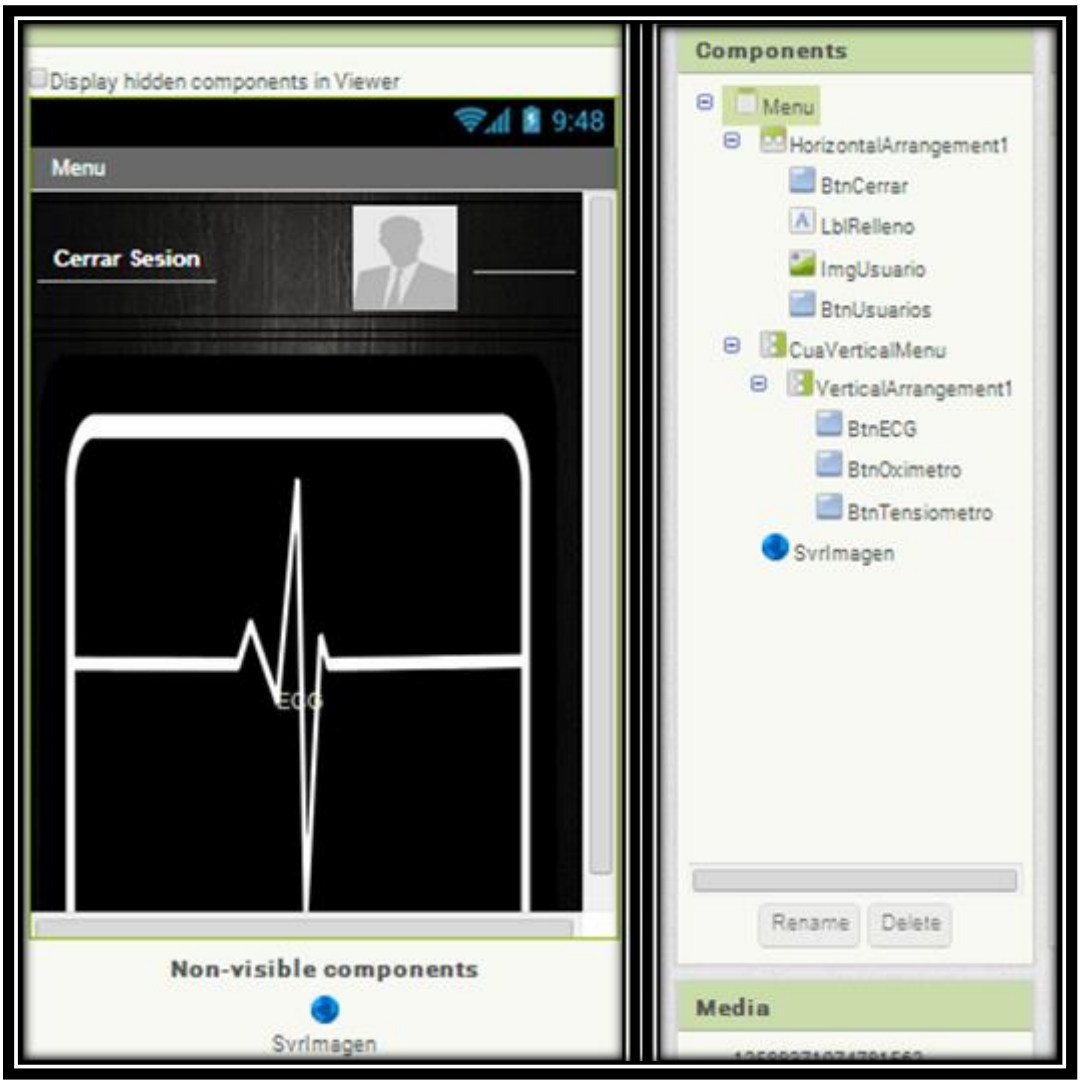
3.3.4 Pantalla de menú.

La pantalla Menú será capaz de:

- Presentar un menú con las diferentes opciones de las señales biológicas que se quiera adquirir.
- Mostar la foto y el nombre del usuario que inicio la sesión.
- Contener botón para cerrar la sesión actual y otro para modificar la información básica del usuario.

Para el diseño de la interfaz gráfica no se necesitan ningún elemento diferente a los descritos anteriormente por lo que el diseño en el App inventor quedara de la siguiente manera (FIGURA 107):

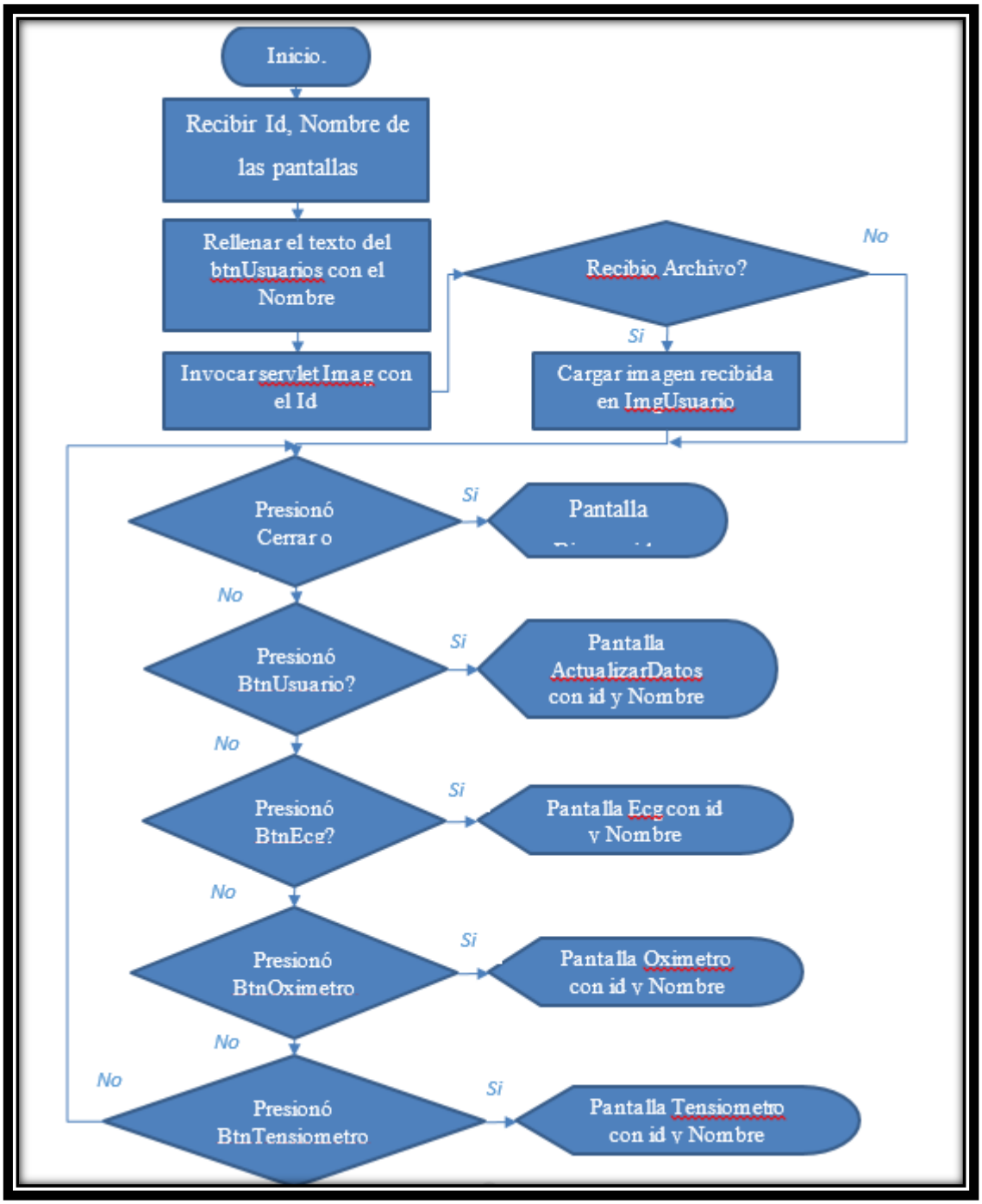
FIGURA 107. Diseño de la pantalla Menú.



Fuente: Autor.

La programación de los bloques de App inventor obedece al diagrama de flujo de la FIGURA 108

FIGURA 108. Diagrama de flujo de la pantalla Menú.

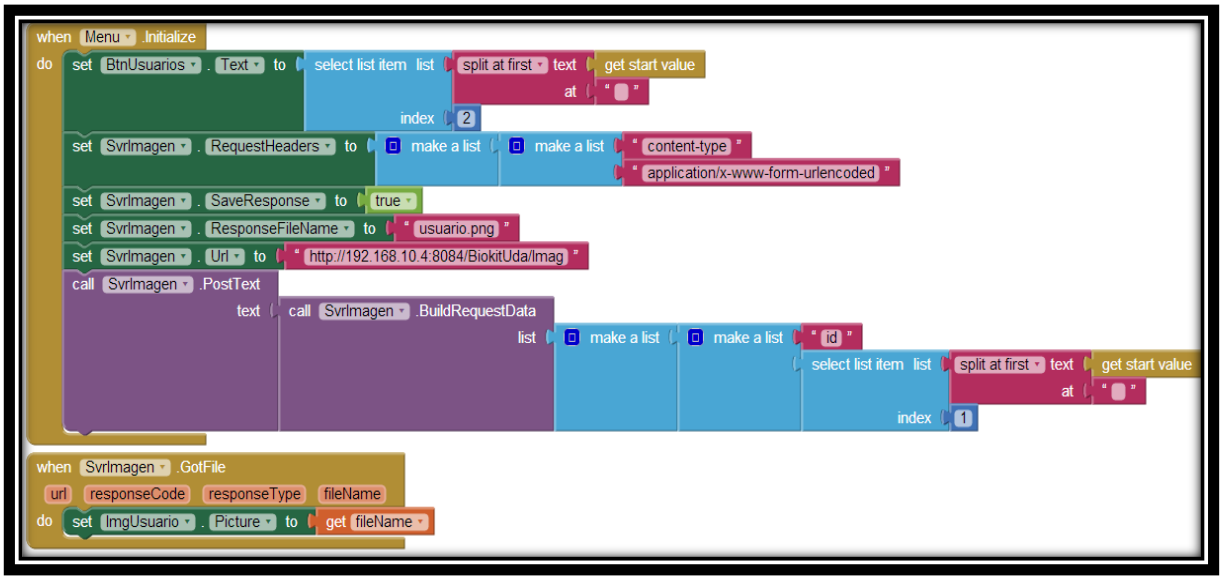


Fuente: Autor.

Para llevar a cabo el diagrama de flujo de la figura; en primera instancia debemos recibir los datos con los que fue llamada la presente pantalla y separarlos con la herramienta **Split** ya que vienen de la forma “código Nombre Apellido”. Se necesita

separar el código de usuario porque es un parámetro del servlet CargaImag, éste servlet nos devolverá la imagen correspondiente al código de usuario, dicha imagen se almacenara en la memoria del teléfono con el nombre "usuario.png" y se mostrará en el componente ImgUsuario, el bloque de este proceso se ilustra en la FIGURA 109.

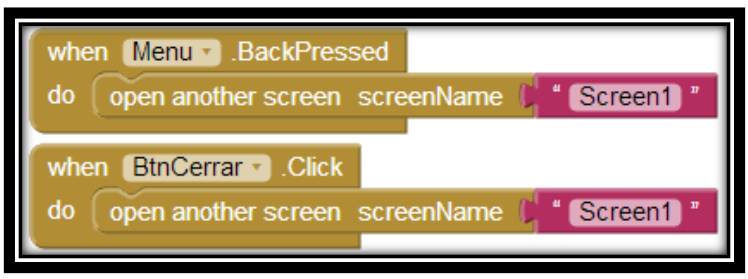
FIGURA 109. Bloque de Inicialización de la pantalla Menú.



Fuente: Autor.

En caso de ser presionado el botón "Cerrar Sesión" o "Atrás" se dará paso nuevamente a la pantalla de bienvenida o inicio (Screen1), los bloques se pueden observar en la FIGURA 110

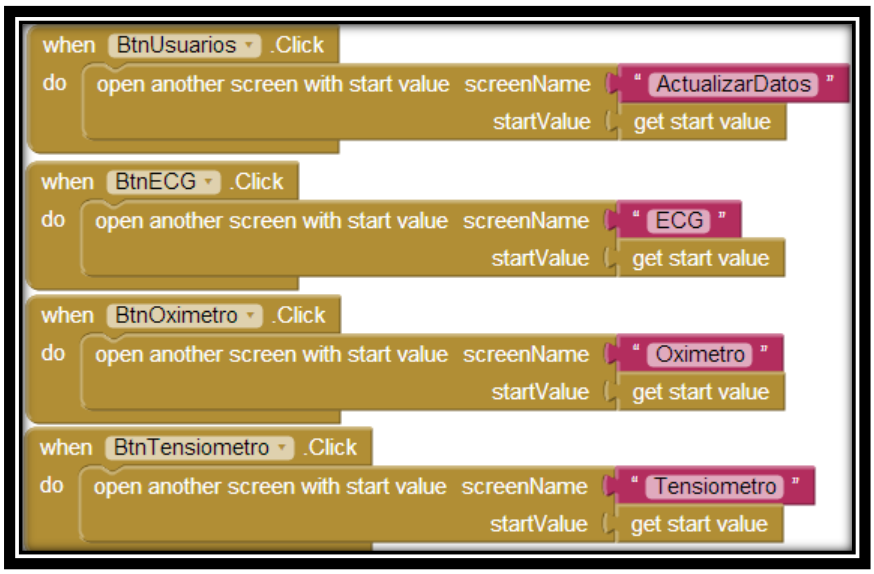
FIGURA 110. Bloques del evento Click de los botones Atrás y Cerrar.



Fuente: Autor.

De manera similar, los diferentes botones restantes nos mostraran las pantallas del Electrocardiograma, Oxímetro de pulso y Tensiómetro, llamándolos a estos con la misma información que ha sido invocada la pantalla menú. Además, el botón Usuarios nos dará la posibilidad de abrir una pantalla llamada ActualizarDatos, en la que se puede modificar la información básica del usuario como también su foto. En la FIGURA 111 se puede ilustrar lo explicado.

FIGURA 111. Bloques del evento Click de los componentes BtnUsuarios, BtnEcg, BtnOximetro, BtnTensiometro.



Fuente: Autor.

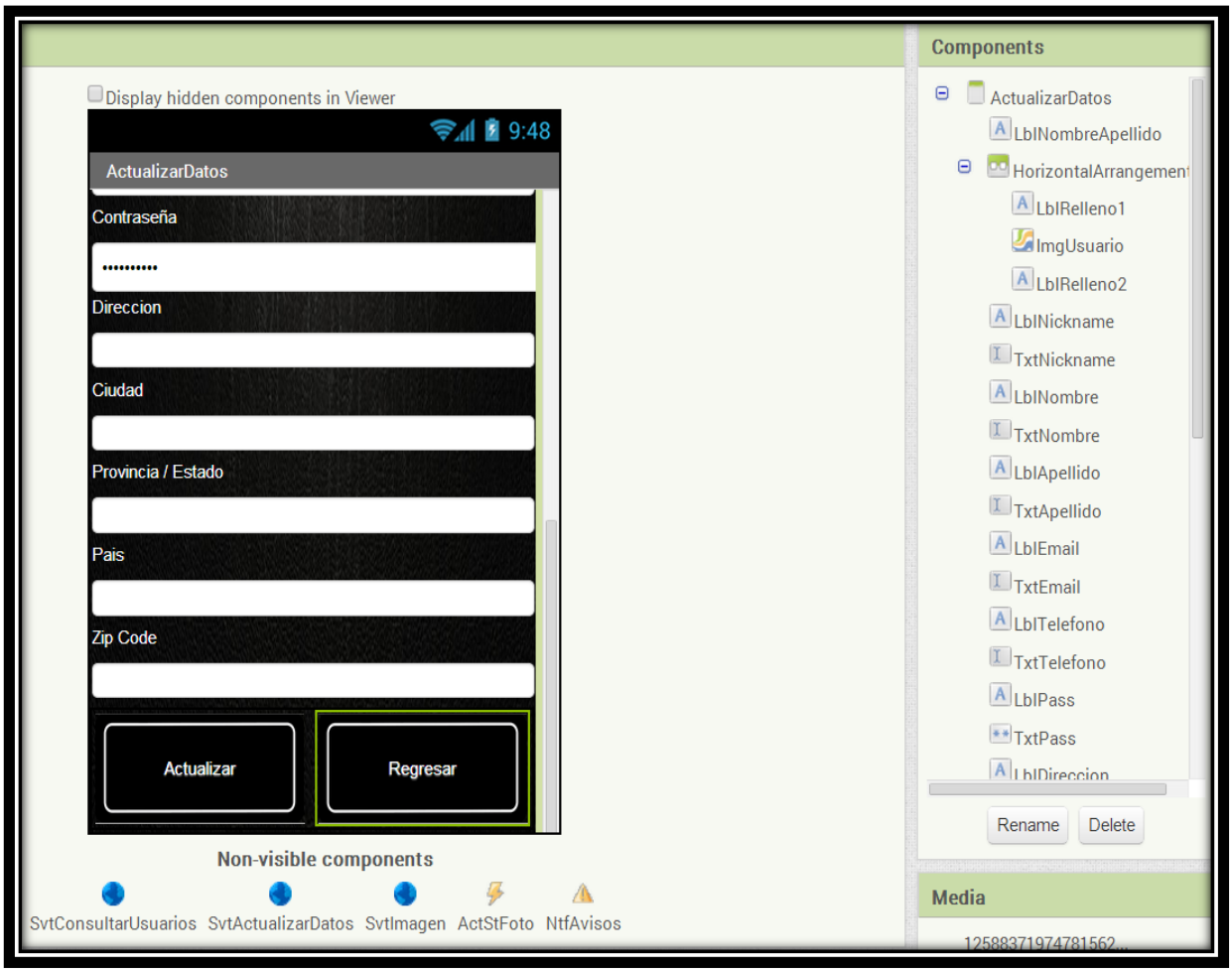
3.3.5 Pantalla ActualizarDatos.

La pantalla para la actualización de los datos personales del usuario cumplirá con los siguientes objetivos:

- Mostrar los datos actuales en las diferentes cajas de texto.
- Visualizar la fotografía actual del usuario, y ser capaz de registrar una fotografía nueva a elección del usuario.
- Un botón para la actualización de datos de usuario, a su vez, la aplicación pedirá la clave actual para guardar cambios.
- Un botón para regresar al menú principal.

No se necesita ningún componente nuevo para el diseño de la pantalla por lo que quedará como se muestra en la FIGURA 112:

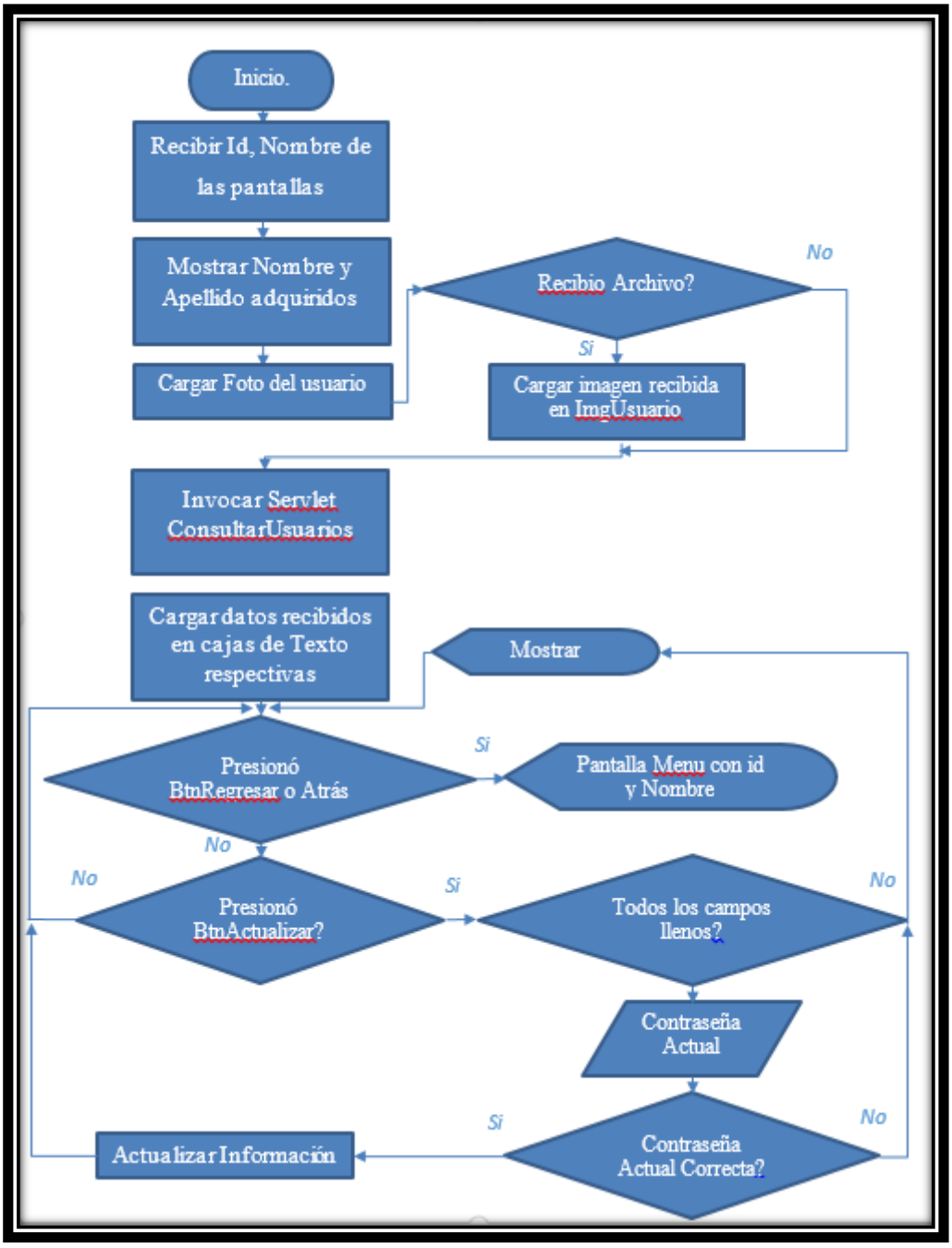
FIGURA 112. Diseño de la pantalla ActualizarDatos.



Fuente: Autor.

Para cumplir con los objetivos de la pantalla deberá cumplir el esquema del diagrama de flujo de la FIGURA 113:

FIGURA 113. Diagrama de flujo de la pantalla ActualizarDatos.



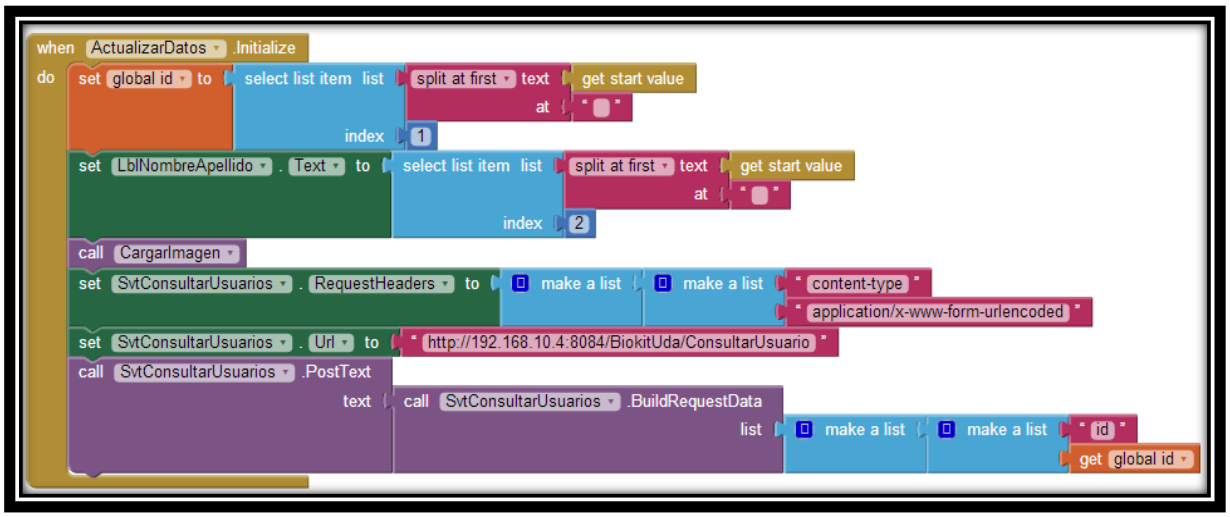
Fuente: Autor.

Para cumplir con el diagrama de flujo de la figura; en el evento “initialize” de la pantalla (FIGURA 114), se debe primero adquirir los datos pasados por la pantalla invocadora como los son el id, nombre y apellido del usuario y guardarlos en variables globales de la pantalla, realizado esto, se invocará al proceso llamado CargarImagen

(FIGURA 115) encargado de llamar al servlet Imag con el parámetro Id, que devuelve la foto del usuario para ser mostrada en el componente ImgUsuario.

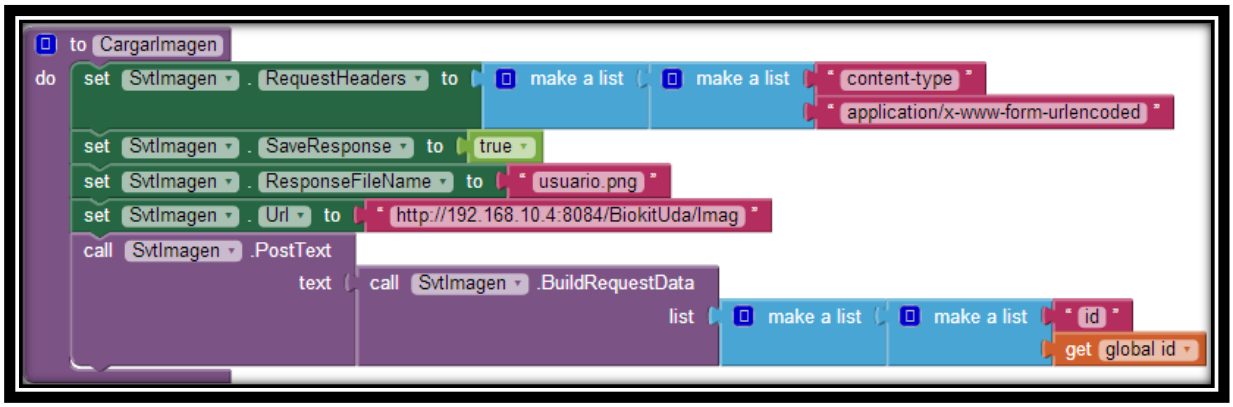
El proceso continúa invocando al servlet ConsultarUsuario con el parámetro id, que devolverá todos los datos del usuario, que serán cargados en las respectivas cajas de texto (FIGURA 116).

FIGURA 114. Evento Initialize de la pantalla ActualizarDatos.



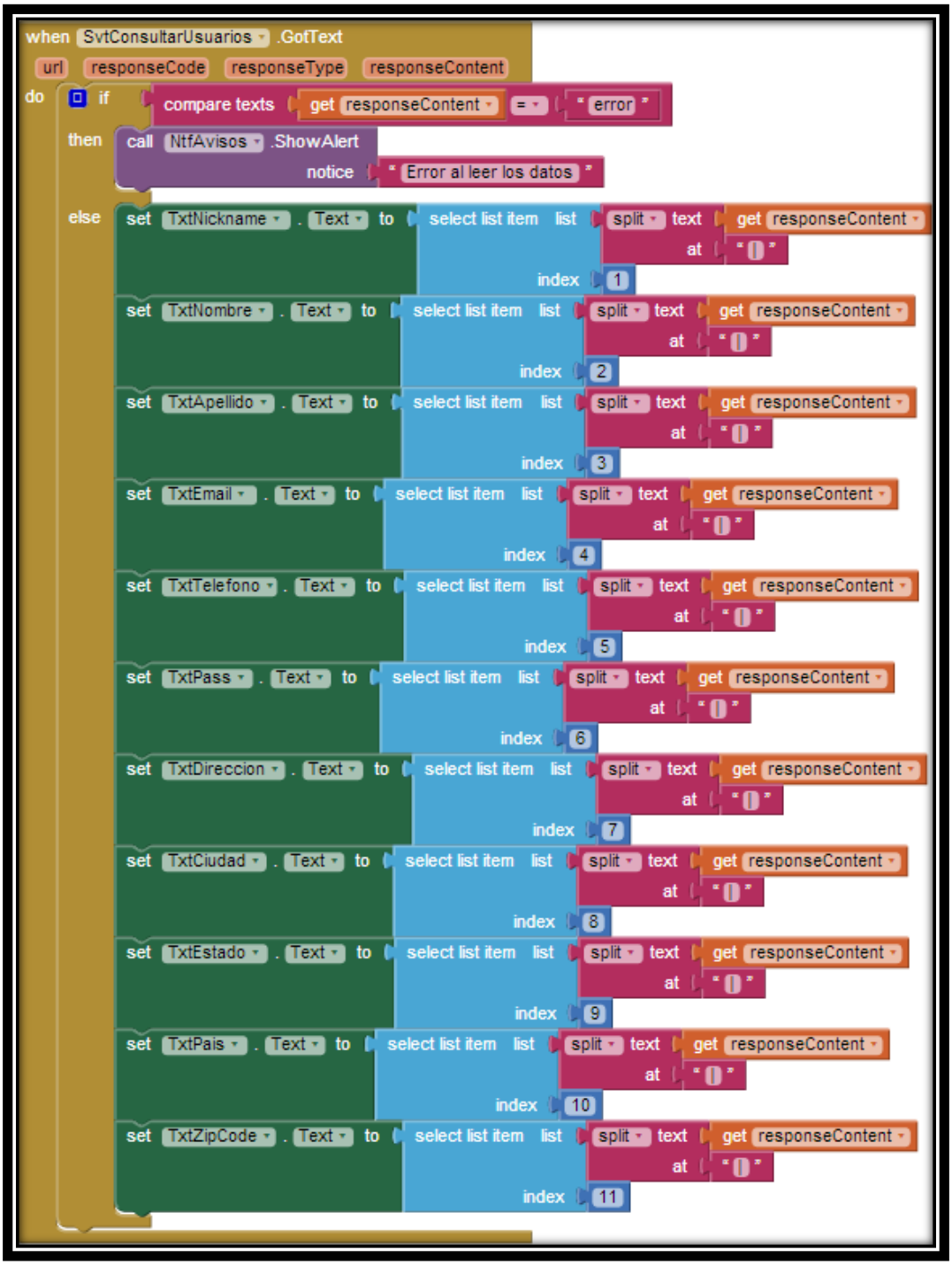
Fuente: Autor.

FIGURA 115. Proceso CargarImagen.



Fuente: Autor.

FIGURA 116. Evento GotText del Componente SvtConsultarUsuario.

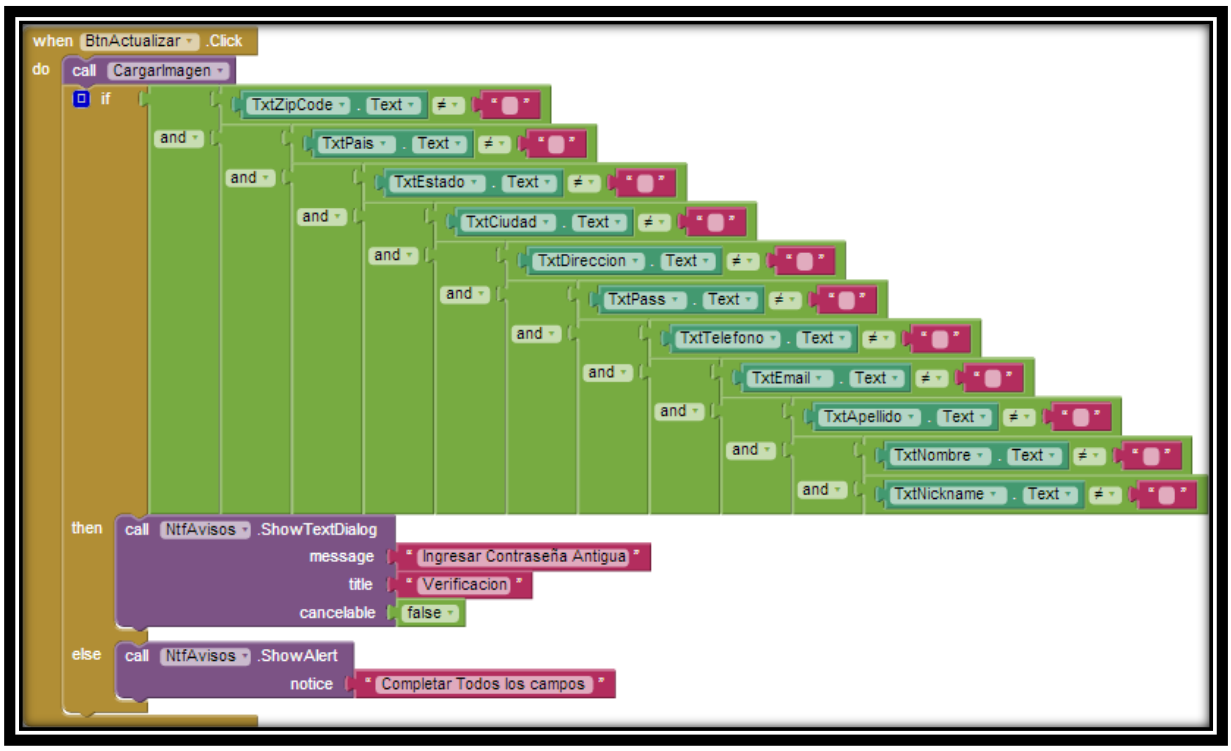


Fuente: Autor.

Si se presiona el botón “Actualizar” (FIGURA 117) se actualizará la imagen del usuario con el proceso “CargarImagen” y preguntará si todos los campos de información están llenos, de estarlos, se requerirá la contraseña actual del usuario, esta información se comprobará en el evento AfterTextInput de la notificación “NtfAvisos” llamando al proceso “Actualizar” (FIGURA 118).

