



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E
INSTRUMENTACIÓN

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO EN ELECTRÓNICA E
INSTRUMENTACIÓN

AUTORES: GÓMEZ MORALES OSCAR WLADIMIR
SILVA LUZURIAGA JAIME ISAAC

TEMA: “DISEÑO E IMPLEMENTACIÓN DE UN
POLISOMNOGRAFO CON TRANSMISIÓN DE DATOS
INALÁMBRICOS”

DIRECTOR: PHD. SÁNCHEZ LUIS
CODIRECTORA: ING. GUERRÓN NANCY

LATACUNGA, OCTUBRE 2014

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICADO

PhD. Luis Sánchez (DIRECTOR)

Ing. Nancy Guerrón (CODIRECTORA)

CERTIFICAN

Que el trabajo titulado “DISEÑO E IMPLEMENTACIÓN DE UN POLISOMNÓGRAFO, CON TRANSMISIÓN DE DATOS INALÁMBRICOS”, realizado por los Señores Oscar Wladimir Gómez Morales y Jaime Isaac Silva Luzuriaga, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la Universidad de las Fuerzas Armadas ESPE.

Debido a que constituye un trabajo de excelente contenido científico y aplicable para el desarrollo profesional, SI recomiendan su aplicación.

El mencionado trabajo consta de un documento empastado y un disco compacto el cual contiene los archivos en formato portátil de Acrobat (pdf). Autorizan a los Señores Oscar Wladimir Gómez Morales y Jaime Isaac Silva Luzuriaga, que lo entregue al Ing. Nancy Guerrón, en su calidad de Director de Carrera.

Latacunga, Octubre del 2014

PhD. Luis Sánchez
DIRECTOR

Ing. Nancy Guerrón
CODIRECTORA

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

DECLARACIÓN DE RESPONSABILIDAD

OSCAR WLADIMIR GÓMEZ MORALES

JAIME ISAAC SILVA LUZURIAGA

DECLARAMOS QUE:

El proyecto de grado denominado “DISEÑO E IMPLEMENTACIÓN DE UN POLISOMNÓGRAFO, CON TRANSMISIÓN DE DATOS INALÁMBRICOS”, ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan el pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Octubre del 2014

Oscar Wladimir Gómez Morales

C.C.: 050335862-4

Jaime Isaac Silva Luzuriaga

C.C.: 050261156-9

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros: Oscar Wladimir Gómez Morales
Jaime Isaac Silva Luzuriaga

Autorizamos a la UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE, la publicación, en la biblioteca virtual de la Institución del Trabajo “DISEÑO E IMPLEMENTACIÓN DE UN POLISOMNÓGRAFO, CON TRANSMISIÓN DE DATOS INALÁMBRICOS”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, Octubre 2014

Oscar Wladimir Gómez Morales
C.C.:050335862-4

Jaime Isaac Silva Luzuriaga
C.C.: 050261156-9

DEDICATORIA

Dedico este proyecto de tesis principalmente a Dios y a la Virgen de agua Santa, por darme la vida y sus bendiciones por permitirme concluir un paso más en mi vida profesional. A mis padres, José y Julia por ser el pilar más importante y por demostrarme siempre su amor y apoyo incondicional. A mi hermano Washington por su apoyo cuando sentía desmayar.

A todas y cada una de las personas que ayudaron directa e indirectamente en la culminación de este proyecto.

Oscar.

DEDICATORIA

Dedico este tema de tesis a toda la comunidad politécnica en especial a José Buchelli, Diego Ortiz, Jaqueline Llanos, Nancy Guerrón, quienes de verdad tienen la vocación de ser docentes y no son egoístas con sus conocimientos porque sin ellos no se haría realidad este proyecto. Dedico a los docentes Jaqueline Llanos, Diego Ortiz, Nancy Guerrón, Julio Acosta y David Rivas que son una inspiración para mi persona a seguir en la investigación científica. Quiero reconocer que sin el apoyo incondicional de mi familia este logro no hubiera sido posible, ya que todos fueron esa base para poder hacer este sueño una realidad.

Isaac

AGRADECIMIENTO

Deseo expresar mis más profundos agradecimiento a Dios y a la Virgen de agua Santa, por las bendiciones recibidas en el transcurso de mi vida.

A mis Padres, José Gómez y a Julia Morales, mi eterno agradecimiento ya que me apoyaron incondicionalmente sin ellos este logro no hubiera sido posible, ya que los mismos fueron esa base para poder hacer este sueño una realizad.

A mi Hermano Washington, por su constancia y ejemplo.

De manera especial al PhD. Luis Sánchez y a la Ing. Nancy Guerrón quienes en calidad de tutores, aportaron con sus conocimientos y experiencia.

A mis amigos, en especial a Isaac por acompañarme en el desarrollo de la tesis y brindarme su amistad y confianza.

Y a todas las personas que de una u otro forma aportaron para que el presente proyecto haya llegado a su feliz término.

Oscar.

AGRADECIMIENTO

Agradezco a Dios por hacer posible mis peticiones mantenerme con vida bajo sus bendiciones, guiar mi camino, y permitirme culminar una meta más en la vida.

A mi familia, quienes con cariño y esmero supieron motivarme para siempre seguir adelante y no rendirme por las adversidades, les debo todo lo que hoy en día soy; en especial a mi madre Mirian Luzuriaga por estar siempre a mi lado en los malos momentos, en los peores, y en los fracasos ya que con su presencia y apoyo me ayudo a surgir.

A mis hermanas Verónica, Gabriela y Fernanda quien con su trabajo y consejos supieron enseñarme cómo enfrentar la vida, mismas palabras que me motivaron a seguir adelante, aprecio y agradezco por su infinito apoyo, y gracias por permitirme ser un ejemplo para sus hijos.

A mi hermosa Madeleine por estar a mi lado eres lo que me inspira y motiva para luchar en la vida.

Oscar quiero darte las gracias por acompañarme con tu apoyo incondicional para seguir adelante en los momentos difíciles del proyecto. A mis amigos que con su apoyo supieron darme palabras de aliento cuando más lo necesitaba y un agradecimiento muy especial para Maricela Paz quien me apoyo mucho y supo brindarme una amistad sincera, estoy eternamente agradecido contigo.

A la ingeniera Nancy Guerrón y al PhD. Luis Sánchez por compartir sus conocimientos quien supo guiarnos desde un inicio en la elaboración de este proyecto.

Isaac

ÍNDICE DE CONTENIDO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA.....	i
CERTIFICADO	ii
DECLARACIÓN DE RESPONSABILIDAD	iii
AUTORIZACIÓN.....	iv
DEDICATORIA.....	v
AGRADECIMIENTO	vii
ÍNDICE DE CONTENIDO	ix
ÍNDICE DE TABLAS.....	xii
ÍNDICE DE FIGURAS.....	xiii
ÍNDICE DE ECUACIONES	xvi
RESUMEN.....	xvii
ABSTRACT	xviii
CAPÍTULO 1.....	1
1 FUNDAMENTOS TEÓRICOS	1
1.1 Antecedentes.....	1
1.1.1 Históricos	1
1.1.2 La actividad eléctrica del cerebro	1
1.1.3 Técnica de Polisomnografía	3
1.1.4 Cuando realizar un Polisomnograma.	4
1.1.5 Evaluación del SAHOS mediante Polisomnógrafo	6
1.1.6 La polisomnografía.....	8
1.2 Variables Electrofisiológicas	9
1.2.1 Electrocardiograma	9
1.2.2 Presión sanguínea	13
1.2.3 Pulsioximetría.....	16
1.2.4 Espirómetro.....	19
1.3 Estándar Bluetooth (802.15.1)	23
1.4 Arduino UNO	28
1.4.1 Pines especiales de entrada y salida	28
1.4.2 Programación.....	30

	x
1.5	Plataforma Android30
1.6	Plataforma de desarrollo Java31
CAPÍTULO 2.....32	
2	DISEÑO E IMPLEMENTACIÓN.....32
2.1	Diagrama de bloques del proyecto a realizar32
2.2	Implementación del ECG33
2.2.1	Electrodos34
2.2.2	Amplificador de Instrumentación34
2.2.3	Implementación de los Filtros Analógicos.....38
2.2.4	Acondicionamiento de la señal.41
2.3	Medición de la presión sanguínea arterial.....43
2.3.1	Método Oscilométrico.....43
2.3.2	Brazalete inflable.....44
2.3.3	Bomba de aire.....45
2.3.4	Electroválvula.....45
2.3.5	Sensor de presión MPX5050GP.....46
2.3.6	Acondicionamiento del sensor de presión MPX5050GP.....46
2.3.7	Descripción del Circuito Acondicionador.48
2.4	Diseño y Construcción del SPO2.....51
2.4.1	Diseño del Hardware.....54
2.4.2	Diseño del convertor de Corriente a Voltaje.....55
2.4.3	Diseño del filtro56
2.4.4	Diseño del Amplificador final.57
2.4.5	Diseño del circuito de control para activación de los leds59
2.4.6	Diseño del circuito generador de las señales de control.....61
2.4.7	Diseño del Generador de Pulsos.....62
2.5	Diseño del Espirómetro.....64
2.5.1	Consideraciones de Diseño.....65
2.5.2	Adquisición de la señal.....65
2.5.3	Acondicionamiento de la señal.66
2.6	Artefactos67
2.6.1	ECG67
2.6.2	Presión Sanguínea.....67

	xi
2.6.3 SPO2	67
2.6.4 Espirometría.....	68
2.7 Modulo Bluetooth HC-06 para la Comunicación Serial	68
2.8 Diseño del Software.....	69
2.8.1 Entorno de desarrollo para Arduino.....	70
2.8.2 Desarrollo de la aplicación en Android	71
CAPÍTULO 3.....	78
3 PRUEBAS Y RESULTADOS OBTENIDOS	78
3.1 Pruebas del ECG.....	78
3.2 Pruebas de la Presión Sanguínea	80
3.3 Pruebas del SPO2	82
3.4 Pruebas del Espirometría	84
CAPÍTULO 4.....	86
4 Análisis Económico.....	86
4.1 Proyectos futuros.....	89
CAPÍTULO 5.....	90
5 CONCLUSIONES Y RECOMENDACIONES	90
5.1 CONCLUSIONES	90
5.2 RECOMENDACIONES.....	92
6 REFERENCIAS BIBLIOGRÁFICAS.....	94
ANEXOS	97

ÍNDICE DE TABLAS

Tabla 1.1: Edificación del sueño.	3
Tabla 1.2: Evaluación de la respiración	4
Tabla 1.3: Valores normales del ECG. Frecuencia cardiaca Latidos por minuto (b.p.m).	12
Tabla 1.4: Valores normales de la Presión Arterial	16
Tabla 1.5: Valores de la Hipertensión	16
Tabla 1.6: Características del sensor según el fabricante	17
Tabla 1.7: Ventajas y desventajas de los siguientes tipos de Espirómetros.	20
Tabla 1.8: Capacidad Pulmonar valores normales.....	23
Tabla 1.9: Volumen Pulmonar valores normales	23
Tabla 1.10: Características de la tarjeta Arduino UNO	29
Tabla 2.1 Descripción de los pines del sensor Nellcor.	51
Tabla 2.2 Relación entre el color y la tensión de umbral de Leds.	53
Tabla 2.3: Características técnicas del sensor de oximetría	53
Tabla 2.4: Tabla de verdad para las señales de control.	61
Tabla 2.5: Características del Bluetooth HC-06.	69
Tabla 2.6: Funciones comunes de la barra de herramientas del entorno de desarrollo para Arduino.....	71
Tabla 3.1: Datos del Voluntario	79
Tabla 3.2: Datos del ECG	80
Tabla 3.3: Datos de la Presión Sanguínea.....	82
Tabla 3.4: Datos del SPO2	83
Tabla 3.5: Datos del Espirómetro.....	85
Tabla 4.1: Costo total del Polisomnógrafo inalámbrico	87

ÍNDICE DE FIGURAS

Figura 1.1: Derivación I.....	10
Figura 1.2: Derivación II.....	10
Figura 1.3: Derivación III.....	11
Figura 1.4: Triangulo de Einthoven.	11
Figura 1.5: Señal de un ECG.....	12
Figura 1.6: Esfigmomanómetro para medir presiones sanguíneas.	14
Figura 1.7: Espectro electromagnético.....	17
Figura 1.8: Esquema de un espirómetro de turbina A) aspas fijas para dirigir el flujo de aire, B) Hélice (turbina), C) Eje de la hélice, D) Sensor óptico para el movimiento de la hélice.....	22
Figura 1.9: Espirómetro de turbina.....	22
Figura 1.10: Aplicaciones del Bluetooth	24
Figura 1.11: Estructura de Piconet.	26
Figura 1.12: Estructura de Scatternet.	26
Figura 1.13: Tarjeta Arduino UNO y su puerto de programación.	30
Figura 2.1: Diagrama de bloques general para la implementación de un Polisomnógrafo con transmisión de datos inalámbricos.....	32
Figura 2.2: Diagrama de Bloques del ECG.	33
Figura 2.3: Diagrama Esquemático de la adquisición de la señal del ECG. .	33
Figura 2.4: Diagrama del Amplificador de Instrumentación basado en el integrado TL084.	35
Figura 2.5: Amplificador de Instrumentación pasa banda basado en el integrado TL084.	37
Figura 2.6: Filtro rechaza banda.	38
Figura 2.7: Filtro rechaza banda con los valores del diseño.....	39
Figura 2.8: Filtro de Buttherworth.....	39
Figura 2.9: Filtro Buttherworth con los valores del diseño.	41
Figura 2.10: Etapa de acondicionamiento de señal.	42
Figura 2.11: Seguidor de Tensión TL084.....	42
Figura 2.12: Diagrama de bloque de adquisición de la presión.....	44
Figura 2.13: Brazaletes inflables.	44
Figura 2.14: Bomba de aire.	45
Figura 2.15: Electroválvula.	46
Figura 2.16: Sensor de presión MPX5050GP.	46
Figura 2.17: Circuito de conexión del sensor MPX5050GP.....	47
Figura 2.18: Amplificador de la señal de Oscilación.....	49
Figura 2.19: Respuesta de Frecuencia del Filtro.....	50
Figura 2.20: Señal PB en las salida del sensor de Presión.	50
Figura 2.21: Señal de oscilación extraída de la salida del amplificador de la Figura 2.20.	51

Figura 2.22: Sensor de Oximetría marca Nellcor.	51
Figura 2.23: Esquema interno del sensor de Oximetría.	52
Figura 2.24: Diagrama de bloques del SPO2.....	55
Figura 2.25: Conversor de Corriente a Voltaje diseñado.....	55
Figura 2.26: Filtro Pasabajos diseñado.....	57
Figura 2.27: Amplificador de Instrumentación diseñado.....	58
Figura 2.28: Esquema del Puente H.	59
Figura 2.29: Circuito de control de activación de los leds.	60
Figura 2.30: Circuito generador de señales de control para el puente H.	62
Figura 2.31: Esquema del temporizador.	62
Figura 2.32: Generador de pulsos diseñado.	64
Figura 2.33: Diagrama de bloque general del Espirómetro.	65
Figura 2.34: El sensor fotodiodo detecta el movimiento de la turbina.	66
Figura 2.35: Circuito de acondicionamiento del espirómetro.....	66
Figura 2.36: Módulo Bluetooth HC-06.....	68
Figura 2.37: Conexión del Arduino UNO y el módulo Bluetooth HC-06.....	69
Figura 2.38: Entorno de desarrollo de Arduino.....	70
Figura 2.39: Selección de la carpeta y dirección del proyecto.....	72
Figura 2.40: Selección de un nuevo proyecto.	72
Figura 2.41: Selección de la carpeta Android.....	72
Figura 2.42: Selección de la carpeta Android.....	73
Figura 2.43: Creación de un nuevo proyecto Android.	74
Figura 2.44: Creación de un nuevo proyecto Android.	74
Figura 2.45: Configuración para la creación del icono para el proyecto.	75
Figura 2.46: Creación de una actividad para un nuevo proyecto Android	75
Figura 2.47: Creación de una actividad para un nuevo proyecto Android. ...	76
Figura 2.48: Proyecto finalizado en Android.....	76
Figura 2.49: Aplicación creada para la Polisomnografía	77
Figura 2.50: Ventana principal de la Aplicación del Polisomnógrafo	77
Figura 2.51: Ventana del ECG	77
Figura 2.52: Ventana de datos para el SPO2, Presión Sanguínea y Flujo de Aire	77
Figura 3.1: Señal obtenida del ECG implementado, mostrada en un osciloscopio.	78
Figura 3.2: Señal del ECG de una voluntaria	79
Figura 3.3: Señal del ECG adquirida en la Tablet	79
Figura 3.4: Señal obtenida del circuito de Presión Sanguínea implementado mostrada en un osciloscopio	81
Figura 3.5: Señal de presión de un voluntario.....	81
Figura 3.6: Señal de la Presión Sistólica y Diastólica adquirida en la Tablet.	81
Figura 3.7: Señal obtenida del circuito de Oximetría implementado mostrada en un osciloscopio	82

	xv
Figura 3.8: Señal del SPO2 de un voluntario.	83
Figura 3.9: Señal del SPO2 adquirida en una Tablet	83
Figura 3.10: Mascarilla acoplada al Sensor de turbina e infrarrojo.	84
Figura 3.11: Señal obtenida en el circuito de espirometría mostrada en un osciloscopio.	84
Figura 3.12: Señal del Espirómetro de un Voluntario.	85
Figura 3.13: Señal de flujo de aire adquirida en la Tablet.	85
Figura 4.1: Monitor de paciente multiparámetros	86
Figura 0.1 Símbolo eléctrico y terminales de un amplificador operacional .	104

ÍNDICE DE ECUACIONES

$V_0V_2 - V_1 = G = R_2R_12R_3RG + 1$ Ecuación (2.1)	36
$f_{corte} = 12\pi RC$ Ecuación (2.2)	36
$Conteo = VXdcr - VrlVrh - Vrl * 210 - 1$ Ecuación (2.3)	47
$resolucion = Presión\ maxima - Presión\ minima$ Ecuación (2.4) ...	48
$fp1 = 12\pi R_2C_2$ Ecuación (2.5)	49
$fp2 = 12\pi R_6C_3$ Ecuación (2.6)	49
$V_{out} = I_p.R_21 + R_3R_1$ Ecuación (2.7)	56
$Av = 1 + RfRi$ Ecuación (2.8) $f = 12\pi RC$	56
$V_0V_1 - V_2 = 1 + 2RRp$ Ecuación (2.9)	58
$VO = 1 + 2RRp$ $V_1 - V_2 = K V_1 - V_2$ Ecuación (2.10).....	58
$Talto \approx 0.7R_1 + R_2C_1$ Ecuación (2.11)	63
$FP = Número\ de\ Pulsos * Tiempo$ $Numero\ de\ aspas\ de\ la\ turbina$ Ecuación (2.12)	66

RESUMEN

El presente proyecto consiste en el diseño y construcción de un equipo de Polisomnografía, este equipo consta de sensores para la adquisición de las señales Biofísicas y Bioeléctricas del cuerpo humano, las cuales se van acondicionar para poder transmitir las inalámbricamente vía Bluetooth para la visualización respectiva de los datos en un dispositivo móvil que contenga la plataforma Android. Estas señales ayudan en la detección de las enfermedades producidas durante las fases del sueño, como el SAHOS que es una enfermedad de ahogamiento de las personas mientras duermen conocida como hipoapneas o apneas obstructoras del sueño, que son la principal causa de desarrollo de células cancerígenas en cuerpo humano, debido a que durante la noche se impide el paso necesario de oxígeno a las células las cuales van muriendo y convirtiéndose en células malignas o cancerígenas; hay que tomar en cuenta que el cáncer si no es tratado con anticipación puede provocar la muerte.

Palabras Clave: Polisomnografía, Enfermedades del sueño, SAHOS, Dispositivos móviles, Conexiones inalámbricas.

ABSTRACT

This project involves the design and construction of a Polysomnography team, this team consists of sensors for the acquisition of Biophysical and bioelectric signals of the human body, which will put to transmit wirelessly via Bluetooth to the respective visualization data on a mobile device containing the Android platform. These signals help in the detection of diseases produced during sleep stages as OSAHS is a disease of people drowning while sleeping or hypopneas known as obstructive sleep apnea, which are the leading cause of cancer cell development in human body, because overnight the necessary passage of oxygen to the cells which die and become malignant or cancerous cells is prevented; must take into account that cancer if not treated early can lead to death.

Keywords: Polysomnography, Sleep Disorders, OSAHS, Mobile Devices, Wireless.

CAPÍTULO 1

1 FUNDAMENTOS TEÓRICOS

1.1 Antecedentes

1.1.1 Históricos

Ha existido un gran interés científico en el sueño, con los descubrimientos de la actividad eléctrica del cerebro, los sistemas respiratorios, cardíacos y el sueño de movimientos oculares rápidos. Estos descubrimientos, en el campo de la medicina del sueño han existido por sólo cerca de cuatro décadas. Y aún está en crecimiento, a medida que se tiene nuevos descubrimientos, nuevos tratamientos; y las ciencias básicas están ayudando a entender la complejidad del sueño y sus trastornos. [1]

1.1.2 La actividad eléctrica del cerebro

En 1875, el fisiólogo escocés Richard Catón demostró ritmos eléctricos en el cerebro de animales. Sin embargo, no fue sino hasta 1928 que el psiquiatra alemán Hans Berger registró la actividad eléctrica del cerebro humano y demostró diferencias claras entre los ritmos cuando los sujetos estaban despiertos y cuando dormían. Él acuñó el término electroencefalogramas para referirse a las señales que registraba.

Posteriormente, los elementos fundamentales de los patrones de ondas de sueño cerebral fueron descritos por Harvey, en la Universidad de Harvard, y por Kleitman, en la Universidad de Chicago, entre 1937 y 1939.

En 1949, después de la Segunda Guerra Mundial, Moruzzi publicó un texto clásico "Brain Stem Reticular Formation and Activation of the EEG" y concluyó que la transición de un estado del sueño a uno de alerta presenta una aparente ruptura de la sincronización de descarga de los elementos de la corteza cerebral, así como una alteración marcada en el EEG debido al cambio de ondas lentas con alto voltaje por una actividad veloz de bajo voltaje.

En 1953, Nathaniel Kleitman, profesor de fisiología de la Universidad de Chicago, y Eugene Aserinsky, un estudiante de fisiología, publicaron la descripción del sueño MOR, para ello utilizaron el Electrooculograma.

Posteriormente, Dement y Kleitman describieron la variación cíclica de los patrones electroencefalográficos.

La investigación que enfatiza el registro de toda la noche de sueño se inició realmente en los años sesenta con el nombre de polisomnografía.

En 1972, Christian Guilleminault, un neurólogo y psiquiatra francés, se unió al grupo de Standford. Con su llegada se hizo rutinaria la utilización de sensores respiratorios y cardiacos durante los estudios del sueño; el término polisomnografía se acuñó en 1974, desde 1975 se logró que se considerara a la polisomnografía como una prueba diagnóstica y tuviera una retribución económica por parte de las aseguradoras.

Al principio del 2007, American Academy of Sleep Medicine emitió las últimas normas, donde se modifica la estadificación a sueño MOR y no MOR (estadios N1, N2 y N3), Tabla 1.1 y se establece las recomendaciones para tener un estudio óptimo de sueño.

Concomitantemente y a lo largo de la historia se fueron describiendo algunas alteraciones del sueño, y es así como, en 1880, Jean Baptiste Edouard Gelineau describió la narcolepsia de las raíces griegas narcosis (sopor, entorpecimiento) y lepsis (posición).

Lo que posteriormente sería el trastorno de sueño por excelencia del siglo XX, la apnea obstructiva del sueño, fue descrita en 1836, no por un clínico, sino por el novelista Charles Dickens. En su serie Posthumous papers of the Pickwick Club, en la que describe a un muchacho obeso, con excesiva somnolencia diurna y fuerte roncador.

Posteriormente, Peretz Lavie publicaron muchas referencias sobre la apnea del sueño (SAHOS o síndrome de apnea hipopnea obstructiva del sueño).

La Asociación Americana de Sueño comenzó en 1975 con cinco miembros y al 2005 contaba con 2.600 médicos certificados en sueño, 766 programas acreditados y asistencia a la reunión anual de unas 5.000 personas de todo el mundo. [1]

1.1.3 Técnica de Polisomnografía

El registro de los estudios de sueño requiere la adquisición de las siguientes mediciones fundamentales:

- Electroencefalograma (EEG).
- Electrooculograma (EOG).
- Electromiograma de superficie (EMG).
- Electrocardiograma (ECG).
- Los niveles de oxígeno en la sangre.
- El volumen de aire que entra y sale de los pulmones durante la respiración

Estas mediciones permiten estadificar el sueño en dos etapas R: movimientos oculares rápidos MOR y no MOR.

- Estadios N1, N2 y N3 Tabla 1.1

Tabla 1.1: Edificación del sueño.

	CARACTERÍSTICAS
R	Bajo voltaje desincronizado, brotes de movimientos oculares rápidos en el EOG y supresión de actividad EMG.
N1	Sueño ligero, ocurre al inicio del sueño y transiciones la noche, Actividad de bajo voltaje, ondas del vertex.
N2	Husos de sueño y complejos K en el EEG.
N3	Sueño de ondas lentas, ondas de alto voltaje y dos o pocos ciclos por segundo.

Y la evaluación de la respiración se puede mencionar varios términos Tabla 1.2.

Tabla 1.2: Evaluación de la respiración

TERMINO	DEFINICIÓN
Apnea	Cesación de la respiración durante 10 segundos.
Hipopnea	Disminución en la amplitud de la respiración en 30% - 50%, asociada con saturación del 4%, acompañada o no de un despertar.
Desaturación	Disminución del 3% - 4% en la saturación de oxígeno.
Apnea obstructiva	Evento respiratorio por obstrucción de la vía aérea.
Apnea central	Eventos respiratorios causados por ausencia del esfuerzo respiratorio.
Apnea mixta	Eventos respiratorios con características de obstructivo y central.
Respiración periódica	Patrón regular en el que alterna incremento de la ventilación con disminución o ausencia de ésta. Por ejemplo, respiración de Cheyne – Stokes.

[2]

Adicionalmente el equipo de polisomnografía permite realizar otras pruebas diagnósticas, como el test de latencia múltiple del sueño (MSLT) para los pacientes en quienes se sospecha narcolepsia. Otra prueba que se puede realizar es el test de mantenimiento del estado alerta, el cual se ha empleado con los pilotos.

Debe anotarse que los equipos de polisomnografía modernos permiten la realización de grabación de video, para así poder documentar múltiples trastornos asociados con el sueño.

1.1.4 Cuando realizar un Polisomnograma.

Existen múltiples casos y características que llevan a la conveniencia de practicar un Polisomnograma. Algunos de las más frecuentes son:

1. En pacientes con sospechas de trastornos respiratorios durante el sueño, como apnea obstructiva del sueño, apnea central, síndrome de **Hipoventilación**, respiración periódica o enfermedad neuromuscular.
2. En pacientes con alta probabilidad **Pretest**, como roncadores, **Hipersomnes**, obesos o con apneas presenciadas.
3. En la evaluación pre y posquirúrgica **Otorrinolaringológica**, por ejemplo en pacientes con obstrucción de la vía aérea superior con sospecha de SAHOS.
4. En el seguimiento de pacientes con SAHOS luego del tratamiento con dispositivo intraoral o con CPAP (presión continua positiva en la vía aérea).
5. En la evaluación con pacientes con insomnio.
6. En el estudio con pacientes con sospecha de narcolepsia que presentan **Hipersomnio** diurno.
7. En la evaluación laboral de personas que desarrollan actividades por turnos; por ejemplo, celadores, médicos, conductores y pilotos.
8. En la evaluación de **Parasomnias**, entre las que encontramos algunas asociadas tanto al sueño no MOR (despertares confusos, sonambulismo, terrores nocturnos) como al sueño MOR (desorden de comportamiento asociado al MOR, parálisis del sueño aislada recurrente, pesadillas).
9. En la evaluación de diferentes patologías, como los trastornos disociativos relacionados con el sueño, la catatrenia y el síndrome de la cabeza por explotar.
10. En el estudio de los trastornos alimentarios asociados al sueño.
11. En la evaluación de la impotencia.
12. En el estudio de pacientes con hipertensión pulmonar y **Eritrocitosis de etiología** no establecida.
13. En la determinación del valor de presión positiva requerido para la corrección de los eventos respiratorios, cuando esté indicado.
14. En el estudio de pacientes con insuficiencia cardíaca congestiva que cumplan alguna de las siguientes condiciones:
 - **Angina** nocturna.

- Sueño intranquilo.
 - Clasificación de la **NYHA** III o IV a pesar del tratamiento.
 - $p\text{CO}_2$ baja.
 - Utilización de desfibrilador.
15. En pacientes con enfermedad coronaria, hipertensión arterial o **Arritmias** de difícil control y síntomas sugestivos de SAHOS.
 16. En pacientes con enfermedad cerebrovascular y síntomas sugestivos de SAHOS.
 17. En el seguimiento de pacientes que presenten modificación del 10% en su peso corporal.
 18. En la evaluación previa a una cirugía **Bariátrica** y luego de llegar al peso ideal.
 19. En la evaluación de movimientos anormales de las piernas.

1.1.5 Evaluación del SAHOS mediante Polisomnógrafo

Las investigaciones acerca de la polisomnografía han tomado gran interés en especial por la aparición del síndrome de apneas-hipoapneas obstructivas del sueño (SAHOS) es una de las más frecuentes enfermedades del sueño.

Consiste en episodios repetidos de pausas respiratorias durante el sueño, como consecuencia de una alteración anatómico-funcional de la vía aérea superior que lleva al colapso de la misma, asociadas a una reducción de la saturación de oxígeno y cambios en el EEG.

Se define al SAHOS como un cuadro de somnolencia excesiva, ronquidos y apneas, con trastornos cognitivo-conductuales, respiratorios, cardíacos, metabólicos e inflamatorios, relacionado con la morbilidad asociada a complicaciones cardiorrespiratorias y neurológicas y con una estrecha relación con los accidentes de tránsito.