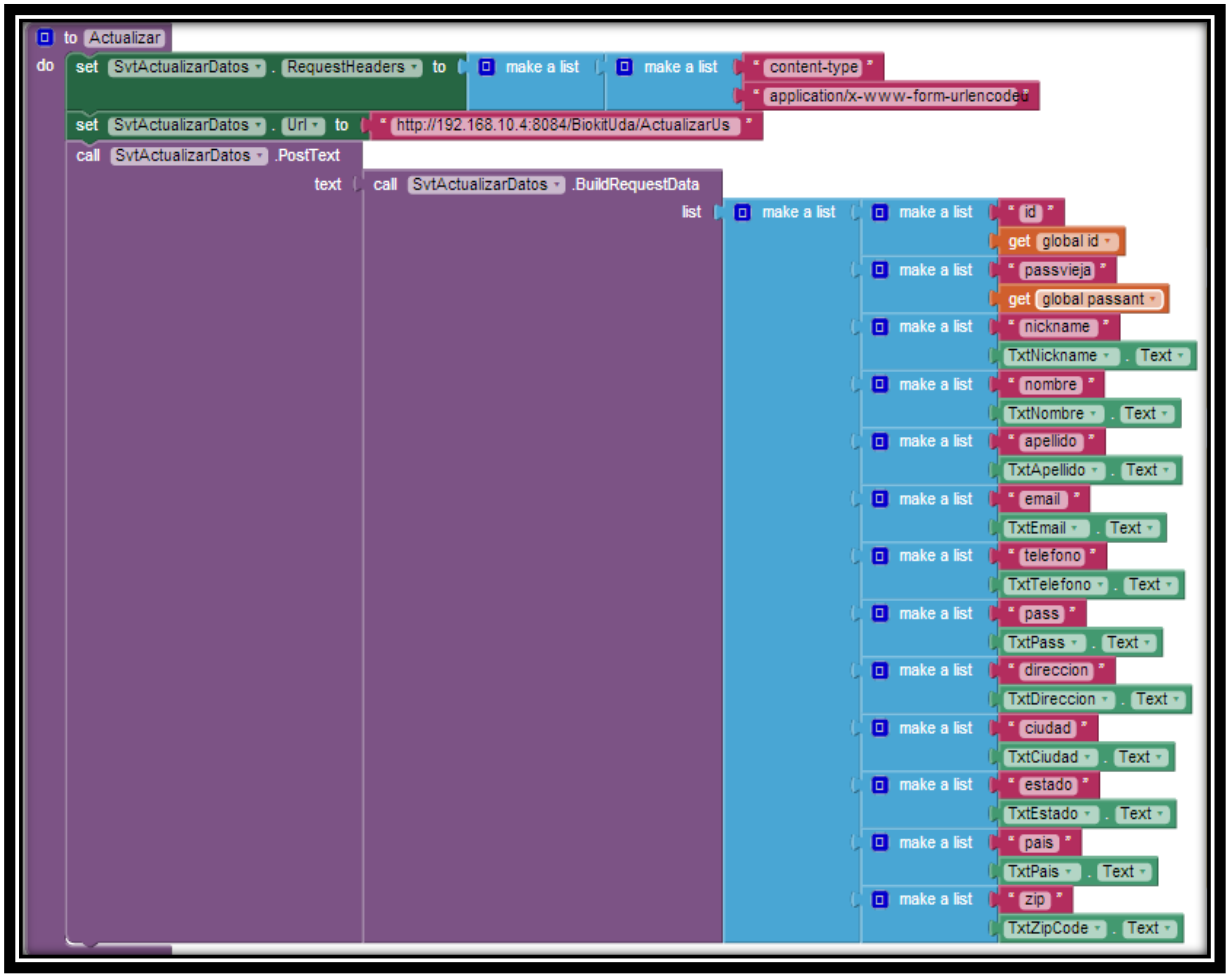
El procedimiento “Actualizar” se encarga de invocar al servlet ActualizarUs pasando los parámetros de Id, passvieja, y todos los campos de las cajas de texto (FIGURA 119), si la contraseña en el parámetro passvieja es igual a la contraseña actual del usuario entonces los datos ingresados remplazaran a los anteriores, caso contrario se visualizará un mensaje de error por contraseñas no correspondientes.

FIGURA 117. Evento Click del componente BtnActualizar.



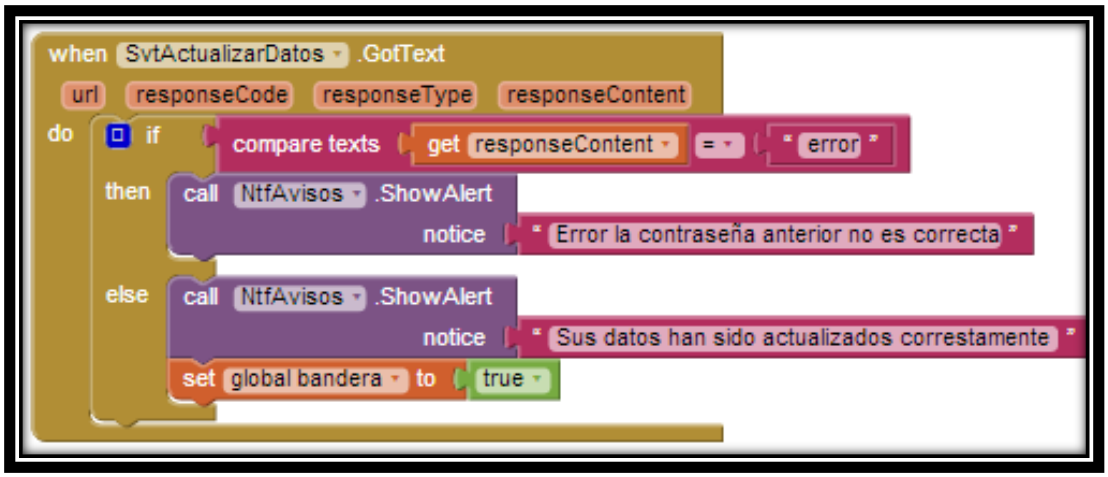
Fuente: Autor.

FIGURA 118. Proceso Actualizar.



Fuente: Autor.

FIGURA 119. Evento GotText del componente SvtActualizarDatos.



Fuente: Autor.

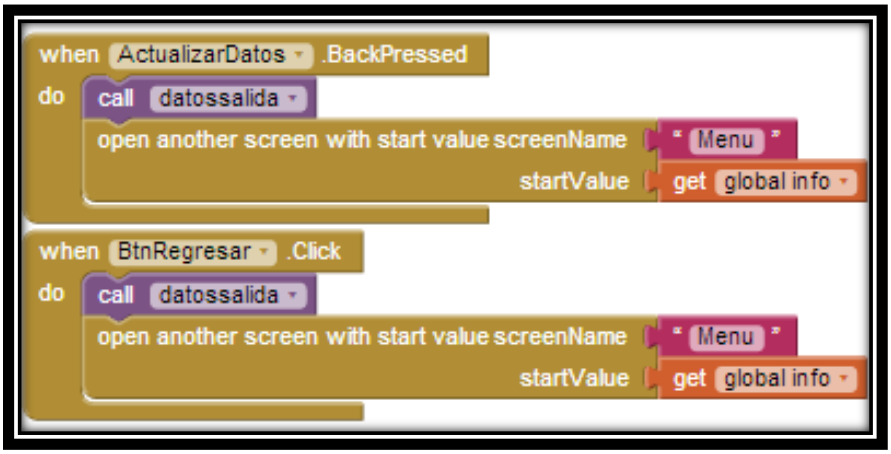
Al presionar sobre la imagen de usuario (ImgUsuario) llamaremos a un navegador web por medio del ActivityStarter "ActStFoto" que cargará una página web con el parámetro id del usuario, esta página permitirá cambiar la foto de Usuario (FIGURA 120).

FIGURA 120. Evento TouchDown del componente ImgUsuario.

Fuente: Autor.

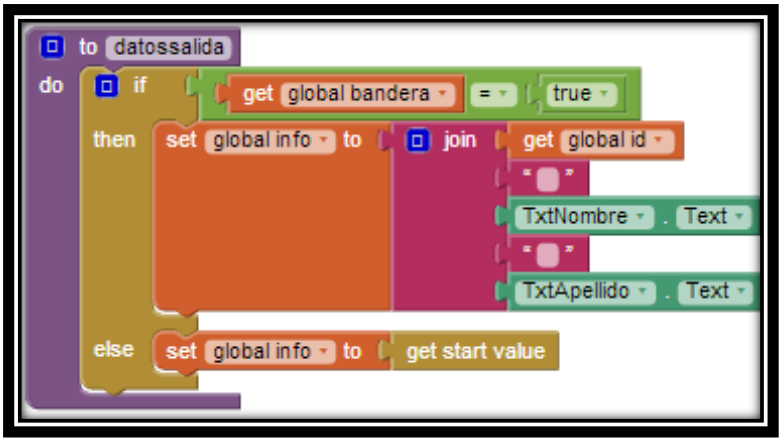
Si se quiere regresar a la pantalla Menú basta con presionar el botón regresar (BtnRegresar) o el botón Atrás del sistema, estos llamaran a un proceso llamado "datossalida" (FIGURA 121), que se encarga de comprobar si los datos de las cajas fueron modificados, para pasar como parámetro a la pantalla menú, de esta forma se mantendrá actualizado el nombre del usuario en todas las pantallas (FIGURA 122).

FIGURA 121. Evento de presión de los componentes BtnRegresar y Atrás del sistema.



Fuente: Autor.

FIGURA 122. Proceso datossalida.



Fuente: Autor.

3.3.6 Pantalla ECG:

La pantalla del Electrocardiograma cumple con las siguientes características:

- Establecer conexión por medio de Bluetooth con el dispositivo electrónico de toma de señales biológicas.
- Enviar comandos para iniciar y detener la adquisición de datos de las tres derivaciones propias del ECG.
- Generar una gráfica en tiempo real de los datos adquiridos.

- Una opción para suprimir la información recolectada hasta el momento.
- Guardar los adquiridos en la base de datos del servidor.
- Mostrar el electrocardiograma definitivo al final de la adquisición.
- Una opción que permite regresar al menú principal

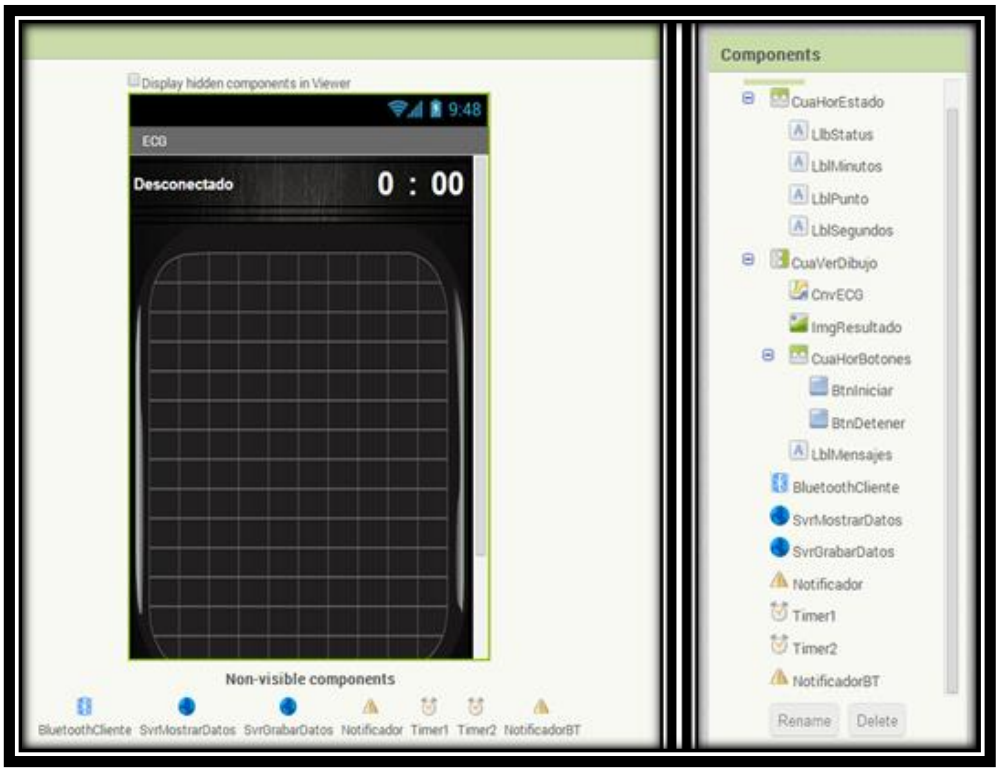
Para cumplir con los objetivos propuestos, es necesario implementar el componente de temporizador en App Inventor.

3.3.6.1 Clock:

Se encuentra en la sección de **User Interface** dentro del diseño de App inventor, este componente cumple con las funciones que se le designa cada cierto tiempo definido por el usuario arbitrariamente o por programación.

Conocido lo necesario para el correcto cumplimiento de las tareas de la pantalla, el diseño gráfico se muestra en la figura (FIGURA 123).

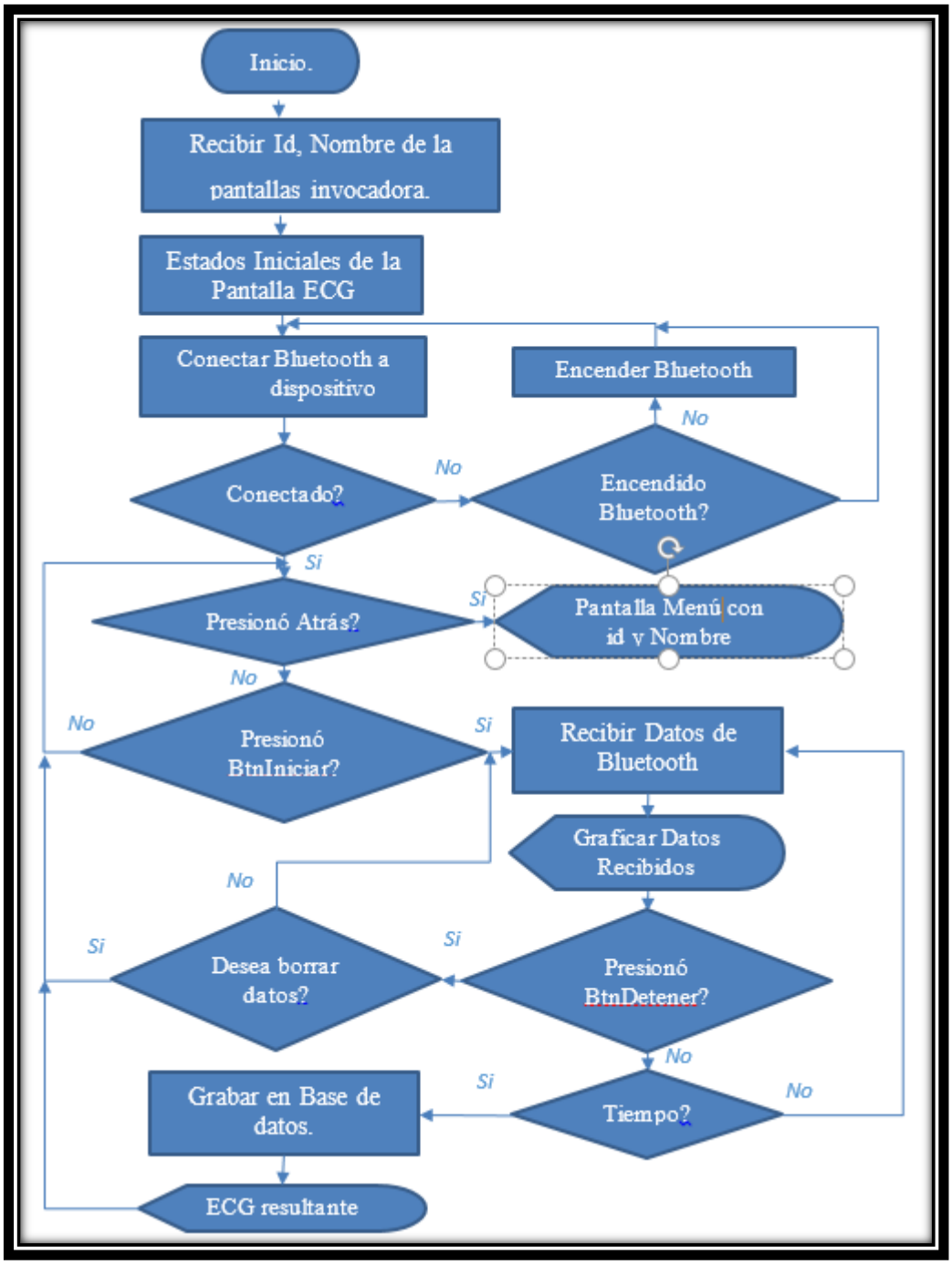
FIGURA 123. Diseño de la Pantalla ECG.



Fuente: Autor.

El programa en el diagrama de bloques de App Inventor obedece al diagrama de flujo ilustrado en la FIGURA 124:

FIGURA 124. Diagrama de Flujo de la Pantalla ECG.



Fuente: Autor

Para cumplir íntegramente con el diagrama de flujo propuesto en la sección de bloques de App Inventor, en primer lugar, se debe recibir los datos con los que fue invocada la pantalla y grabarla en variables globales, a continuación, se deben inicializar los

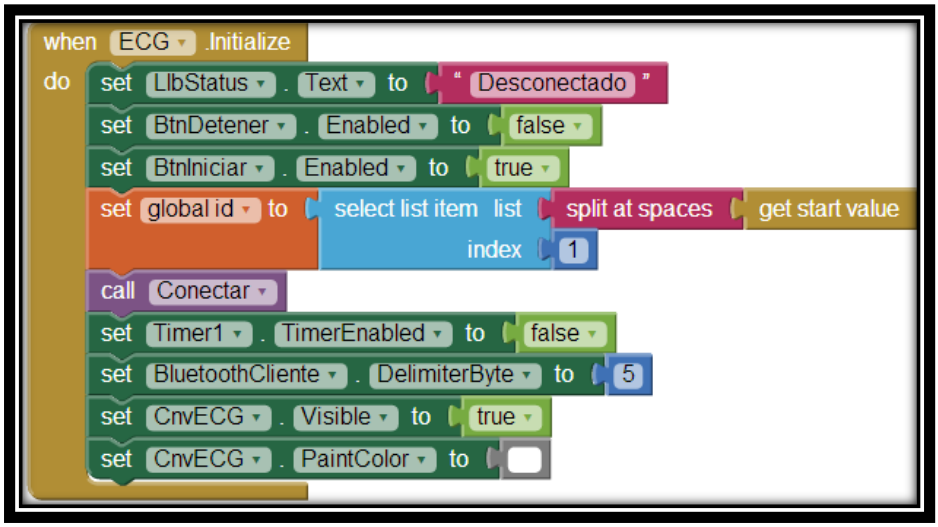
estados de arranque de programa como lo son banderas y variables, se debe deshabilitar el Timer y llamar a un procedimiento llamado **Conectar.**

El procedimiento **Conectar** de la (FIGURA 125): está encargado de inicializar el Bluetooth y establecer una conexión con el dispositivo encargado de receptar las señales, en caso de no establecerse la conexión se pedirá por medio de un ActivityStarter el encendido del bluetooth e para intentar una vez el procedimiento de **Conectar (**FIGURA 126 y FIGURA 127**).**

FIGURA 125. Inicializar variables en Pantalla ECG.

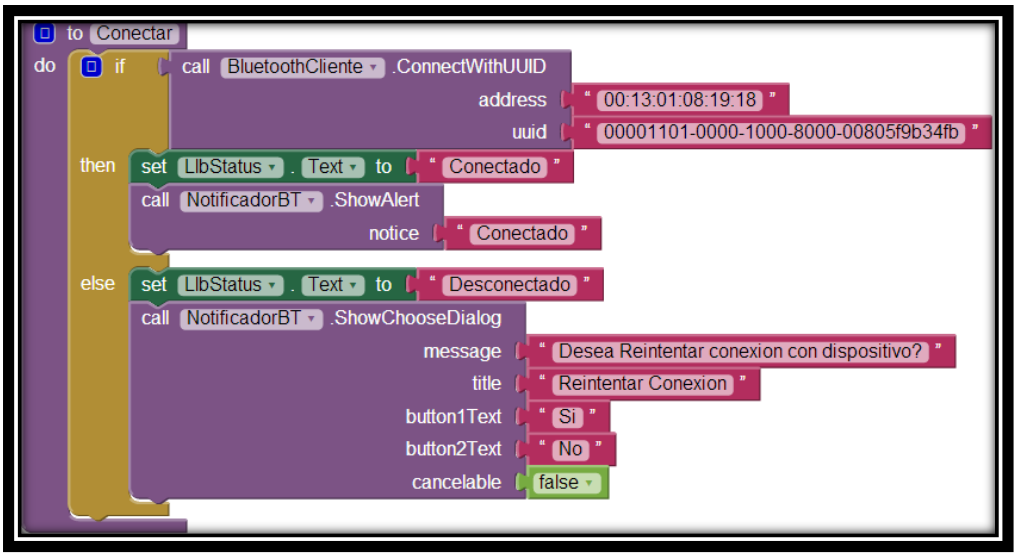
Fuente: Autor.

FIGURA 126.Evento Initialize de la Pantalla ECG.



Fuente: Autor.

FIGURA 127.Procedimiento Conectar.



Fuente: Autor.

Si se presiona el Botón “iniciar” (BtnIniciar) se enviará una señal al dispositivo para que inicie la recolección de datos provenientes del módulo del electrocardiograma, simultáneamente, se habilita el Timer y el botón de “detener” (BtnDetener), lógicamente, se deshabilita el botón de Iniciar (BtnIniciar) como se muestra en la figura.

El Timer está configurado para generar un evento cada milisegundo ya que de esta manera se puede mejorar la calidad de dibujo de las 3 derivaciones del electrocardiograma, dicho evento verifica si el tiempo de adquisición de datos del electrocardiograma ha culminado, de no ser este el caso, pregunta si existen datos en el buffer de recepción del Bluetooth, si cumple con esta última condición recoge los datos y los almacena en una variable global, donde van concatenados todos los datos recibidos.

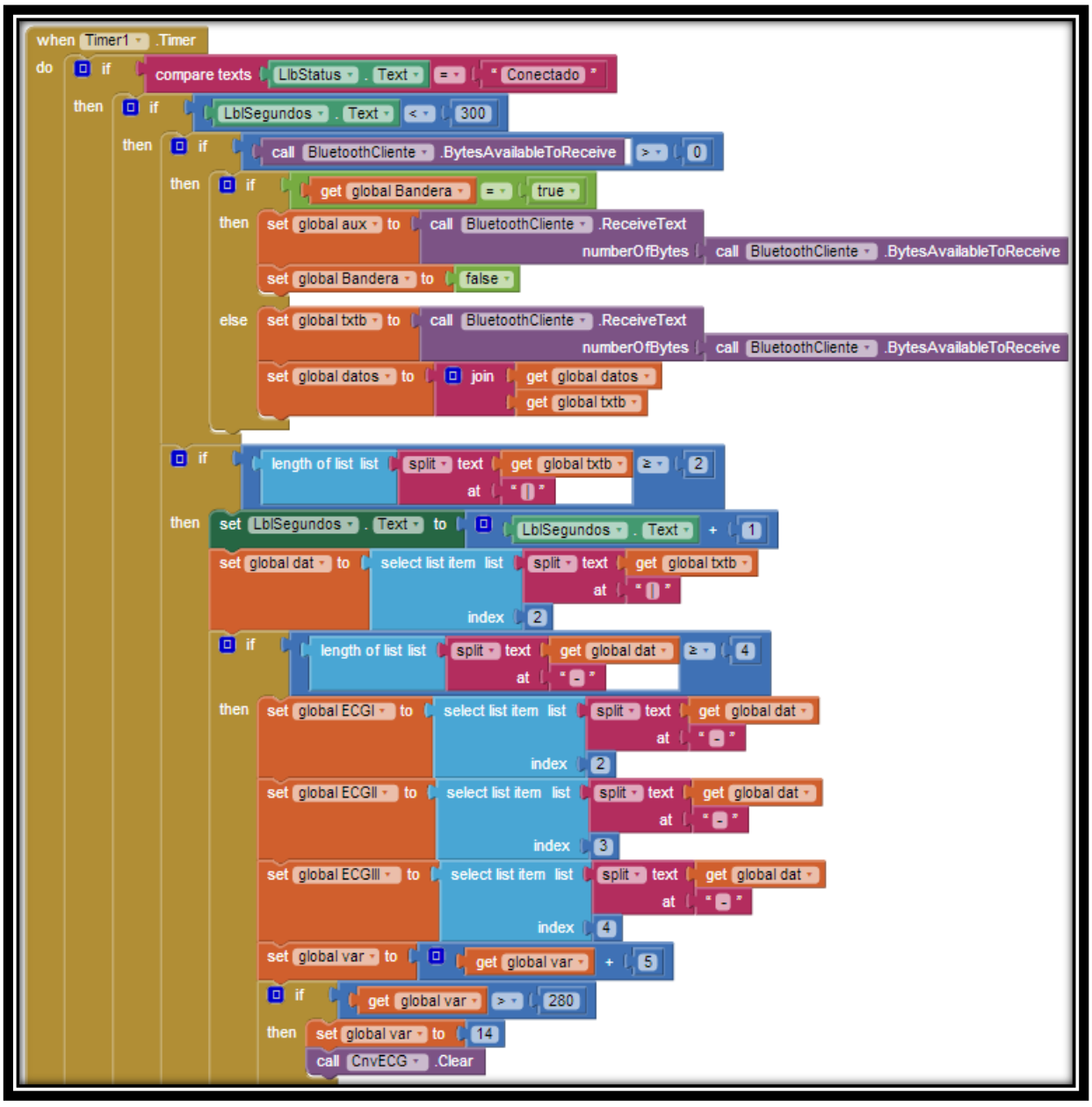
Estos datos almacenados son primeramente separados ya que se reciben de la siguiente manera: "|ECGI-ECGII-ECGIII|" los caracteres "|" separan los datos recolectados en el mismo tiempo, y los caracteres "-" separan la primera, segunda y tercera derivación del electrocardiograma.

Separados los datos se grafican uno a uno en el componente "CnvECG".

Si el tiempo transcurrido ha finalizado, la información recolectada se almacenará en la base de datos por medio del servlet **GrabarECG** pasándole los parámetros de **id** de usuario y **datos** recolectados, también se llamara al servlet **MostrarECGSimple** encargado de devolver el resultado final de la información tratada.

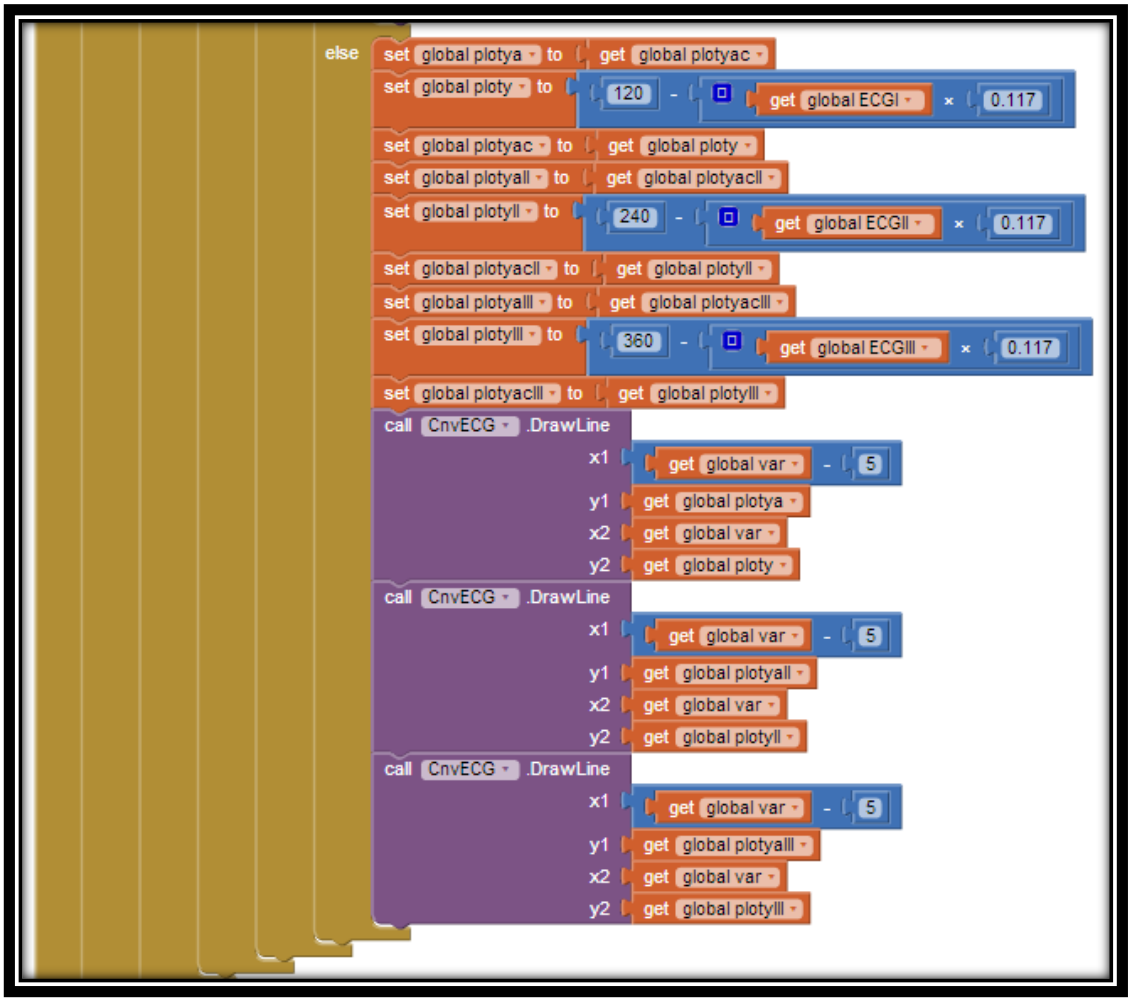
Todos los procesos anteriormente citados se ilustran en la FIGURA 128 y FIGURA 129 y FIGURA 130:

FIGURA 128.Evento Timer del Timer Parte 1.



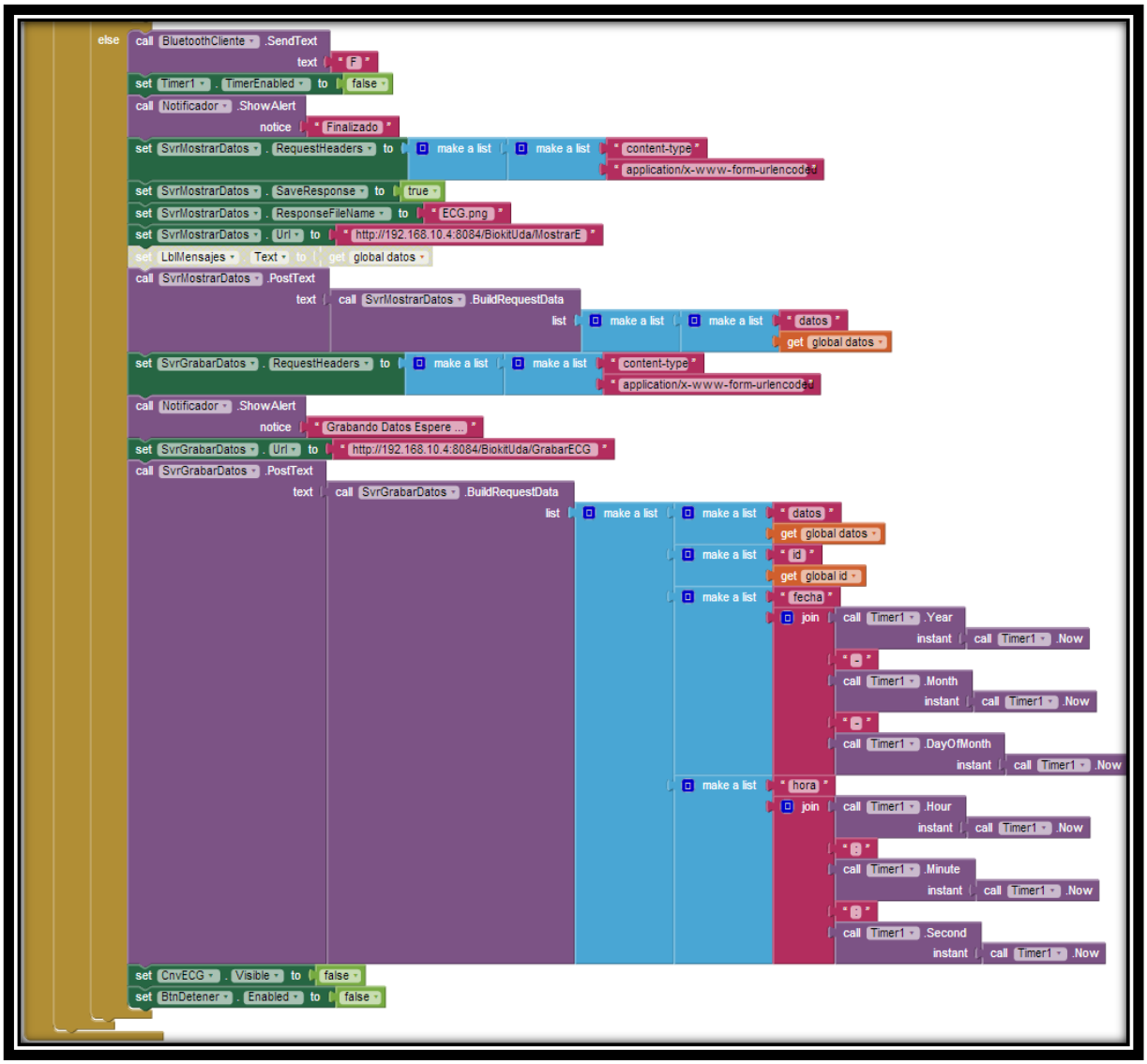
Fuente: Autor.

FIGURA 129. Evento Timer del Timer Parte 2.



Fuente: Autor.

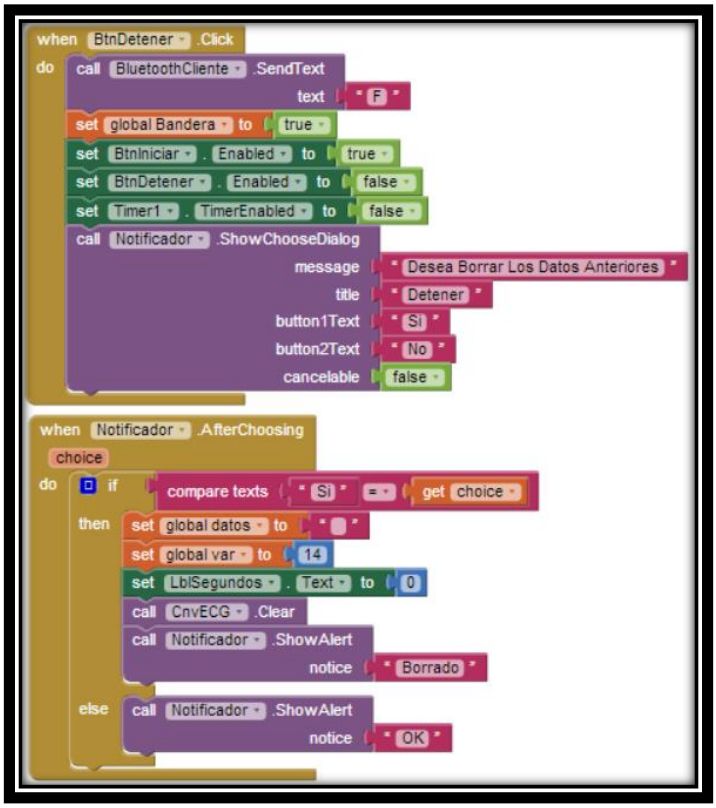
FIGURA 130.Evento Timer del Timer Parte 3.



Fuente: Autor.

Mientras el Timer está habilitado, también se habilita el Botón Detener (**BtnDetener**), el cual, si es presionado, se encarga de enviar una señal al módulo de ECG para que deje de emitir datos y luego pregunta si se quiere conservar los datos recolectados hasta el momento o borrarlos, con el objetivo de tomar nuevamente la información caso contrario continuará la toma de datos en curso (FIGURA 131).

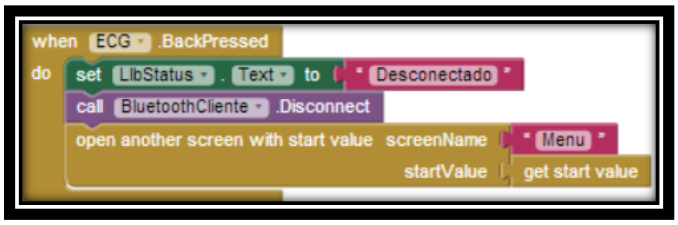
FIGURA 131. Evento Click del componente BtnDetener y su respectivo notificador.



Fuente: Autor.

Como todas las pantallas al presionarse el Botón “Atrás” del sistema regresará al menú principal con la única diferencia de que finalizará la conexión Bluetooth (FIGURA 132).

FIGURA 132. Evento BackPressed del botón Atrás del sistema.



Fuente: Autor.

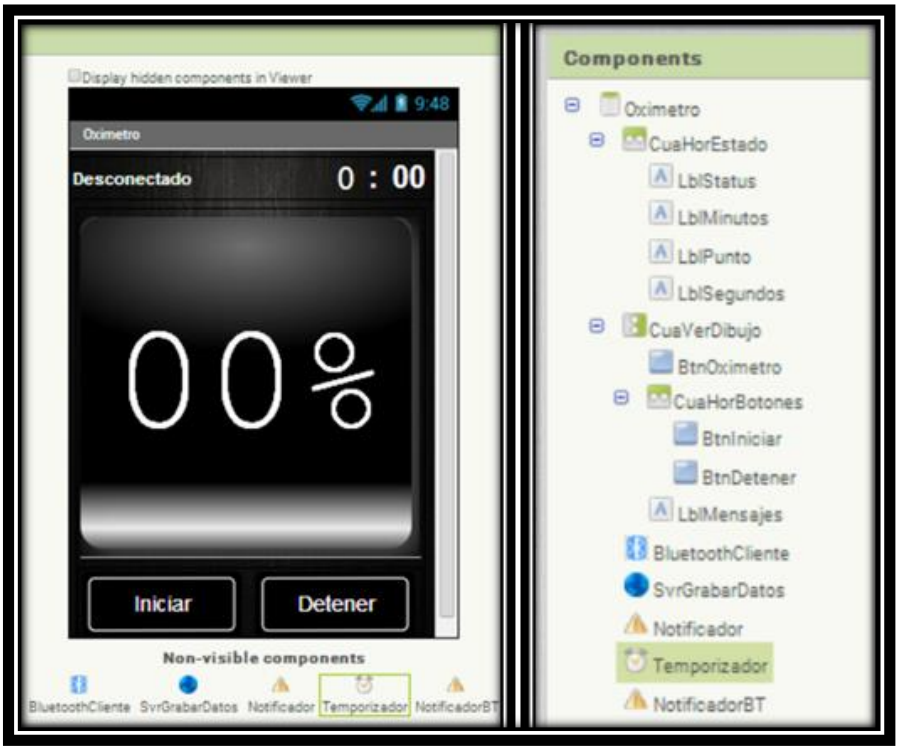
3.3.7 Pantalla Oxímetro.

La pantalla de Oxímetro cumple con las siguientes características

- Establecer conexión por medio de Bluetooth con el dispositivo electrónico de toma de señales biológicas.
- Enviar comandos para iniciar y detener la adquisición del porcentaje de oxígeno en la sangre.
- Mostrar en pantalla el porcentaje del oxígeno en la sangre.
- Una opción para suprimir la información recolectada hasta ese momento.
- Guardar los adquiridos en la base de datos del servidor.
- Una opción que permite regresar al menú principal

La pantalla Oxímetro es muy similar a la de electrocardiograma con la diferencia de que los datos serán mostrados en una Etiqueta y no en componente canvas. El diseño de la pantalla se muestra en la FIGURA 133.

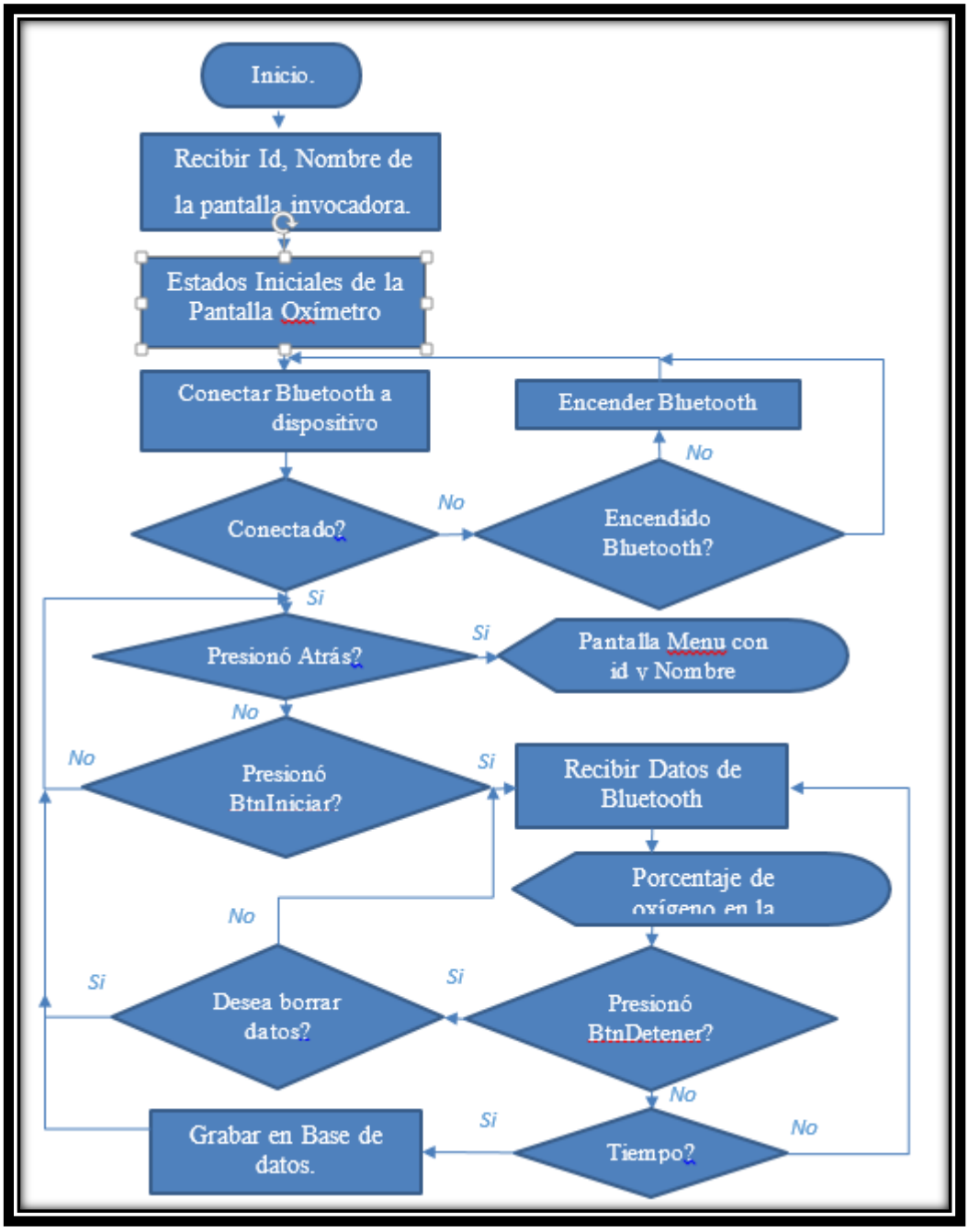
FIGURA 133. Diseño de la Pantalla Oxímetro.



Fuente: Autor.

El programa en el diagrama de bloques de App Inventor obedece al diagrama de flujo ilustrado en la FIGURA 134:

FIGURA 134. Diagrama de Flujo de la Pantalla Oxímetro.

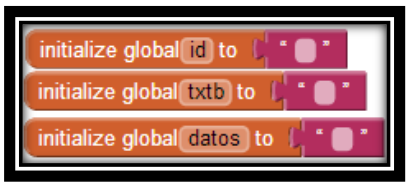


Fuente: Autor.

Para cumplir íntegramente con el diagrama de flujo propuesto en la sección de bloques de App Inventor, en primer lugar se debe recibir los datos con los que fue invocada la pantalla y grabarla en variables globales, luego se deben inicializar los estados de arranque de programa como lo son botones, etiquetas y variables, se debe deshabilitar el Timer y llamar a un procedimiento llamado **Conectar.**

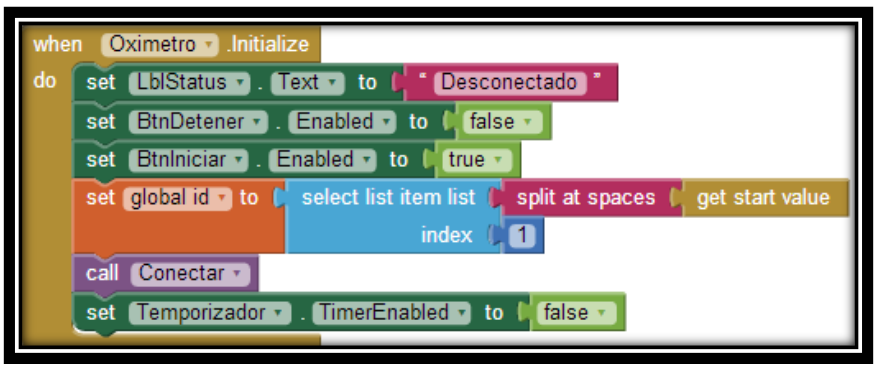
El procedimiento **Conectar** de la figura: es el mismo de la pantalla **ECG** está encargado de inicializar el Bluetooth (FIGURA 135 FIGURA 136) y establecer una conexión con el dispositivo encargado de receptar las señales, en caso de no establecerse la conexión se pedirá por medio de un ActivityStarter el encendido del bluetooth e para intentar una vez el procedimiento de **Conectar (**FIGURA 137).

FIGURA 135. Inicializar variables en Pantalla Oxímetro.



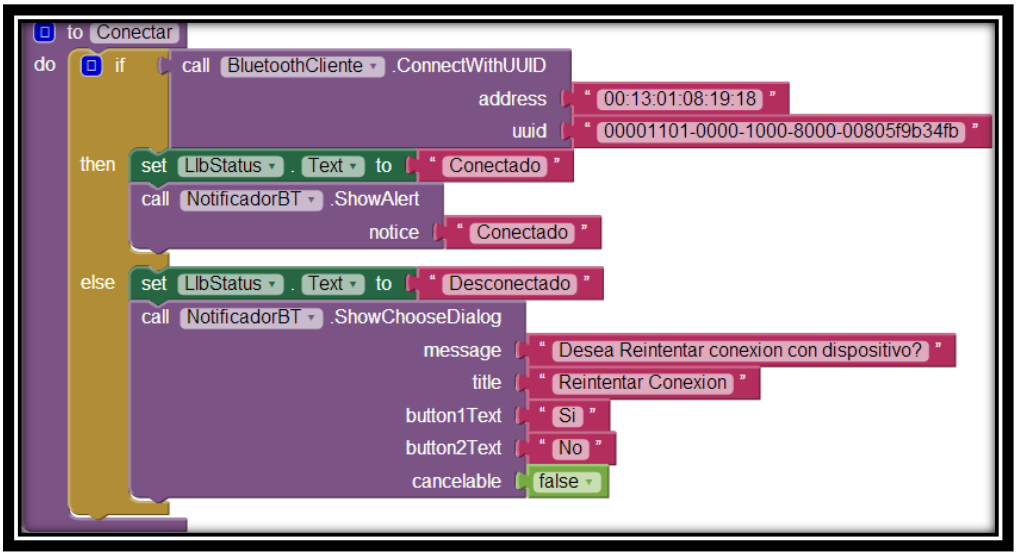
Fuente: Autor.

FIGURA 136. Evento Initialize de la Pantalla Oxímetro.



Fuente: Autor.

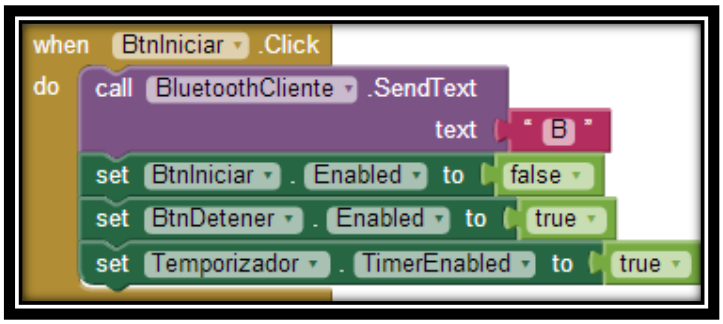
FIGURA 137. Procedimiento Conectar.



Fuente: Autor.

Si se presiona el Botón “iniciar” (BtnIniciar) enviará una señal al dispositivo para que inicie la recolección de datos provenientes del módulo de electrocardiograma, además, se habilita el Timer y el botón de detener (BtnDetener) y se deshabilita el botón de Iniciar (BtnIniciar) como se muestra en la FIGURA 138

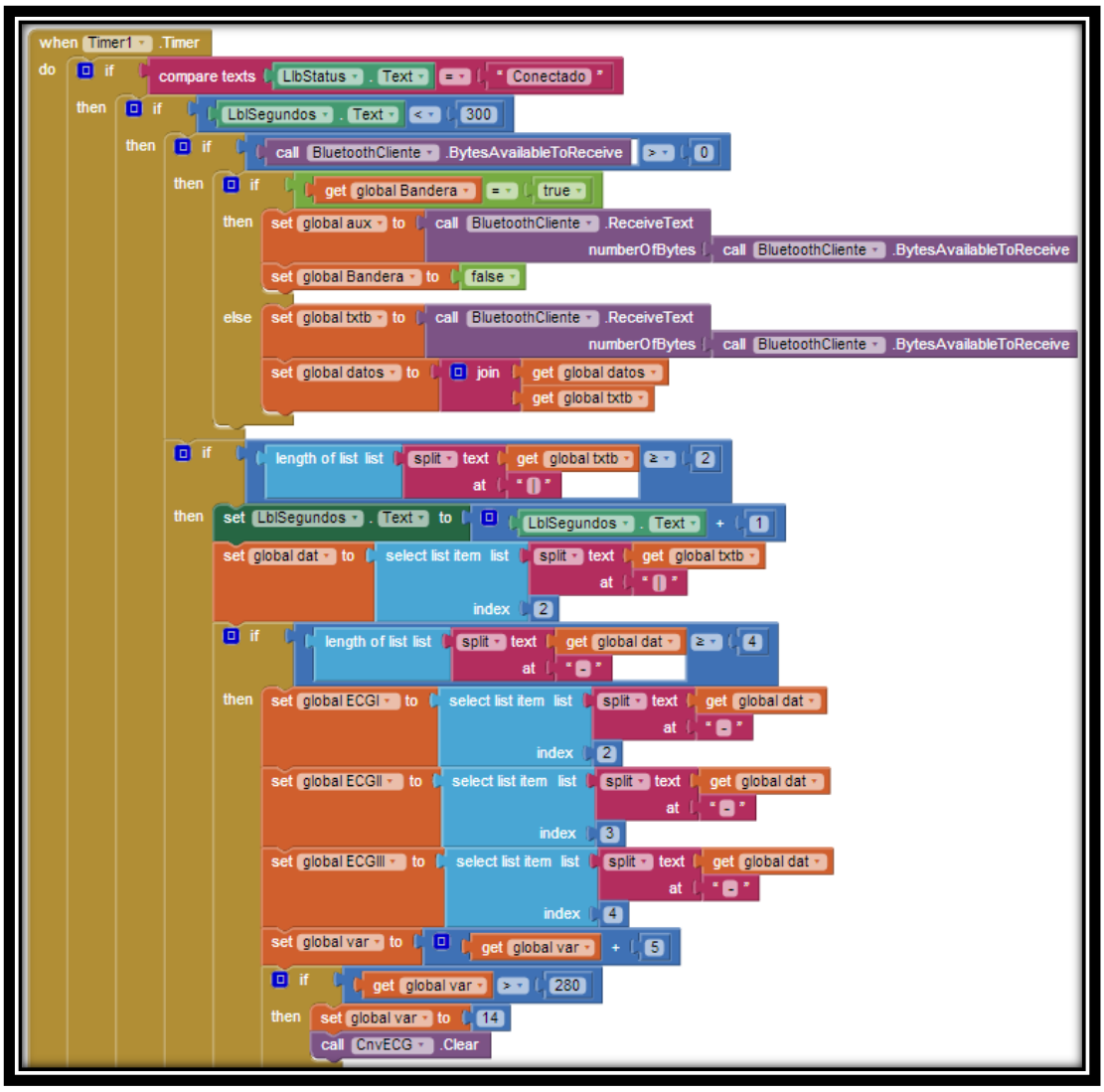
FIGURA 138. Evento Click del componente BtnIniciar.



Fuente: Autor.

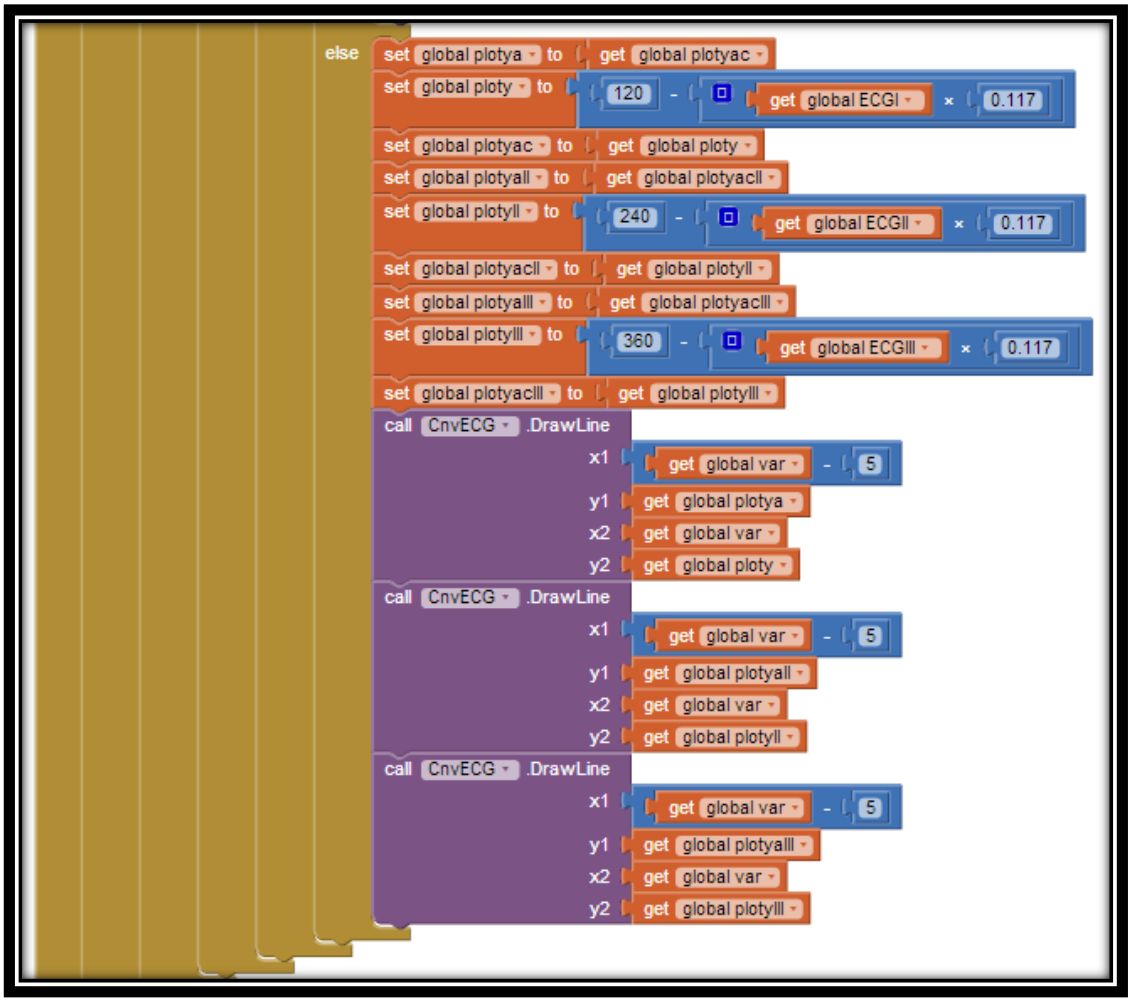
El Timer está configurado para generar un evento cada segundo a diferencia de la pantalla de ECG ya que no es necesario tomar datos con la misma velocidad, en primer lugar se verifica si existen datos en el buffer del Bluetooth, y en caso de haberlos los separa y los muestra en pantalla, ya que los datos se muestran de la forma: I dato1 I dato2 I dato3 (FIGURA 139, FIGURA 140, FIGURA 141).

FIGURA 139. Evento Timer del Timer Parte 1.



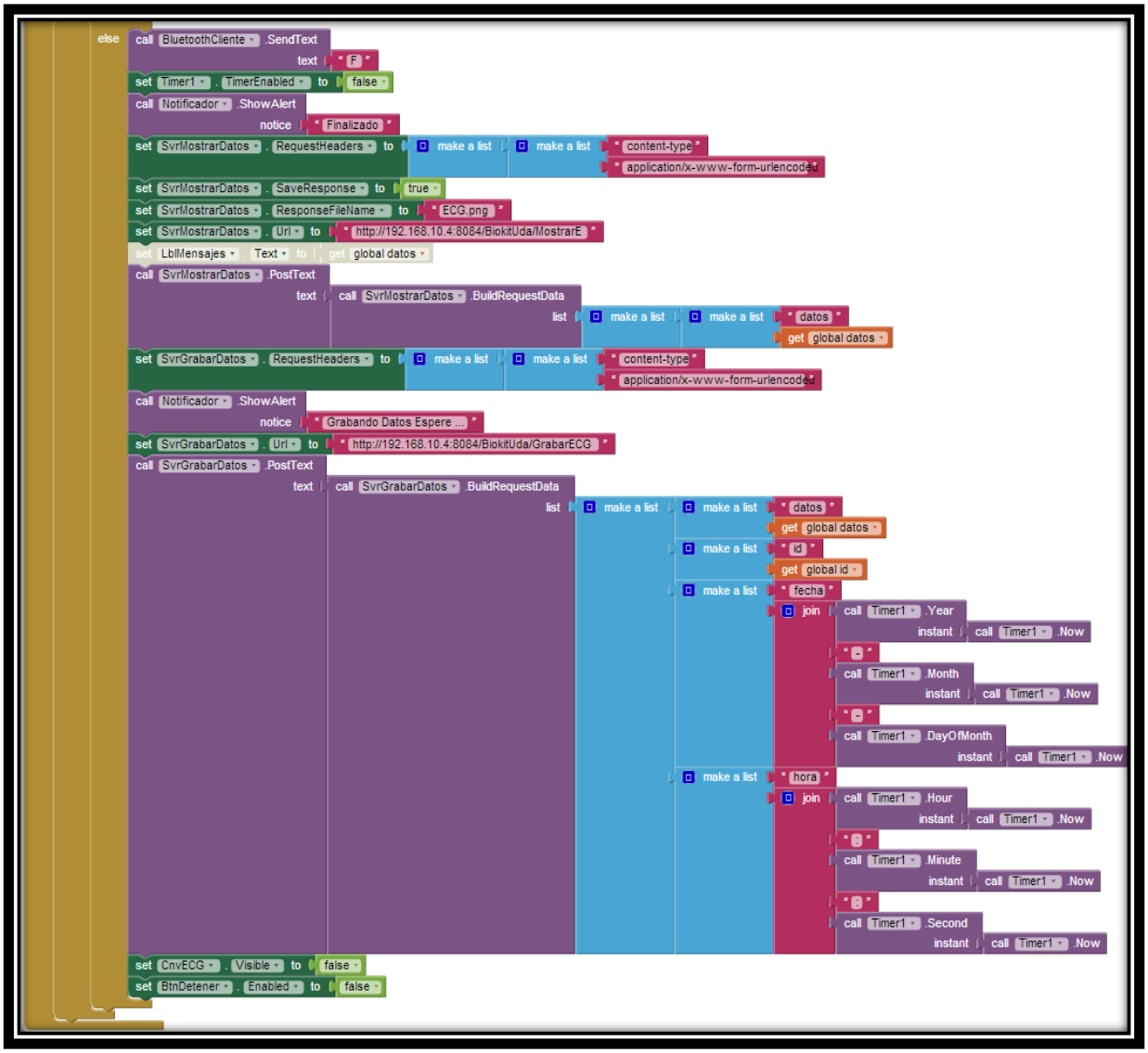
Fuente: Autor.

FIGURA 140. Evento Timer del Timer Parte 2.



Fuente: Autor.

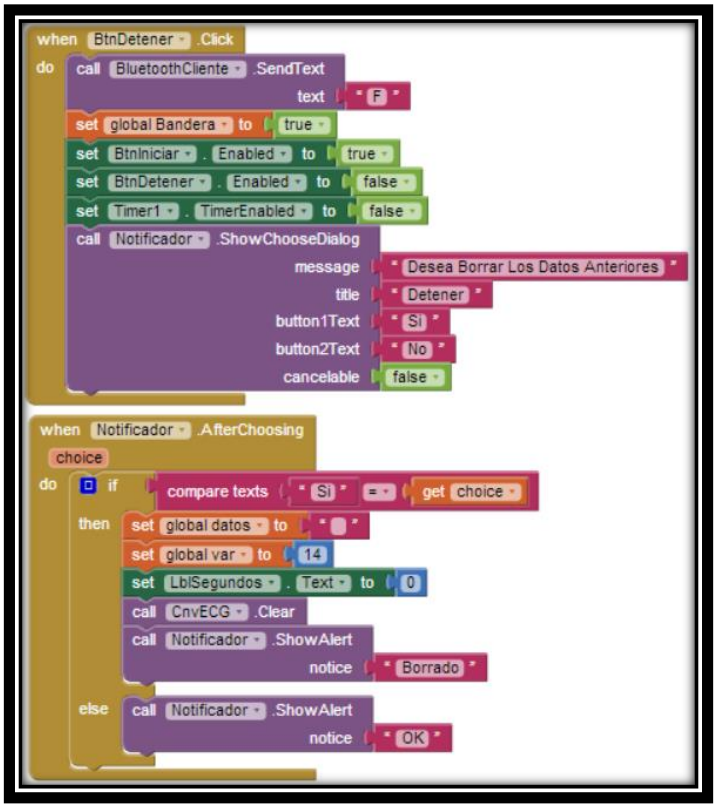
FIGURA 141. Evento Timer del Timer Parte 3.



Fuente: Autor.

Mientras el Timer este habilitado, también se habilita el botón “Detener” (**BtnDetener**), el cual, si es presionado, se encarga de enviar una señal al módulo de ECG para que deje de emitir datos y pregunta si quiere conservar los datos recolectados hasta el momento o borrarlos, con el objetivo de tomar nuevamente la información, caso contrario continuará la toma de datos en curso (FIGURA 142).

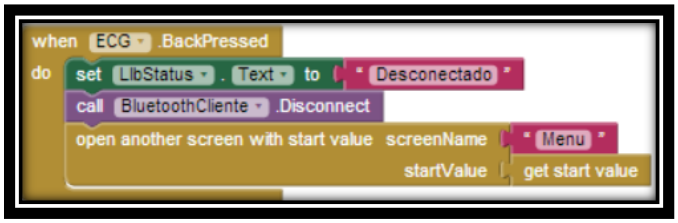
FIGURA 142. Evento Click del componente BtnDetener y su respectivo notificador.



Fuente: Autor.

Como todas las pantallas, al presionarse el Botón "Atrás" del sistema regresará al menú principal con la única diferencia de que finalizará la conexión Bluetooth (FIGURA 143).

FIGURA 143. Evento BackPressed del botón Atrás del sistema.



Fuente: Autor.

CAPÍTULO 4.

SOFTWARE DEL SERVICIO WEB

4.1 JAVA.

Lo que hoy conocemos como el software de nombre JAVA fue inicialmente desarrollado por James Gosling, perteneciente a la empresa Sun Microsystems en 1990. Inicialmente nombrado como OAK, el nombre fue modificado pues ya estaba registrado por otra empresa.

Gosling básicamente quería crear un lenguaje de programación que solucionara los fallos del lenguaje C y C++, posteriormente él y su equipo optaron por crear un nuevo lenguaje de programación mucho más sencillo capaz de ser ejecutado en cualquier entorno, tomando las características esenciales de un leguaje robusto y eliminando funciones no imprescindibles.

Posteriormente, Bill Joy, cofundador de Sun, modifico el diseño del lenguaje para poder ejecutarlo junto con internet y de esta manera hacerlo competitivo con el entonces líder de software Microsoft, y es así como en 1995 JAVA fue presentado con una enorme acogida por sus características de independencia y sencillez.

4.1.1 Definición.

Se puede definir a JAVA como un lenguaje de programación del cual se derivan tecnologías de software y plataformas informáticas basados en el mismo lenguaje. Esto nos permite, entre otras cosas, la generación de herramientas de software, juegos, aplicaciones comerciales y como veremos más adelante, aplicaciones dinámicas que interactúan a través de internet con usuarios web (Mestras, 2004).

4.1.2 Características.

JAVA es principalmente un lenguaje de programación orientada a objetos, siendo un "objeto" como una unidad dentro de un programa compuesto por datos almacenados y tareas a realizables, java utiliza objetos en sus interacciones para elaborar programas y aplicaciones.

A su vez, JAVA emplea técnicas tales como Encapsulamiento (ocultamiento de datos de un objeto), Herencia (crear nuevas clases partiendo de una clase ya existente) y polimorfismo (instrucciones sintácticamente iguales para objetos de distinto tipo).
A más de esto, JAVA goza de características tales como:

- No existen violaciones al tipo de dato o variable una vez declarada.
- Gestión automática de tratamiento de memoria
- Multihilo (soporte para ocurrencia de eventos).
- Constructores deslindados de la arquitectura del ordenador.
- Conjunto de bibliotecas que posibilitan:
    - Interfaces Graficas en 2D y 3D
    - Conectividad
    - Funciones Matemáticas.

Por su capacidad de ser ejecutado en múltiples plataformas, y contar con una gama de soporte y opciones para interactuar con el usuario, JAVA constituye un lenguaje idóneo para aplicaciones web.
En resumen podemos enumerar las siguientes ventajas que ofrece JAVA:

- Sencillez.
- Robusto y Seguro.
- Portable.
- Alto Rendimiento
- Dinámico.

SERVLETS, HTML, JSP, NODE js. Conceptos y Características
Para desarrollar programas, proyectos, o aplicaciones en general que interactúen con internet, tenemos que tener claros ciertos conceptos y características de ciertos elementos que intervienen en la programación web, misma que pasamos a analizar a continuación (Mestras, 2004).

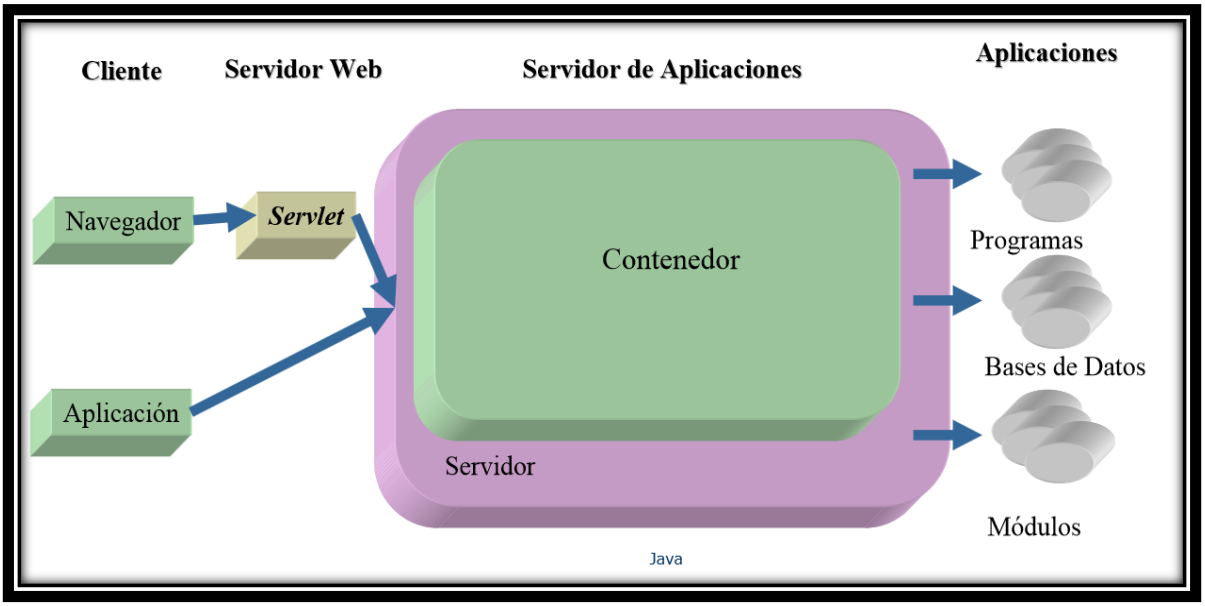
4.1.3 Servlet.

Los Servlets son programas escritos en JAVA que se ejecutan a través de un servidor web, que es un contenedor web que proporciona un medio para la ejecución y almacenamiento de distintos elementos web, en este caso los servlets o páginas HTML (Mestras, 2004) (FIGURA 144).

Su uso común es para crear páginas web dinámicas partiendo de una petición enviada por el navegador web, sus características principales son:

- Independencia de plataforma de lanzamiento.
- Uso más eficiente de memoria.
- Reutilización. (Persistencia).

FIGURA 144. Esquema gráfico de las funciones de un Servlet.



Fuente: (Mestras, 2004).

Al momento de enviar información a un servlet tenemos dos formas de hacerlo: mediante el método GET y el método POST

4.1.3.1 Método POST.

Este método solo esta accesible desde los formularios. Se envían los parámetros de forma implícita junto a la página, es decir, al pasar los parámetros, nosotros no vemos reflejado en ningún sitio qué parámetros son y cuál es su valor (Gonzalez, 2006).

4.1.3.2 Método GET.

A través de este método se envían los parámetros de forma explícita junto a la página, mostrando en la barra de navegación los parámetros y sus valores (Gonzalez, 2006). Son esas largas cadenas que aparecen en algunas páginas en nuestra barra de navegación, por ejm: buscar?id=1806&valor=0987&texto=todo&...

4.2 Servlets Biokit UDA

4.2.1 Servlet ActualizarUs.java.

Actualiza o renueva los datos del usuario, incluyendo su Id y Contraseña (FIGURA 145), para esto, el usuario debe ingresar los datos actuales de Id y contraseña de usuario, una vez ingresados, estos pasan a ser comparados con los datos de Id y contraseña albergados en la base de datos hasta ese momento, si los datos ingresados coinciden con los datos almacenados en la base, el servlet permitirá modificar los siguientes elementos:

- Id. (Numero asignado con el que está registrado el usuario en la base de datos)
- Password. (Clave de acceso).
- Nickname. (Nombre o Seudónimo de reconocimiento de usuario).
- Nombre.
- Apellido.
- Correo Electrónico.
- Teléfono.
- Dirección.
- Ciudad.
- Estado.
- País.
- Código Zip.

Ejemplo:
http://localhost:8084/BiokitUda/ActualizarUs?id=1&passvieja=xavier1006&nickname=xavi&nombre=xavier&apellido=padilla&email=xavi_padilla77@hotmail.com&pass=xavier1006&telefono=+593987899428&direccion=13-22%20129th&ciudad=New%20York&estado=New%20York&pais=USA&zip=11356

FIGURA 145. Ejemplo de Servlet Actualizar.

Fuente: Autor.

4.2.2 Servlet CargarImagen.java.

Recibe un archivo en protocolo multiparte y recibe un parámetro Id con el que se guardara y asignara el archivo recibido. En su ejecución, carga una imagen preseleccionada en el servidor, que se visualiza en la página del usuario como foto de perfil.