El tratamiento de elección de este síndrome es el empleo de la presión positiva continua sobre la vía respiratoria, conocido como CPAP (Continuous Positive Airways Pressure), y la evidencia científica demuestra que esta terapia es efectiva para mejorar la somnolencia y la calidad de vida en

pacientes con SAHOS y, además, es más efectiva que los dispositivos orales para mejorar las apneas y las hipoapneas.

En el caso del SAHOS, el éxito de la terapia depende en gran parte de la gravedad de la sintomatología diurna (somnolencia y trastornos cognitivos diurnos), de la correcta indicación, de la calidad de las máscaras y fundamentalmente de la colaboración del paciente.

Sin embargo, existen casos donde las dificultades de adaptación o las **Comorbilidades**, hacen necesario utilizar nuevos modos ventilatorios, para mejorar la calidad de vida de estas personas enfermas.

Muchas veces el SAHOS coexiste en los pacientes obesos con el síndrome de hiperventilación por obesidad [SHO], por lo que es probable que tal vez sean un mismo proceso en diferentes etapas evolutivas, o que la obesidad sea la que determine una u otra de las patologías dependiendo de otros múltiples factores asociados [**Retrognatia, Hipertrofia amigdalina**, perímetro de cuello, hipotiroidismo, etcétera].

En ocasiones los pacientes requieren distintos niveles de presión a lo largo de la noche o no toleran espirar contra una resistencia elevada.

Otras veces se trata de apneas difíciles de controlar con **CPAP**, tal es el caso de los pacientes con SAHOS que desencadenan un importante número de apneas centrales como consecuencia del tratamiento con CPAP o los pacientes con respiración de **Cheyne Stokes**, en quienes el tratamiento con CPAP no siempre es beneficioso.

Estos casos son definidos como síndrome de apneas del sueño complejo y representan un 15% de los pacientes con apnea del sueño. Estos pacientes pueden beneficiarse con sistemas ventilatorios de más reciente incorporación como el modo ASV [Adaptive Servo Ventilation] que demostró ser un tratamiento eficaz. [3]

1.1.6 La polisomnografía

La polisomnografía es el registro, análisis e interpretación de múltiples parámetros fisiológicos durante el sueño, permitiendo su estudio y el de los trastornos que lo afectan. “El paciente es estudiado durante una noche para saber el grado de somnolencia”, y recibe un cuestionario para saber los principales síntomas que presenta el enfermo; en todo el cuerpo se colocan sensores para medir el tono muscular del mentón, pantorrillas, abdomen, la intensidad de la actividad cerebral, así como para medir el flujo aéreo o ingreso y salida de aire, además se aclara que esta prueba no se le realiza a pacientes con dificultad para conciliar el sueño, sino a quienes duermen pero su descanso no es reparador, y la polisomnografía se efectúa para ver qué razones originan esto.

Por otro lado, en el país es casi nula, por no decir inexistente, la investigación en esta área. Tal vez se hayan realizado algunos trabajos aislados pero no representan un grupo de investigación consolidado en esta área de especial importancia, que a futuro podría representar una ayuda vital para personas que tienen economía y tiempo limitado, permitiéndoles y brindándoles herramientas que les asistan para poder llevar una vida de mejor calidad.

Este proyecto pretende la iniciación a nivel local de la investigación en esta área, por medio de la exposición de sus principios generales, algunas pruebas experimentales cuyos resultados indiquen la posibilidad de extraer información correspondiente a condiciones específicas mediante una medición de ECG, Saturación de oxígeno en la sangre, Presión arterial, Flujo de aire en los pulmones y finalmente la implementación de un Polisomnógrafo con transmisión de datos inalámbricos, que motive a próximos estudiantes e investigadores, a desarrollar de una manera más amplia y profunda, todos los aspectos vinculados con la Polisomnografía. [4]

1.2 Variables Electrofisiológicas

Son las propiedades eléctricas de células y tejidos biológicos. Incluye medidas de cambio de voltaje o corriente eléctrica en una variedad amplia de escalas, desde el simple canal iónico de proteínas hasta órganos completos como el corazón.

1.2.1 Electrocardiograma

Un electrocardiograma (ECG) es un registro de la actividad eléctrica que tiene lugar en el corazón cada vez que se contrae.

Se ponen electrodos en determinadas zonas del cuerpo del paciente y mediante el uso de diversas combinaciones de estos electrodos se observan 12 vistas diferentes de la misma actividad eléctrica en el papel cuadriculado de ECG. Cada vista del corazón se llama derivación electrocardiográfica.

Si bien se coloca electrodos en ambas muñecas y el tobillo izquierdo del paciente para obtener las derivaciones estándares y aumentadas, los electrodos en realidad pueden ponerse en cualquier parte de las respectivas extremidades o en la parte superior e inferior del torso y se registrara la misma vista del Corazón. Un cuarto electrodo se coloca en el tobillo derecho para estabilizar el ECG, pero este electrodo no participa en la formación de derivaciones.

Derivaciones estándares.

Las derivaciones estándares se llaman derivaciones bipolares porque están compuestas por dos electrodos, uno negativo y uno positivo, y el ECG registra la diferencia de potencial eléctrico entre ellos.

- a) **DERIVACIÓN I.** La derivación I se forma con el electrodo del brazo derecho, que se designa como negativo, y el brazo izquierdo, que se considera positivo, obsérvese la Figura 1.1.

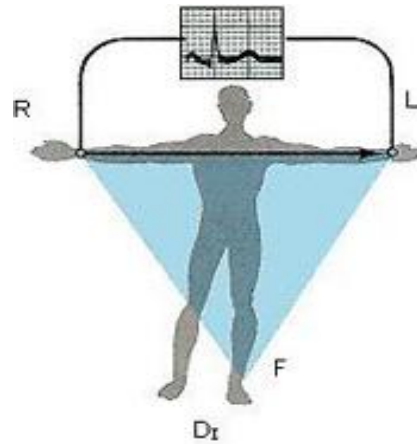


Figura 1.1: Derivación I
[5]

- b) **DERIVACIÓN II.** La derivación II se forma con el electrodo del brazo derecho, que se designa como negativo, y el de la pierna izquierda, que se considera positivo, Obsérvese la Figura 1.2.

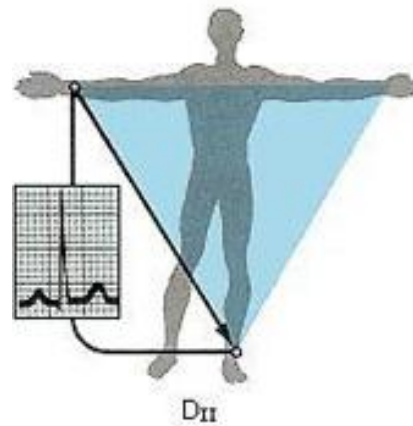


Figura 1.2: Derivación II
[5]

- c) **DERIVACION III.** Se forma con el electrodo del brazo izquierdo, que se designa como negativo, y el de la pierna izquierda, que se considera positivo. Obsérvese la Figura 1.3.

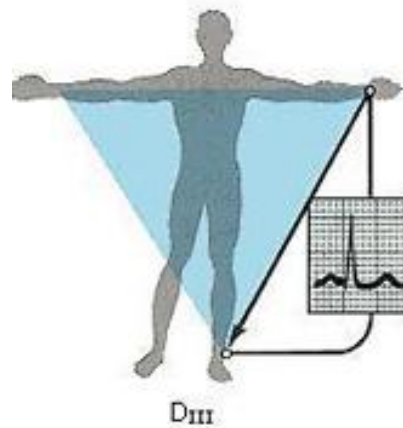


Figura 1.3: Derivación III.
[5]

Las tres derivaciones estándares forman un triángulo sobre el cuerpo y tienen una relación matemática entre sí, como lo describió Einthoven Figura 1.4: la altura o profundidad de los registros de la derivación I más de la derivación III es igual a la altura o profundidad del registro en la derivación II. Obsérvese la Figura 1.4

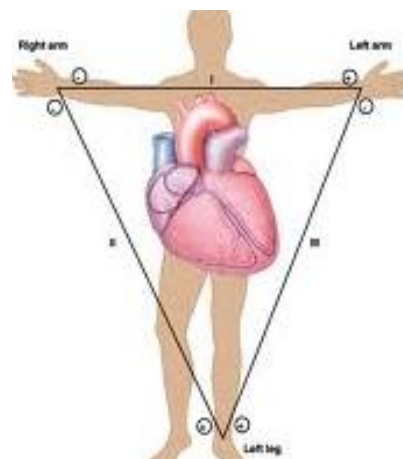


Figura 1.4: Triángulo de Einthoven.
[5]

Los valores que entrega el examen del electrocardiograma son múltiples dependiendo la edad Tabla 1.3.

Tabla 1.3: Valores normales del ECG. Frecuencia cardiaca Latidos por minuto (b.p.m).

Edad	Rango b.p.m	Valor Normal b.p.m
Neonato	95 – 150	123
1 – 2 meses	121 – 179	149
3 – 5 meses	106 – 186	141
6 – 11 meses	109 – 169	134
1 – 2 años	89 – 151	119
3 – 4 años	73 – 137	108
5 – 7 años	65 – 133	100
8 – 11 años	62 – 130	91
12 – 15 años	30 – 119	85
Adultos	60 – 100	80

[6]

La señal resultante de un ECG es la Figura 1.5 donde se indica todas las ondas relacionadas en el tiempo-nivel y su máxima amplitud.

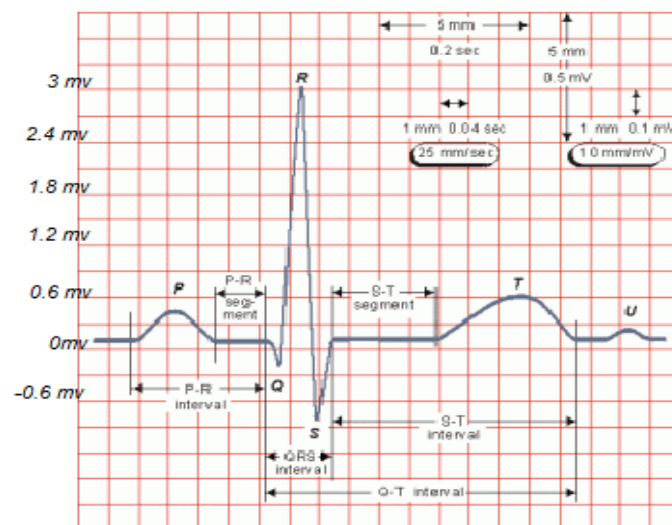


Figura 1.5: Señal de un ECG

[5]

1.2.2 Presión sanguínea

La presión sanguínea es la fuerza ejercida por la sangre contra las paredes vasculares por unidad de superficie. Depende del flujo sanguíneo, equivalente al gasto cardíaco, y de las resistencias periféricas que se oponen a él, especialmente la ejercida por las arteriolas.

El control continuo de la presión sanguínea se ejerce por el sistema nervioso autónomo, especialmente por el sistema simpático. Los impulsos recibidos tanto por los receptores de presión situados en las arterias carótidas y en la aorta, como por los centros cerebrales superiores, se transmiten a través del hipotálamo al sistema nervioso autónomo, que responde modificando la frecuencia cardíaca y la resistencia de los vasos, alterando en definitiva la intensidad del riego sanguíneo en los tejidos.

En ciertos estados patológicos, la presión de la sangre se eleva por encima de las cifras normales, algunas veces tales aumentos llegan a ser una fuente de peligros inminentes para la vida. Esta elevación de la presión arterial no es una enfermedad en sí misma, pero si un signo o una manifestación de un proceso patológico, igual que la fiebre no es una enfermedad pero si un índice de medición externa de una enfermedad interna. Podría decirse en consonancia tanto de la razón como de la ciencia, que la elevación de la presión arterial es un proceso COMPENSATORIO o CONSERVADOR por el cual se mantiene la adecuada circulación de la sangre a los tejidos, a pesar de los incrementos de resistencias u obstrucciones al flujo de sangre. Si la presión de la sangre no se eleva cuando la obstrucción al flujo de sangre se incrementa, el resultado inevitable sería la muerte por fallo circulatorio. Para acercar más la analogía entre fiebre y elevación de la presión arterial podría afirmarse que una elevación de la presión sanguínea es un mal necesario para mantener la vida en circunstancias adversas del flujo de sangre, igual que la fiebre es una reacción desagradable, pero que sin embargo, conduce al cuerpo a luchar contra la invasión de la enfermedad. Pero tanto los grados de fiebre como las cifras de la presión arterial deben permanecer entre los límites de seguridad, por tanto deben hacerse los esfuerzos necesarios para reducir tanto los grados como las cifras cuando ellos alcanzan proporciones

alarmantes. Debería tenerse en mente que una drástica reducción de la fiebre o de la presión arterial, por drogas, está lleno de graves consecuencias, y no debería ser intentado. Quitar la causa por la cual el cuerpo reacciona con fiebre o elevaciones de la presión arterial es el método ideal de tratamiento.

La elevación de la presión arterial, también conocida como HIPERTENSIÓN, parece ser más común en nuestros días que antiguamente. No se puede negar que la civilización moderna ha traído consigo gran cantidad de factores adversos que tienden a elevar la presión de la sangre. El hecho de vivir continuamente con ansiedad, codicia, ambición, métodos artificiales de vida, falta de fe religiosa, excesos de todo tipo, horarios irregulares, comida poco saludable y muchos vicios peculiares de la civilización moderna, juegan indudablemente un papel importante en su desarrollo. El esfigmomanómetro Figura 1.6 es el instrumento habitual que se utiliza para hacer esta medida de forma conveniente y sin ningún dolor.

Durante un ciclo completo de bombeo, la presión en el corazón y el sistema circulatorio experimenta un máximo (cuando la sangre es bombeada del corazón) y un mínimo (cuando el corazón se relaja y se llena de la sangre procedente de las venas). El esfigmomanómetro se emplea para medir estas presiones extremas.



Figura 1.6: Esfigmomanómetro para medir presiones sanguíneas.

Su uso se basa en el hecho de que el flujo de sangre en las arterias no es siempre estacionario. Cuando las arterias se encogen y el gasto es grande, el flujo se hace turbulento. Este flujo es ruidoso y puede oírse con un estetoscopio.

En el esfigmomanómetro se mide con un manómetro o un indicador de presión la presión manométrica en un brazalete arrollado alrededor del brazo. Al principio, la presión del brazalete se aumenta hasta que la arteria humeral queda totalmente cerrada. La presión del brazalete se reduce entonces lentamente, mientras se utiliza un estetoscopio para escuchar los ruidos de la arteria humeral por debajo de la parte aprisionada por el brazalete. Cuando la presión llega a un valor ligeramente inferior a la presión sistólica (máxima) producida por el corazón, la arteria se abrirá brevemente. Como solo está parcialmente abierta, la velocidad de la sangre es elevada y el flujo es turbulento y ruidoso. El ruido resultante se oye como una especie de latido (Ruidos de Korotkoff).