4.2.3 Servlet ComUsuario.java.

Comprueba si el Nickname de usuario con el que se está registrando ha sido ocupado por otro usuario, en caso de no estarlo se permitirá el acceso y mostrara el nombre de usuario, caso contrario se ejecutara un mensaje de error (FIGURA 146).

Parámetros:

- **User:** es el campo que se requiere comprobar su existencia previa.

Ejemplo:

http://localhost:8084/BiokitUda/ComUsuario?user=xavi

FIGURA 146. Ejemplo de Servlet ComUsuario.

Fuente: Autor.

4.2.4 Servlet ComEmail.java.

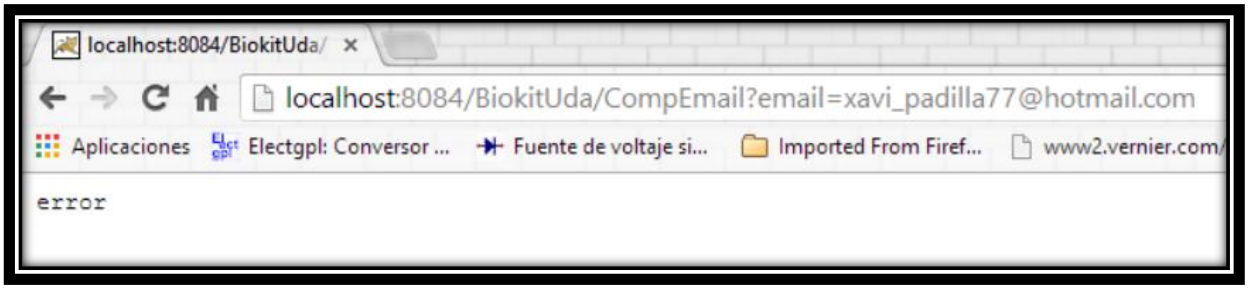
Comprueba si el E-Mail de usuario con el que se está registrando ha sido ocupado por otro usuario, en caso de no estarlo se permitirá el acceso y mostrara el nombre de usuario, caso contrario se ejecutara un mensaje de error (FIGURA 147).

Parámetros:

- **email:** es el campo que se requiere comprobar su existencia previa.

Ejemplo:
http://localhost:8084/BiokitUda/CompEmail?email=xavi_padilla77@hotmail.com

FIGURA 147. Ejemplo de Servlet ComEmail.java.



Fuente: Autor.

4.2.5 Servlet ConsultarUsuario.java.

Devuelve todos los datos de usuario (FIGURA 148) separados por el símbolo” dichos parámetros son los siguientes:

- Id. (Numero asignado con el que está registrado el usuario en la base de datos)
- Password. (Clave de acceso).
- Nickname. (Nombre o Seudónimo de reconocimiento de usuario).
- Nombre.
- Apellido.
- Correo Electrónico.
- Teléfono.
- Dirección.

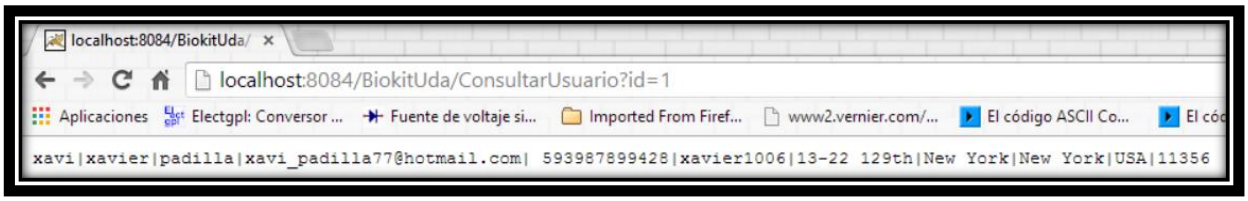
- Ciudad.
- Estado.
- País.
- Código Zip.

Parámetros:

- **id:** número de identificación con la que fue creada la cuenta de usuario.

Ejemplo:
http://localhost:8084/BiokitUda/ConsultarUsuario?id=1

FIGURA 148. Ejemplo de Servlet ConsultarUsuario.java.



Fuente: Autor.

4.2.6 Servlet GrabarECG.java.

Graba los parámetros propios del Electrocardiograma (fecha, hora, Id, ECG), en una sola cadena (string) separado cada parámetro y cada string por guiones y barras respectivamente; para posteriormente separar y asignar los datos particulares de cada string en cada ubicación correspondiente en la tabla de la base de datos (toquenizar), tomando como referencia el carácter barra para los strings y el carácter guion para los datos individuales a ser asignados en la base en un segundo toquenizado (FIGURA 149).

Parámetros:

- **Datos:** Son los datos del ECG que se requieren grabar en la base de datos, cada grupo se dividen por el caracter "|", y dentro de cada grupo se separan las derivaciones por el caracter "-".

- **Id:** número de identificación con la que fue creada la cuenta de usuario.
- **Fecha:** Fecha con la que se requiere grabar los datos adquiridos.
- **Hora:** Hora con la que se requiere grabar los datos adquiridos.

Ejemplo:
http://localhost:8084/BiokitUda/GrabarECG?datos=|-321-324-526|-215-412-854|-652-236-512|&id=1&fecha=2014-03-05&hora=9:13:12

FIGURA 149. Ejemplo de Servlet GrabarECG.java.

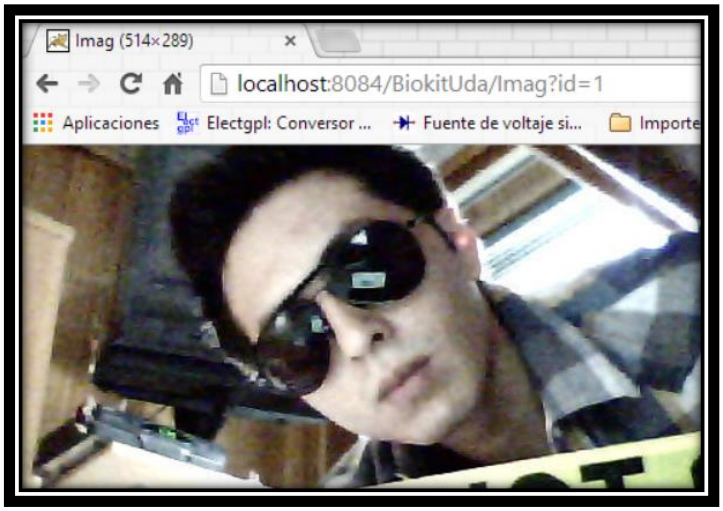
Fuente: Autor.

4.2.7 Servlet Imag.java.

Devuelve la foto de perfil del usuario previamente seleccionada y cargada, con solo ingresar el Id (FIGURA 150).

Ejemplo:
http://localhost:8084/BiokitUda/Imag?id=1

FIGURA 150. Ejemplo de Servlet Imag.java.



Fuente: Autor.

Parámetros.

- **Id:** número de identificación con la que fue creada la cuenta de usuario.

4.2.8 Servlet MostrarECGSimple.java.

Lee, desde la base de datos, el Id, la fecha, la hora ingresados y devuelve una imagen (.png) con los valores del ECG del momento correspondiente a los datos ingresados (FIGURA 151).

Ejemplo:

http://localhost:8084/BiokitUda/MostrarECGSimple?User=1&Fecha=2013-12-20&Hora=17:56:15

FIGURA 151. Ejemplo de Servlet MostrarECGSimple.java.

Fuente: Autor.

4.2.9 Servlet Perfil.java.

Devuelve una página web al ingresar el Id del usuario, (FIGURA 152) dicha página posee y exhibe los siguientes elementos:

- Fotografía de Perfil.
- Nombre.
- Apellido.
- Correo Electrónico.
- Teléfono.
- Dirección.
- Ciudad.
- Estado.
- País.
- Código Zip.

Ejemplo:

http://localhost:8084/BiokitUda/Perfil?id=1

FIGURA 152. Ejemplo de Servlet Perfil.java.

Fuente: Autor.

4.2.10 Servlet Registro.java.

Creado únicamente para la creación de nuevos usuarios, con datos inexistentes en la base de datos de usuarios, se ingresan directamente los datos del mismo sin ningún tipo de restricción (no se requiere Password), se ingresan todos los parámetros enumerados anteriormente excepto el numero Id que se asigna automáticamente (FIGURA 153).

Ejemplo:

http://localhost:8084/BiokitUda/Registro?user=pabloperez&nombre=pablo&apellido=perez&email=pperez@hotmail.com&pass=pablo&telefono=+593987012365&direccion=Av%20Americas&ciudad=Cuenca&estado=Azuay&pais=Ecuador&zip=10101

FIGURA 153. Ejemplo de Servlet Registro.java.

Fuente: Autor.

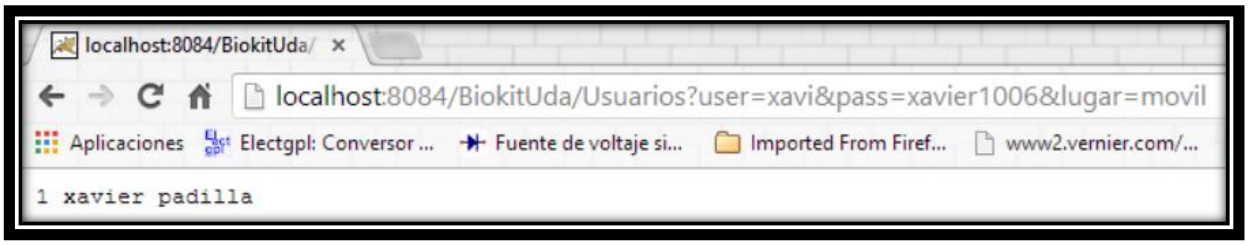
4.2.11 Servlet Usuarios.java.

Comprueba si el usuario es administrador o cliente, lo que, en la práctica, corresponde a la distinción entre Medico o Paciente, esto se realiza a través de la pantalla de Login donde se ingresa el Nickname de usuario y la contraseña del mismo, y a su vez, devuelve la página del usuario en el caso de web, o la Id, nombre y apellido del usuario en una hoja de texto al ingresar desde el móvil (FIGURA 154 y FIGURA 155).

Ejemplo:

http://localhost:8084/BiokitUda/Usuarios?user=xavi&pass=xavier1006&lugar=movil

FIGURA 154. Ejemplo de Servlet Usuarios.java.

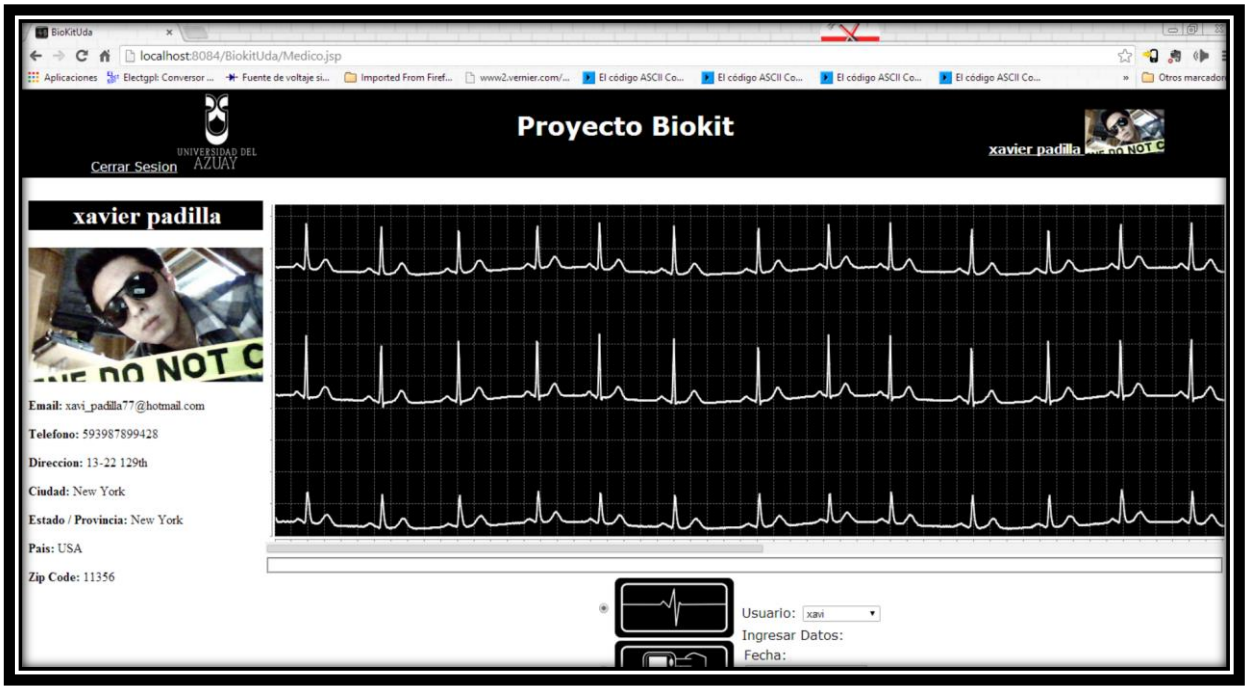


Fuente: Autor.

Ejemplo:

http://localhost:8084/BiokitUda/Usuarios?user=xavi&pass=xavier1006&lugar=web

FIGURA 155. Ejemplo de Servlet ConsultarUsuario.java aplicado en dispositivo.



Fuente: Autor.

4.3 Páginas Web.

4.3.1 HTML.

(Hyper Text Markup Languaje) Es el lenguaje usado para definir páginas web mediante etiquetas en las que se especifican elementos como textos o gráficos componentes de la página web. Se lo construye desde un editor de texto para páginas web y se lo almacena con la extensión. HTML o .htm.

Este lenguaje consta de etiquetas escritas entre los símbolos <> con una implicación en cada letra que va dentro de estos, por ejm <B> para texto en negrita (bold). <P> para un párrafo o <A> para un enlace, etc.

Entre sus ventajas podemos enumerar:

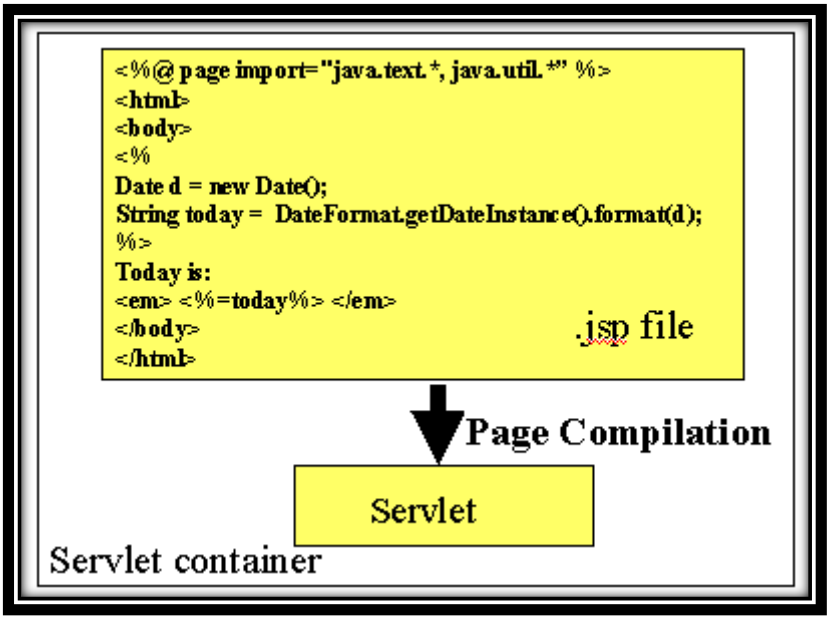
- Fácil de entender.
- Pocas etiquetas.
- Compatibles con cualquier visualizador.
- Ficheros pequeños y veloces.
- No requiere programas adicionales.

Lamentablemente tiene un número limitado de etiquetas y no es posible crear más lo que limita las opciones de diseño, a más de carecer una vista previa del producto final (Alvarez, 2001).

4.3.2 JSP.

(JAVA Server Pages) Es una tecnología de software diseñada para crear páginas web mediante programación JAVA, lo que nos permite ejecutarlas en múltiples servidores gracias a la característica multiplataforma con la que cuenta JAVA.
Las paginas JSP están escritas en código HTML/XML basadas en etiquetas especiales que programan los archivos de ordenes o procesamiento (scripts) JAVA.
El móvil; de las paginas JSP radica en los servlets, que como ya hemos visto, son programas JAVA que se ejecutan a través de un servidor, de ahí que podemos ver a JSP como una manera simplificada de construir servlets, pues el servidor interpreta el código de la página JSP y construye el código JAVA del servlet que se necesite, dicho servlet a su vez, genera el documento HTML que se verá en la pantalla de navegación (FIGURA 156).

FIGURA 156. Esquema de funcionamiento del móvil JSP.



Fuente: (Álvarez, 2002).

Entre sus ventajas, JSP posee un lenguaje de propósito general, haciendo posible su uso en otras aplicaciones aparte de las web, junto con su portabilidad por heredar la multiplataformidad de JAVA, JSP constituye una herramienta muy útil para la construcción de aplicaciones mayormente web (Álvarez, http://www.desarrolloweb.com, 2002).

4.3.3 JavaScript y NODEjs.

Java Script es un lenguaje de programación cuya utilidad radica en que los programas creados pueden ser insertados en una página web, dándonos la posibilidad de crear diferentes efectos o interactuar con los usuarios.

Entre sus características podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

Hay dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, el JavaScript propiamente dicho, denominado Navigator JavaScript. Y un JavaScript que se ejecuta en el servidor y se denomina LiveWire JavaScript

Node.js es una plataforma construida en tiempo de ejecución de JavaScript de Chrome para construir fácilmente aplicaciones rápidas de red escalables. Node.js utiliza un modelo de Entrada / Salida dirigida por eventos, dándole características de ligereza y eficiencia, ideal para aplicaciones en tiempo real de datos intensivos que se ejecutan a través de dispositivos distribuidos. NODEjs es un servidor de aplicaciones complementario de Java Script pues genera un entorno de ejecución para este, del lado del servidor (desarrolloweb.com, 2008).

4.3.4 Encriptación Multiparte.

Al enviar un formulario, el navegador interpreta esta acción como un intento por enviar de red correctamente envuelto en una estructura de mensaje de protocolo TCP/IP a través del protocolo HTTP. Al enviar los datos, se puede utilizar ya sea el método POST o GET para enviar datos a través de dicho protocolo HTTP.

El método POST le pide a su navegador construir un mensaje HTTP y poner todo el contenido en el encabezado del mensaje (una forma muy útil de hacer las cosas, más seguros y también flexibles).El método GET tiene algunas limitaciones sobre representación de datos y la longitud.

Al enviar un archivo, es necesario contar protocolo HTTP para enviar un archivo con varias características de información en su interior. De esta manera es posible enviar sistemáticamente datos para el receptor y deje que abra el archivo con el formato actual. Este es un requisito del protocolo HTTP. No se debería enviar archivos utilizando parámetros ENCTYPE por defecto, porque el receptor podría encontrar problemas al leerlo

Mediante la utilización de estos métodos se asegura de que algunos algoritmos de seguridad funcionan en tus mensajes. Esta información también es utilizada por routers de nivel de aplicación con el propósito de efectuar la labor de servidores de seguridad para datos externos (Morillo, 2012).

4.4 Páginas Web BiokitUda.

El proyecto propuesto está divido en páginas web destinadas al usuario y al médico, aunque poseen ciertas diferencias que radican mayormente en el acceso a la información la mayoría se utilizan en ambos casos y las pasamos a enumerar a continuación con sus respectivos parámetros que la componen dentro del contexto de programación:

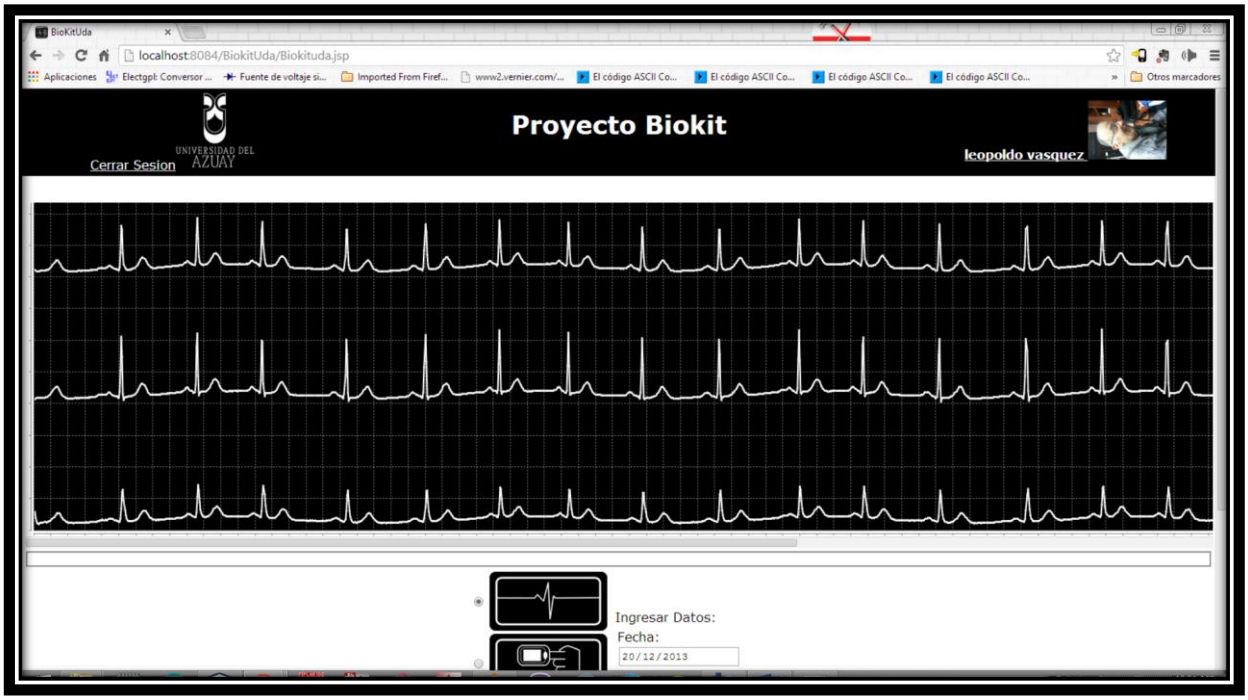
4.4.1 Biokituda.jsp.

Es la página de cada usuario del sistema, en donde se podrán realizar las consultas de los datos previamente adquiridos, además se podrá cambiar la foto de perfil al gusto de cada usuario (FIGURA 157).

Ejemplo:

http://localhost:8084/BiokitUda/Biokituda.jsp

FIGURA 157. Biokituda.jsp.



Fuente: Autor.

4.4.2 Medico.jsp.

Es una página exclusiva para los médicos para que puedan revisar los datos adquiridos por los diferentes pacientes, ya que en esta contaran con una lista desplegable que mostrara todos los usuarios registrados en el sistema (FIGURA 158).

Además y al igual que la página de usuarios (BiokitUda.jsp), se podrá cambiar la foto de perfil del médico y cerrar la respectiva sesión (FIGURA 159 y FIGURA 160).

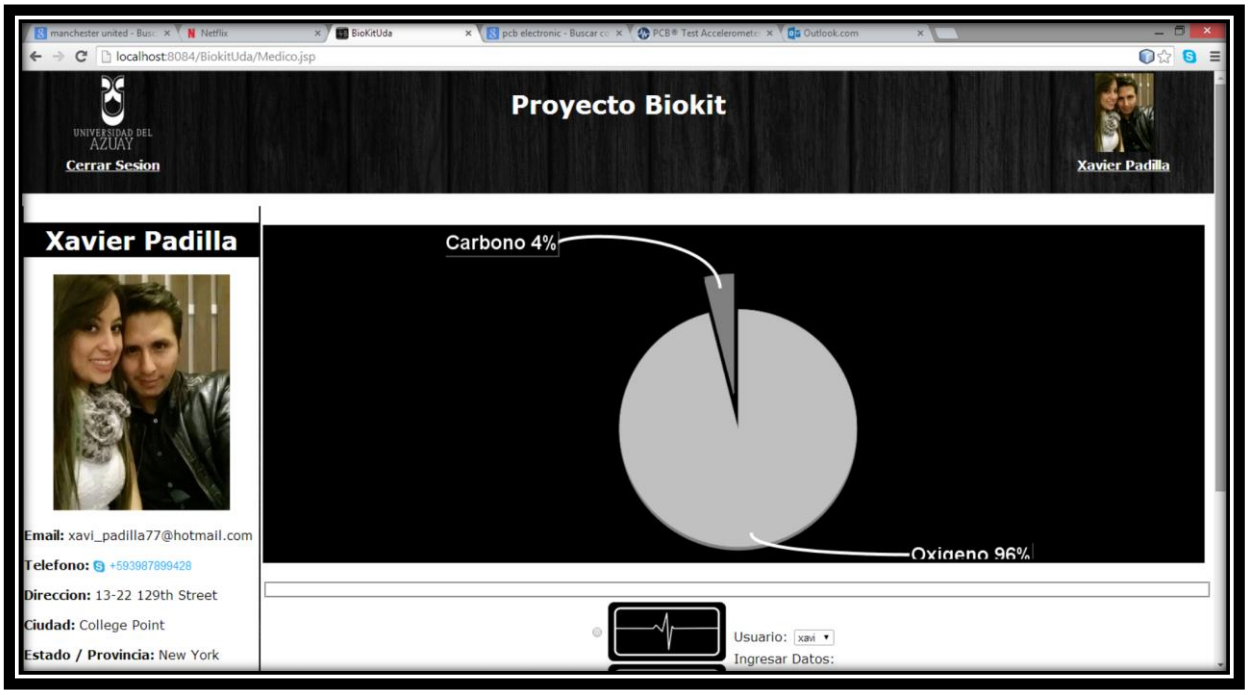
Ejemplo:

http://localhost:8084/BiokitUda/Medico.jsp

FIGURA 158.Medico.jsp.

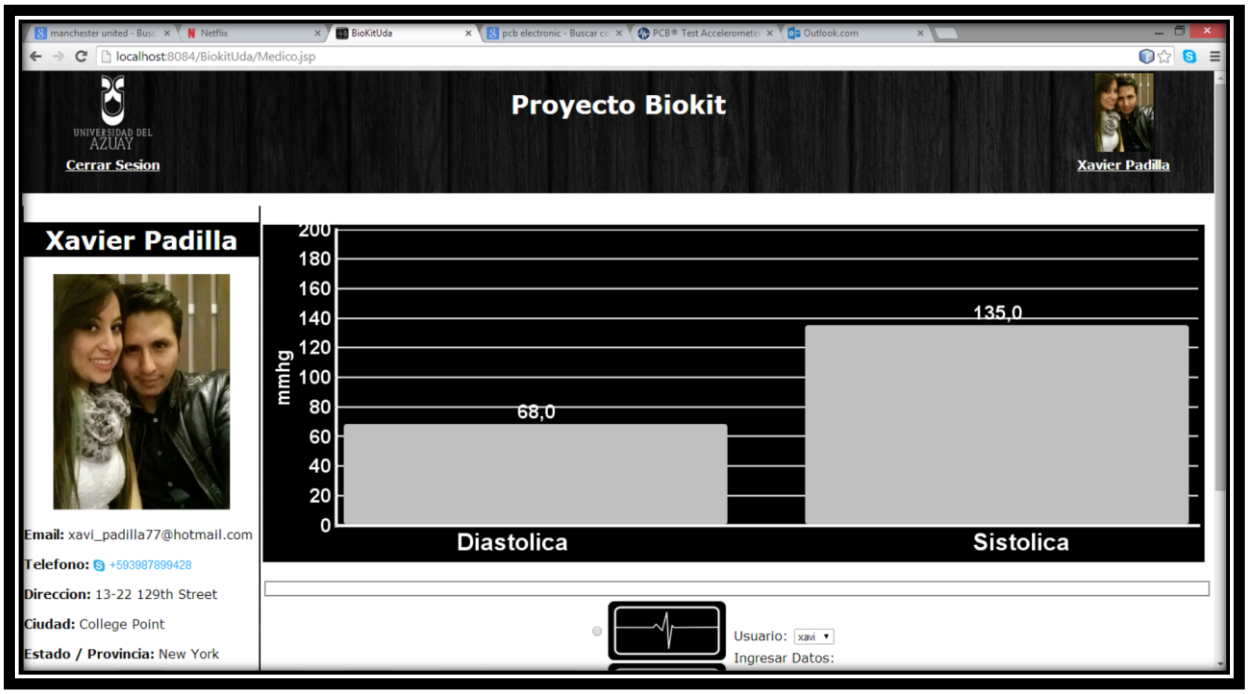
Fuente: Autor.

FIGURA 159. Medico.jsp.



Fuente: Autor.

FIGURA 160. Medico.jsp.



Fuente: Autor.

4.4.3 CargaIma.jsp.

Carga y exhibe las imágenes utilizadas en las fotografías de perfil de usuario. Verifica que la sesión sea previamente abierta y evita que los datos se muestren en el URL como medida de seguridad (FIGURA 161).

http://localhost:8084/BiokitUda/CargaIma.jsp

FIGURA 161. CargaIma.jsp.

Fuente: Autor.

4.4.4 Cerrar.jsp.

No es exactamente una página visible como la conceptualizamos, estrictamente es el código que utilizamos para cerrar las cesiones de usuario.

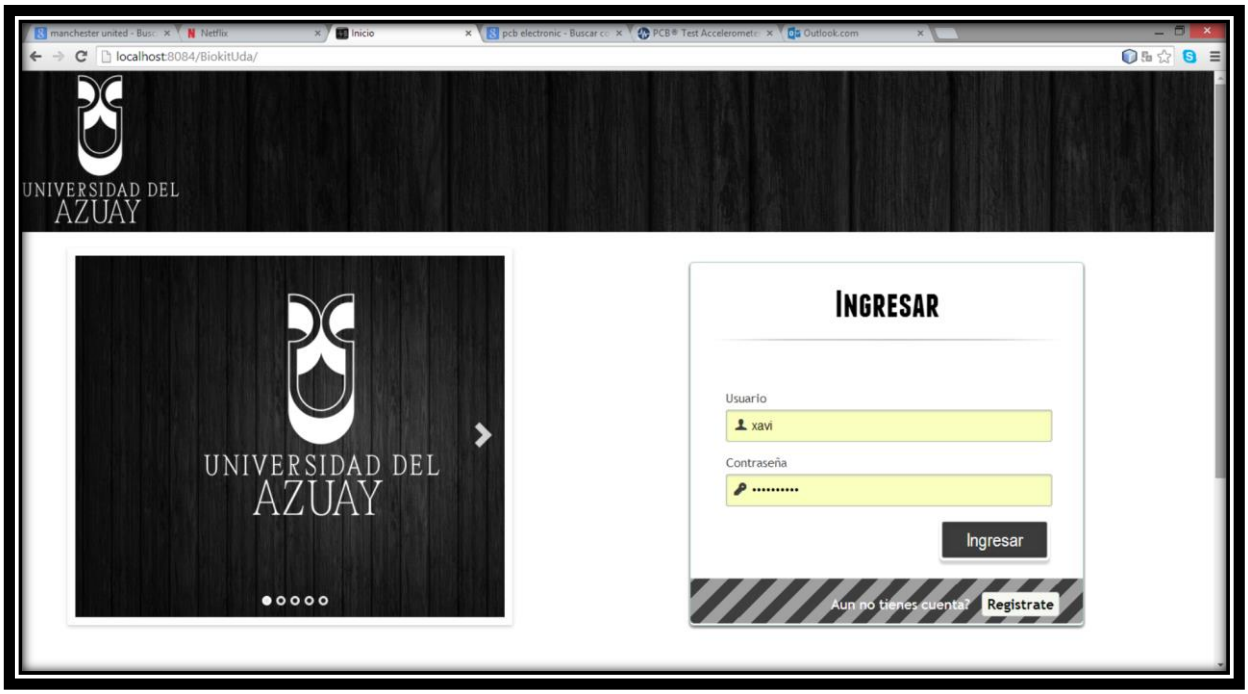
4.4.5 Imagen.jsp.

Efectúa una labor de bypass entre la página de login del biokit y la página dedicada al médico o administrador.

4.4.6 Index.jsp.

Es la página principal de bienvenida donde se ingresa el nombre de usuario y la contraseña (FIGURA 162)

FIGURA 162. Index.jsp.



Fuente: Autor.

CAPÍTULO 5.

BASE DE DATOS

5.1 Software para manejo de datos.

5.1.1 Base de datos.

Una base de datos (siglas en ingles DB). Es un conjunto ordenado de información almacenado en un espacio de memoria en un ordenador, al cual diferentes usuarios designados pueden acceder o modificar a través de un programa gestor.
Las bases de datos tradicionales suelen organizarse por campos registros y archivos. Un campo es un fragmento único de información, un registro es un sistema completo de campos, a su vez el archivo constituye una colección de registros. Por esta razón podemos decir que una base de datos es a la final un sistema electrónico ordenado de archivos (Martinez, 2002).

5.1.2 MySQL.

En la década de los 80 Michael Widenius, programador de complejas aplicaciones basadas en el lenguaje de programación BASIC, al no tener a su alcance un sistema satisfactorio de almacenamiento de datos comenzó a construir uno propio. En 1995 con la ayuda de David Axmark, Widenius presento el resultado de su trabajo: El lenguaje SQL para datos, cuyas siglas en ingles significan “Lenguaje estructurado de consulta”, siendo ahora el estándar más utilizado para trabajar con bases de datos relacionales; junto con la posibilidad de acceso a través de internet. Así como inicio el gestor MySQL junto con la empresa a la que se le atribuye su creación y propiedad: MySQL AB.
La gran evolución de MySQL desde su inicio se debe en gran parte a las peticiones y sugerencias de usuarios del producto alrededor del mundo, que a su vez son tomadas en cuenta por parte de la empresa sueca MySQL AB que continuamente mejora el código original a través de sus programadores (McGraw-Hill, 2008).

5.1.2.1 Definición.

MySQL puede ser definido como un gestor o servidor de bases de datos relacionales. Para agregar, acceder y almacenar datos en una base de datos se requiere una aplicación que sirva como administrador de dicha base de datos que nos permita interactuar con dicha base, que es la función que desempeña MySQL.

Como ya mencionamos, una base de datos es, dicho de forma simple, un conjunto ordenado de información, al hablar de base de datos relacionados, nos referimos a un almacenamiento de datos en tablas separadas unas de otras, en lugar de colocarse la información en un solo lugar, utilizamos tablas enlazadas entre sí por diferentes condiciones, esto nos permite combinar datos de diferentes tablas y acceder a la información de forma más flexible y veloz (Bravo, 2007).

5.1.2.2 Características de MySQL.

Una de las principales características de MySQL es la de ser "Open Source", término en ingles que significa que relativamente cualquier persona en el mundo puede descargar el software de MySQL desde Internet y utilizarlo de forma gratuita. De hecho, de ser necesario, cualquier usuario puede cambiar el código fuente de acuerdo a sus necesidades, pues MySQL utiliza una licencia llamada GPL (Licencia Publica General), con la posibilidad de extenderse a una licencia comercial según ciertas condiciones.

A más de esto, podemos mencionar de forma general, que MySQL como tal, posee rapidez de respuesta, relativa seguridad por medio de un sistema de Password, y facilidad de uso; debido a que este programa fue desarrollado originalmente para manejar grandes bases de datos de forma más veloz que los demás servidores existentes hasta ese momento.

Entre los detalles de carácter técnico del servidor, MySQL FIGURA 166 es un sistema basado en la metodología cliente/servidor, el servidor como tal está elaborado a través del leguaje SQL multihilo, cuenta con gran variedad de APIs (Interfaces de aplicación), junto con diversas bibliotecas enlazables entre sí, constituyendo un sistema eficiente, robusto y de fácil uso (mysql.com, 2011).

FIGURA 163. Logo de MySQL y productos relacionados.

Fuente: (mysql.com, 2011).

5.1.2.3 Diagrama o modelo entidad-relación.

A veces denominado por sus siglas en inglés, E-R "Entity relationship", o del español DER "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades (Ochando, 2014).
(, FIGURA 168, FIGURA 169.

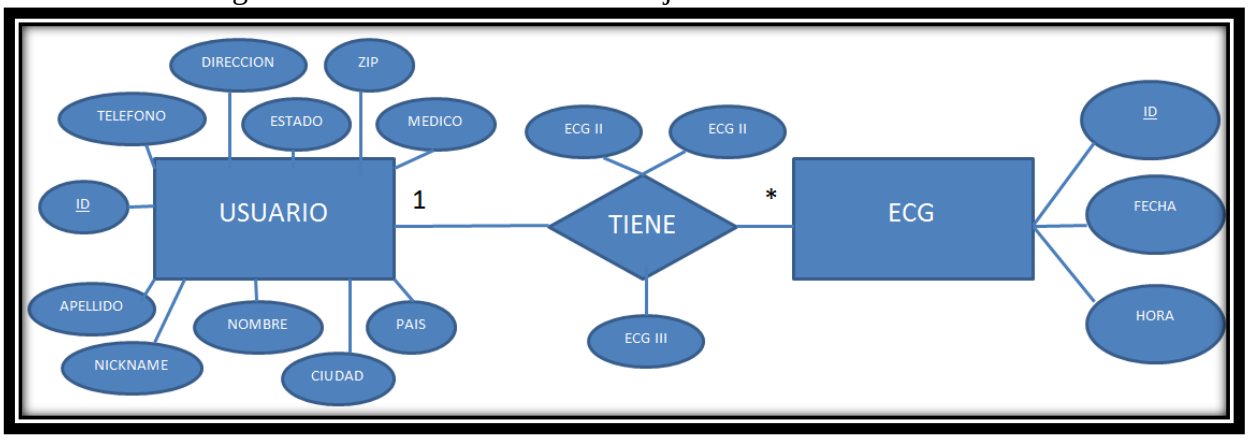
5.2 Desarrollo.

El manejo de los datos recolectados de los distintos servicios que se brinda se debe manejar de tal manera que se optimice siempre la cantidad de información ingresada en la base de datos por lo que se diseñó la base de datos de la siguiente manera:

5.2.1 Electrocardiograma.

La base de datos del servicio del electrocardiograma se maneja siguiendo el diagrama entidad relación descrito en la FIGURA 167 sugiriendo que por cada usuario se pueden tener diferentes registros de las 3 derivaciones del electrocardiograma, estos registros tendrán su propio identificador y su propia fecha y hora.
En el relacionador "Tiene" se asigna el identificador de la tabla del electrocardiograma al identificador de la tabla usuario, optimizando de esta manera la información que se agregue por cada usuario.

FIGURA 164.Diagrama entidad Relación del Manejo de datos en ECG.

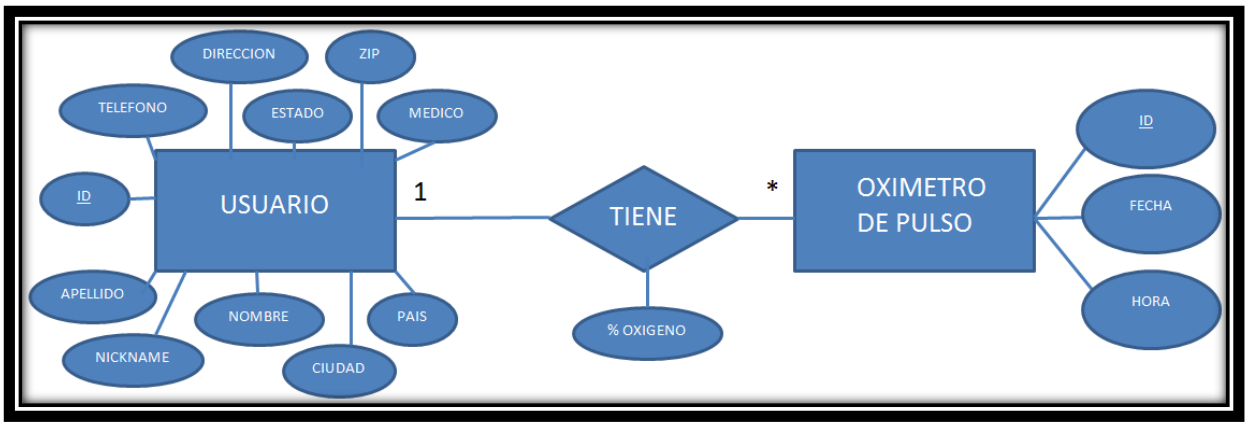


Fuente: Autor.

5.2.2 Oxímetro.

La base de datos del servicio del oxímetro se maneja siguiendo el diagrama entidad relación descrito en la FIGURA 168 sugiriendo que por cada usuario se pueden tener diferentes registros de los niveles de oxígeno de la sangre, estos registros tendrán su propio identificador y su propia fecha y hora.
En el relacionador "Tiene" se asigna el identificador de la tabla del oxímetro al identificador de la tabla usuario optimizando de esta manera la información que se agregue por cada usuario.

FIGURA 165.Diagrama entidad Relación del Manejo de datos en Oxímetro de Pulso.



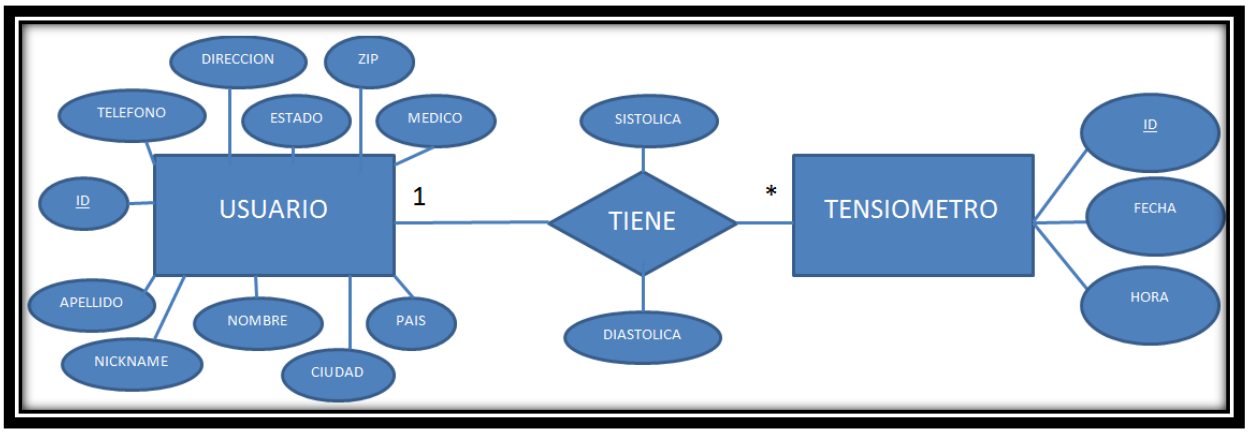
Fuente: Autor.

5.2.3 Tensiómetro.

La base de datos del servicio del tensiómetro se maneja siguiendo el diagrama entidad relación descrito en la FIGURA 169 sugiriendo que por cada usuario se pueden tener diferentes registros de la presión arterial diastólica y sistólica, estos registros tendrán su propio identificador y su propia fecha y hora.

En el relacionador “Tiene” se asigna el identificador de la tabla del tensiómetro al identificador de la tabla usuario optimizando de esta manera la información que se agregue por cada usuario.

FIGURA 166. Diagrama entidad Relación del Manejo de datos en Tensiómetro.



Fuente: Autor.

CONCLUSIONES

Obviamente aspectos del presente tema de tesis que tienen que ver con su utilidad, su aplicabilidad o su alcance dentro de determinadas circunstancias; son temas de opinión personal por parte de las personas que de alguna manera han tenido acceso a esta información, y de hecho, se puede decir que estos aspectos prácticamente caen en la categoría de lo subjetivo. Por lo expuesto, sobra decir que al momento de exponer las conclusiones, que de hecho, puede decirse que es una de las más importantes, no se incluirán este tipo de temas, sino más bien, los tópicos que tiene que ver con las experiencias, problemáticas, recomendaciones, y demás aspectos objetivos que vinieron surgiendo conforme se desarrollaba el proyecto, mismos de los que, en un todo, se puede decir que es de lo que se trata, la ingeniería como tal. Dichos eventos, se pasan a listar a continuación, todos con una breve reseña de los detalles que los conforman.

- **Ruido 60hz** Lamentablemente, uno de los mayores problemas que surgieron al momento del desarrollo del proyecto, fue que, al trabajar con señal eléctricas provenientes de órganos del cuerpo humano y por ende señales eléctricas muy débiles, se filtraba excesiva cantidad de ruido en cada una de las fases de desarrollo, lo que hasta cierto punto era previsible y atenuado mediante filtros "tradicionales" por así decirlos, sin embargo surgió un tipo de ruido que contaminaba la pureza de la señal proveniente de la mismísima fuente de alimentación, o dicho de otra forma, del tomacorriente convencional conocido por todos. Como es bien sabido, esta fuente proporciona energía eléctrica de corriente alterna que trabaja con 60 Hz de frecuencia, lo que significa que , a más de obviamente proveer la energía para el funcionamiento del equipo, filtraba ruido en la señal; para dicha contingencia resultó necesario implementar un filtro especial conocido cono filtro NOTCH activo, que en resumen, se caracteriza por rechazar una frecuencia determinada que este interfiriendo con un circuito, en nuestro caso la frecuencia es de 60Hz generada por la fuente de energía.

- **Utilidad de App Inventor.** A estas alturas de la exposición, y fuera de la importancia que tiene o pudiera llegar a tener el proyecto, sin lugar a dudas

queda claro que el entorno de desarrollo que responde al nombre de **App inventor** es una herramienta de software extremadamente útil y con ventajas adicionales igualmente innegables, sin embargo, derivado de la experiencia en el desarrollo del tema, se puede afirmar que el software en cuestión es útil para relativamente sencillos, pero comienza a verse escaso de recursos al momento del desarrollo de proyectos de mayor envergadura, pues no es capaz de procesar una mayor cantidad de señales en un determinado instante de tiempo con la misma velocidad o frecuencia que lo hiciera si se trabajara con una menor cantidad de señales en el mismo instante de tiempo. Si bien es cierto, el lenguaje de programación de App Inventor es sumamente cómodo, y por ende útil, al momento de trabajar o desarrollar proyectos de mayores requerimientos se sugiere echar mano de entornos de programación Android de bajo nivel, séase software como Android Studio o Eclipse.

- **Programación Arduino** En cuanto a la programación de microcontrolador, hay un punto muy parecido al anterior, y es que la versatilidad de Arduino permite también una programación de microcontrolador en bajo nivel, también extremadamente útil para propósitos de mayor detalle tales como adquisición de señales, convertidores AD y datos en general.

- **Filtros Digitales** Sin ahondar demasiado, un filtro es un elemento que solo permite pasar a través de él, la parte útil de una determinada señal, el asunto en cuestión es que al existir muchos tipos de filtros o técnicas de filtraje, a veces puede escogerse algún tipo de filtro que a la final no resulte tan conveniente, por esta razón, y en cuanto al desarrollo del proyecto se refiere, es altamente recomendable utilizar filtros digitales de referencia los que provienen de elementos precisos como microcontroladores, por encima de los filtros analógicos tradicionales, ya que estos últimos no son capaces de generar una señal lo suficientemente útil, lo que en consecuencia sacrifica velocidad en lo que respecta a adquisición de datos y se traduce en perdida de los mismos.

BIBLIOGRAFÍA.

ALVAREZ, M. A. (1 de Enero de 2001). *http://www.desarrolloweb.com*. (desarrolloweb.com) Recuperado el Junio de 2014, de http://www.desarrolloweb.com/articulos/que-es-html.html

ÁLVAREZ, M. A. (8 de Julio de 2002). *http://www.desarrolloweb.com*. (desarrolloweb.com/articulos) Recuperado el Junio de 2014, de http://www.desarrolloweb.com/articulos/831.php

ÁLVAREZ, M. A. (8 DE JULIO DE 2002). *HTTP://WWW.DESARROLLOWEB.COM*. OBTENIDO DE http://www.desarrolloweb.com/articulos/831.php aplicaciones-android.org. (8 de Septiembre de 2011). *http://www.aplicaciones-android.org*. Recuperado el Mayo de 2014, de http://www.aplicaciones-android.org/sistema-operativo-android/

AZUAY, U. d. (2005). *http://www.uazuay.edu.ec*. (uazuay.edu.ec) Recuperado el Julio de 2014, de http://www.uazuay.edu.ec/analisis/El%20modelo%20relacional.pdf

BRANDAN, N. A. (2008). *http://docs.moodle.org/*. Recuperado el Enero de 2014, de http://docs.moodle.org/all/es/images_es/5/5b/Hemoglobina.pdf

BRAVO, I. M. (2007). *http://indira-informatica.blogspot.com/*. (indira-informatica.blogspot.com/) Recuperado el Julio de 2014, de http://indira-informatica.blogspot.com/2007/09/qu-es-mysql.html

BRITO, R. (11 de febrero de 2013). *es.slideshare.net*. (slideshare) Recuperado el Abril de 2014, de http://es.slideshare.net/yanri94/sensores-de-presin

CUEVA, I. A. (2008). *http://www.monografias.com/*. Recuperado el Marzo de 2014, dehttp://www.monografias.com/trabajos90/electromedicina/electromedicina.shtmldesarrolloweb.com. (2008). *http://www.desarrolloweb.com*. (desarrolloweb.com) Recuperado el Junio de 2014, de http://www.desarrolloweb.com/javascript/#quees

GEOSALUD. (s.f.). *http://www.geosalud.com*. Obtenido de http://www.geosalud.com/hipertension/procedimiento_ha.htm

GONZALEZ,E. (2006). *http://www.aprenderaprogramar.com*. (aprenderaprogramar.com) Recuperado el Junio de 2014, de http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=527:get-y-post-html-method-formas-de-envio-de-datos-en-

formulario-diferencias-y-ventajas-ejemplos-cu00721b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

HONEYWELL. (Noviembre de 2008). *http://sensing.honeywell.com/*. (Honeywell) Recuperado el Abril de 2014, de http://sensing.honeywell.com/index.php/ci_id/49876/la_id/1/document/1/re_id/0 *http://www.geosalud.com/*. (s.f.). (Webmaster GeoSalud) Recuperado el 20 de abril de 2014, de http://www.geosalud.com/hipertension/procedimiento_ha.htm

KUBOTA., J. C. (16 de Agosto de 2009). *http://es.scribd.com/*. (scribd) Recuperado el 20 de Febrero de 2014, de http://es.scribd.com/doc/18665760/Elementos-de-Espectroscopia-y-Ley-de-Lambert-Beer

MARTINEZ, J. A. (13 de Agosto de 2002). *http://www.monografias.com*. (monografias.com) Recuperado el Julio de 2014, de http://www.monografias.com/trabajos11/basda/basda.shtml

MCGRAW-HILL, t. (30 de 10 de 2008). *http://www.mailxmail.com*. (mailxmail.com) Recuperado el Julio de 2014, de http://www.mailxmail.com/curso-mysql-informatica/mysql-origenes-historia

MESTRAS, J. P. (2004). *http://www.fdi.ucm.es*. (Dep. Sistemas Informáticos y Programación. Universidad Complutense Madrid) Recuperado el Junio de 2014, de http://www.fdi.ucm.es/profesor/jpavon/poo/02IntroJava.pdf

MORILLO, A. E. (2012). *http://recsi2012.mondragon.edu/es*. (mondragon.edu/es) Recuperado el Junio de 2014, de http://recsi2012.mondragon.edu/es/programa/recsi2012_submission_31.pdf

MULTSTOCK. (2012). *http://www.multstock.com.* Obtenido de http://www.multstock.com.br/nossos-produtos/oximetro-de-pulso-md300c/ mysql.com. (2011). *http://dev.mysql.com*. (mysql.com) Recuperado el Julio de 2014, de http://dev.mysql.com/doc/refman/5.0/es/features.html

NUÑEZ, K. (18 de Julio de 2013). *es.slideshare.net*. (es.slideshare.net) Recuperado el Mayo de 2014, de http://es.slideshare.net/karenonunez/sistema-operativo-android-versiones-historia

OCHANDO, M. B. (20 de febrero de 2014). *http://ccdoc-basesdedatos.blogspot.com*. Recuperado el Julio de 2014, de http://ccdoc-basesdedatos.blogspot.com/2013/02/modelo-entidad-relacion-er.html

OSUNA, I. O. (Julio de 2012). *http://www.elandroidelibre.com*. Recuperado el Junio de 2014, de http://www.elandroidelibre.com/2012/03/app-inventor-de-nuevo-disponible-para-que-crees-tus-aplicaciones.html?aceptarCookies=023fa46f41713fa1bb43049828b49cef7472e293

PASQUIER, D. R. (19 de Enero de 2013). *http://www.medicinapreventiva.com.ve/*. (medicinapreventiva.com.ve) Recuperado el Marzo de 2014, de http://www.medicinapreventiva.com.ve/auxilio/signos/tension.htm

PÉREZ, V. (23 de septiembre de 2010). *Laguia2000*. Obtenido de http://matematica.laguia2000.com/general/ley-de-beer-lambert

RAMON,A. X. (13 de Marzo de 2013). *blogspot.com*. Obtenido de http://alexalteno1.blogspot.com/2013/03/triangulo-de-einthoven.html

RIEGO,A.R. (24 de Enero de 2013). *https://sites.google.com/site/appinventormegusta*. Recuperado el Junio de 2014, de https://sites.google.com/site/appinventormegusta/primeros-pasos

RIEGO,A.R. (24 de Enero de 2013). *https://sites.google.com/site/appinventormegusta*. Recuperado el Junio de 2014, de https://sites.google.com/site/appinventormegusta/conceptos

RIEGO,A.R. (24 de Enero de 2013). *https://sites.google.com/site/appinventormegusta*. Recuperado el Junio de 2014, de https://sites.google.com/site/appinventormegusta/instalacion

ROBREDO.,A. R. (1993). *Microelectronica 92.* (U. d. Cantabria, Ed.) Santander, Espana: Graficas Calima S.A. Recuperado el Abril de 2014, de http://books.google.com.ec/books?id=lFoQLyduGC4C&pg=PA114&dq=piezorresistivo+modulo&hl=es&sa=X&ei=n8DNU8PcHKzesAStkYCIBA&ved=0CE4Q6wEwCQ#v=onepage&q=piezorresistivo%20modulo&f=false

SARDIÑAS, M. R. (2008). *http://www.monografias.com/*. Recuperado el 4 de abril de 2014, de http://www.monografias.com/trabajos58/medicion-tension-arterial/medicion-tension-arterial.shtml

SARDIÑAS, M. R. (2008). *http://www.monografias.com/*. Recuperado el 4 de abril de 2014, de http://www.monografias.com/trabajos58/medicion-tension-arterial/medicion-tension-arterial2.shtml

SAVÓN, D. T. (31 de Enero de 2007). *revistaciencias.com*. Recuperado el Abril de 2014, de http://www.revistaciencias.com/publicaciones/EEZpklkZZkOTohDvCL.php

SYLVIA PALACIOS M., C. Á. (2010). *http://www.scielo.cl/*. Recuperado el 10 de Febrero de 2014, de http://www.scielo.cl/pdf/rcher/v26n1/art10.pdf

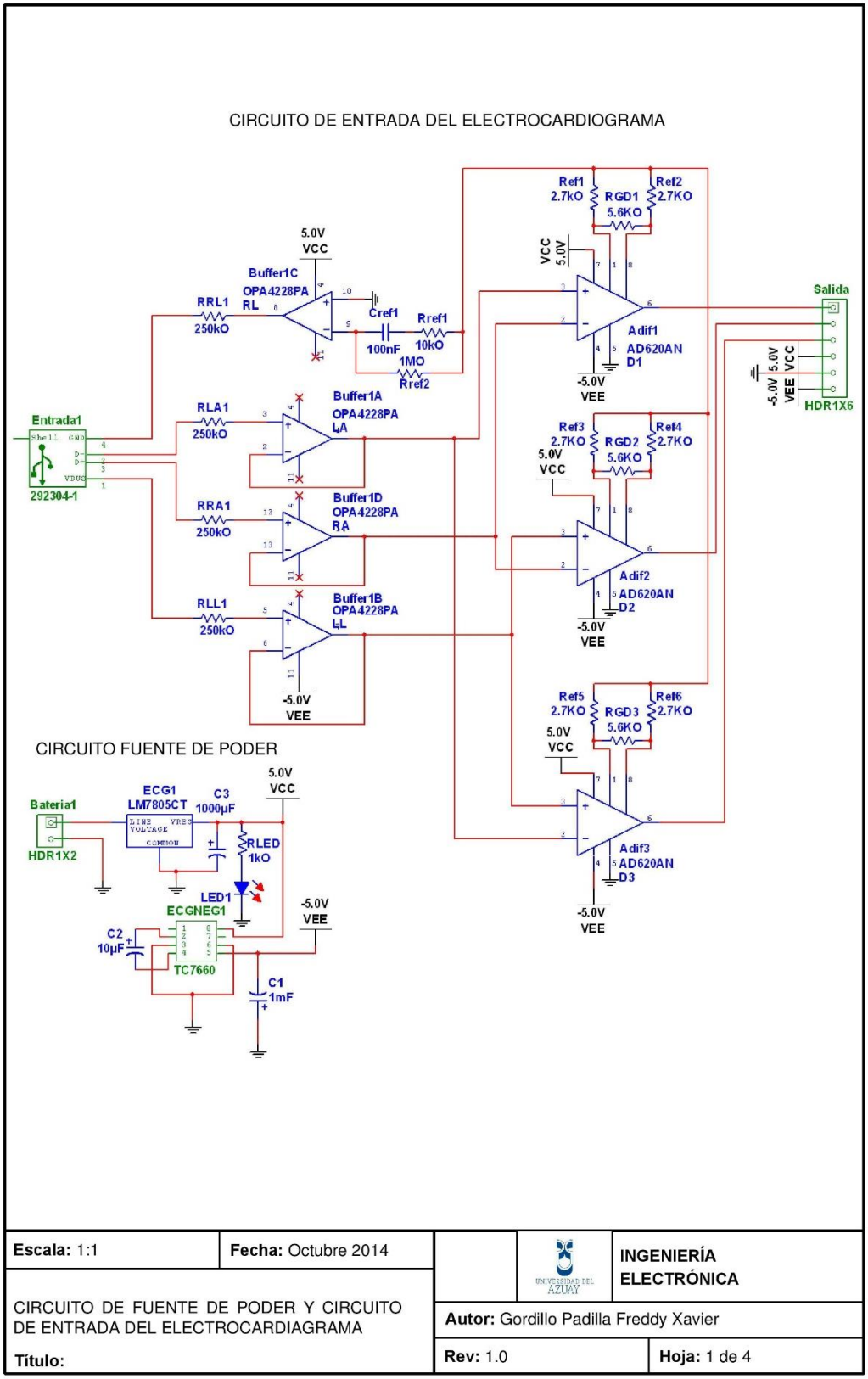
TOLOZA, A. M. (2005). *http://www.dalcame.com*. (Dalcame) Recuperado el 10 de Octubre de 2014, de http://www.dalcame.com/ecg.html#.U80zBeN5NUU. 1.

TOLOZA, A. M. (2005). *http://www.dalcame.com/*. (Grupo de investigacion biomedica DALCAME) Recuperado el Enero de 2014, de http://www.dalcame.com/ecg.html#.U80zBeN5NUU

UNIVERSIDAD CES. (2007). *http://its.uvm.edu*. (Universidad CES) Recuperado el Mayo de 2014, de http://its.uvm.edu/medtech/Tmodule_es.html

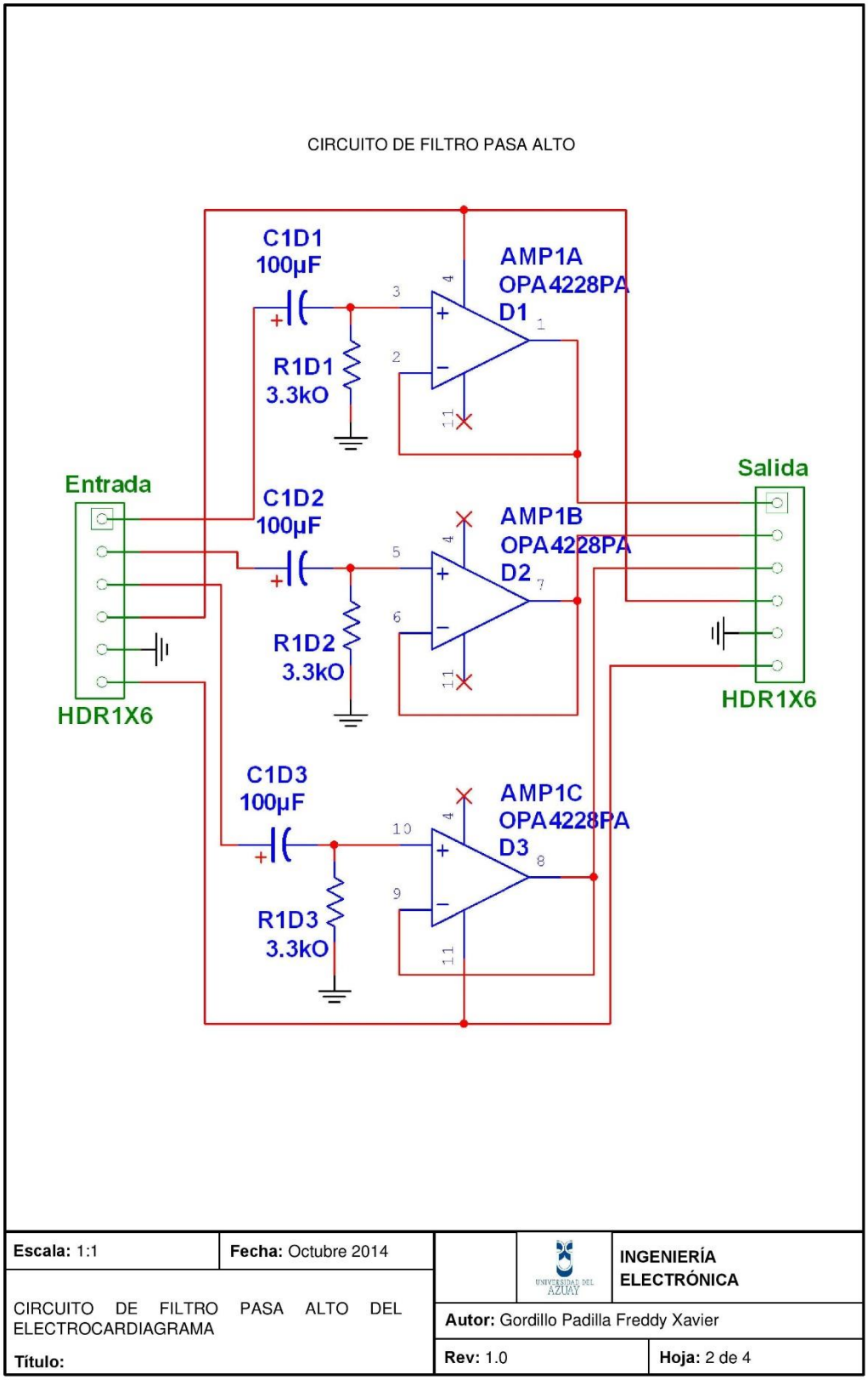
ANEXOS.

ANEXO 1: Circuito de Entrada del Electrocardiograma.

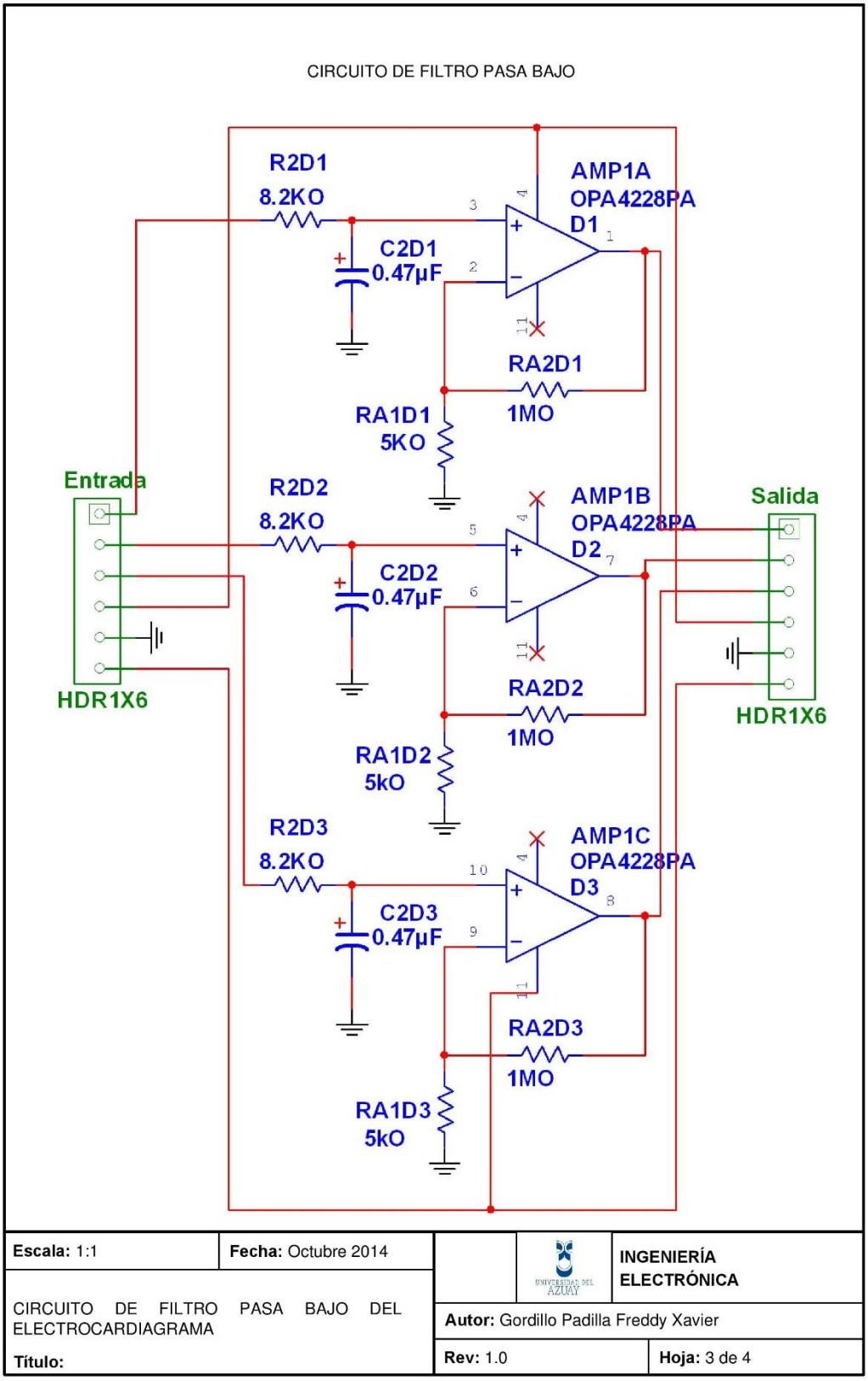


Escala: 1:1	**Fecha:** Octubre 2014		INGENIERÍA ELECTRÓNICA
CIRCUITO DE FUENTE DE PODER Y CIRCUITO DE ENTRADA DEL ELECTROCARDIAGRAMA		**Autor:** Gordillo Padilla Freddy Xavier	
Título:		**Rev:** 1.0	**Hoja:** 1 de 4

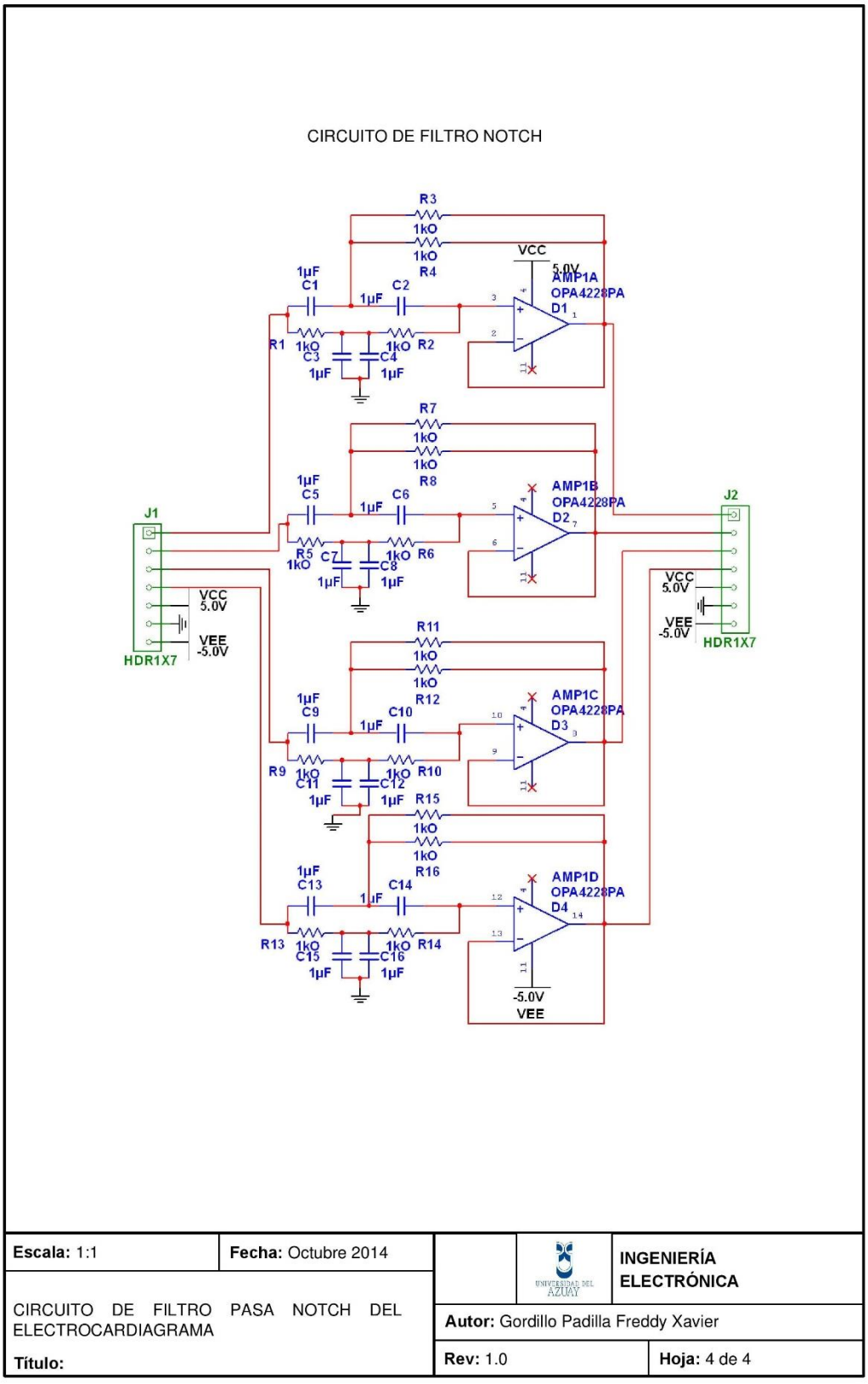
ANEXO 2: Circuito de Filtro Pasa Alto del Electrocardiograma.