Cuando se sigue bajando la presión del brazalete, la arteria permanece abierta durante períodos más prolongados del ciclo cardíaco, pero se halla aún cerrada en la parte de presión diastólica (mínima), por consiguiente, se oyen sonidos, pero estos se ven interrumpidos por períodos de silencio. Cuando la presión del brazalete llega a la presión diastólica, la arteria permanece abierta durante todo el ciclo del corazón. A esta presión el flujo es todavía turbulento y ruidoso (particularmente en la presión diastólica) pero ahora el sonido es continuo. Así pues, tanto la presión sistólica como la diastólica pueden medirse sin necesidad de canulación.

Las presiones sanguíneas se expresan habitualmente como razones de presiones sistólica/diastólica. Los datos típicos para un adulto sano en reposo son aproximadamente 120/80 en mm/Hg y 16/11 en Kpa. La frontera para la alta presión sanguínea (hipertensión) se define generalmente como 140/90 en mm/Hg y 19/12 en Kpa. Las presiones por encima de este nivel requieren atención médica, porque una alta presión sanguínea prolongada puede causar lesiones en el corazón o en otros órganos antes de que una persona se dé cuenta de este problema. En los últimos años se ha dado un gran relieve

a la realización de exámenes en masa para descubrir a la gente que tiene presión sanguínea elevada sin saberlo.

Es muy importante destacar que la presión varía de latido a latido, durante el día y la noche, frente a situaciones cotidianas como caminar, hablar por teléfono o realizar ejercicios. Por lo tanto, la variación de los valores de la presión arterial Tabla 1.4, es un fenómeno normal. La hipertensión es una enfermedad muy común en todo el mundo que afecta a más del 20 por ciento de los adultos entre 40 y 65 años y casi al 50 por ciento en las personas de más de 65 años; los valores de la hipertensión se puede observar en la Tabla 1.5. [8].

Tabla 1.4: Valores normales de la Presión Arterial

	Sistólica	Diastólica
Óptima	< 120 mm/Hg	< 80 mm/Hg
Normal	< 130 mm/Hg	< 85 mm/Hg
Normal – Alta	130 – 139 mm/Hg	85 – 89 mm/Hg

[9]

Tabla 1.5: Valores de la Hipertensión

	Sistólica	Diastólica
Leve	140 – 159 mm/Hg	90 – 99 mm/Hg
Moderada	160 – 179 mm/Hg	100 – 109 mm/Hg
Severa	≥ 180 mm/Hg	≥ 110

[9]

1.2.3 Pulsioximetría

La pulsimetría es una técnica de monitorización continua de la saturación de oxígeno de la sangre arterial. El pulsioxímetro determina la saturación arterial de O_2 (saO_2) a partir de la medición de la tasa de absorción de la luz de

la sangre arterial; la mayor diferencia en los espectros de absorción entre los dos tipos de hemoglobina se da en el rango de dos longitudes de onda específicas, 660nm (roja) y 940nm (infrarroja), ya que la oxihemoglobina absorbe más luz infrarroja mientras que la desoxihemoglobina absorbe más luz roja. Las longitudes de onda a las que emiten los leds pueden tener alguna pequeña variación dependiendo del fabricante, tal como se muestra en la Tabla 1.6, pero son generalmente de este orden, el rojo está en el rango de 630 – 660nm y el infrarrojo 800 – 940nm, adicional en la Figura 1.7 se puede observar el espectro visible a la región del espectro electromagnético que el ojo humano puede percibir, no existe límites exactos en la percepción del ojo humano pero un ojo normal responde a longitudes de onda de 400 a 700nm.

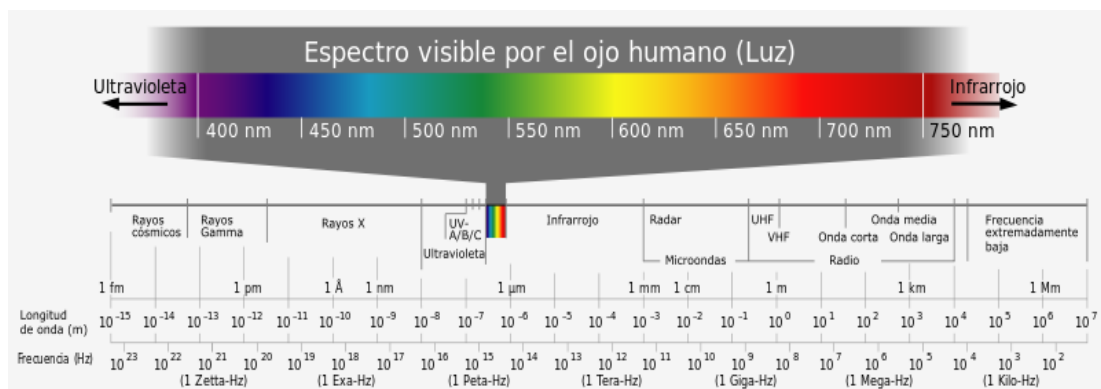


Figura 1.7: Espectro electromagnético

[10]

Tabla 1.6: Características del sensor según el fabricante

Sensor	Longitud de onda	Potencia
Rojo	662nm	1.8mW
Infrarrojo	905nm	2.0mW (Nellcor, Datex, CSI, BCI)
	940nm	1.5mW (Ohmeda, Novamatrix)

[11]

La comparación entre la absorción correspondiente a cada longitud de onda permite calcular el porcentaje de **Hemoglobina** oxigenada. La

absorción de los tejidos incluye sangre arterial, capilar y venosa en porcentajes que no se conocen. Para determinar el porcentaje de hemoglobina oxigenada correspondiente a la sangre arterial se utiliza un programa informático relativamente simple, el cual mide solo la absorción simultánea con la oscilación del pulso arterial. La absorción de luz por parte de la hemoglobina que no está asociada al pulso arterial, o sea, el valor de absorción de fondo que corresponde a la sangre mezclada, se resta y permanece sólo el valor de la sangre arterial.

La técnica descrita entrega, por lo tanto, un valor que es sólo una inferencia y no una medición directa. Existen numerosas posibilidades de error que deben ser tenidas en cuenta.

La primera es el mal contacto entre el diodo emisor de luz, la célula fotoeléctrica, y la piel del paciente, por desplazamiento del sensor. Especialmente en el caso de pacientes agitados, los sensores puestos en los dedos duran poco tiempo en posición y la lectura de saturación se hace intermitente.

Una Segunda causa frecuente de mala lectura es **Vasoconstricción** periférica, debido a compromiso **Hemodinámico** del paciente o a frío. En efecto, la respuesta termorreguladora al frío incluye vasoconstricción marcada de manos, pies, nariz y orejas, que son lugares frecuentes de localización del sensor. La disminución de la amplitud de pulso hace menos fiable la lectura del sistema y posibles errores en la determinación de la saturación de oxígeno.

Una tercera causa es el exceso de luz ambiente sobre el sensor, a que la variable medida es la absorción de luz. Si en el ambiente hay luz de las mismas longitudes de onda que las que registra el sistema, entonces las lecturas serán erróneas.

La existencia de campos electromagnéticos, como los emitidos por una unidad de electrocirugía, pueden afectar a la lectura de saturación de oxígeno de forma marcada.

Una última causa, el cable que se conecta el sensor al paciente puede haber perdido continuidad por el uso repetido y resultar en una señal intermitente o errónea.

Existen comunicaciones anecdóticas sobre daños a pacientes originados en el contacto de los sensores de oximetría con la piel, por temperatura o por contacto eléctrico. Se trata sin embargo, de eventos excepcionales. En general esta tecnología no representa riesgo para los pacientes.

A pesar de sus limitaciones la Oximetría de pulso proporciona información sobre la llegada efectiva de oxígeno a los tejidos, o sea, al punto final de la función cardiorrespiratoria. De este modo, refleja globalmente la indemnidad de los sistemas de transferencia de oxígeno a los pulmones, de estos a la sangre, y la llegada efectiva de esta a los tejidos periféricos. Esta información es importante, su costo es bajo y su riesgo prácticamente cero. [12]

1.2.4 Espirómetro

La espirometría es una prueba médica de tamizaje que va medir varios aspectos de la función respiratoria y del pulmón. Se lleva a cabo utilizando un espirómetro, un dispositivo especial que registra la cantidad de aire que un sujeto inhala o exhala así como la velocidad a la cual dicho aire es desplazado hacia fuera o dentro del pulmón. [13] El objetivo de la espirometría es valorar la función pulmonar, y en general es una exploración básica de los pacientes con sospecha de patologías respiratorias.

Existen varios tipos de espirómetros pero se clasifican en dos grupos: 1) Aquellos que registran la cantidad de aire exhalado o inhalado en un determinado intervalo de tiempo (espirómetros de volumen) y 2) aquellos que miden que tan rápido fluye el aire cuando se desplaza hacia adentro o hacia afuera del pulmón, conforme se incrementa el volumen de gas inhalado o exhalado (espirómetros de flujo). [13]

Dentro de los espirómetros de volumen tenemos el de campana o de agua, de pistón y de fuelle; dentro de los espirómetros de flujo tenemos los

neumotacógrafos de tipo fleisch, lilly y desechable, espirómetros de hilo caliente, de ultrasonidos y el espirómetro de turbina, obsérvese la Tabla 1.7.

Tabla 1.7: Ventajas y desventajas de los siguientes tipos de Espirómetros.

Tipo de espirómetro	Ventajas	Desventajas
De agua o de campana	Fácil de usar. Fiable, preciso y reproducible. Proporciona copia en papel.	No se puede transportar. Requiere mantenimiento por técnicos. Difícil de limpiar si se contamina. Si no tiene microprocesador, deben hacerse los cálculos manualmente.
De pistón o de fuelle	Fácil de usar. Fiable, preciso y reproducible. Proporciona copia en papel.	Transporte difícil por su tamaño. Difícil de limpiar si se contamina. Si no tiene microprocesador, deben hacerse los Cálculos manualmente. Puede descalibrarse si se mueve.
Neumotacógrafo	Fácil de usar. Fiable, preciso y reproducible. Ligero y de reducido tamaño. Fácilmente transportable.	Puede afectarse por la temperatura o por la condensación. Necesita una impresora o un ordenador para imprimir las curvas. Requiere limpieza cuidadosa.
De Hilo caliente	Fácil de usar. Fiable y reproducible. Ligero y de reducido tamaño. Fácilmente transportable. Fácil de limpiar. Relativamente barato.	Puede afectarse por la Temperatura. Necesita una impresora o un ordenador para imprimir las curvas. Puede no ser siempre preciso.
De ultrasonidos	Fácil de usar. Fiable y reproducible.	Necesita una impresora

CONTINUA 

	Ligero y de reducido tamaño. Fácilmente transportable. Fácil de limpiar. No tiene partes móviles.	o un ordenador para imprimir las curvas. Relativamente caro.
De Turbina	Fácil de usar. Reproducibile. Ligero y de reducido tamaño. Fácilmente transportable. Fácil de limpiar. Relativamente barato.	Si el diseño no es bueno, puede infraestimar o supraestimar las medidas. Necesita una impresora o un ordenador para imprimir las curvas. Infraestima los volúmenes a flujos bajos. Puede no ser siempre preciso.

[14]

Los espirómetros descritos dibujan directamente una gráfica volumen-tiempo. A partir de esta gráfica pueden determinarse manualmente algunos índices espirométricos de interés, tales como la capacidad vital forzada (FVC) o el volumen espirado durante el primer segundo (FEV_1). Otros índices, por ejemplo el flujo en el instante en el que se ha expirado el 50% de FVC ($FEF_{50\% FVC}$) o el valor medio de tiempo de tránsito (MTT), exigen el cálculo mediante circuitos electrónicos analógicos u ordenador y las condiciones de normalización exigen que el error del volumen medido con el espirómetro sea inferior al 2%. Para cubrir estas necesidades de manera óptima se eligió el espirómetro de turbina.

Espirómetro de turbina

Este tipo de espirómetros tienen un cabezal con un eje sobre el que gira una pequeña hélice, en los extremos del cabezal hay unas aspas fijas que ordenan el flujo de aire al penetrar en el cabezal. El flujo de aire hace girar la

hélice, y las aspas de esta interrumpen una fuente de luz infrarroja en cada paso que hagan. La velocidad de giro de la hélice es proporcional al flujo, y por tanto, a más flujo, más veces se interrumpirá la señal luminosa, esta información se dirige al microprocesador que en función de las revoluciones de la hélice calcula el flujo y luego por integración los volúmenes, Figura 1.86 y Figura 1.97. Este tipo de espirómetros están muy extendidos actualmente ya que son relativamente baratos.

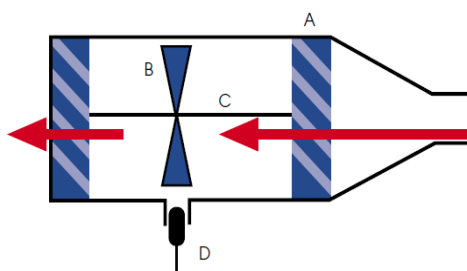


Figura 1.8: Esquema de un espirómetro de turbina A) aspas fijas para dirigir el flujo de aire, B) Hélice (turbina), C) Eje de la hélice, D) Sensor óptico para el movimiento de la hélice.

[14]



Figura 1.9: Espirómetro de turbina
Elaborado por: Gómez O, Silva I.

La espirometría consta de una serie de pruebas respiratorias sencillas, bajo circunstancias controladas, que miden la magnitud absoluta de las capacidades pulmonares como se explica en la Tabla 1.8, y los volúmenes pulmonares en la Tabla 1.9, y la rapidez con que éstos pueden ser movilizados (flujos aéreos), los resultados se representan en forma numérica fundamentados en cálculos sencillos y en forma de impresión gráfica.

Tabla 1.8: Capacidad Pulmonar valores normales

Capacidad inspiratoria (CI)	3500mL aproximadamente
Capacidad residual funcional (CRF)	2300mL aproximadamente
Capacidad vital (CV)	Alrededor de 4 litros
Capacidad pulmonar total (CPT)	6 litros aproximadamente

[15]

Tabla 1.9: Volumen Pulmonar valores normales

Volumen de corriente o tidal (VC o VT)	500mL
Volumen de reserva inspiratorio (VRI)	3000mL
Volumen de reserva espiratorio (VRE)	1100mL
Volumen residual (VR)	1200mL

[15]

1.3 Estándar Bluetooth (802.15.1)

La tecnología Bluetooth define un estándar de comunicaciones inalámbricas de corto alcance mediante señales de radiofrecuencia que permite la transmisión de datos y voz. El desarrollo de esta tecnología parte de los laboratorios de Ericsson Mobile Communications, que en 1994 decidieron desarrollar una tecnología que permitiera la conexión mediante un interfaz radio, de consumo, coste y tamaño reducido, de los teléfonos móviles con sus accesorios. Una vez iniciado el proyecto se pensó que esta tecnología podía utilizarse para eliminar los cables que interconectan dispositivos como impresoras, teclados, ordenadores, teléfonos móviles, etc., de forma transparente y amigable para los usuarios. En febrero de 1998 se creó el SIG (Special Interest Group), formado por Ericsson, Nokia, IBM, Toshiba e Intel, para promover un estándar abierto para el interfaz radio y un conjunto de protocolos y de interoperatividad que permitiese la compatibilidad entre los equipos de los diversos fabricantes. Ya en el 2003 el SIG agrupa a más de 2000 compañías fabricantes de chips, desarrolladores de software o de sistemas electrónicos. La extensión de los modos de utilización de esta tecnología permite su uso en pequeñas redes de datos, engloba dentro de las

denominadas WPAN (Wireless Personal Area Networks), junto con otras tecnologías como HomeRF 802.15 o UWB (Ultra Wide Band). Bluetooth se presenta como uno de los candidatos para el desarrollo de redes Adhoc o redes híbridas.