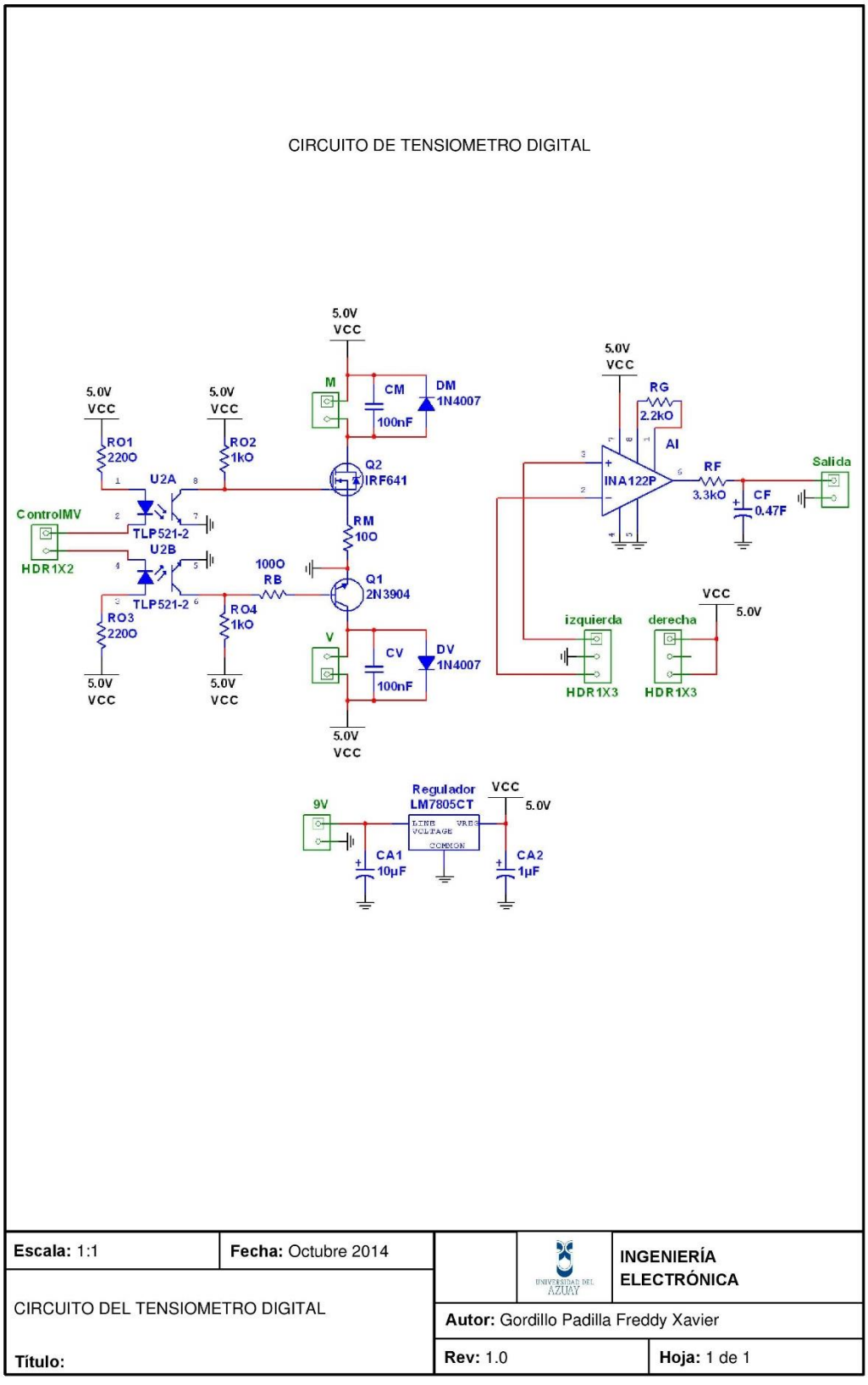
ANEXO 3: Circuito de Filtro Pasa Bajo del Electrocardiograma.



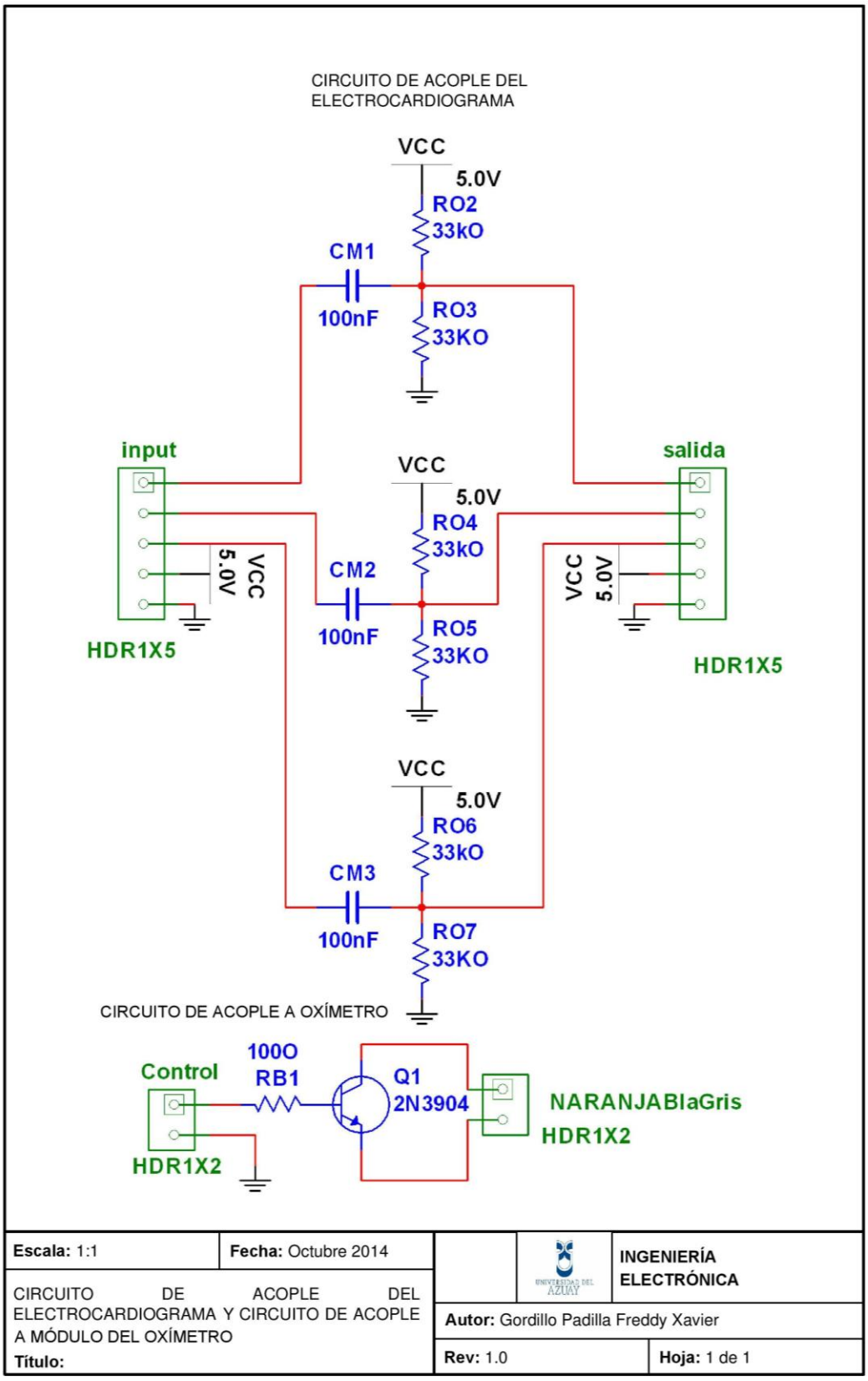
ANEXO 4: Circuito de Filtro Notch del Electrocardiograma.



ANEXO 5: Circuito de Tensiómetro Digital.



ANEXO 6: Circuito de Acople del Electrocardiograma y Módulo de Oxímetro.



ANEXO 7: Código de Index.html

```
1
2  <!DOCTYPE HTML>
3  <HTML lang="en">
4      <head>
5          <meta charset="UTF-8" />
6          <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
7          <title>Inicio</title>
8          <meta name="viewport">
9          <meta content="no-cache">
10         <link href="img/ico1.png" type="image/x-icon" rel="shortcut icon" />
11         <link rel="stylesheet" type="text/css" href="css/demo.css" />
12         <link rel="stylesheet" type="text/css" href="css/style.css" />
13         <link rel="stylesheet" type="text/css" href="css/style2.css" />
14         <link rel="stylesheet" type="text/css" href="css/animate-custom.css" />
15         <script type="text/javascript" src="js/modernizr.custom.04022.js"></script>
16                                                                                                  <link
href='http://fonts.googleapis.com/css?family=Open+Sans+Condensed:700,300,300italic'
rel='stylesheet' type='text/css'/>
17         <style type="text/css">
18             <!--
19             body {
20                 font: 100%/1.4 Verdana, Arial, Helvetica, sans-serif;
21                 background-color: #000;
22                 margin: 0;
23                 padding: 0;
24                 color: #000;
25                 background-image:url(img/map.png);
26             }
27
28             ul, ol, dl {
29                 padding: 0;
30                 margin: 0;
31             }
32             h1, h2, h3, h4, h5, h6, p {
33                 margin-top: 0;
34                 padding-right: 15px;
35                 padding-left: 15px;
36             }
37             a img {
38                 border: none;
39             }
40
41             a:link {
42                 color:#414958;
43                 text-decoration: underline;
44             }
45             a:visited {
46                 color: #4E5869;
47                 text-decoration: underline;
48             }
49             a:hover, a:active, a:focus {
50                 text-decoration: none;
51             }
52
53             .container {
54                 width: 100%;
55                 height: 100%;
56                 background-color: #FFF;
57                 margin: 0 auto;
58                 color: #FFF;
59                 background-image: none;
60                 text-align: center;
61             }
62
63             .header {
64                 background-color: #000;
65                 background-image:url(img/map.png);
66                 text-align:left;
67             }
68             .cajas_texto {
69                 background-color: #FFF;
70                 text-align:left;
71                 font: 85%/1.4 Verdana, Arial, Helvetica, sans-serif;
72             }
73             .botones {
74                 background-color: #000;
75                 text-align: center;
76                 font: 85%/1.4 Verdana, Arial, Helvetica, sans-serif , bold;
77                 width: auto;
```

```
            color: #FFF;
            padding-right: 15px;
            padding-left: 15px;
            left: 80px;
            position: relative;
        }
        .icono {
            background-color: #FFF;
            text-align: center;
            font: 85%/1.4 Verdana, Arial, Helvetica, sans-serif , bold;
            width: auto;
            color: #000;
            padding-right: 15px;
            padding-left: 15px;
            left: 60px;
            position: relative;
        }

        .sidebar1 {
            float: right;
            width: 55%;
            height: auto;
            background-color: #FFFFFF;
            padding-bottom: 100px;
            color: #000;
            text-align:left;
        }
        .content {
            padding: 10px 0;
            width: 45%;
            float: right;
        }

        .content ul, .content ol {
            padding: 0 15px 15px 40px;
        }

        ul.nav {
            list-style: none;
            border-top: 1px solid #666;
            margin-bottom: 15px;
            background-image: url(../tesis/leather_black.jpg);
        }
        ul.nav li {
            border-bottom: 1px solid #666;
        }
        ul.nav a, ul.nav a:visited {
            padding: 5px 5px 5px 15px;
            display: block;
            text-decoration: none;
            background-color: #FFFFFF;
            color: #000;
        }
        ul.nav a:hover, ul.nav a:active, ul.nav a:focus {
            background-color: #FFFFFF;
            color: #FFF;
        }

        .footer {
            padding: 10px 0;
            background-color: #000000;
            background-image:url(img/map.png);
            position: relative;
            clear: both;
            font-family: Verdana, Geneva, sans-serif;
            color: #FFF;
            text-align: center;
        }

        .fltrt {
            float: right;
            margin-left: 8px;
        }
        .fltlft {
            float: left;
            margin-right: 8px;
        }
        .clearfloat {
            clear:both;
            height:0;
            font-size: 1px;
            line-height: 0px;
        }
```

```
161             -->
162         </style></head>
164     <body>
166         <div class="container">
167             <div class="header">
168                 <img src="img/uda.png" width="200" height="200">
171             </div>
172             <div class="sidebar1">
173                 <p> </p>
174                 <div id="container_demo" >
175                     <a class="hiddenanchor" id="toregister"></a>
176                     <a class="hiddenanchor" id="tologin"></a>
177                     <div id="wrapper">
178                         <div id="login" class="animate form">
179                             <form  action="Usuarios?lugar=web" method="POST">
180                                 <h1>Ingresar</h1>
181                                 <p>
182                                    <label for="username" class="uname" data-icon="u" > Usuario
</label>
183                                           <input id="user" name="user" required type="text"
placeholder="JuanPerez"/>
184                                 </p>
185                                 <p>
186                                        <label for="password" class="youpasswd" data-icon="p">
Contraseña </label>
187                                         <input id="pass" name="pass" required type="password"
placeholder="ejemplo: X8df!90E" />
188                                 </p>
190                                 <p class="login button">
191                                     <input type="submit" value="Ingresar" />
192                                 </p>
193                                 <p class="change_link">
194                                     Aun no tienes cuenta?
195                                     <a href="#toregister" class="to_register">Registrate</a>
196                                 </p>
197                             </form>
198                         </div>
200                         <div id="register" class="animate form">
201                             <form  action="mysuperscript.php" autocomplete="on">
202                                 <h1> Registrar </h1>
203                                 <p>
204                                             <label for="usernamesignup" class="uname" data-
icon="u">Usuario</label>
205                                     <input id="usernamesignup" name="usernamesignup" required
type="text" placeholder="mysuperusername690" />
206                                 </p>
207                                 <p>
208                                      <label for="emailsignup" class="youmail" data-icon="e" >
Email</label>
209                                          <input id="emailsignup" name="emailsignup" required
type="email" placeholder="mysupermail@mail.com"/>
210                                 </p>
211                                 <p>
212                                          <label for="passwordsignup" class="youpasswd" data-
icon="p">Contraseña </label>
213                                     <input id="passwordsignup" name="passwordsignup" required
type="password" placeholder="eg. X8df!90EO"/>
214                                 </p>
215                                 <p>
216                                    <label for="passwordsignup_confirm" class="youpasswd" data-
icon="p">Repetir Contraseña </label>
217                                <input id="passwordsignup_confirm" name="passwordsignup_confirm"
required type="password" placeholder="eg. X8df!90EO"/>
218                                 </p>
219                                 <p class="signin button">
220                                     <input type="submit" value="Sign up"/>
221                                 </p>
222                                 <p class="change_link">
223                                     Ya eres miembro ?
224                                     <a href="#tologin" class="to_register">Ingresar</a>
225                                 </p>
226                             </form>
227                         </div>
229                     </div>
230                 </div>
231             </div>
232             <div class="content">
233                 <div class="sp-slideshow">
235                      <input id="button-1" type="radio" name="radio-set" class="sp-selector-1"
checked="checked" />
236                     <label for="button-1" class="button-label-1"></label>
237                      <input id="button-2" type="radio" name="radio-set" class="sp-selector-2"
/>
```

```
238                     <label for="button-2" class="button-label-2"></label>
239                      <input id="button-3" type="radio" name="radio-set" class="sp-selector-3"
/>
240                     <label for="button-3" class="button-label-3"></label>
241                      <input id="button-4" type="radio" name="radio-set" class="sp-selector-4"
/>
242                     <label for="button-4" class="button-label-4"></label>
243                      <input id="button-5" type="radio" name="radio-set" class="sp-selector-5"
/>
244                     <label for="button-5" class="button-label-5"></label>
245                     <label for="button-1" class="sp-arrow sp-a1"></label>
246                     <label for="button-2" class="sp-arrow sp-a2"></label>
247                     <label for="button-3" class="sp-arrow sp-a3"></label>
248                     <label for="button-4" class="sp-arrow sp-a4"></label>
249                     <label for="button-5" class="sp-arrow sp-a5"></label>
250                     <div class="sp-content">
251                         <div class="sp-parallax-bg"></div>
252                         <ul class="sp-slider clearfix">
253                             <li><img src="img/image1.png" alt="image01" /></li>
254                             <li><img src="img/image2.png" alt="image02" /></li>
255                             <li><img src="img/image3.png" alt="image03" /></li>
256                             <li><img src="img/image4.png" alt="image04" /></li>
257                             <li><img src="img/image5.png" alt="image05" /></li>
258                         </ul>
259                     </div>
260                 </div>
261             </div>
262             <div class="footer">
263                 <table border="0" cellpadding="0" align="center">
264                     <tbody>
265                         <tr>
266                             <td><img src="img/electrocardiograma.png" width="200" height="150"
/></td>
267                               <td><img src="img/BioKitUda_clip_image001.png" alt="" width="63"
height="137" /></td>
268                             <td><img src="img/oximetro.png" width="200" height="150" /></td>
269                               <td><img src="img/BioKitUda_clip_image001.png" alt="" width="63"
height="137" /></td>
270                            <td><img src="img/tensiometro.png" width="200" height="150" /></td>
271                         </tr>
272                     </tbody>
273                 </table>
274                 <p>Derechos Reservados..</p>
275             </div>
276         </div>
277     </body>
278 </html>
```

ANEXO 8: Código de BiokitUda.jsp

```
<%
    response.setHeader("Cache-Control", "no-store"); //HTTP 1.1
    response.setHeader("Pragma", "no-cache"); //HTTP 1.0
    response.setDateHeader("Expires", 0); %>
  <%@   page   contentType="text/html;   charset=utf-8"   language="java"   import="java.sql.*"
errorPage="" session="true"%>
     <!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%
    HttpSession sesion = request.getSession();
    String id = (String) sesion.getAttribute("id");
    String nombre = (String) sesion.getAttribute("nombre");
    String apellido = (String) sesion.getAttribute("apellido");
    if (id == null || nombre == null || apellido == null) {
        response.sendRedirect("CerrarSesion.jsp");
    } else {
        sesion.setAttribute("user", id);
        sesion.setAttribute("modo", "web");
        sesion.setAttribute("nuevo", "no");
    }
%>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <link href="img/ico1.png" type="image/x-icon" rel="shortcut icon" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>BioKitUda</title>
        <style type="text/css">
            <!--
            body {
                font: 100%/1.4 Verdana, Arial, Helvetica, sans-serif;
                background-color: #42413C;
                margin: 0;
                padding: 0;
                color: #000;
            }
            ul, ol, dl {
                padding: 0;
                margin: 0;
            }
            h1, h2, h3, h4, h5, h6, p {
                margin-top: 0;
                padding-right: 15px;
                padding-left: 15px;
            }
            a img {
                border: none;
            }

            a:link {
                color: #FFF;
                text-decoration: underline;
            }
            a:visited {
                color: #FFF;
                text-decoration: underline;
            }
            a:hover, a:active, a:focus {
                text-decoration: none;
            }

            .container {
                width: 100%;
                height: 100%;
                background-color: #FFF;
                margin: 0 auto;
            }

            .header {
                background-color: #000;
                background-image: url(img/map.png);
                color: #FFF;
                text-align: center;
            }

            .sidebar1 {
                float: left;
                max-width: 30%; min-width: 20%;
                height: 100%;
```

```
78                  background-color: #FFF;
79                  text-align: center;
80
81              }
82              .content {
83                  background-color: #FFF;
84                  width: 80%;
85                  float: left;
86                  text-align: left;
87              }
88
89              .footer {
90                  padding: 10px 0;
91                  background-color: #000;
92                  background-image: url(img/map.png);
93              }
94
95              .fltrt {
96                  float: right;
97                  margin-left: 8px;
98              }
99              .fltlft {
100                 float: left;
101                 margin-right: 8px;
102             }
103             .clearfloat {
104                 clear:both;
105                 height:0;
106                 font-size: 1px;
107                 line-height: 0px;
108             }
109             -->
110         </style></head>
111
112     <script language="javascript">
113         function abrir(Fd, Hd, Us) {
114             var Tb;
115             if (document.formulario.Tablas[0].checked) {
116                  parent.principal.location.href = "MostrarECGSimple?Fecha=" + Fd + "&Hora=" +
Hd + "&User=" + Us;
117             }
118             if (document.formulario.Tablas[1].checked) {
119                 parent.principal.location.href = "MostrarOximetroWeb?fecha=" + Fd + "&hora=" +
Hd + "&id=" + Us;
120             }
121             if (document.formulario.Tablas[2].checked) {
122                parent.principal.location.href = "MostrarTensiometroWeb?fecha=" + Fd + "&hora="
+ Hd + "&id=" + Us;
123             }
124
125         }
126     </script>
127     <body>
128         <div class="container">
129             <div class="header">
130                 <table width="100%" border="0">
131                     <tr>
132                     <td width="15%"><img src="img/uda.png" width="100" height="100"/></a></td>
133                       <td width="70%"><div align="center"><h1>Proyecto Biokit</h1> </div></td>
134                         <td width="15%"><img src='usuarios/${id}.jpg' style="max-width: 100px;
max-height: 100px" /></a></td>
135                     </tr>
136                 </table>
137                 <table width="100%" border="0">
138                     <tr>
139                       <td width="15%"><a href="CerrarSesion.jsp"><b>Cerrar Sesion</b></a></td>
140                         <td width="70%"><div align="center"><h1></h1> </div></td>
141                               <td width="15%"><a href="cargaima.jsp" id="enlace"><b>${nombre}
${apellido} </b></a></td>
142                     </tr>
143                 </table>
144                 <p> </p>
145                 <!-- end .header --> </div>
146             <div class="sidebar1" >
147                   <iframe src="PerfilUsuario?user=xavi" name="perfil" width=100% height=752px
frameborder="0"></iframe>
148             </div>
149             <div class="content">
150                 <h1>
151
152                 </h1>
153                 <table width=100% border="0" >
154                     <tr>
```

```
155                                 <td><iframe src="" name="principal" width=100% height="450"
frameborder="0"></iframe></td>
156                         </tr>
157                         <tr>
158                             <td>
159                                 <form name="formulario">
160                                     <table width="200" border="0" align="center">
161                                         <tr>
162                                             <td><table border="0">
163                                                     <tr>
164                                                         <td><input type="radio" name="Tablas"
value="ecg" id="ecg"checked="checked" onclick="eventofechas();"/></td>
165                                                           <td><img src="img/ecg8x4.png" width="150"
height="75" alt="ecg" /></td>
166                                                     </tr>
167                                                     <tr>
168                                                         <td><input type="radio" name="Tablas"
value="oximetro" id="oximetro" onclick="eventofechas();"/></td>
169                                                            <td><img src="img/oximetro8x4.png"
width="150" height="75" alt="oximetro" /></td>
170                                                     </tr>
171                                                     <tr>
172                                                         <td><input type="radio" name="Tablas"
value="tensiometro" id="tensiometro" onclick="eventofechas();"/></td>
173                                                         <td><img src="img/tensiometro8x4.png"
width="150" height="75" alt="tensiometro" /></td>
174                                                     </tr>
175                                                 </table></td>
176                                             <fieldset>
177                                                 <td><table width="200" border="0" align="left">
178                                                         <tr>
179                                                             <td>Usuario:
180                                                                     <select name="sUser"
id="sUser"></select></td>
181                                                         </tr>
182                                                         <tr>
183                                                             <td><table border="0" align="left">
184                                                                     <thead>
185                                                                         Ingresar Datos:
186                                                                     </thead>
187                                                                     <tbody>
188                                                                         <tr>
189                                                                             <td>Fecha:
190                                                                                 <input type="date"
name="fec" id="fec"/></td>
191                                                                             <td>
192                                                                             </td>
193                                                                         </tr>
194                                                                         <tr>
195                                                                             <td>Hora:
196                                                                               <select name="sPro"
id="sPro" ></select> </td>
197                                                                             <td></td>
198                                                                         </tr>
199                                                                         <tr></tr>
200                                                                     </tbody>
201                                                                 </table></td>
202                                                         </tr>
203                                                         <tr>
204                                                                   <td><input type="button"
onClick="abrir(document.formulario.fec.value,                       document.formulario.sPro.value,
document.formulario.sUser.value);" value="Abrir"></input></td>
205                                                         </tr>
206                                                     </table>
207                                             </fieldset>
208                                             </td>
209                                         </tr>
210                                     </table>
211                                 </form>
212                             </td>
213                         </tr>
214                 </table>
215
216                                                          <script    type="text/javascript"
src="http://192.168.10.2:3000/socket.io/socket.io.js"></script>
217                 <script type="text/javascript">
218                                                                                 var io =
io.connect("http://192.168.10.2:3000/");
219
220                                                                               var sUser =
document.getElementById('sUser');
221                                                 var sPro = document.getElementById('sPro');
222                                                 var sDis = document.getElementById('sDis');
```

```
223
224                                                 var fecha = document.getElementById('fec');
225                                                     fecha.setAttribute("onchange",
"eventofechas()");
226                                                   fecha.setAttribute("onkeypress",
"eventofechas()");
227                                                     sUser.setAttribute("onchange",
"eventousuarios()");
228
229
230                                                 function eventousuarios() {
231                                                     parent.perfil.location.href =
"PerfilUsuario?user=" + sUser.options[sUser.selectedIndex].text;
232                                                     eventofechas();
233                                                 }
234
235                                                 function eventofechas() {
236                                                 if (document.formulario.Tablas[0].checked)
{
237                                                         io.emit("horas", {fecha:
fecha.value, id: sUser.value});
238                                                     }
239                                                 if (document.formulario.Tablas[1].checked)
{
240                                                         io.emit("horasoximetro",
{fecha: fecha.value, id: sUser.value});
241                                                     }
242                                                 if (document.formulario.Tablas[2].checked)
{
243                                                       io.emit("horastensiometro",
{fecha: fecha.value, id: sUser.value});
244                                                     }
245                                                 }
246                                                 function hora() {
247
248                                                 }
249                                                 io.on("usuarios", function(data) {
250                                                     var totDep = data.length;
251                                                     sUser.length = totDep;
252
253                                                       for (var i = 0; i < totDep;
i++) {
254                                                           sUser.options[i] = new
Option(data[i].nickname, data[i].id);
255                                                     }
256                                                     ;
257
258                                                     eventofechas();
259                                                 });
260
261                                                 io.on("horas", function(data) {
262                                                     var totPro = data.length;
263                                                     sPro.length = totPro;
264
265                                                       for (var i = 0; i < totPro;
i++) {
266                                                           sPro.options[i] = new
Option(data[i].hora, data[i].hora);
267                                                     }
268                                                     ;
269
270                                                 });
271                                               io.on("horasoximetro", function(data)
{
272                                                     var totPro = data.length;
273                                                     sPro.length = totPro;
274
275                                                       for (var i = 0; i < totPro;
i++) {
276                                                           sPro.options[i] = new
Option(data[i].hora, data[i].hora);
277                                                     }
278                                                     ;
279
280                                                 });
281
282                                                   io.on("horastensiometro",
function(data) {
283                                                     var totPro = data.length;
284                                                     sPro.length = totPro;
285
286                                                       for (var i = 0; i < totPro;
i++) {
```

```
287                                                     sPro.options[i] = new
Option(data[i].hora, data[i].hora);
288                                                 }
289                                                 ;
290
291                                             });
292
293                 </script>
294             </div>
295             <div class="footer">
296                 <p> </p>
297                 <p> </p>
298                 <p> </p>
299                 <p> </p>
300
301             </div>
302         </div>
303     </body>
304 </html>
```

ANEXO 9: Código de Cargaima.jsp

```
<%
    response.setHeader("Cache-Control", "no-store"); //HTTP 1.1
    response.setHeader("Pragma", "no-cache"); //HTTP 1.0
    response.setDateHeader("Expires", 0);  %>
<%@ page contentType="text/html; charset=utf-8" language="java" import="java.sql.*"
errorPage="" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%
    HttpSession sesion = request.getSession();
    String user_id = (String) sesion.getAttribute("user");
    String modo = (String) sesion.getAttribute("modo");
    String nuevo = (String) sesion.getAttribute("nuevo");
    sesion.setAttribute("user", user_id);
    sesion.setAttribute("modo", "web");
%>
<%
    String Us = "CargarImagen?ID=" + user_id + "&modo=" + modo;
    String ima;
    if (nuevo.equals("si")) {
        ima = "img/NoUserImage.jpg";
    } else {
        ima = "usuarios/" + user_id + ".jpg";
    };
%>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Cargar Imagen</title>
        <style type="text/css">
            <!--
            body {
                font: 100%/1.4 Verdana, Arial, Helvetica, sans-serif;
                background-color: #000;
                margin: 0;
                padding: 0;
                color: #000;
                background-image: url(img/map.png);
            }
            ul, ol, dl {
                padding: 0;
                margin: 0;
            }
            h1, h2, h3, h4, h5, h6, p {
                margin-top: 0;
                padding-right: 15px;
                padding-left: 15px;
                font-family: Verdana, Arial, Helvetica, sans-serif;
                text-align: center;
            }
            a img {
                border: none;
            }

            a:link {
                color: #fff;
                text-decoration: underline;
            }
            a:visited {
                color: #6E6C64;
                text-decoration: underline;
            }
            a:hover, a:active, a:focus {
                text-decoration: none;
            }

            .container {
                width: 480px;
                background-color: #FFF;
                margin: 0 auto;
            }

             .header {
                background-color: #000;
                font-family: Verdana, Geneva, sans-serif;
                font-size: 36px;
                font-style: normal;
                line-height: normal;
                color: #FFF;
```

```
                font-weight: bold;
                background-image: url(img/map.png);
            }

            .content {
                padding: 10px 0;
                font-family: Verdana, Arial, Helvetica, sans-serif;
            }

            .footer {
                padding: 10px 0;
                background-color: #000000;
                color: #FFF;
                background-image: url(img/map.png);
            }
            .fltrt {
                float: right;
                margin-left: 8px;
            }
            .fltlft {
                float: left;
                margin-right: 8px;
            }
            .clearfloat {
                clear:both;
                height:0;
                font-size: 1px;
                line-height: 0px;
            }
            .container .header table tr td {
                text-align: center;
                font-family: "Arial Black", Gadget, sans-serif;
            }
            -->
        </style></head>

    <body>

        <div class="container">

            <div class="header">
                <table width="485" border="0" cellpadding="000">
                    <tr>
                        <td width="178"><img src="img/uda.png" alt="UDA" name="logo"
width="178" height="180" id="logo" /></td>
                        <td width="301">BIOKIT</td>
                    </tr>
                </table>

                <hr /></div>

            <div class="content">
                <h1>Seleccionar Imagen</h1>
                <p><img src="<%= ima%>" style="max-width: 229px; max-height: 229px" /></p>
                <form action=<%= Us%> method="post" enctype="multipart/form-data"
name="formularioimagen" id="formularioimagen">
                    <p>

                        <input type="file" name="CargarImag" id="CargarImag" />
                    </p>
                    <p>
                        <input type="submit" name="Subir" id="Subir" value="Enviar" />
                    </p>
                </form>
            </div>
            <div class="footer">
                <p>Biokit UDA derechos reservados</p>

            </div>
        </div>
    </body>
</html>

```

ANEXO 10: Código de CerrarSesion.jsp

```jsp
<%
    response.setHeader("Cache-Control", "no-store"); //HTTP 1.1
    response.setHeader("Pragma", "no-cache"); //HTTP 1.0
    response.setDateHeader("Expires", 0);; %>
<%@ page session="true" %>
<%
    HttpSession sesionOk = request.getSession();
    sesionOk.invalidate();
    response.sendRedirect("index.html");
%>
```

ANEXO 11: Código de ActualizarUs.

```
 1 package Servlets;
 2
 3 import java.io.*;
 4 import java.sql.*;
 5 import javax.servlet.*;
 6 import javax.servlet.http.*;
 7
 8 public class ActualizarUs extends HttpServlet
 9 {
10    private Connection conexion = null;
11    private Statement sentencia = null;
12    private ResultSet conjuntoResultados = null;
13    private String controlador = "com.mysql.jdbc.Driver";
14    private String url = "jdbc:mysql://localhost/proyecto";
15    private String usuario = "root";
16    private String clave = "xavier1006";
17    private String texto = "";
18
19    public void init(ServletConfig conf) throws ServletException
20    {
21       super.init(conf);
22    }
23
24    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         String id=request.getParameter("id");
27         String passvieja=request.getParameter("passvieja");
28         String nickname=request.getParameter("nickname");
29         String nombre=request.getParameter("nombre");
30         String apellido=request.getParameter("apellido");
31         String email=request.getParameter("email");
32         String pass=request.getParameter("pass");
33         String telefono=request.getParameter("telefono");
34         String direccion=request.getParameter("direccion");
35         String ciudad=request.getParameter("ciudad");
36         String estado=request.getParameter("estado");
37         String pais=request.getParameter("pais");
38         String zip=request.getParameter("zip");
39        try
40       {
41          Class.forName(controlador);
42          conexion = DriverManager.getConnection(url, usuario, clave);
43       }
44       catch (ClassNotFoundException e)
45       {
46          System.err.println("No se puede cargar el controlador JDBC: " + e);
47          System.exit(1);
48       }
49       catch (SQLException e)
50       {
51          System.err.println("No se puede conectar con la base de datos: " + e);
52          if (conexion != null)
53          {
54             try
55             {
56                conexion.close();
57             }
58             catch (SQLException ee)
59             {
60                System.err.println("No puede cerrar la conexion: " + ee);
61             }
62          }
63          System.exit(2);
64       }
65       try
66       {
67                   String consulta = "SELECT * FROM usuarios WHERE id='"+ id +"' and
contrasena='"+passvieja+"';";
68          sentencia = conexion.createStatement();
69          conjuntoResultados = sentencia.executeQuery(consulta);
70          boolean registros = conjuntoResultados.first();
71
72       if (!registros)
73       {
74           texto="error";
75           sentencia.close();
76          response.setContentType("text/plain");
77          response.setContentLength(texto.length());
78          PrintWriter salida = response.getWriter();
```

```
 79          salida.println(texto);
 80       }
 81       else{
 82                   String actualizar = "update usuarios set nickname='"+ nickname +"'
,nombre='"+nombre+"',apellido='"+apellido+"',email='"+email+"',Telefono='"+telefono+"',contrasena
='"+pass+"',direccion='"+direccion+"',ciudad='"+ciudad+"',estado='"+estado+"',pais='"+pais+"',zip
='"+zip+"' where id="+id+";";
 83          sentencia = conexion.createStatement();
 84          sentencia.executeUpdate(actualizar);
 85          sentencia.close();
 86          response.setContentType("text/plain");
 87          texto="OK";
 88          response.setContentLength(texto.length());
 89          PrintWriter salida = response.getWriter();
 90          salida.println(texto);
 91        }
 92
 93       }
 94       catch (SQLException e)
 95       {
 96          System.err.println("Error de SQL: " + e);
 97          if (conexion != null)
 98          {
 99             try
100             {
101                 conexion.close();
102             }
103             catch (SQLException ee)
104             {
105                System.err.println("No puede cerrar la conexion: " + ee);
106             }
107          }
108          System.exit(3);
109       }
110
111       if (conexion != null)
112       {
113          try
114          {
115             conexion.close();
116          }
117          catch (SQLException ee)
118          {
119             System.err.println("No puede cerrar la conexion: " + ee);
120          }
121       }
122     }
123
124    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
125    {
126        processRequest(req,res);
127    }
128
129   public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
130    {
131       processRequest(req,res);;
132    }
133 }
```

ANEXO 12: Código de CargarImagen.

```
 1 package Servlets;
 2
 3 import java.io.File;
 4 import java.io.IOException;
 5 import java.io.PrintWriter;
 6 import java.util.List;
 7 import javax.servlet.ServletException;
 8 import javax.servlet.http.HttpServlet;
 9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12 import org.apache.commons.fileupload.FileItem;
13 import org.apache.commons.fileupload.disk.DiskFileItemFactory;
14 import org.apache.commons.fileupload.servlet.ServletFileUpload;
15
16 public class CargarImagen extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18     private static final String UPLOAD_DIRECTORY = "usuarios";
19     private static final int MEMORY_THRESHOLD   = 1024 * 1024 * 3;  // 3MB
20     private static final int MAX_FILE_SIZE      = 1024 * 1024 * 40; // 40MB
21     private static final int MAX_REQUEST_SIZE   = 1024 * 1024 * 50; // 50MB
22     String id="";
23
24     protected void doPost(HttpServletRequest request,
25             HttpServletResponse response) throws ServletException, IOException {
26         id=request.getParameter("ID");
27         String modo =request.getParameter("modo");
28         HttpSession session = request.getSession(true);
29         String mode =(String) session.getAttribute("modo");
30         if (!ServletFileUpload.isMultipartContent(request)) {
31             PrintWriter writer = response.getWriter();
32             writer.println("Error: el tipo de requerimento debe ser multiparte
enctype=multipart/form-data.");
33             writer.flush();
34             return;
35         }
36         DiskFileItemFactory factory = new DiskFileItemFactory();
37         factory.setSizeThreshold(MEMORY_THRESHOLD);
38         factory.setRepository(new File(System.getProperty("java.io.tmpdir")));
39         ServletFileUpload upload = new ServletFileUpload(factory);
40         upload.setFileSizeMax(MAX_FILE_SIZE);
41         upload.setSizeMax(MAX_REQUEST_SIZE);
42         String uploadPath = getServletContext().getRealPath("") + File.separator +
UPLOAD_DIRECTORY;
43         File uploadDir = new File(uploadPath);
44         if (!uploadDir.exists()) {
45             uploadDir.mkdir();
46         }
47         try {
48             @SuppressWarnings("unchecked")
49             List<FileItem> formItems = upload.parseRequest(request);
50
51             if (formItems != null && formItems.size() > 0) {
52                 for (FileItem item : formItems) {
53                     if (!item.isFormField()) {
54                         String fileName = id + ".jpg";
55                         String filePath = uploadPath + File.separator + fileName;
56                         File storeFile = new File(filePath);
57                         item.write(storeFile);
58                     }
59                 }
60             }
61         } catch (Exception ex) {
62             request.setAttribute("message",
63                     "There was an error: " + ex.getMessage());
64         }
65         if (mode.equals("web")){
66                             response.sendRedirect("Biokituda.jsp");
67                          }
68                         else {
69
response.sendRedirect("http://192.168.10.4:8383/BiokitU/index.html?id="+id);
70                              }
71
72     }
73 }
```

ANEXO 13: Código de ComUsuario.

```
1  package Servlets;
2
3  import java.io.*;
4  import java.sql.*;
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7
8  public class ComUsuario extends HttpServlet
9  {
10    private Connection conexion = null;
11    private Statement sentencia = null;
12    private ResultSet conjuntoResultados = null;
13    private final String controlador = "com.mysql.jdbc.Driver";
14    private final String url = "jdbc:mysql://localhost/proyecto";
15    private final String usuario = "root";
16    private final String clave = "xavier1006";
17    private String texto = "";
18    String user="";
19    String id="";
20
21    @Override
22    public void init(ServletConfig conf) throws ServletException
23    {
24       super.init(conf);
25    }
26
27    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28            throws ServletException, IOException {
29        user=request.getParameter("user");
30        try
31       {
32          Class.forName(controlador);
33          conexion = DriverManager.getConnection(url, usuario, clave);
34       }
35       catch (ClassNotFoundException e)
36       {
37          System.err.println("No se puede cargar el controlador JDBC: " + e);
38          System.exit(1);
39       }
40       catch (SQLException e)
41       {
42          System.err.println("No se puede conectar con la base de datos: " + e);
43          if (conexion != null)
44          {
45             try
46             {
47                conexion.close();
48             }
49             catch (SQLException ee)
50             {
51                System.err.println("No puede cerrar la conexion: " + ee);
52             }
53          }
54          System.exit(2);
55       }
56       try
57       {
58          String consulta = "SELECT nickname FROM usuarios WHERE nickname='"+ user +"';";
59          sentencia = conexion.createStatement();
60          conjuntoResultados = sentencia.executeQuery(consulta);
61          boolean registros = conjuntoResultados.first();
62          if (!registros)
63             {
64                 texto="ok";
65             }
66          else{
67                 conjuntoResultados.beforeFirst();
68                 texto="error";
69              }
70          sentencia.close();
71          response.setContentType("text/plain");
72          response.setContentLength(texto.length());
73          PrintWriter salida = response.getWriter();
74          salida.println(texto);
75       }
76       catch (SQLException e)
77       {
78          System.err.println("Error de SQL: " + e);
79          if (conexion != null)
```

```
80          {
81             try
82             {
83                 conexion.close();
84             }
85             catch (SQLException ee)
86             {
87                System.err.println("No puede cerrar la conexion: " + ee);
88             }
89          }
90          System.exit(3);
91       }
92
93       if (conexion != null)
94       {
95          try
96          {
97             conexion.close();
98          }
99          catch (SQLException ee)
100         {
101            System.err.println("No puede cerrar la conexion: " + ee);
102         }
103      }
104    }
105
106    @Override
107    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
108    {
109        processRequest(req,res);
110    }
111
112    private void mostrarResultados(ResultSet conjuntoResultados) throws SQLException
113    {
114       boolean registros = conjuntoResultados.next();
115
116       if (!registros)
117       {
118          System.out.println("No hay datos");
119          return;
120       }
121
122       try
123       {
124          do
125          {
126             porFila(conjuntoResultados);
127          } while (conjuntoResultados.next());
128       }
129       catch (SQLException e)
130       {
131          System.err.println("Error al mostrar los datos: " + e);
132          if (conexion != null)
133          {
134             try
135             {
136                conexion.close();
137             }
138             catch (SQLException ee)
139             {
140                System.err.println("No puede cerrar la conexion: " + ee);
141             }
142          }
143          System.exit(4);
144       }
145    }
146
147    private void porFila(ResultSet conjuntoResultados) throws SQLException
148    {
149
150         id = conjuntoResultados.getString(1);
151         String nombre = conjuntoResultados.getString(2);
152
153         texto = nombre;
154    }
155
156    @Override
157    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
158    {
159       processRequest(req,res);
160    }}
```

ANEXO 14: Código de CompEmail.

```
1  package Servlets;
2
3  import java.io.*;
4  import java.sql.*;
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7
8  public class CompEmail extends HttpServlet
9  {
10    private Connection conexion = null;
11    private Statement sentencia = null;
12    private ResultSet conjuntoResultados = null;
13    private final String controlador = "com.mysql.jdbc.Driver";
14    private final String url = "jdbc:mysql://localhost/proyecto";
15    private final String usuario = "root";
16    private final String clave = "xavier1006";
17    private String texto = "";
18    String email="";
19    String id="";
20
21    public void init(ServletConfig conf) throws ServletException
22    {
23       super.init(conf);
24    }
25
26    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27             throws ServletException, IOException {
28        email=request.getParameter("email");
29        try
30       {
31          Class.forName(controlador);
32          conexion = DriverManager.getConnection(url, usuario, clave);
33       }
34       catch (ClassNotFoundException e)
35       {
36          System.err.println("No se puede cargar el controlador JDBC: " + e);
37          System.exit(1);
38       }
39       catch (SQLException e)
40       {
41          System.err.println("No se puede conectar con la base de datos: " + e);
42          if (conexion != null)
43          {
44             try
45             {
46                conexion.close();
47             }
48             catch (SQLException ee)
49             {
50                System.err.println("No puede cerrar la conexion: " + ee);
51             }
52          }
53          System.exit(2);
54       }
55       try
56       {
57          String consulta = "SELECT email FROM usuarios WHERE email='"+ email +"';";
58          sentencia = conexion.createStatement();
59          conjuntoResultados = sentencia.executeQuery(consulta);
60          boolean registros = conjuntoResultados.first();
61
62          if (!registros)
63             {
64             texto="ok";
65             }
66          else{
67             conjuntoResultados.beforeFirst();
68             texto="error";
69             }
70          sentencia.close();
71          response.setContentType("text/plain");
72          response.setContentLength(texto.length());
73          PrintWriter salida = response.getWriter();
74          salida.println(texto);
75       }
76       catch (SQLException e)
77       {
78          System.err.println("Error de SQL: " + e);
79          if (conexion != null)
```

```
80          {
81             try
82             {
83                 conexion.close();
84             }
85             catch (SQLException ee)
86             {
87                System.err.println("No puede cerrar la conexion: " + ee);
88             }
89          }
90          System.exit(3);
91       }
92
93       if (conexion != null)
94       {
95          try
96          {
97             conexion.close();
98          }
99          catch (SQLException ee)
100         {
101            System.err.println("No puede cerrar la conexion: " + ee);
102         }
103      }
104    }
105
106    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
107    {
108        processRequest(req,res);
109    }
110
111    private void mostrarResultados(ResultSet conjuntoResultados) throws SQLException
112    {
113       boolean registros = conjuntoResultados.next();
114
115       if (!registros)
116       {
117          System.out.println("No hay datos");
118          return;
119       }
120
121       try
122       {
123          do
124          {
125             porFila(conjuntoResultados);
126          } while (conjuntoResultados.next());
127       }
128       catch (SQLException e)
129       {
130          System.err.println("Error al mostrar los datos: " + e);
131          if (conexion != null)
132          {
133             try
134             {
135                conexion.close();
136             }
137             catch (SQLException ee)
138             {
139                System.err.println("No puede cerrar la conexion: " + ee);
140             }
141          }
142          System.exit(4);
143       }
144    }
145
146    private void porFila(ResultSet conjuntoResultados) throws SQLException
147    {
148         id = conjuntoResultados.getString(1);
149         String nombre = conjuntoResultados.getString(2);
150         texto = nombre;
151    }
152
153    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
154    {
155       processRequest(req,res);;
156    }
157 }
```

ANEXO 15: Código de GrabarECG.

```
1  package Servlets;
2
3
4  import java.io.*;
5  import java.sql.*;
6  import javax.servlet.*;
7  import javax.servlet.http.*;
8  import java.util.StringTokenizer;
9
10 public class GrabarECG extends HttpServlet
11 {
12    private Connection conexion = null;
13    private Statement sentencia = null;
14    private ResultSet conjuntoResultados = null;
15    private String controlador = "com.mysql.jdbc.Driver";
16    private String url = "jdbc:mysql://localhost/proyecto";
17    private String usuario = "root";
18    private String clave = "xavier1006";
19    private String texto = "";
20    String datos="";
21    String id="";
22    String [] puntos;
23    double[] ECGI;
24    double[] ECGII;
25    double[] ECGIII;
26    int numPuntos=0;
27    int punto=0;
28    String fecha="";
29    String hora="";
30    double d=0.0;
31
32    public void init(ServletConfig conf) throws ServletException
33    {
34       super.init(conf);
35    }
36
37    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
38            throws ServletException, IOException {
39         datos=request.getParameter("datos");
40         id=request.getParameter("id");
41         fecha=request.getParameter("fecha");
42         hora=request.getParameter("hora");
43         StringTokenizer tokens = new StringTokenizer(datos);
44          numPuntos=tokens.countTokens();
45          puntos=new String[numPuntos];
46          ECGI=new double[numPuntos];
47          ECGII=new double[numPuntos];
48          ECGIII=new double[numPuntos];
49          punto=0;
50          tokens.nextToken();
51          for(int j=0;j<numPuntos-2;j++){
52             puntos[j]=tokens.nextToken();
53             StringTokenizer tokendos = new StringTokenizer(puntos[j],"*");
54             while(tokendos.hasMoreTokens()){
55                 ECGI[j]=Double.parseDouble(tokendos.nextToken());
56                 ECGII[j]=Double.parseDouble(tokendos.nextToken())+1024;
57                 ECGIII[j]=Double.parseDouble(tokendos.nextToken())+2048;
58             }
60          }
61       try
62       {
63          Class.forName(controlador);
64          conexion = DriverManager.getConnection(url, usuario, clave);
65       }
66       catch (ClassNotFoundException e)
67       {
68          System.err.println("No se puede cargar el controlador JDBC: " + e);
69          System.exit(1);
70       }
71       catch (SQLException e)
72       {
73          System.err.println("No se puede conectar con la base de datos: " + e);
74          if (conexion != null)
75          {
76             try
77             {
78                conexion.close();
79             }
80             catch (SQLException ee)
```

```
 81             {
 82                System.err.println("No puede cerrar la conexion: " + ee);
 83             }
 84          }
 85          System.exit(2);
 86       }
 87       try
 88       {
 89
 90          for(int j=0;j<numPuntos-2;j++){
 91              String consulta = "INSERT INTO ecg (id, ECGI, ECGII, ECGIII, fecha, hora) VALUES
('" + id + "', '"+ ECGI[j] + "', '" + ECGII[j] + "', '"+ ECGIII[j] + "', '"+ fecha + "', '" + hora
+ "');";
 92             sentencia = conexion.createStatement();
 93             sentencia.executeUpdate(consulta);
 94
 95          }
 96          sentencia.close();
 97          response.setContentType("text/plain");
 98          texto="OK";
 99          response.setContentLength(texto.length());
100          PrintWriter salida = response.getWriter();
101          salida.println(texto);
102          //texto = user;
103       }
104       catch (SQLException e)
105       {
106          System.err.println("Error de SQL: " + e);
107          if (conexion != null)
108          {
109             try
110             {
111                 conexion.close();
112             }
113             catch (SQLException ee)
114             {
115                System.err.println("No puede cerrar la conexion: " + ee);
116             }
117          }
118          System.exit(3);
119       }
120
121       if (conexion != null)
122       {
123          try
124          {
125             conexion.close();
126          }
127          catch (SQLException ee)
128          {
129             System.err.println("No puede cerrar la conexion: " + ee);
130          }
131       }
132     }
133
134    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
135    {
136        processRequest(req,res);
137    }
138
139
140    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
141    {
142       processRequest(req,res);;
143    }
144 }
145
146
```

ANEXO 16: Código de GrabarOximetro.

```
 1 package Servlets;
 2
 3 import java.io.*;
 4 import java.sql.*;
 5 import javax.servlet.*;
 6 import javax.servlet.http.*;
 7
 8 public class GrabarOximetro extends HttpServlet {
 9
10     private Connection conexion = null;
11     private Statement sentencia = null;
12     private ResultSet conjuntoResultados = null;
13     private String controlador = "com.mysql.jdbc.Driver";
14     private String url = "jdbc:mysql://localhost/proyecto";
15     private String usuario = "root";
16     private String clave = "xavier1006";
17     private String texto = "";
18     String datos = "";
19     String id = "";
20     String[] puntos;
21     double[] ECGI;
22     double[] ECGII;
23     double[] ECGIII;
24     int numPuntos = 0;
25     int punto = 0;
26     String fecha = "";
27     String hora = "";
28     double d = 0.0;
29     PreparedStatement spPrepararsentecias;
30
31     public void init(ServletConfig conf) throws ServletException {
32         super.init(conf);
33     }
34
35     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
36             throws ServletException, IOException {
37         datos = request.getParameter("oxigeno");
38         id = request.getParameter("id");
39         fecha = request.getParameter("fecha");
40         hora = request.getParameter("hora");
41
42         try {
43             Class.forName(controlador);
44             conexion = DriverManager.getConnection(url, usuario, clave);
45         } catch (ClassNotFoundException e) {
46             System.err.println("No se puede cargar el controlador JDBC: " + e);
47             System.exit(1);
48         } catch (SQLException e) {
49             System.err.println("No se puede conectar con la base de datos: " + e);
50             if (conexion != null) {
51                 try {
52                     conexion.close();
53                 } catch (SQLException ee) {
54                     System.err.println("No puede cerrar la conexion: " + ee);
55                 }
56             }
57             System.exit(2);
58         }
59         try {
60
61                 String sentenciaInsert = "INSERT INTO oximetro (id,datos,fecha,hora) values
(?,?,?,?);";
62
63             //conexion.prepareStatement(sentenciaInsert);
64             spPrepararsentecias = conexion.prepareStatement(sentenciaInsert);
65             spPrepararsentecias.setString(1, id);
66             spPrepararsentecias.setString(2, datos);
67             spPrepararsentecias.setString(3, fecha);
68             spPrepararsentecias.setString(4, hora);
69             spPrepararsentecias.executeUpdate();
70             spPrepararsentecias.close();
71
72             response.setContentType("text/plain");
73             texto = "OK";
74             response.setContentLength(texto.length());
75             PrintWriter salida = response.getWriter();
76             salida.println(texto);
77             //texto = user;
78         } catch (SQLException e) {
```

```
79              System.err.println("Error de SQL: " + e);
80              if (conexion != null) {
81                  try {
82                      conexion.close();
83                  } catch (SQLException ee) {
84                      System.err.println("No puede cerrar la conexion: " + ee);
85                  }
86              }
87              System.exit(3);
88          }
89
90          if (conexion != null) {
91              try {
92                  conexion.close();
93              } catch (SQLException ee) {
94                  System.err.println("No puede cerrar la conexion: " + ee);
95              }
96          }
97      }
98
99      public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
100         processRequest(req, res);
101     }
102
103     public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
104         processRequest(req, res);;
105     }
106 }
107
```

ANEXO 17: Código de GrabarTensiometro.

```
 1 package Servlets;
 2
 3 import java.io.*;
 4 import java.sql.*;
 5 import javax.servlet.*;
 6 import javax.servlet.http.*;
 7
 8 public class GrabarTensiometro extends HttpServlet {
 9
10     private Connection conexion = null;
11     private Statement sentencia = null;
12     private ResultSet conjuntoResultados = null;
13     private String controlador = "com.mysql.jdbc.Driver";
14     private String url = "jdbc:mysql://localhost/proyecto";
15     private String usuario = "root";
16     private String clave = "xavier1006";
17     private String texto = "";
18     String datos = "";
19     String id = "";
20     String[] puntos;
21     double[] ECGI;
22     double[] ECGII;
23     double[] ECGIII;
24     int numPuntos = 0;
25     int punto = 0;
26     String fecha = "";
27     String hora = "";
28     double d = 0.0;
29     PreparedStatement spPrepararsentecias;
30
31     public void init(ServletConfig conf) throws ServletException {
32         super.init(conf);
33     }
34
35     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
36             throws ServletException, IOException {
37         String diastolica = request.getParameter("diastolica");
38         String sistolica = request.getParameter("sistolica");
39         id = request.getParameter("id");
40         fecha = request.getParameter("fecha");
41         hora = request.getParameter("hora");
42
43         try {
44             Class.forName(controlador);
45             conexion = DriverManager.getConnection(url, usuario, clave);
46         } catch (ClassNotFoundException e) {
47             System.err.println("No se puede cargar el controlador JDBC: " + e);
48             System.exit(1);
49         } catch (SQLException e) {
50             System.err.println("No se puede conectar con la base de datos: " + e);
51             if (conexion != null) {
52                 try {
53                     conexion.close();
54                 } catch (SQLException ee) {
55                     System.err.println("No puede cerrar la conexion: " + ee);
56                 }
57             }
58             System.exit(2);
59         }
60         try {
61          String sentenciaInsert = "INSERT INTO tensiometro (id,diastolica,sistolica,fecha,hora)
values (?,?,?,?,?);";
62             spPrepararsentecias = conexion.prepareStatement(sentenciaInsert);
63             spPrepararsentecias.setString(1, id);
64             spPrepararsentecias.setString(2, diastolica);
65             spPrepararsentecias.setString(3, sistolica);
66             spPrepararsentecias.setString(4, fecha);
67             spPrepararsentecias.setString(5, hora);
68             spPrepararsentecias.executeUpdate();
69             spPrepararsentecias.close();
70
71             response.setContentType("text/plain");
72             texto = "OK";
73             response.setContentLength(texto.length());
74             PrintWriter salida = response.getWriter();
75             salida.println(texto);
76         } catch (SQLException e) {
77             System.err.println("Error de SQL: " + e);
78             if (conexion != null) {
```

```
79                 try {
80                     conexion.close();
81                 } catch (SQLException ee) {
82                     System.err.println("No puede cerrar la conexion: " + ee);
83                 }
84             }
85             System.exit(3);
86         }
87 
88         if (conexion != null) {
89             try {
90                 conexion.close();
91             } catch (SQLException ee) {
92                 System.err.println("No puede cerrar la conexion: " + ee);
93             }
94         }
95     }
96 
97     public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
98         processRequest(req, res);
99     }
100 
101    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
102         processRequest(req, res);;
103     }
104 }
105 
```

ANEXO 18: Código de Imag.

```java
package Servlets;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Imag extends HttpServlet {

    String id = "";

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        id = request.getParameter("id");
        ServletContext sc = getServletContext();
        String im = "/usuarios/" + id + ".jpg";
        String filename = sc.getRealPath(im);
        String mimeType = sc.getMimeType(filename);
        if (mimeType == null) {
            sc.log("Could not get MIME type of " + filename);
            response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
            return;
        }
        response.setContentType(mimeType);
        File file = new File(filename);
        response.setContentLength((int) file.length());
        FileInputStream in = new FileInputStream(file);
        OutputStream out = response.getOutputStream();
        byte[] buf = new byte[1024];
        int count = 0;
        while ((count = in.read(buf)) >= 0) {
            out.write(buf, 0, count);
        }
        in.close();
        out.close();
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }/
}
```

ANEXO 13: Código de MostrarE.

```java
package Servlets;

import java.awt.BasicStroke;
import java.awt.*;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import java.sql.SQLException;
import java.util.StringTokenizer;
```

```
22 import org.jfree.chart.axis.NumberAxis;
23 import org.jfree.chart.axis.NumberTickUnit;
24 import org.jfree.chart.axis.TickUnits;
25
26 public class MostrarE extends HttpServlet {
27
28     String datos = "";
29     String[] puntos;
30     static int numPuntos = 0;
31     double[] ECGI;
32     double[] ECGII;
33     double[] ECGIII;
34     static int punto = 0;
35     double d = 0.0;
36
37     private XYDataset generaDatos() throws SQLException {
38
39         XYSeries serie1 = new XYSeries("% ECGI");
40         XYSeries serie2 = new XYSeries("% ECGII");
41         XYSeries serie3 = new XYSeries("% ECGIII");
42         for (int j = 0; j < numPuntos - 2; j++) {
43             serie1.add(j + 1, ECGI[j]);
44             serie2.add(j + 1, ECGII[j]);
45             serie3.add(j + 1, ECGIII[j]);
46         }
47         XYSeriesCollection xyseriescollection = new XYSeriesCollection();
48         xyseriescollection.addSeries(serie1);
49         xyseriescollection.addSeries(serie2);
50         xyseriescollection.addSeries(serie3);
51
52         return xyseriescollection;
53     }
54
55     private static JFreeChart generaGrafico(XYDataset xydataset) throws IOException {
56         JFreeChart jfreechart = ChartFactory.createXYLineChart(
57                 "", "", "",
58                 xydataset, PlotOrientation.VERTICAL,
59                 false, true, false);
60
61         XYPlot xyplot = (XYPlot) jfreechart.getPlot();
62         jfreechart.setBorderPaint(Color.BLACK);
63         jfreechart.setBorderVisible(false);
64         jfreechart.removeLegend();
65         jfreechart.removeLegend();
66         jfreechart.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
67
68         xyplot.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
69         xyplot.getRenderer().setSeriesPaint(0, Color.white);
70         xyplot.getRenderer().setSeriesPaint(1, Color.white);
71         xyplot.getRenderer().setSeriesPaint(2, Color.white);
72         if (numPuntos > 0) {
73             NumberAxis axisdominio = (NumberAxis) xyplot.getDomainAxis();
74             NumberTickUnit ntu2 = new NumberTickUnit(20);
75             NumberTickUnit ntu3 = new NumberTickUnit(20);
76             TickUnits unidades = new TickUnits();
77             unidades.add(ntu2);
78             unidades.add(ntu3);
79             axisdominio.setStandardTickUnits(unidades);
80             axisdominio.setTickLabelsVisible(false);
81             axisdominio.setMinorTickMarksVisible(true);
82             axisdominio.setMinorTickCount(5);
83             xyplot.setDomainAxis(axisdominio);
84                     XYLineAndShapeRenderer xylineandshaperenderer = (XYLineAndShapeRenderer)
xyplot.getRenderer();
85             BasicStroke result = new BasicStroke(2.0f);
86             xylineandshaperenderer.setStroke(result);
87         }
88         NumberAxis axis = (NumberAxis) xyplot.getRangeAxis();
89         axis.setTickLabelsVisible(false);
90         NumberTickUnit ntu = new NumberTickUnit(256);
91         axis.setTickUnit(ntu);
92         axis.setMinorTickMarksVisible(true);
93         axis.setMinorTickCount(5);
94         axis.setTickLabelsVisible(false);
95         xyplot.setRangeAxis(axis);
96         xyplot.setDomainMinorGridlinesVisible(true);
97         xyplot.setRangeMinorGridlinesVisible(true);
98         xyplot.setDomainMinorGridlinePaint(Color.GRAY);
99         xyplot.setRangeMinorGridlinePaint(Color.GRAY);
100        float[] style = {2, 1};
101              xyplot.setRangeMinorGridlineStroke(new BasicStroke(0.5f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
```

```
102             xyplot.setDomainMinorGridlineStroke(new  BasicStroke(0.5f,  BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
103                 xyplot.setDomainGridlineStroke(new  BasicStroke(1.0f,  BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
104                  xyplot.setRangeGridlineStroke(new  BasicStroke(1.0f,  BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
105         xyplot.setDomainGridlinePaint(Color.LIGHT_GRAY);
106         xyplot.setRangeGridlinePaint(Color.LIGHT_GRAY);
107         return jfreechart;
108     }
109 
110      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
111         datos = request.getParameter("datos");
112         StringTokenizer tokens = new StringTokenizer(datos);
113         numPuntos = tokens.countTokens();
114         puntos = new String[numPuntos];
115         ECGI = new double[numPuntos];
116         ECGII = new double[numPuntos];
117         ECGIII = new double[numPuntos];
118         punto = 0;
119         tokens.nextToken();
120         for (int j = 0; j < numPuntos - 2; j++) {
121             puntos[j] = tokens.nextToken();
122             StringTokenizer tokendos = new StringTokenizer(puntos[j], "*");
123             while (tokendos.hasMoreTokens()) {
124                 ECGI[j] = Double.parseDouble(tokendos.nextToken());
125                 ECGII[j] = Double.parseDouble(tokendos.nextToken()) + 1024;
126                 ECGIII[j] = Double.parseDouble(tokendos.nextToken()) + 2048;
127             }
128         }
129         response.setContentType("image/png");
130         OutputStream out = response.getOutputStream();
131         XYDataset dataset = generaDatos();
132         JFreeChart grafico = generaGrafico(dataset);
133         ChartUtilities.writeChartAsPNG(out, grafico, 4801, 801);
134         out.close();
135     }
136 
137     @Override
138     protected void doGet(HttpServletRequest request, HttpServletResponse response)
139             throws ServletException, IOException {
140         try {
141             processRequest(request, response);
142         } catch (SQLException ex) {
143 
144         }
145     }
146 
147     @Override
148     protected void doPost(HttpServletRequest request, HttpServletResponse response)
149             throws ServletException, IOException {
150         try {
151             processRequest(request, response);
152         } catch (SQLException ex) {
153 
154         }
155     }
156 
157     @Override
158     public String getServletInfo() {
159         return "Short description";
160     }
161 }
162 
```

ANEXO 19: Código de MostrarECGSimple.

```
package Servlets;

import java.awt.BasicStroke;
import java.awt.Color;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.axis.TickUnits;

public class MostrarECGSimple extends HttpServlet {

    private Connection conexion = null;
    private String controlador = "com.mysql.jdbc.Driver";
    private String url = "jdbc:mysql://localhost/proyecto";
    private String usuario = "root";
    private String clave = "xavier1006";
    int indice = 0;
    double[] ECGI;
    double[] ECGII;
    double[] ECGIII;
    static String[] fecha;
    int i = 0;
    static int longitud;
    String fechaecg = "";
    String horaecg = "";
    String user = "";

    private void obtenerDatos() throws SQLException {
        try {
            Class.forName(controlador);
            conexion = DriverManager.getConnection(url, usuario, clave);
        } catch (ClassNotFoundException e) {
            System.err.println("No se puede cargar el controlador JDBC: " + e);
            System.exit(1);
        } catch (SQLException e) {
            System.err.println("No se puede conectar con la base de datos: " + e);
            if (conexion != null) {
                try {
                    conexion.close();
                } catch (SQLException ee) {
                    System.err.println("No puede cerrar la conexion: " + ee);
                }
            }
            System.exit(2);
        }

        try {
                    PreparedStatement obtenerDirecciones = conexion.prepareStatement("select
ecgi,ecgii,ecgiii,fecha,hora from ecg where fecha='" + fechaecg + "' and id=" + user + " and hora='"
+ horaecg + "';");
            ResultSet resultado = obtenerDirecciones.executeQuery();
            resultado.last();
            indice = resultado.getRow();
            longitud = indice;
            resultado.first();
            ECGI = new double[indice + 1];
            ECGII = new double[indice + 1];
            ECGIII = new double[indice + 1];
            fecha = new String[indice + 1];
            i = 0;
```

```
78              while (resultado.next()) {
79                  ECGI[i] = Double.parseDouble(resultado.getString(1));
80                  ECGII[i] = Double.parseDouble(resultado.getString(2));
81                  ECGIII[i] = Double.parseDouble(resultado.getString(3));
82                  fecha[i] = resultado.getString(4) + "\n" + resultado.getString(5);
83                  i = i + 1;
84              }
85          } finally {
86              conexion.close();
87          }
88      }
89
90      private XYDataset generaDatos() throws SQLException {
91          obtenerDatos();
92          XYSeries serie1 = new XYSeries("% ECGI");
93          XYSeries serie2 = new XYSeries("% ECGII");
94          XYSeries serie3 = new XYSeries("% ECGIII");
95          for (int j = 0; j < indice - 1; j++) {
96              serie1.add(j + 1, ECGI[j]);
97              serie2.add(j + 1, ECGII[j]);
98              serie3.add(j + 1, ECGIII[j]);
99          }
100         XYSeriesCollection xyseriescollection = new XYSeriesCollection();
101         xyseriescollection.addSeries(serie1);
102         xyseriescollection.addSeries(serie2);
103         xyseriescollection.addSeries(serie3);
104         return xyseriescollection;
105     }
106
107     private static JFreeChart generaGrafico(XYDataset xydataset) {
108         JFreeChart jfreechart = ChartFactory.createXYLineChart(
109                 "", "", "",
110                 xydataset, PlotOrientation.VERTICAL,
111                 false, true, false);
112         XYPlot xyplot = (XYPlot) jfreechart.getPlot();
113         jfreechart.setBorderPaint(Color.BLACK);
114         jfreechart.setBorderVisible(false);
115         jfreechart.removeLegend();
116         xyplot.setBackgroundPaint(Color.BLACK);
117         xyplot.getRenderer().setSeriesPaint(0, Color.white);
118         xyplot.getRenderer().setSeriesPaint(1, Color.white);
119         xyplot.getRenderer().setSeriesPaint(2, Color.white);
120         if (longitud > 0) {
121             NumberAxis axisdominio = (NumberAxis) xyplot.getDomainAxis();
122             NumberTickUnit ntu2 = new NumberTickUnit(20);
123             NumberTickUnit ntu3 = new NumberTickUnit(20);
124             TickUnits unidades = new TickUnits();
125             unidades.add(ntu2);
126             unidades.add(ntu3);
127             axisdominio.setStandardTickUnits(unidades);
128             axisdominio.setTickLabelsVisible(false);
129             axisdominio.setMinorTickMarksVisible(true);
130             axisdominio.setMinorTickCount(5);
131             xyplot.setDomainAxis(axisdominio);
132                     XYLineAndShapeRenderer xylineandshaperenderer = (XYLineAndShapeRenderer)
xyplot.getRenderer();
133             BasicStroke result = new BasicStroke(2.0f);
134             xylineandshaperenderer.setStroke(result);
135         }
136         NumberAxis axis = (NumberAxis) xyplot.getRangeAxis();
137         axis.setTickLabelsVisible(false);
138         NumberTickUnit ntu = new NumberTickUnit(256);
139         axis.setTickUnit(ntu);
140         axis.setMinorTickMarksVisible(true);
141         axis.setMinorTickCount(5);
142         axis.setTickLabelsVisible(false);
143         xyplot.setRangeAxis(axis);
144         xyplot.setDomainMinorGridlinesVisible(true);
145         xyplot.setRangeMinorGridlinesVisible(true);
146         xyplot.setDomainMinorGridlinePaint(Color.GRAY);
147         xyplot.setRangeMinorGridlinePaint(Color.GRAY);
148         float[] style = {2, 1};
149               xyplot.setRangeMinorGridlineStroke(new BasicStroke(0.5f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
150              xyplot.setDomainMinorGridlineStroke(new BasicStroke(0.5f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
151                   xyplot.setDomainGridlineStroke(new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
152                    xyplot.setRangeGridlineStroke(new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 5.0f, style, 0.0f));
153         xyplot.setDomainGridlinePaint(Color.LIGHT_GRAY);
154         xyplot.setRangeGridlinePaint(Color.LIGHT_GRAY);
155         return jfreechart;
```

```
156     }
157 
158     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
159             throws ServletException, IOException, SQLException {
160         i = 0;
161         longitud = 0;
162         indice = 0;
163         fechaecg = request.getParameter("Fecha");
164         horaecg = request.getParameter("Hora");
165         user = request.getParameter("User");
166         response.setContentType("image/png");
167         OutputStream out = response.getOutputStream();
168         XYDataset dataset = generaDatos();
169         JFreeChart grafico = generaGrafico(dataset);
170         ChartUtilities.writeChartAsPNG(out, grafico, 4325, 435);
171         out.close();
172     }
173 
174     @Override
175     protected void doGet(HttpServletRequest request, HttpServletResponse response)
176             throws ServletException, IOException {
177         try {
178             processRequest(request, response);
179         } catch (SQLException ex) {
180 
181         }
182     }
183 
184     @Override
185     protected void doPost(HttpServletRequest request, HttpServletResponse response)
186             throws ServletException, IOException {
187         try {
188             processRequest(request, response);
189         } catch (SQLException ex) {
190 
191         }
192     }
193 
194     @Override
195     public String getServletInfo() {
196         return "Short description";
197     }
198 }
199 
```