En la Figura 1.10 se muestra un conjunto de aplicaciones posibles mediante la tecnología Bluetooth. Básicamente se trata de la sustitución de los cables por un enlace radio creado mediante Bluetooth. Mediante esta tecnología se puede acceder desde un ordenador, una cámara fotográfica o cualquier otro dispositivo electrónico a otro dispositivo Bluetooth situado a un teléfono móvil como punto de acceso a la red GSM/GPRS o UMTS. También permite la interconexión de ordenadores creando redes Adhoc. Otra de las aplicaciones es la sustitución de los cables, RS-232, audio, etc., que conectan distintos dispositivos electrónicos entre sí. [16]

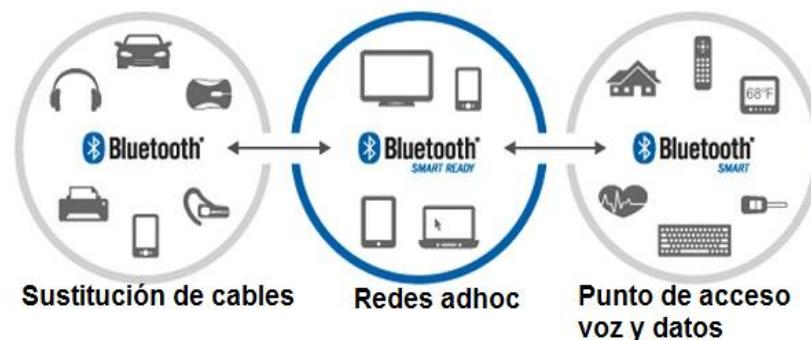


Figura 1.10: Aplicaciones del Bluetooth
[17]

Las especificaciones definen un enlace radio que permite establecer enlaces de corto alcance, hasta unos 10 metros u opcionalmente hasta algunos centenares de metros, de voz mediante enlaces síncronos (SCO), y de datos bidireccionales mediante enlaces asíncronos (ACL). Cada canal de voz puede soportar una tasa de transferencia de 64 Kbits/s en cada sentido, suficiente para la transmisión de voz. Un canal asíncrono puede transmitir hasta 721 Kbits/s en una dirección y 56 Kbits/s en la dirección opuesta en enlaces asimétricos. Por otro lado, para una conexión asíncrona es posible soportar 432,6 Kbits/s en ambas direcciones si el enlace es simétrico.

La frecuencia de radio de operación es la banda ISM de 2,4 GHz, la misma que las redes WLAN 802.11b y 802.11g. Funcionalmente el sistema Bluetooth ha sido diseñado para sustituir los cables a bajo coste. Comparativamente con las redes WLAN el sistema Bluetooth tiene la misma cobertura, menor velocidad, pero un coste de producción mucho menor. WLAN está diseñado como complemento y/o sustituido de las redes de datos cableadas. En cambio, Bluetooth está diseñado para la sustitución de los cables de interconexión de los dispositivos.

Debido a que la banda ISM es de libre uso, cumpliendo ciertas restricciones, la señal de radiofrecuencia del sistema Bluetooth deberá estar preparada para que las múltiples interferencias que se pudieran producir no mermen su capacidad. Para ello Bluetooth utiliza el método de salto de frecuencia debido a que esta tecnología puede ser integrada en equipos de baja potencia y bajo costos. Este sistema divide la banda de frecuencia en varios canales de salto: los transceptores, durante la conexión, van cambiando de uno a otro canal de salto de manera **Pseudoaleatoria**. La potencia de transmisión se especifica según tres tipos de clases de dispositivos: 1 mW para alcances inferiores a 5 metros, 2,5 mW para alcances de hasta 30 metros, y hasta 100 mW para cobertura de hasta 300 metros.

La topología de las redes Bluetooth puede ser punto a punto, o punto a multipunto, con todos los dispositivos iguales. Si un equipo se encuentra dentro del radio de cobertura de otro, este puede establecer conexión con cualquiera de ellos. El control del enlace lo asume la unidad que ha iniciado la conexión según un protocolo de maestro-esclavo. De todas formas, los dispositivos pueden intercambiar el control de la conexión pasando el dispositivo que actúa como maestro a esclavo y viceversa. Un dispositivo que actúa como maestro puede estar conectado de forma simultánea hasta con siete dispositivos esclavos Bluetooth. Cuando dos o más dispositivos Bluetooth establecen una conexión a través de un único dispositivo que actúa como maestro forman una Piconet. Cada Piconet establece una secuencia de salto de frecuencia que depende de la dirección y de un reloj interno del dispositivo maestro. También existen diversos estados de bajo consumo en

los que los dispositivos esclavos son aparcados, aunque se mantienen sincronizados, a la espera de la reiniciación del enlace tal y como se muestra en la Figura 1.11. Los dispositivos que no establecen ningún tipo de conexión están en un estado de espera.

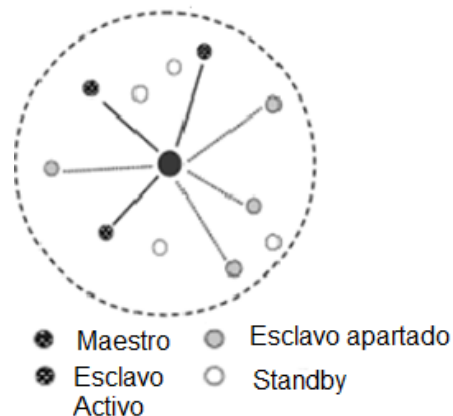


Figura 1.11: Estructura de Piconet.
[16]

Las unidades Bluetooth pueden establecer potencialmente comunicaciones entre ellas cuando las aplicaciones que los gestionan lo requieran, de modo que en una misma zona pueden existir distintos dispositivos que establezcan comunicaciones simultáneas entre sí. Esto provocará que se creen varias Piconets, cada una con una frecuencia de salto distinto en áreas de cobertura superpuestas, tal y como se muestran en la Figura 1.12.

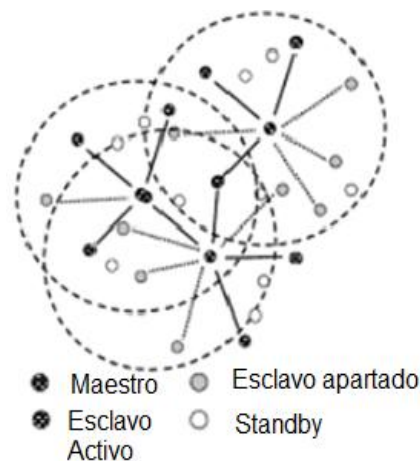


Figura 1.12: Estructura de Scatternet.
[16]

A un grupo de Piconets se le denomina Scatternet. Debido a que individualmente cada Piconet tiene un salto de frecuencia diferente, diferentes Piconets, pueden coincidir sin interferirse. De todas formas si en un área reducida se establecen muchas Piconets, más de 20 de forma simultánea a máxima velocidad, el rendimiento de cada una de ellas disminuirá. Las distintas Piconets pueden intercomunicarse puesto que un dispositivo puede ser maestro de una Piconet y esclavo de otra, o esclavo de dos Piconets distintas.

La seguridad de las transmisiones y de los enlaces entre dispositivos se obtiene mediante:

1. Saltos de frecuencia pseudoaleatorios que dificultan que dispositivos ajenos a la red puedan interceptar o ver el tráfico de información.
2. Autenticación, que permite a un usuario controlar la conectividad sólo para dispositivos especificados.
3. Encriptación, mediante el uso de claves secretas.

La implantación del sistema Bluetooth ha sido más difícil de lo que los impulsores tenían previsto. La falta de apoyo en algunos momentos de alguno de los grandes fabricantes y la competencia que se estableció con el IEEE, con sus estándares 802.11 y 802.15, provocó una ralentización en la aparición de productos de forma masiva. Sin embargo, el acuerdo para adoptar la tecnología Bluetooth como parte del estándar IEEE 802.15.1 propició que los primeros productos de consumo comenzasen a distribuirse en el año 2002. Es en el año 2003 en que la implantación comienza a ser masiva, primero en equipos de gama alta, ordenadores portátiles, cámaras de video, impresoras, etc., y más tarde en equipos destinados a todos los usuarios: adaptadores de puerto serie, teléfonos móviles, agendas electrónicas, etc.

El potencial de productos en los que la tecnología es aplicable es enorme, pues cualquier sistema que utilice la transmisión de datos de hasta 700 Kbits/s puede incorporar Bluetooth para eliminar los cables de conexión. Los miembros del SIG están trabajando en la ampliación del estándar para adaptarlo a las nuevas necesidades multimedia y conexiones de datos de alta

velocidad. Para ello, se pretende ampliar el interfaz radio, manteniendo compatibilidad con los productos anteriores, para alcanzar velocidades de transmisión de varios Mbits/s.

1.4 Arduino UNO

Arduino Uno (Anexo C) es una placa con un microcontrolador de la marca Atmel y con toda la circuitería del soporte, que incluye reguladores de tensión, un puerto USB conectado a un módulo adaptador USB-serie que permite programar el microcontrolador desde cualquier PC de manera cómoda y también hacer pruebas de comunicación con el propio chip.

Cada uno de los 14 pines digitales se puede usar como entrada o salida. Funcionan a 5V, cada pin puede suministrar hasta 40mA.

Cada uno de los pines digitales dispone de una resistencia de pull-up interna de entre 20K Ω y 50K Ω que esta desconectada, salvo que se indique lo contrario.

La tarjeta también dispone de 6 pines de entrada analógicos que trasladan a un conversor analógico digital de 10 bits.

1.4.1 Pines especiales de entrada y salida

- RX Y TX: Se usan para transmisiones serie de señales TTL.
- Interrupciones externas: Los pines 2 y 3 están configurados para generar una interrupción en el Atmega328. Las interrupciones pueden dispararse cuando se encuentra un valor bajo en estas entradas y con flancos de subida o bajada de la entrada.
- PWM: Arduino dispone de 6 salidas destinadas a la generación de señales PWM de hasta 8 bits.
- SPI: Los pines 10,11,12 y 13 pueden utilizarse para llevar a cabo comunicaciones SPI, que permite trasladar información full dúplex en un entorno Maestro/Esclavo.
- I²C: Permite establecer comunicaciones a través de un bus I²C. El bus I²C es un producto de Phillips para interconexión de sistemas

embebidos. Actualmente se puede encontrar una gran diversidad de dispositivos que utilizan esta interfaz, desde pantallas LCD, Memorias EEPROM, Sensores.

Puede alimentarse a través del propio cable USB o mediante una fuente de alimentación externa, como puede ser un pequeño transformador o, por ejemplo una pila de 9V. Los límites están entre los 6 y 12 voltios. Como única restricción hay que saber que si la placa se alimenta con menos de 7V la salida del regulador de tensión a 5V puede dar menos que este voltaje y si sobrepasamos los 12 voltios, probablemente dañaremos la placa.

Hay que tener en cuenta que podemos medir el voltaje presente en el Jack directamente desde Vin y GND marcados sobre la placa. Se puede visualizar de mejor manera las características en la Tabla 1.10.

Tabla 1.10: Características de la tarjeta Arduino UNO

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada – Salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz.

1.4.2 Programación.

El Arduino UNO puede ser programado con la última versión del software Arduino.

El programa se implementará haciendo uso del entorno de programación propio de Arduino y se transferirá empleando un cable USB por el puerto de programación como se ve en la Figura 1.13. Si bien en el caso de la placa USB no es preciso utilizar una fuente de alimentación externa, ya que el propio cable USB la proporciona.

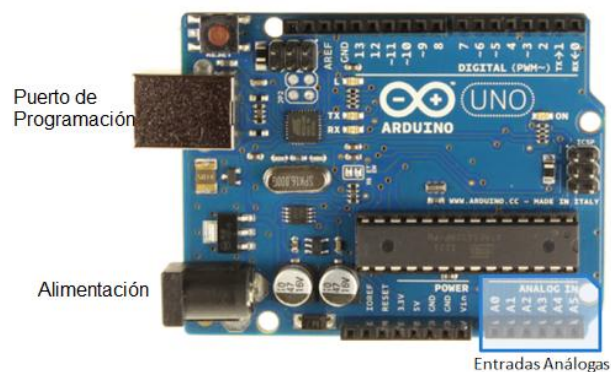


Figura 1.13: Tarjeta Arduino UNO y su puerto de programación.

[19]

1.5 Plataforma Android

Android es un sistema operativo para dispositivos móviles. Está basado en GNU/Linux e inicialmente fue desarrollado por Google. La presentación de la plataforma Android se realizó el 5 de noviembre de 2007 junto con la fundación Open Handset Alliance, un consorcio de 48 compañías de hardware, software y telecomunicaciones comprometidas a la promoción de estándares abiertos para dispositivos móviles.

Esta plataforma permite el desarrollo de aplicaciones por terceros (personas ajenas a Google), para lo cual, los desarrolladores deben de escribir código gestionado en el lenguaje de programación Java y controlar los dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google, es decir, escribir programas en C u otros lenguajes, utilizando o no las bibliotecas de Google (compilándolas a código nativo de ARM). Sin embargo, este esquema de desarrollo no es oficialmente soportado por Google.

La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto.

1.6 Plataforma de desarrollo Java

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de las aplicaciones, y un conjunto de bibliotecas estándar que ofrecen una funcionalidad común.

Desde 2006, la versión actual de la Plataforma Java Standard Edition se le conoce como Java SE 6 como versión externa, y 1.6 como versión interna. Sin embargo, se prefiere el término versión 6. Una visión general de la multitud de tecnologías que componen la Plataforma Java puede encontrarse en la página de documentación del JDK.

Las características principales que nos ofrece Java respecto a cualquier otro lenguaje de programación, son:

- Es simple.
- Es orientado a objetos.
- Es distribuido.
- Es robusto.
- Es seguro.
- Es portable.

CAPÍTULO 2

2 DISEÑO E IMPLEMENTACIÓN

En este capítulo se detalla los pasos a seguir para la fabricación de cada una de las placas que constituye el Polisomnógrafo y la correcta utilización de los sensores para captar las señales biofísicas y bioeléctricas del cuerpo humano.

2.1 Diagrama de bloques del proyecto a realizar

El proyecto tiene varias etapas funcionales como se puede visualizar a continuación en la Figura 2.1.