ANEXO 20: Código de MostrarOximetro.

```java
package Servlets;

import java.awt.*;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import java.sql.SQLException;
import org.jfree.chart.labels.StandardPieSectionLabelGenerator;
import org.jfree.chart.plot.PiePlot;
import org.jfree.data.general.DefaultPieDataset;

public class MostrarOximetro extends HttpServlet {

      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
        String oxigeno = request.getParameter("oxigeno");
        DefaultPieDataset pieDataset = new DefaultPieDataset();
        pieDataset.setValue("Oxigeno", Integer.parseInt(oxigeno));
        pieDataset.setValue("Carbono", 100 - Integer.parseInt(oxigeno));
        JFreeChart chart = ChartFactory.createPieChart("Nivel de SaO2 en la Sangre",
                pieDataset,
                false,
                true,
                false
        );

        PiePlot plot = (PiePlot) chart.getPlot();
        chart.getTitle().setPaint(Color.WHITE);
        chart.getTitle().setFont(new Font("Arial", Font.BOLD, 50));
        plot.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        chart.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        plot.setOutlineVisible(false);
        plot.setSectionPaint(0, Color.LIGHT_GRAY);
        plot.setSectionPaint(1, Color.GRAY);
        plot.setLabelFont(new Font("Arial", Font.BOLD, 25));
        plot.setLabelLinkStroke(new BasicStroke(4f));
        plot.setLabelLinkPaint(Color.WHITE);
        plot.setLabelBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        plot.setLabelPaint(Color.WHITE);
        plot.setLabelShadowPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        plot.setLabelOutlinePaint(new Color(0xFF, 0xFF, 0xFF, 0));
        plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0} {2}"));
        plot.setExplodePercent(0, 0.3);
        plot.setShadowPaint(Color.DARK_GRAY);
        plot.setInteriorGap(0.0);
        plot.setCircular(true);
        try {
            response.setContentType("image/png");
            OutputStream out = response.getOutputStream();
            ChartUtilities.writeChartAsPNG(out, chart, 800, 800);
            out.close();
        } catch (IOException e) {
            System.err.println("Problem occurred creating chart.");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {

        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
```

ANEXO 21: Código de MostrarOximetroWeb.

```
 1 package Servlets;
 2
 3 import java.awt.*;
 4 import java.io.PrintWriter;
 5 import java.sql.Connection;
 6 import java.sql.DriverManager;
 7 import java.sql.ResultSet;
 8 import java.sql.Statement;
 9 import java.io.IOException;
10 import java.io.OutputStream;
11 import javax.servlet.ServletException;
12 import javax.servlet.http.HttpServlet;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15 import org.jfree.chart.ChartFactory;
16 import org.jfree.chart.ChartUtilities;
17 import org.jfree.chart.JFreeChart;
18 import java.sql.SQLException;
19 import org.jfree.chart.labels.StandardPieSectionLabelGenerator;
20 import org.jfree.chart.plot.PiePlot;
21 import org.jfree.data.general.DefaultPieDataset;
22
23 public class MostrarOximetroWeb extends HttpServlet {
24
25     private Connection conexion = null;
26     private Statement sentencia = null;
27     private ResultSet conjuntoResultados = null;
28     private final String controlador = "com.mysql.jdbc.Driver";
29     private final String url = "jdbc:mysql://localhost/proyecto";
30     private final String usuario = "root";
31     private final String clave = "xavier1006";
32     private String texto = "";
33     String datos = "";
34
35      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
36         String id = request.getParameter("id");
37         String fecha = request.getParameter("fecha");
38         String hora = request.getParameter("hora");
39         try {
40             Class.forName(controlador);
41             conexion = DriverManager.getConnection(url, usuario, clave);
42         } catch (ClassNotFoundException e) {
43             System.err.println("No se puede cargar el controlador JDBC: " + e);
44             System.exit(1);
45         } catch (SQLException e) {
46             System.err.println("No se puede conectar con la base de datos: " + e);
47             if (conexion != null) {
48                 try {
49                     conexion.close();
50                 } catch (SQLException ee) {
51                     System.err.println("No puede cerrar la conexion: " + ee);
52                 }
53             }
54             System.exit(2);
55         }
56         try {
57             String consulta = "SELECT datos from oximetro where id='" + id + "' and fecha='" +
fecha + "' and hora='" + hora + "';";
58             sentencia = conexion.createStatement();
59             conjuntoResultados = sentencia.executeQuery(consulta);
60             boolean registros = conjuntoResultados.first();
61             if (!registros) {
62                 texto = "error";
63                 sentencia.close();
64                 response.setContentType("text/plain");
65                 response.setContentLength(texto.length());
66                 PrintWriter salida = response.getWriter();
67                 salida.println(texto);
68             } else {
69                 conjuntoResultados.beforeFirst();
70                 conjuntoResultados.next();
71                 String oxigeno = conjuntoResultados.getString(1);
72                 DefaultPieDataset pieDataset = new DefaultPieDataset();
73                 pieDataset.setValue("Oxigeno", Integer.parseInt(oxigeno));
74                 pieDataset.setValue("Carbono", 100 - Integer.parseInt(oxigeno));
75                 JFreeChart chart = ChartFactory.createPieChart("",
76                         pieDataset,
77                         false,
```

```
78                          true,
79                          false
80                  );
81                  PiePlot plot = (PiePlot) chart.getPlot();
82                  plot.setBackgroundPaint(Color.BLACK);
83                  chart.setBackgroundPaint(Color.BLACK);
84                  plot.setOutlineVisible(false);
85                  plot.setSectionPaint(0, Color.LIGHT_GRAY);
86                  plot.setSectionPaint(1, Color.GRAY);
87                  plot.setShadowXOffset(0D);
88                  plot.setLabelFont(new Font("Arial", Font.BOLD, 25));
89                  plot.setLabelLinkStroke(new BasicStroke(4f));
90                  plot.setLabelLinkPaint(Color.WHITE);
91                  plot.setLabelBackgroundPaint(Color.BLACK);
92                  plot.setLabelPaint(Color.WHITE);
93                  plot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0} {2}"));
94                  plot.setExplodePercent(0, 0.3);
95                  plot.setInteriorGap(0.01);
96                  try {
97                      response.setContentType("image/png");
98                      OutputStream out = response.getOutputStream();
99                      ChartUtilities.writeChartAsPNG(out, chart, 1200, 430);
100                     out.close();
101                 } catch (IOException e) {
102                     System.err.println("Error");
103                 }
104
105             }
106
107         } catch (SQLException e) {
108             System.err.println("Error de SQL: " + e);
109             if (conexion != null) {
110                 try {
111                     conexion.close();
112                 } catch (SQLException ee) {
113                     System.err.println("No puede cerrar la conexion: " + ee);
114                 }
115             }
116             System.exit(3);
117         }
118
119         if (conexion != null) {
120             try {
121                 conexion.close();
122             } catch (SQLException ee) {
123                 System.err.println("No puede cerrar la conexion: " + ee);
124             }
125         }
126
127     }
128
129     @Override
130     protected void doGet(HttpServletRequest request, HttpServletResponse response)
131             throws ServletException, IOException {
132         try {
133             processRequest(request, response);
134         } catch (SQLException ex) {
135
136         }
137     }
138
139     @Override
140     protected void doPost(HttpServletRequest request, HttpServletResponse response)
141             throws ServletException, IOException {
142         try {
143             processRequest(request, response);
144         } catch (SQLException ex) {
145
146         }
147     }
148
149     @Override
150     public String getServletInfo() {
151         return "Short description";
152     }
153 }
154
```

ANEXO 22: Código de MostrarTensiometro.

```
package Servlets;

import java.awt.*;
import java.io.IOException;
import java.io.OutputStream;
import java.sql.SQLException;
import java.text.DecimalFormat;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.axis.TickUnits;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.RectangleInsets;

public class MostrarTensiometro extends HttpServlet {
    String datos = "";
    double[] puntos;
    int numPuntos = 0;
    int punto = 0;
    double d = 0.0;
    static Image imagen = Toolkit.getDefaultToolkit().getImage("chart.png");

     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
        String diastolica = request.getParameter("diastolica");
        String sistolica = request.getParameter("sistolica");
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.setValue(Integer.parseInt(diastolica), "Diastolica", "Diastolica");
        dataset.setValue(Integer.parseInt(sistolica), "Sistolica", "Sistolica");
        JFreeChart chart = ChartFactory.createBarChart("Nivel de Presion Sanguinea",
                "", "mmhg", dataset, PlotOrientation.VERTICAL, false, true, false);
        chart.setBackgroundPaint(Color.BLACK);
        chart.setBorderPaint(Color.BLACK);
        chart.getTitle().setPaint(Color.WHITE);
        chart.getTitle().setFont(new Font("Arial", Font.BOLD, 50));
        CategoryPlot plot = chart.getCategoryPlot();
        plot.setOutlineVisible(false);
        plot.setAxisOffset(new RectangleInsets(0D, 0D, 0D, 0D));
        plot.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        chart.setBackgroundPaint(new Color(0xFF, 0xFF, 0xFF, 0));
        BarRenderer renderer = (BarRenderer) plot.getRenderer();
        renderer.setBarPainter(new CustomBarPainter());
        renderer.setShadowVisible(false);
          GradientPaint gradientGray = new GradientPaint(0.0F, 0.0F, Color.LIGHT_GRAY, 0.0F,
0.0F, Color.LIGHT_GRAY);
        renderer.setSeriesPaint(0, gradientGray);
          GradientPaint gradientRed = new GradientPaint(0.0F, 0.0F, Color.GRAY, 0.0F, 0.0F,
Color.GRAY);
        renderer.setSeriesPaint(1, gradientRed);
        renderer.setItemMargin(-0.6D);
        renderer.setBaseItemLabelsVisible(true);
         renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGenerator("{2}", new
DecimalFormat("#0.0#")));
        renderer.setBaseItemLabelFont(new Font("Arial", Font.BOLD, 40));
        renderer.setBaseItemLabelPaint(Color.WHITE);
        renderer.setItemMargin(-1);
        plot.setRenderer(renderer);
        ValueAxis rango = plot.getRangeAxis();
        rango.setTickLabelPaint(Color.WHITE);
        rango.setLabelPaint(Color.WHITE);
        rango.setRange(0, 200);
        rango.setLabelFont(new Font("Arial", Font.BOLD, 25));
        rango.setTickLabelFont(new Font("Arial", Font.BOLD, 25));
        NumberTickUnit ntu2 = new NumberTickUnit(20);
        TickUnits unidades = new TickUnits();
        unidades.add(ntu2);
        rango.setStandardTickUnits(unidades);
        rango.setAxisLineStroke(new BasicStroke(4f));
```

```
76          rango.setAxisLinePaint(Color.WHITE);
77          CategoryAxis cAxis = plot.getDomainAxis();
78          cAxis.setLowerMargin(0.01D);
79          cAxis.setUpperMargin(0.01D);
80          cAxis.setTickMarksVisible(false);
81          cAxis.setAxisLinePaint(Color.white);
82          cAxis.setLabelPaint(Color.WHITE);
83          cAxis.setTickLabelPaint(Color.WHITE);
84          cAxis.setTickLabelFont(new Font("Arial", Font.BOLD, 30));
85          cAxis.setLabelFont(new Font("Arial", Font.BOLD, 25));
86          cAxis.setAxisLineStroke(new BasicStroke(4f));
87          plot.setRangeGridlinePaint(Color.WHITE);
88          plot.setRangeGridlineStroke(new BasicStroke(2f));
89          plot.setDomainGridlinePaint(Color.WHITE);
90          plot.setDomainGridlineStroke(new BasicStroke(2f));
91          try {
92              response.setContentType("image/png");
93              OutputStream out = response.getOutputStream();
94              ChartUtilities.writeChartAsPNG(out, chart, 800, 800);
95              out.close();
96          } catch (IOException e) {
97              System.err.println("Problem occurred creating chart.");
98          }
99      }
100
101     @Override
102     protected void doGet(HttpServletRequest request, HttpServletResponse response)
103             throws ServletException, IOException {
104         try {
105             processRequest(request, response);
106         } catch (SQLException ex) {
107
108         }
109     }
110
111     @Override
112     protected void doPost(HttpServletRequest request, HttpServletResponse response)
113             throws ServletException, IOException {
114         try {
115             processRequest(request, response);
116         } catch (SQLException ex) {
117
118         }
119     }
120
121     @Override
122     public String getServletInfo() {
123         return "Short description";
124     }
125 }
```

ANEXO 23: Código de MostrarTensiometroWeb.

```
 1 package Servlets;
 2
 3 import java.awt.*;
 4 import java.io.IOException;
 5 import java.io.OutputStream;
 6 import java.io.PrintWriter;
 7 import java.sql.Connection;
 8 import java.sql.DriverManager;
 9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import java.sql.Statement;
12 import java.text.DecimalFormat;
13 import javax.servlet.ServletException;
14 import javax.servlet.http.HttpServlet;
15 import javax.servlet.http.HttpServletRequest;
16 import javax.servlet.http.HttpServletResponse;
17 import org.jfree.chart.ChartFactory;
18 import org.jfree.chart.ChartUtilities;
19 import org.jfree.chart.JFreeChart;
20 import org.jfree.chart.axis.CategoryAxis;
21 import org.jfree.chart.axis.NumberTickUnit;
22 import org.jfree.chart.axis.TickUnits;
23 import org.jfree.chart.axis.ValueAxis;
24 import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
25 import org.jfree.chart.plot.CategoryPlot;
26 import org.jfree.chart.plot.PlotOrientation;
27 import org.jfree.chart.renderer.category.BarRenderer;
28 import org.jfree.data.category.DefaultCategoryDataset;
29 import org.jfree.ui.RectangleInsets;
30
31 public class MostrarTensiometroWeb extends HttpServlet {
32
33     private Connection conexion = null;
34     private Statement sentencia = null;
35     private ResultSet conjuntoResultados = null;
36     private final String controlador = "com.mysql.jdbc.Driver";
37     private final String url = "jdbc:mysql://localhost/proyecto";
38     private final String usuario = "root";
39     private final String clave = "xavier1006";
40     private String texto = "";
41     String datos = "";
42
43      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, SQLException {
44         String id = request.getParameter("id");
45         String fecha = request.getParameter("fecha");
46         String hora = request.getParameter("hora");
47         try {
48             Class.forName(controlador);
49             conexion = DriverManager.getConnection(url, usuario, clave);
50         } catch (ClassNotFoundException e) {
51             System.err.println("No se puede cargar el controlador JDBC: " + e);
52             System.exit(1);
53         } catch (SQLException e) {
54             System.err.println("No se puede conectar con la base de datos: " + e);
55             if (conexion != null) {
56                 try {
57                     conexion.close();
58                 } catch (SQLException ee) {
59                     System.err.println("No puede cerrar la conexion: " + ee);
60                 }
61             }
62             System.exit(2);
63         }
64         try {
65             String consulta = "SELECT diastolica, sistolica from tensiometro where id='" + id
+ "' and fecha='" + fecha + "' and hora='" + hora + "';";
66             sentencia = conexion.createStatement();
67             conjuntoResultados = sentencia.executeQuery(consulta);
68             boolean registros = conjuntoResultados.first();
69             if (!registros) {
70                 texto = "error";
71                 sentencia.close();
72                 response.setContentType("text/plain");
73                 response.setContentLength(texto.length());
74                 PrintWriter salida = response.getWriter();
75                 salida.println(texto);
76             } else {
77                 conjuntoResultados.beforeFirst();
```

```
78                conjuntoResultados.next();
79                String diastolica = conjuntoResultados.getString(1);
80                String sistolica = conjuntoResultados.getString(2);
81                DefaultCategoryDataset dataset = new DefaultCategoryDataset();
82                dataset.setValue(Integer.parseInt(diastolica), "Diastolica", "Diastolica");
83                dataset.setValue(Integer.parseInt(sistolica), "Sistolica", "Sistolica");
84                JFreeChart chart = ChartFactory.createBarChart("",
85                        "", "mmhg", dataset, PlotOrientation.VERTICAL, false, true, false);
86                chart.setBackgroundPaint(Color.BLACK);
87                chart.setBorderPaint(Color.BLACK);
88                CategoryPlot plot = chart.getCategoryPlot();
89                plot.setOutlineVisible(false);
90                plot.setAxisOffset(new RectangleInsets(0D, 0D, 0D, 0D));
91                BarRenderer renderer = (BarRenderer) plot.getRenderer();
92                renderer.setBarPainter(new CustomBarPainter());
93                renderer.setShadowVisible(false);
94                 GradientPaint gradientGray = new GradientPaint(0.0F, 0.0F, Color.LIGHT_GRAY,
0.0F, 0.0F, Color.LIGHT_GRAY);
95                renderer.setSeriesPaint(0, gradientGray);
96                  GradientPaint gradientRed = new GradientPaint(0.0F, 0.0F, Color.LIGHT_GRAY,
0.0F, 0.0F, Color.LIGHT_GRAY);
97                renderer.setSeriesPaint(1, gradientRed);
98                renderer.setItemMargin(-0.6D);
99                renderer.setBaseItemLabelsVisible(true);
100             renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGenerator("{2}",
new DecimalFormat("#0.0#")));
101               renderer.setBaseItemLabelFont(new Font("Arial", Font.BOLD, 25));
102               renderer.setBaseItemLabelPaint(Color.WHITE);
103               renderer.setItemMargin(-1);
104               plot.setRenderer(renderer);
105               ValueAxis rango = plot.getRangeAxis();
106               rango.setTickLabelPaint(Color.WHITE);
107               rango.setLabelPaint(Color.WHITE);
108               rango.setRange(0, 200);
109               rango.setLabelFont(new Font("Arial", Font.BOLD, 25));
110               rango.setAxisLineStroke(new BasicStroke(4f));
111               rango.setAxisLinePaint(Color.WHITE);
112               rango.setTickLabelFont(new Font("Arial", Font.BOLD, 25));
113               NumberTickUnit ntu2 = new NumberTickUnit(20);
114               TickUnits unidades = new TickUnits();
115               unidades.add(ntu2);
116               rango.setStandardTickUnits(unidades);
117               CategoryAxis cAxis = plot.getDomainAxis();
118               cAxis.setLowerMargin(0.01D);
119               cAxis.setUpperMargin(0.01D);
120               cAxis.setTickMarksVisible(false);
121               cAxis.setAxisLinePaint(Color.white);
122               cAxis.setLabelPaint(Color.WHITE);
123               cAxis.setTickLabelPaint(Color.WHITE);
124               cAxis.setTickLabelFont(new Font("Arial", Font.BOLD, 30));
125               cAxis.setLabelFont(new Font("Arial", Font.BOLD, 25));
126               cAxis.setAxisLineStroke(new BasicStroke(4f));
127               plot.setBackgroundPaint(Color.BLACK);
128               plot.setRangeGridlinePaint(Color.WHITE);
129               plot.setDomainGridlinePaint(Color.WHITE);
130               plot.setRangeGridlinePaint(Color.WHITE);
131               plot.setRangeGridlineStroke(new BasicStroke(2f));
132               plot.setDomainGridlinePaint(Color.WHITE);
133               plot.setDomainGridlineStroke(new BasicStroke(2f));
134               try {
135                   response.setContentType("image/png");
136                   OutputStream out = response.getOutputStream();
137                   ChartUtilities.writeChartAsPNG(out, chart, 1200, 430);
138                   out.close();
139               } catch (IOException e) {
140                   System.err.println("Problem occurred creating chart.");
141               }
142           }
143       } catch (SQLException e) {
144           System.err.println("Error de SQL: " + e);
145           if (conexion != null) {
146               try {
147                   conexion.close();
148               } catch (SQLException ee) {
149                   System.err.println("No puede cerrar la conexion: " + ee);
150               }
151           }
152           System.exit(3);
153       }
154
155       if (conexion != null) {
156           try {
157               conexion.close();
```

```
158             } catch (SQLException ee) {
159                 System.err.println("No puede cerrar la conexion: " + ee);
160             }
161         }
162 
163     }
164 
165     @Override
166     protected void doGet(HttpServletRequest request, HttpServletResponse response)
167             throws ServletException, IOException {
168         try {
169             processRequest(request, response);
170         } catch (SQLException ex) {
171 
172         }
173     }
174 
175     @Override
176     protected void doPost(HttpServletRequest request, HttpServletResponse response)
177             throws ServletException, IOException {
178         try {
179             processRequest(request, response);
180         } catch (SQLException ex) {
181 
182         }
183     }
184 
185     @Override
186     public String getServletInfo() {
187         return "Short description";
188     }
189 }
```

ANEXO 24: Código de Perfil.

```java
package Servlets;

import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Perfil extends HttpServlet {

    private Connection conexion = null;
    private Statement sentencia = null;
    private ResultSet conjuntoResultados = null;
    private final String controlador = "com.mysql.jdbc.Driver";
    private final String url = "jdbc:mysql://localhost/proyecto";
    private final String usuario = "root";
    private final String clave = "xavier1006";
    private String texto = "";
    String user = "";
    String pass = "";

    public void init(ServletConfig conf) throws ServletException {
        super.init(conf);
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        user = request.getParameter("user");
        try {
            Class.forName(controlador);
            conexion = DriverManager.getConnection(url, usuario, clave);
        } catch (ClassNotFoundException e) {
            System.err.println("No se puede cargar el controlador JDBC: " + e);
            System.exit(1);
        } catch (SQLException e) {
            System.err.println("No se puede conectar con la base de datos: " + e);
            if (conexion != null) {
                try {
                    conexion.close();
                } catch (SQLException ee) {
                    System.err.println("No puede cerrar la conexion: " + ee);
                }
            }
            System.exit(2);
        }
        try {

            String consulta = "SELECT * FROM usuarios WHERE nickname='" + user + "';";
            sentencia = conexion.createStatement();
            conjuntoResultados = sentencia.executeQuery(consulta);
            boolean registros = conjuntoResultados.first();
            if (!registros) {
                texto = "error";
                sentencia.close();
                response.setContentType("text/plain");
                response.setContentLength(texto.length());
                PrintWriter salida = response.getWriter();
                salida.println(texto);
            } else {
                conjuntoResultados.beforeFirst();
                conjuntoResultados.next();
                String id = conjuntoResultados.getString(1);
                String nombre = conjuntoResultados.getString(3);
                String apellido = conjuntoResultados.getString(4);
                String email = conjuntoResultados.getString(5);
                String direccion = conjuntoResultados.getString(7);
                String ciudad = conjuntoResultados.getString(8);
                String estado = conjuntoResultados.getString(9);
                String pais = conjuntoResultados.getString(10);
                String zip = conjuntoResultados.getString(11);
                String telefono = conjuntoResultados.getString(12);
                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out = response.getWriter();
                try {
                    out.println("<!DOCTYPE html>");
                    out.println("<html>");
                    out.println("<head>");
                    out.println("<title>Perfil</title>");
                    out.println("</head>");
                    out.println("<style type='text/css'>");
```

```
 80                     out.println("h1 {\n"
 81                             + " background-color: #000;\n"
 82                             + " color: #FFF;\n"
 83                             + " text-align: center;\n"
 84                             + "}\n" + ".container {\n"
 85                             + "width: 300px;\n"
 86                             + "overflow: hidden;}"
 87                     );
 88                     out.println("body {\n"
 89                             + " font: 100%/1.4 Verdana, Arial, Helvetica, sans-serif;\n"
 90                             + " background-color: #FFF;\n"
 91                             + " margin: 0;\n"
 92                             + " padding: 0;\n"
 93                             + " color: #000;\n"
 94                             + "}");
 95                     out.println("</style>");
 96                     out.println("<body background-color: #000>");
 97                       out.println("<div class='container'> <h1>" + nombre + " " + apellido +
"</h1>");
 98                           out.println("<div align='center'><img src='usuarios/" + id + ".jpg'
style='max-width: 300px; max-height: 300px'/></div>");
 99                     out.println("<p><b>Email: </b>" + email + "</p>");
100                     out.println("<p><b>Telefono: </b>" + telefono + "</p>");
101                     out.println("<p><b>Direccion: </b>" + direccion + "</p>");
102                     out.println("<p><b>Ciudad: </b>" + ciudad + "</p>");
103                     out.println("<p><b>Estado / Provincia: </b>" + estado + "</p>");
104                     out.println("<p><b>Pais: </b>" + pais + "</p>");
105                     out.println("<p><b>Zip Code: </b>" + zip + "</p></div>");
106                     out.println("</body>");
107                     out.println("</html>");
108                 } finally {
109                     out.close();
110                 }
111             }
112         } catch (SQLException e) {
113             System.err.println("Error de SQL: " + e);
114             if (conexion != null) {
115                 try {
116                     conexion.close();
117                 } catch (SQLException ee) {
118                     System.err.println("No puede cerrar la conexion: " + ee);
119                 }
120             }
121             System.exit(3);
122         }
123         if (conexion != null) {
124             try {
125                 conexion.close();
126             } catch (SQLException ee) {
127                 System.err.println("No puede cerrar la conexion: " + ee);
128             }
129         }
130     }
131 
132     public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
133         processRequest(req, res);
134     }
135 
136     public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
137         processRequest(req, res);;
138     }
139 }
140 
```

ANEXO 25: Código de Registro.

```
 1 package Servlets;
 2
 3 import java.io.*;
 4 import java.sql.*;
 5 import javax.servlet.*;
 6 import javax.servlet.http.*;
 7
 8 public class Registro extends HttpServlet {
 9
10     private Connection conexion = null;
11     private Statement sentencia = null;
12     private ResultSet conjuntoResultados = null;
13     private String controlador = "com.mysql.jdbc.Driver";
14     private String url = "jdbc:mysql://localhost/proyecto";
15     private String usuario = "root";
16     private String clave = "xavier1006";
17     private String texto = "";
18     String user = "";
19     String nombre = "";
20     String apellido = "";
21     String email = "";
22     String pass = "";
23     String direccion = "";
24     String ciudad = "";
25     String estado = "";
26     String pais = "";
27     String zip = "";
28     String id = "";
29
30     public void init(ServletConfig conf) throws ServletException {
31         super.init(conf);
32     }
33
34     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
35             throws ServletException, IOException {
36         user = request.getParameter("user");
37         nombre = request.getParameter("nombre");
38         apellido = request.getParameter("apellido");
39         email = request.getParameter("email");
40         pass = request.getParameter("pass");
41         direccion = request.getParameter("direccion");
42         ciudad = request.getParameter("ciudad");
43         estado = request.getParameter("estado");
44         pais = request.getParameter("pais");
45         zip = request.getParameter("zip");
46         try {
47             Class.forName(controlador);
48             conexion = DriverManager.getConnection(url, usuario, clave);
49         } catch (ClassNotFoundException e) {
50             System.err.println("No se puede cargar el controlador JDBC: " + e);
51             System.exit(1);
52         } catch (SQLException e) {
53             System.err.println("No se puede conectar con la base de datos: " + e);
54             if (conexion != null) {
55                 try {
56                     conexion.close();
57                 } catch (SQLException ee) {
58                     System.err.println("No puede cerrar la conexion: " + ee);
59                 }
60             }
61             System.exit(2);
62         }
63         try {
64                 String consulta = "INSERT INTO usuarios (nickname, nombre, apellido, email,
contrasena, direccion, ciudad, estado, pais, zip) VALUES ('" + user + "', '" + nombre + "', '" +
apellido + "', '" + email + "', '" + pass + "', '" + direccion + "', '" + ciudad + "', '" + estado
+ "', '" + pais + "', '" + zip + "');";
65             sentencia = conexion.createStatement();
66             sentencia.executeUpdate(consulta);
67             sentencia.close();
68             response.setContentType("text/plain");
69             texto = "OK";
70             response.setContentLength(texto.length());
71             PrintWriter salida = response.getWriter();
72             salida.println(texto);
73         } catch (SQLException e) {
74             System.err.println("Error de SQL: " + e);
75             if (conexion != null) {
76                 try {
```

```
 77                     conexion.close();
 78                 } catch (SQLException ee) {
 79                     System.err.println("No puede cerrar la conexion: " + ee);
 80                 }
 81             }
 82             System.exit(3);
 83         }
 84 
 85         if (conexion != null) {
 86             try {
 87                 conexion.close();
 88             } catch (SQLException ee) {
 89                 System.err.println("No puede cerrar la conexion: " + ee);
 90             }
 91         }
 92     }
 93 
 94     public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
 95         processRequest(req, res);
 96     }
 97 
 98     public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
 99         processRequest(req, res);;
100     }
101 }
```

ANEXO 26: Código de Usuarios.

```
 1 package Servlets;
 2
 3 import java.io.*;
 4 import java.sql.*;
 5 import javax.servlet.*;
 6 import javax.servlet.http.*;
 7
 8 public class Usuarios extends HttpServlet {
 9
10     private Connection conexion = null;
11     private Statement sentencia = null;
12     private ResultSet conjuntoResultados = null;
13     private final String controlador = "com.mysql.jdbc.Driver";
14     private final String url = "jdbc:mysql://localhost/proyecto";
15     private final String usuario = "root";
16     private final String clave = "xavier1006";
17     private String texto = "";
18     String user = "";
19     String pass = "";
20     String lugar = "";
21
22     public void init(ServletConfig conf) throws ServletException {
23         super.init(conf);
24     }
25
26     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27             throws ServletException, IOException {
28         user = request.getParameter("user");
29         pass = request.getParameter("pass");
30         lugar = request.getParameter("lugar");
31         try {
32             Class.forName(controlador);
33             conexion = DriverManager.getConnection(url, usuario, clave);
34         } catch (ClassNotFoundException e) {
35             System.err.println("No se puede cargar el controlador JDBC: " + e);
36             System.exit(1);
37         } catch (SQLException e) {
38             System.err.println("No se puede conectar con la base de datos: " + e);
39             if (conexion != null) {
40                 try {
41                     conexion.close();
42                 } catch (SQLException ee) {
43                     System.err.println("No puede cerrar la conexion: " + ee);
44                 }
45             }
46             System.exit(2);
47         }
48         try {
49
50              String consulta = "SELECT id, nombre, apellido FROM usuarios WHERE nickname='" +
user + "' and contrasena='" + pass + "';";
51             sentencia = conexion.createStatement();
52             conjuntoResultados = sentencia.executeQuery(consulta);
53             boolean registros = conjuntoResultados.first();
54             if (!registros) {
55                 texto = "error";
56                 sentencia.close();
57                 response.setContentType("text/plain");
58                 response.setContentLength(texto.length());
59                 PrintWriter salida = response.getWriter();
60                 salida.println(texto);
61             } else {
62                 conjuntoResultados.beforeFirst();
63                 conjuntoResultados.next();
64                 String id = conjuntoResultados.getString(1);
65                 String nombre = conjuntoResultados.getString(2);
66                 String apellido = conjuntoResultados.getString(3);
67                 texto = id + " " + nombre + " " + apellido;
68                 if (lugar.equals("web")) {
69                     HttpSession session = request.getSession(true);
70                     session.setAttribute("id", id);
71                     session.setAttribute("nombre", nombre);
72                     session.setAttribute("apellido", apellido);
73                     response.sendRedirect("Medico.jsp");
74                 } else {
75                     sentencia.close();
76                     response.setContentType("text/plain");
77                     response.setContentLength(texto.length());
78                     PrintWriter salida = response.getWriter();
```

```
79                      salida.println(texto);
80                      texto = user;
81                  }
82              }
83          } catch (SQLException e) {
84              System.err.println("Error de SQL: " + e);
85              if (conexion != null) {
86                  try {
87                      conexion.close();
88                  } catch (SQLException ee) {
89                      System.err.println("No puede cerrar la conexion: " + ee);
90                  }
91              }
92              System.exit(3);
93          }
94          if (conexion != null) {
95              try {
96                  conexion.close();
97              } catch (SQLException ee) {
98                  System.err.println("No puede cerrar la conexion: " + ee);
99              }
100         }
101     }
102
103     public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
104         processRequest(req, res);
105     }
106
107     public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
108         processRequest(req, res);;
109     }
110 }
111
```

ANEXO 27: Código de Microcontrolador Arduino.

```
const int N = 40;
int dato = 70;
double C1[N + 1] = { -0.0000000000000000, 0.0000054780433854, 0.0001638824035711,
0.0002376755713784, -0.0007932382647990, -0.0021409097603455, 0.0002468297903606,
0.0065366170430009, 0.0061198647936651, -0.0087803792474312, -0.0207214047914443, -
0.0021242249315332, 0.0340195817953215, 0.0317591716500565, -0.0255076645573009, -
0.0646690766288130, -0.0152322636957401, 0.0695678047014489, 0.0677889686315982, -
0.0304055189085567, 0.9077999999999999, -0.0304055189085567, 0.0677889686315982,
0.0695678047014489, -0.0152322636957401, -0.0646690766288130, -0.0255076645573009,
0.0317591716500565, 0.0340195817953215, -0.0021242249315332, -0.0207214047914443, -
0.0087803792474312, 0.0061198647936651, 0.0065366170430009, 0.0002468297903606, -
0.0021409097603455, -0.0007932382647990, 0.0002376755713784, 0.0001638824035711,
0.0000054780433854, -0.0000000000000000 };
double V1[N + 1], V2[N + 1], V3[N + 1];
double picos[100];
int pres[100];
int Vd1, Vd2, Vd3, Vten, VOxi;
double d1 = 0.0, d2 = 0.0, d3 = 0.0;
double xo = 0.0, xo1 = 0.0, xo2 = 0.0, yo = 0.0, yo1 = 0.0, yo2 = 0.0;
double xt = 0.0, xt1 = 0.0, xt2 = 0.0, yt = 0.0, yt1 = 0.0, yt2 = 0.0;
int cont = 0;
void setup() {
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  for (int i = 0; i <= N; i++) {
    V1[i] = 0;
    V2[i] = 0;
    V3[i] = 0;
  };
  for (int i = 0; i < 100; i++) {
    picos[i] = 0;
    pres[i] = 0;
  };
}

void loop() {
  if (Serial.available() > 0) {
    dato = Serial.read();
  }
  else {
    if (dato == 67) {
      digitalWrite(5, HIGH);
      VOxi = analogRead(A4);
      xo = VOxi * 1.0;
      yo = 0.000435306 * xo + 0.000870613 * xo1 + 0.000435306 * xo2 + 1.94012 * yo1 - 0.941865 *
yo2;
      xo2 = xo1;
      xo1 = xo;
      yo2 = yo1;
      yo1 = yo;
      double porcen = (yo / 400) * 100;
      Serial.println((int) porcen);
    }
    if (dato == 68) {
      double mayor = 0;
      int indi = 0;
      for (int i = 4; i < cont; i++) {
        if (picos[i] > mayor) {
          mayor = picos[i];
          indi = i;
        }
      };
      Serial.println("****");
      Serial.println((int)((double)pres[5] * 0.21941685546875));
      Serial.println((int)((double)(3 * pres[indi] - 2 * pres[5]) * 0.21941685546875));
      Serial.println("****");
      dato = 70;
    }
    else if (dato == 66) {
      Vten = analogRead(A3);
      digitalWrite(3, HIGH);
      digitalWrite(4, HIGH);
      xt = Vten * 1.0;
      yt = 0.0625699 * xt + 0.0 * xt1 - 0.0625699 * xt2 + 1.81304 * yt1 - 0.87486 * yt2;
```

```
      if (yt1 > yt && yt1 > yt2 && (yt * 100) > 70 && xt1 > 20) {
        picos[cont] = yt1;
        pres[cont] = xt1;
        cont ++;
      };
      xt2 = xt1;
      xt1 = xt;
      yt2 = yt1;
      yt1 = yt;
      Serial.println(Vten);
    }
    else if (dato == 65) {
      Vd1 = analogRead(A0);
      Vd2 = analogRead(A1);
      Vd3 = analogRead(A2);
      V1[0] = Vd1 * 1.0;
      V2[0] = Vd2 * 1.0;
      V3[0] = Vd3 * 1.0;
      d1 = 0;
      d2 = 0;
      d3 = 0;
      for (int i = 0; i <= N ; i++) {
        d1 = d1 + V1[i] * C1[i];
        d2 = d2 + V2[i] * C1[i];
        d3 = d3 + V3[i] * C1[i];
      };
      for (int i = N; i >= 1 ; i--) {
        V1[i] = V1[i - 1];
        V2[i] = V2[i - 1];
        V3[i] = V3[i - 1];

      };
      String salida;
      salida = String(String((int)d1) + "*" + String((int)d2) + "*" + String((int)d3));
      Serial.println(salida);
    }
    else {
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      cont = 0;
      xt2 = 0;
      xt1 = 0;
      yt2 = 0;
      yt1 = 0;
    }
  }
}
```