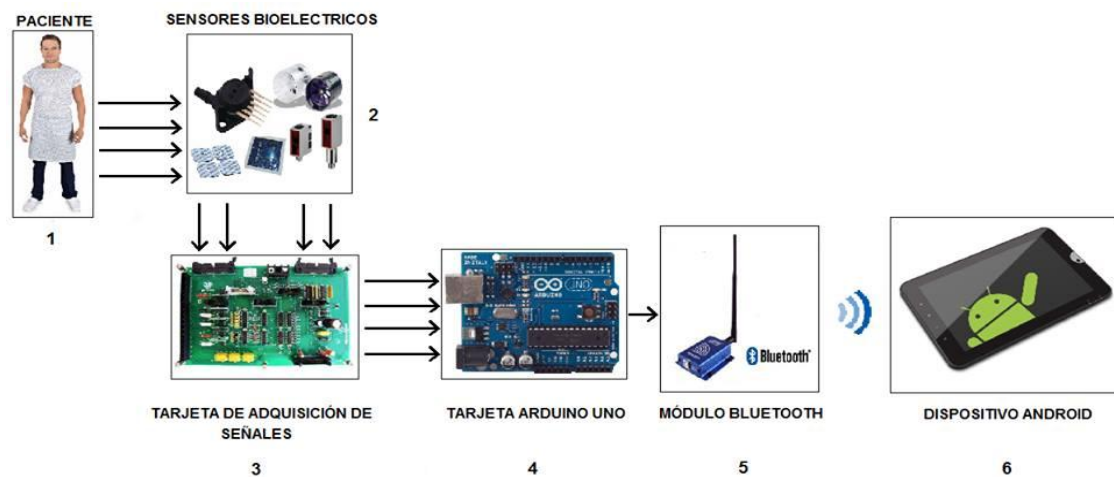


Figura 2.1: Diagrama de bloques general para la implementación de un Polisomnógrafo con transmisión de datos inalámbricos.

Elaborado por: Gómez O, Silva I.

En el primer y segundo bloque se encuentra el paciente a quien se le colocan los sensores bioeléctricos en posiciones determinadas del cuerpo (1.2) para lograr registrar las señales biofísicas y bioeléctricas producto de la actividad de los signos vitales. A continuación la etapa de amplificación y prefiltrado que se realiza en la tarjeta de adquisición de señales, que permite a la adquisición de las señales captadas del cuerpo humano y la eliminación de la interferencia inherente a la medición de potenciales tan minúsculos de un ECG y la saturación de oxígeno. Después se digitaliza estas señales con

la ayuda de la tarjeta Arduino UNO (1.4), para poder ingresar al módulo Bluetooth y transmitirla inalámbricamente a un dispositivo Android donde se visualizará las señales correspondientes a los signos vitales.

2.2 Implementación del ECG

En el desarrollo del ECG se realizó la adquisición de las señales del corazón con la primera derivación de Einthoven Figura 1.1 la cual no necesita el acople del seguidor de tensión ya que no se genera una corriente que regresa al circuito, para ello se utiliza electrodos, amplificadores de instrumentación filtros analógicos y un amplificador de ganancia como se lo ve en el diagrama de bloques de la Figura 2.2 y en la Figura 2.3 se muestra el circuito de adquisición para que estas señales puedan ingresar al conversor análogo digital de la tarjeta Arduino.

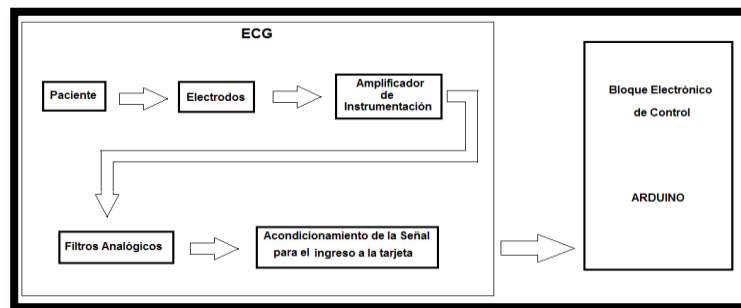


Figura 2.2: Diagrama de Bloques del ECG.
Elaborado por: Gómez O, Silva I.

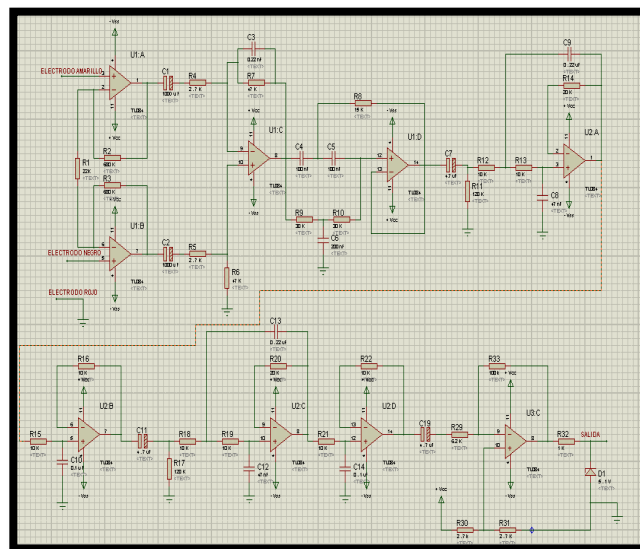


Figura 2.3: Diagrama Esquemático de la adquisición de la señal del ECG.
Elaborado por: Gómez O, Silva I.

2.2.1 Electroodos

Los electrodos, son los elementos primarios de un ECG que convierten las corrientes iónicas del cuerpo humano en corrientes eléctricas, estos cumplen con las siguientes características:

- Baja impedancia.
- No produzca efectos secundarios en el paciente.
- Potencial de contacto estable y pequeño.
- Higiénicos.
- Duradero en el tiempo.
- Transformar corrientes con poca pérdida de información. [20]

Cuando los electrodos no están en una posición estable con respecto a la piel se generan artefactos (2.6.1) que produce un error en la señal del ECG. [20]. Para obtener una señal pura es recomendable rasurar al paciente en las áreas donde se colocarán los electrodos, después limpiar la superficie de la piel con alcohol y al final colocar el gel conductor este último ayuda a disminuir la impedancia producida por la dermis, el movimiento de los músculos generan una fuente potencial de error contaminando la señal del ECG, para ello el paciente debe estar en un perfecto estado de reposo.

2.2.2 Amplificador de Instrumentación

El amplificador de instrumentación Figura 2.4 es un amplificador diferencial, cuya ganancia puede establecerse de forma muy precisa y ha sido optimizado para que opere de acuerdo a sus propias especificaciones aún en un entorno hostil. Es un elemento esencial de los sistemas de medida, en los que se ensambla como un bloque funcional que ofrece características funcionales propias e independientes de los restantes elementos con los que interacciona. [21]

Estos circuitos amplifican la diferencia entre dos señales de entrada y rechazan cualquier señal que sea común a ambas señales. Estos circuitos se utilizan principalmente para amplificar señales diferenciales muy pequeñas en

muchos procesos industriales, medición, adquisición de datos y aplicaciones médicas.

Ante las exigencias de medida que imponen los sensores, estos circuitos deben cumplir unos requisitos generales:

- Ganancia: seleccionable, estable y lineal.
- Entrada diferencial: con CMMR alto.
- Error despreciable debido a las corrientes y tensiones offset.
- Impedancia de entrada alta.
- Impedancia de salida baja. [21]

Para trabajar con la tarjeta Arduino la magnitud de la señal bioeléctrica debe ser amplificada ya que la de un paciente es demasiado pequeña, la cual varía entre los 0.5mV y 4mV en función del tiempo-nivel Figura 1.5, a esta señal se le amplifica dándole una ganancia de 1000.

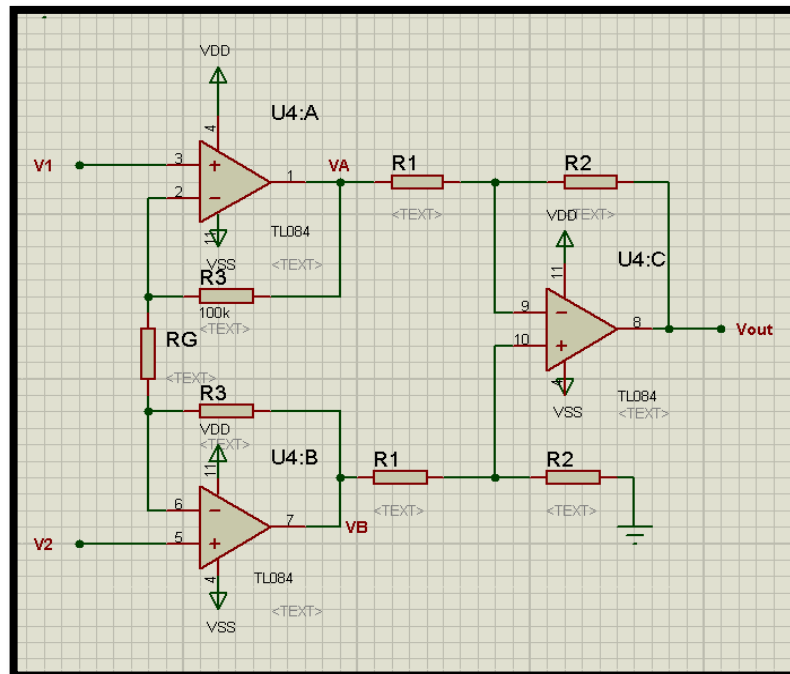


Figura 2.4: Diagrama del Amplificador de Instrumentación basado en el integrado TL084.

Elaborado por: Gómez O, Silva I.

La ganancia del amplificador de instrumentación está dada por la siguiente ecuación:

$$\frac{V_0}{V_2 - V_1} = G = \left(\frac{R_2}{R_1}\right) \left(\frac{2R_3}{R_G} + 1\right) \quad \text{Ecuación (2.1)}$$

Tenemos la ganancia de $G = 1000$ y si $R_3 = 680 \text{ K}\Omega$. $R_G = 22 \text{ K}\Omega$.

$$1000 = \left(\frac{R_2}{R_1}\right) \left(\frac{2 * 680 \text{ K}\Omega}{22 \text{ K}\Omega} + 1\right)$$

$$1000 = \left(\frac{R_2}{R_1}\right) * 63$$

$$\left(\frac{R_2}{R_1}\right) = 16$$

Si $R_2 = 47 \text{ K}\Omega$. $\Rightarrow R_1 = 2.7 \text{ K}\Omega$.

Reemplazamos el resultado final de las resistencias:

$R_3 = 680 \text{ K}\Omega$. $R_G = 22 \text{ K}\Omega$. $R_2 = 47 \text{ K}\Omega$. $R_1 = 2.7 \text{ K}\Omega$.

$$G = \left(\frac{47 \text{ K}\Omega}{2.7 \text{ K}\Omega}\right) \left(\frac{2 * 680 \text{ K}\Omega}{22 \text{ K}\Omega} + 1\right)$$

$$G = 1076.094 \approx 1000$$

Un dato relevante es el rango de frecuencia el cual tenemos que tomar en cuenta las señales eléctricas extracardíacas como es (corriente alterna, movimiento muscular y la respiración) ya que estas pueden interferir en la medición del ECG, para obtener lo más importante de la señal hay que establecer el rango de frecuencia entre 0.05 a 150Hz, al añadir capacitores al diseño del amplificador de instrumentación diseñamos un filtro pasa banda dentro de este amplificador con el fin de evitar señales ajenas al sistema.

$$f_{corte} = \frac{1}{2\pi RC} \quad \text{Ecuación (2.2)}$$

$$f_{corte \text{ inf.}} = 0.05 \text{ Hz.}$$

$$f_{corte \text{ sup.}} = 150 \text{ Hz.}$$

$$C = \frac{1}{2\pi R f_{corte}}$$

$$C_{sup.} = \frac{1}{2\pi R f_{corte}}$$

$$C_{inf.} = \frac{1}{2\pi R f_{inf.}}$$

$$C_{sup.} = \frac{1}{2\pi R f_{sup.}}$$

$$C_{inf.} = \frac{1}{2\pi * 2.7K\Omega * 0.05Hz.}$$

$$C_{sup.} = \frac{1}{2\pi * 47K\Omega * 150Hz.}$$

$$C_{inf.} = 1000 \mu F$$

$$C_{sup.} = 0.22nF$$

Al reemplazar los valores de las resistencias y capacitores calculados se puede establecer el amplificador de instrumentación pasa banda en la Figura 2.5.

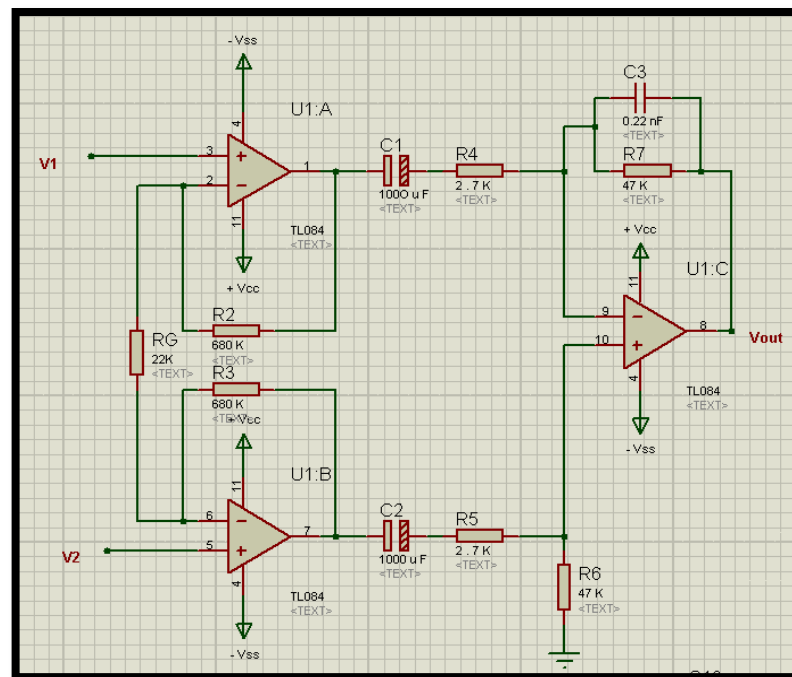


Figura 2.5: Amplificador de Instrumentación pasa banda basado en el integrado TL084.

Elaborado por: Gómez O, Silva I.

Existe una señal de ruido que se induce antes del amplificador de instrumentación, pero el filtro pasa banda diseñado dentro de este amplificador es capaz de eliminar el ruido inducido obteniendo a la salida del amplificador de instrumentación solo la señal amplificada del latido cardíaco.

2.2.3 Implementación de los Filtros Analógicos

El ruido del ambiente puede interferir en la señal del ECG, y para conseguir una señal con el mínimo ruido se ha desarrollado un conjunto de filtros que permite obtener una señal electrocardiográfica aceptable.

Para reducir las frecuencias indeseables o ruido generado por el ambiente que se introduce en nuestra señal electrocardiográfica, se emplea dos tipos de filtros un filtro rechaza banda Figura 2.6 que está centrada en la frecuencia de rechazo de 60Hz. Esta frecuencia es generada por la línea de potencia, este circuito se ve expuesto al ruido ambiental que proviene de las lámparas fluorescentes y otros ruidos que emiten ruido a través de ondas de 60Hz. El diseño de este filtro se encargara de rechazar exclusivamente el ruido de 60Hz. De la línea de potencia 110Ac. El otro filtro es uno pasabanda de (0.5-150Hz), de tal manera que se emplea un filtro de sexto orden Buttherworth Figura 2.8.

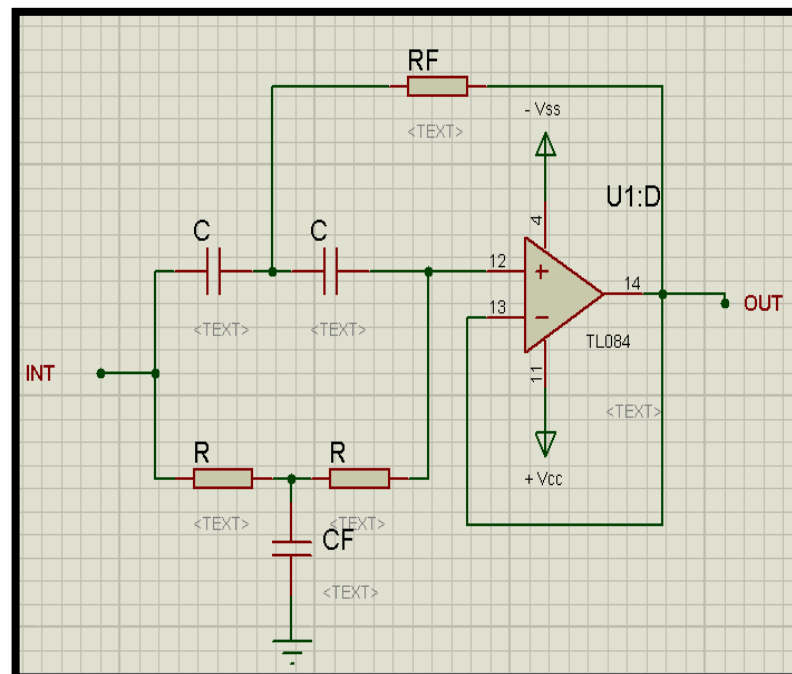


Figura 2.6: Filtro rechaza banda.
Elaborado por: Gómez O, Silva I.

$$Rf = \frac{R}{2} \quad y \quad Cf = 2C$$

$$\text{Si } R = 30K\Omega$$

$$\rightarrow R_f = 15K\Omega.$$

$$\text{Si } C = 100nF$$

$$\rightarrow C_f = 200nF$$

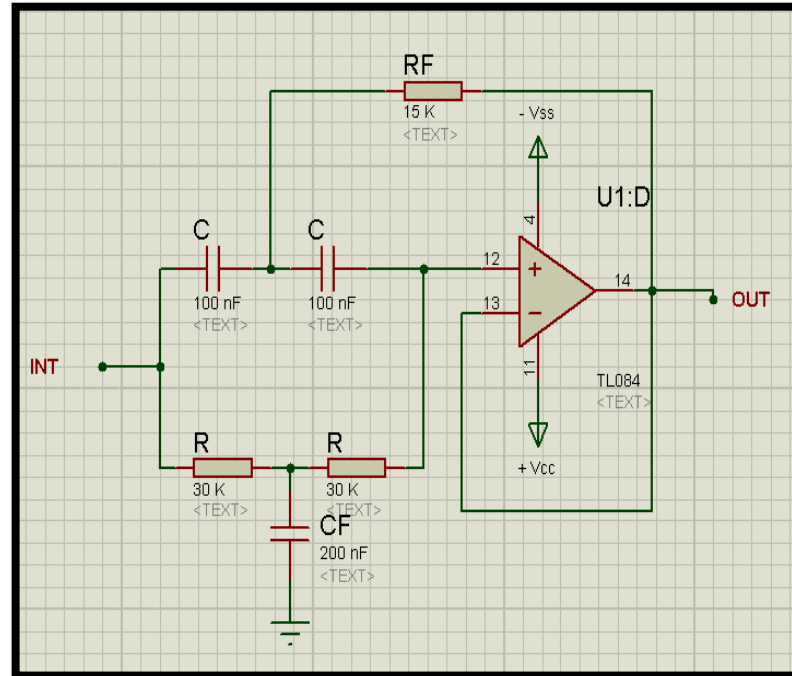


Figura 2.7: Filtro rechaza banda con los valores del diseño.
Elaborado por: Gómez O, Silva I.

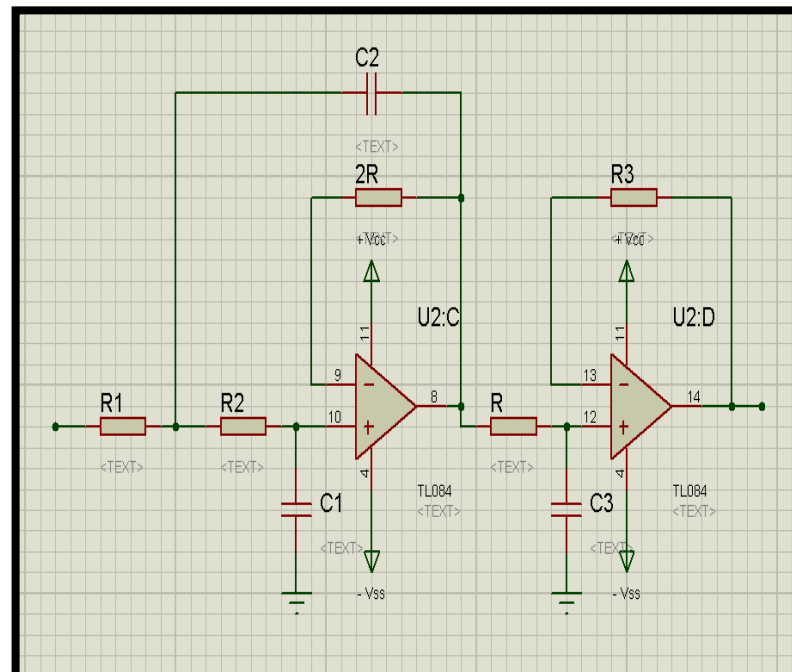


Figura 2.8: Filtro de Butterworth.
Elaborado por: Gómez O, Silva I.

$$C_1 = \frac{1}{2}C_3$$

$$C_2 = 2C_3$$

$$R = \frac{1}{WC_3}$$

$$R = R_1 = R_2 = R_3$$

$$Rf_1 = 2R$$

$$Rf_2 = R$$

Si $C_3 = 0.1\mu F$ y como $f_c = 150\text{Hz}$

$$R = \frac{1}{WC_3} = \frac{1}{2\pi f * C_3}$$

$$R = \frac{1}{2\pi * 150\text{Hz} * 0.1\mu F} = 10\text{K}\Omega.$$

$$R = 10\text{K}\Omega.$$

$$R = R_1 = R_2 = R_3$$

$$\rightarrow R = R_1 = R_2 = R_3 = 10\text{K}\Omega$$

$$Rf_1 = 2R = 2 * 10\text{K}\Omega. = 20\text{K}\Omega.$$

$$\rightarrow Rf_1 = 20\text{K}\Omega.$$

$$Rf_2 = R \rightarrow Rf_2 = 10\text{K}\Omega.$$

$$C_1 = \frac{1}{2}C_3 = \frac{1}{2} * 0.1\mu F = 50\text{nF}$$

$$\rightarrow C_1 = 47\text{nF}$$

$$C_2 = 2C_3 = 2 * 0.1\mu F = 0.2\mu F$$

$$\rightarrow C_2 = 0.22\mu F$$

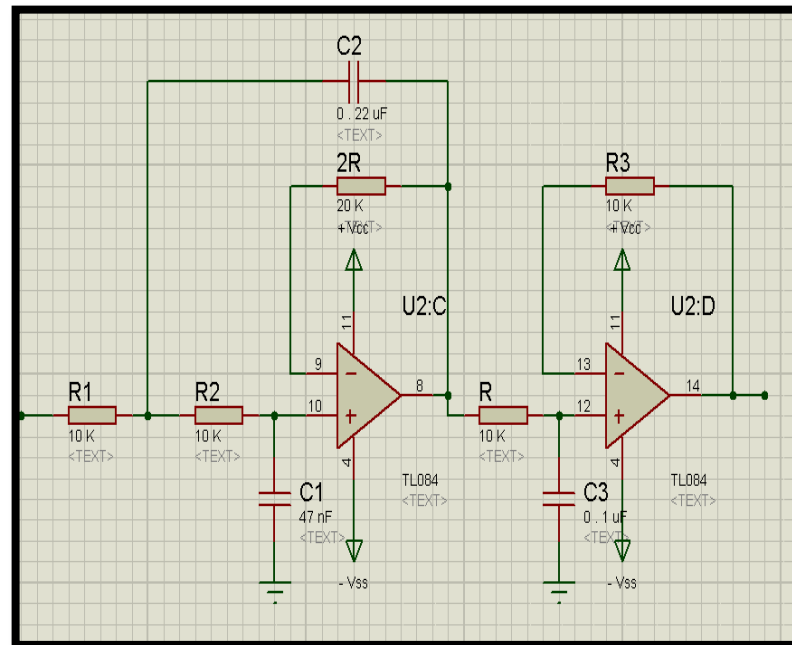


Figura 2.9: Filtro Butterworth con los valores del diseño.
Elaborado por: Gómez O, Silva I.

2.2.4 Acondicionamiento de la señal.

La señal resultante de los filtros analógicos implementados tiene referencia cero, es decir valores de voltajes positivos y negativos, entonces tienen que desplazar la referencia hasta 2.5 V, tomando en cuenta que el nivel de voltaje reconocible por la tarjeta Arduino es TTL: 0 a 5 V, para variar la amplitud de la señal obtenida del ECG se coloca un potenciómetro de 50K Ω como se muestra en la Figura 2.10, para la ganancia de la señal, al variar esta ganancia se evita recortes de los picos de la señal. Para desplazar la señal se utiliza un divisor de voltaje, donde las resistencias son iguales. $R=10K\Omega$

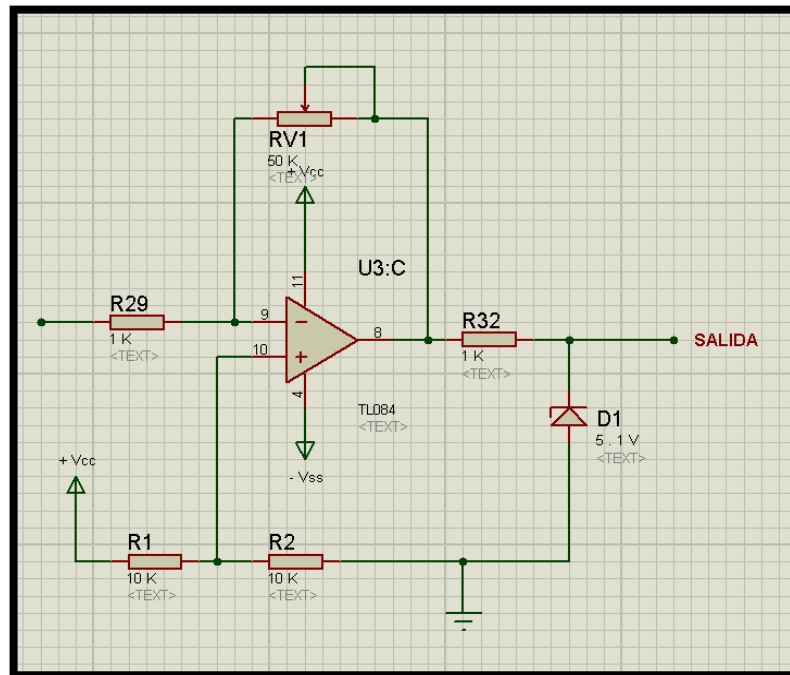


Figura 2.10: Etapa de acondicionamiento de señal.
Elaborado por: Gómez O, Silva I.

Para implementar el amplificador de instrumentación, los filtros analógicos y la etapa de acondicionamiento del presente proyecto se utilizan los amplificadores operacionales TL084, así como se muestra en la Figura 2.11.

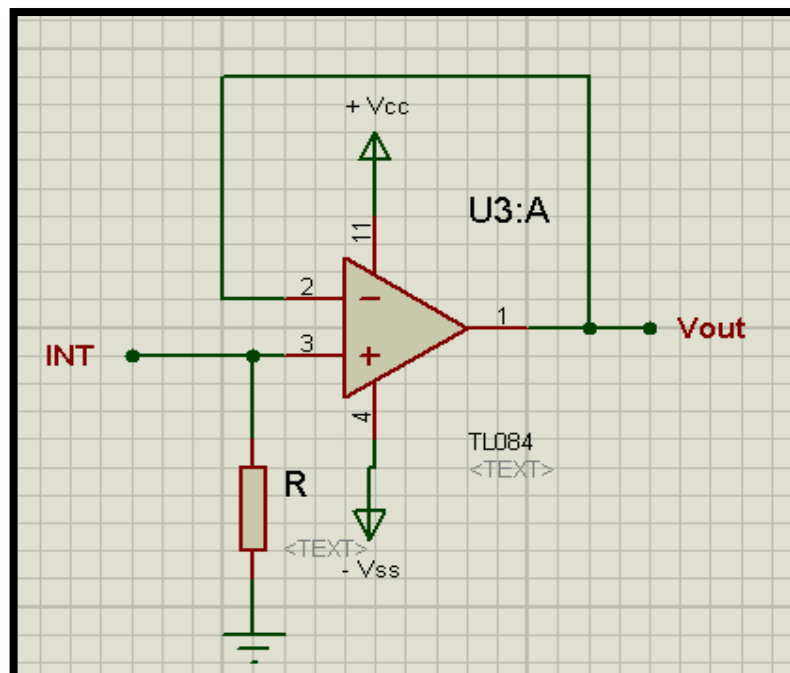


Figura 2.11: Seguidor de Tensión TL084.
Elaborado por: Gómez O, Silva I.

La mayor parte de la adquisición y medida de las señales biofísicas y bioeléctricas del cuerpo humano se realiza mediante circuitos electrónicos, siendo el amplificador operacional un módulo básico de dichos circuitos. Aunque cada vez más, el procesado de la información se realiza con circuitos digitales o sistemas basados en microprocesadores, la conversión de las señales medidas (pulso cardiaco, nivel de oxígeno en la sangre, flujo de aire en los pulmones, presión sanguínea, etc.) en variables eléctricas: corriente o tensión, requiere de circuitos analógicos, donde el amplificador operacional juega un papel fundamental (Anexo D).

2.3 Medición de la presión sanguínea arterial

Para la medición de la presión arterial se utilizó el método Oscilométrico este es uno de los más usados en el desarrollo de equipos de medición arterial.

2.3.1 Método Oscilométrico.

Este método es empleado por la mayoría de los dispositivos de medición automáticos no invasivos. Una extremidad y su vasculatura son comprimidas mediante el uso de un brazalete de compresión inflable.

El principio de medición del método oscilométrico consiste en la medición del cambio de la amplitud de la señal de presión cuando el brazalete es inflado y la onda de presión pasa a través del vaso arterial, produciendo un pulso este a su vez afecta la alteración de la presión en el brazalete, que es el pulso oscilométrico necesario para medir. La técnica oscilométrico requiere una banda para el brazo con una bomba eléctrica, y el medidor de mercurio que en este caso es por software. [22]

La amplitud, de pronto, crece grandemente mientras que el flujo sanguíneo atraviesa la oclusión. En este punto uno se encuentra con la presión sistólica. A medida que la presión del brazalete es reducida incrementa en su amplitud, hasta que alcanza la amplitud máxima y luego disminuye rápidamente. Por lo tanto, la presión sistólica sanguínea arterial y la presión diastólica arterial son obtenidas al identificar la región donde exista este rápido incremento y

decrecimiento en la amplitud de los pulsos respectivamente. La presión sanguínea arterial media se localizará en el punto de máxima oscilación.

La elaboración del equipo de medición de la presión arterial, se basa en el siguiente esquema de funciones que se muestra en la Figura 2.12.

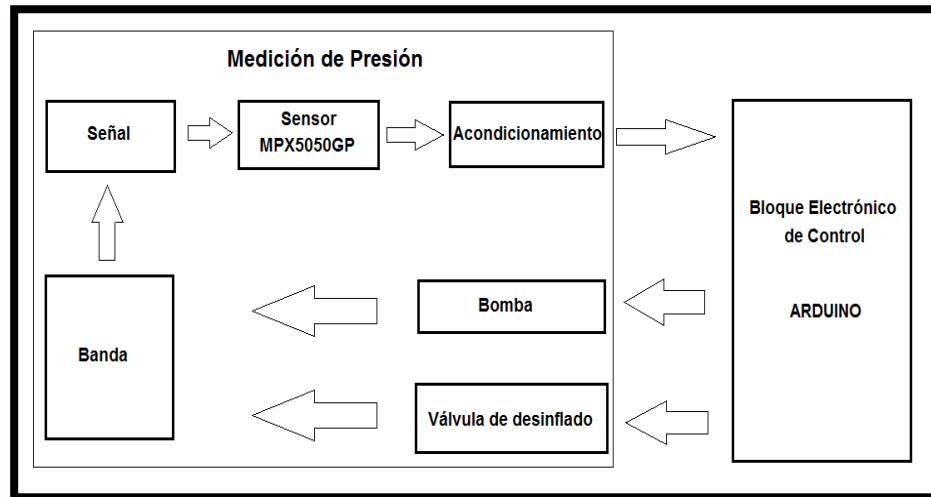


Figura 2.12: Diagrama de bloque de adquisición de la presión.
Elaborado por: Gómez O, Silva I.

2.3.2 Brazaletes inflable

El brazalete utilizado para medir la presión arterial Figura 2.13 de forma oscilométrica, está formado por una banda rectangular por una cámara inflable. De la que parte una manguera hacia los dispositivos de control (bomba, electroválvula y el sensor de presión), y además está formado por una funda flexible, con una sección que contiene dos caras de velcro, permitiéndole así tener una adherencia al momento de rodear el brazo, y ser inflado por la bomba de aire.



Figura 2.13: Brazaletes inflable.
Elaborado por: Gómez O, Silva I.

2.3.3 Bomba de aire

La bomba de aire utilizada para la medición de la presión sanguínea se muestra en la Figura 2.14 y presenta las siguientes características generales:

- Voltaje: 5V.
- Corriente: 120mA.
- Presión: 400mmHg.
- Dimensiones : Cilindro h=6cm, r=1.5 cm
- Ruido: 55 dB.

Más características se observa en el (Anexo I).



Figura 2.14: Bomba de aire.
Elaborado por: Gómez O, Silva I.

2.3.4 Electroválvula.

La electroválvula que se muestra en la Figura 2.15 tiene la función de liberar la presión de aire del sistema. Ésta posee las siguientes características generales:

- Voltaje: 5V.
- Corriente: 120mA.
- Resistencia: 100Ω.
- Dimensiones: 2cm x 1.4 cm x 1.4cm.

Más características se observa en el (Anexo H).



Figura 2.15: Electroválvula.
Elaborado por: Gómez O, Silva I.

2.3.5 Sensor de presión MPX5050GP.

Para adquirir la señal de presión generada en el brazalete de aire se utiliza el sensor de presión MPX5050GP de Motorola que se muestra en la Figura 2.16. Este transforma las vibraciones de presión en una señal eléctrica proporcional que es enviada al conversor A/D del Arduino.

Características:

- Compensación entre temperaturas desde -40 grados hasta 125 grados centígrados.
- Rango de presión de 0 a 50 KPa.
- Sensibilidad de 90mV / Kpa.
- Tiempo de respuesta de 1ms.
- Voltaje de polarización 4.75 Vcd a 5.25 Vcd.
- Corriente suministrada de 7.0mAdc a 10mAdc.



Figura 2.16: Sensor de presión MPX5050GP.
Elaborado por: Gómez O, Silva I.

2.3.6 Acondicionamiento del sensor de presión MPX5050GP.

Refiriéndonos a la Figura 2.17 , el sensor de presión MPX5050GP puede ser conectado ya sea directamente al dispositivo de adquisición de datos o al

circuito acondicionador. Si se conecta el sensor de presión directamente al equipo de adquisición de datos este brinda una señal en la entrada que va de 0.2 Vdc a 0 mmHg hasta 4.7 Vdc a 375 mmHg de presión aplicada; mientras que si se conecta este sensor al circuito acondicionador y este a su vez al equipo de adquisición de datos en la entrada del convertidor aparece una señal que va de 0.005 hasta 3.5 V.

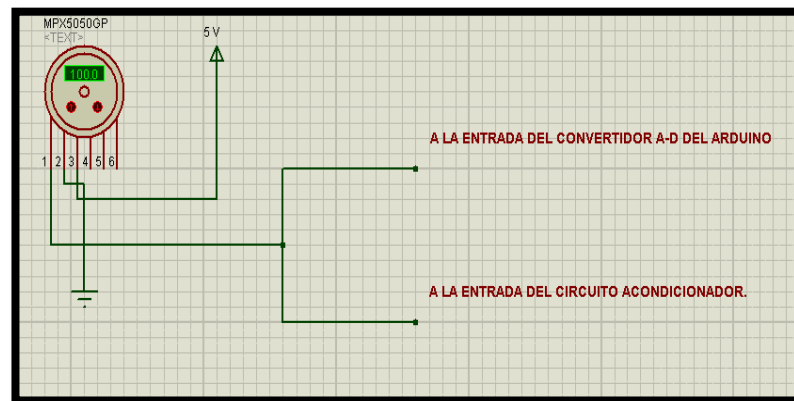


Figura 2.17: Circuito de conexión del sensor MPX5050GP.
Elaborado por: Gómez O, Silva I.

Como el equipo de adquisición de datos tiene un convertidor cuyo voltaje de referencia es de 5 V y además la resolución del convertidor análogo digital es de 10 bits entonces con esto es posible definir el conteo que me envía el convertidor A / D del Arduino de tal manera que se pueda definir los límites de unidades de presión.

Este conteo se define así:

$$Conteo = \frac{VX_{dcr} - V_{rl}}{V_{rh} - V_{rl}} * (2^{10} - 1) \text{ Ecuación (2.3)}$$

Siendo VXdcr: Es el voltaje del transductor.

Vrh y Vrl: Son los voltajes de referencia del convertidor.

El conteo a 0 mmHg, tenemos un voltaje VXdcr = 0.2 V

$$Conteo = \frac{0.2 - 0}{5 - 0} * 1023 \approx 41$$

El conteo a 300 mmHg teniendo un voltaje $V_{Xdcr} = 3.8$ V

$$Conteo = \frac{3.8 - 0}{5 - 0} * 1023 \approx 777$$

Por lo tanto la resolución es $777 - 41 = 736$ conteo. Esto se reduce en una resolución en unidades de presión como:

$$resolucion = \frac{Presión\ maxima - Presión\ mínima}{conteo} \quad Ecuación\ (2.4)$$

$$resolucion = \frac{300\ mmHg - 0\ mmHg}{736} = 0.40\ mmHg$$

El sistema utilizado para la adquisición de datos es de tipo radiométrico pues cuando el sensor de presión es conectado directamente al equipo este es alimentado por el mismo equipo con un voltaje igual al voltaje de referencia (5 Vdc). Es decir que las variaciones de voltaje del sistema de alimentación del equipo de adquisición no tendrán ningún efecto sobre la exactitud de la medición del sensor de presión.

2.3.7 Descripción del Circuito Acondicionador.

La presión del brazalete (PB) es sensada usando un transductor de presión integrado hecho por Motorola. La salida de este sensor puede ser usada de dos maneras, una consiste en medir la presión del brazal, mientras que la otra es de usar esta salida para procesarla analógicamente para así, obtener las oscilaciones o pulsos necesarios para aplicar el método oscilométrico.

Puesto que el sensor usado (MPX5050GP) tiene su propio circuito de acondicionamiento internamente, la presión del brazalete puede ser directamente conectada a un convertidor analógico – digital. El otro método descrito, amplificará y filtrará la señal de presión del brazal para extraer una versión amplificada de las oscilaciones dentro de la señal de presión del brazalete, la cual es causada por la expansión del brazo del sujeto cada vez que la presión en el brazo se incrementa durante la sístole del corazón.

La salida del sensor consta de dos señales, una señal de oscilación (aproximado. 1Hz) sobre la señal de presión del brazalete PB (≤ 0.04 Hz).

Por consiguiente, un filtro paso alto de dos polos es diseñado para bloquear la señal PB antes de amplificar la señal de oscilación. Si la presión PB no es apropiadamente atenuada, el nivel de voltaje base de la señal oscilante no será constante y la amplitud de cada oscilación no tendrá la misma referencia para luego compararlas. La Figura 2.18 muestra el circuito amplificador de la señal de oscilación.

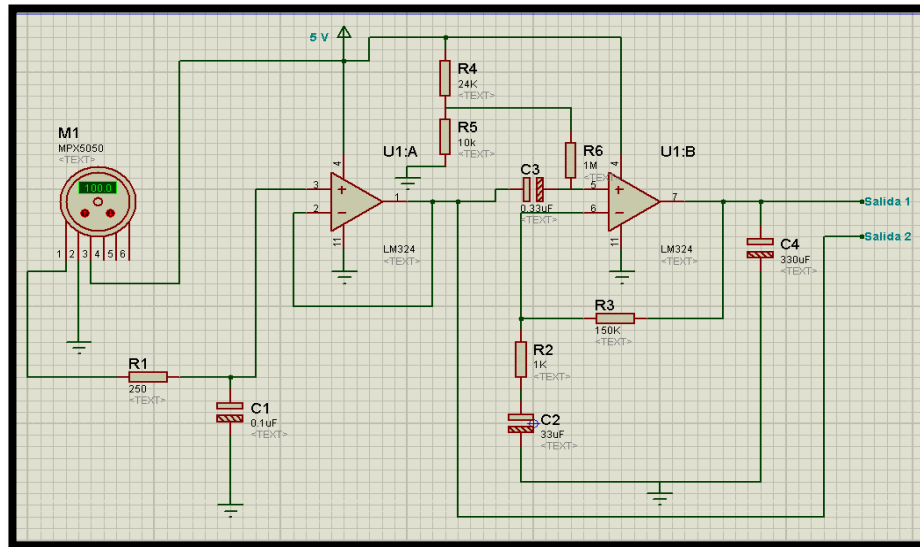


Figura 2.18: Amplificador de la señal de Oscilación.
Elaborado por: Gómez O, Silva I.

El filtro consiste de dos redes RC las cuales determinan dos frecuencias de corte (polos). Estos dos polos son escogidos de tal manera que la señal de oscilación no sea distorsionada ni perdida. Estas dos frecuencias de corte pueden ser aproximadas mediante las siguientes ecuaciones:

$$fp1 = \frac{1}{2\pi R2C2} \text{ Ecuación (2.5)}$$

$$fp2 = \frac{1}{2\pi R6C3} \text{ Ecuación (2.6)}$$

La Figura 2.19 describe la respuesta de frecuencia del filtro. Esta gráfica no incluye la ganancia de amplificación del circuito.

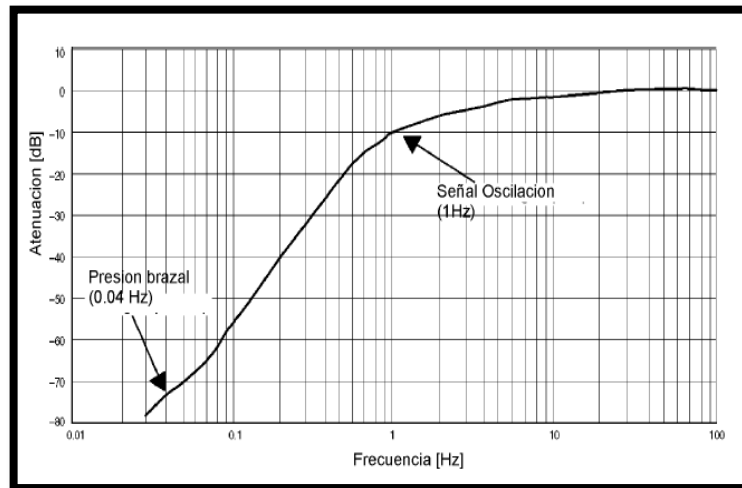


Figura 2.19: Respuesta de Frecuencia del Filtro.
[23]

La señal de oscilación varía de persona a persona, de hecho varía desde menos de 1 mmHg hasta 3 mmHg. A partir de la función de transferencia del MPX5050GP, esto se traduce a un voltaje de salida de 12 mV hasta 36 mV. Debido a que el filtro ofrece una atenuación de 10dB a una señal de oscilación decae a valores de 3.8 mV hasta 11.4 mV respectivamente.

Un factor de amplificación de 150 es escogido en el diseño del circuito para que la señal de oscilación este dentro del límite de salida del amplificador (5 mV hasta 3.5 V). Figura 2.20 muestra la salida tomada directamente del sensor de presión y la Figura 2.21 la señal de oscilación extraída de la salida del amplificador.

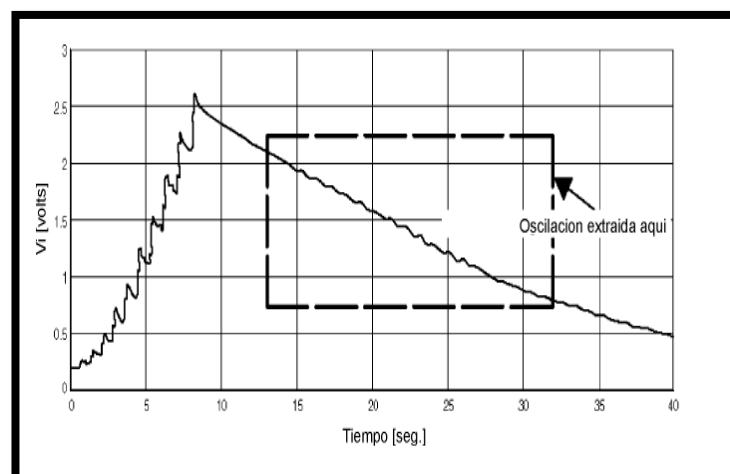


Figura 2.20: Señal PB en las salida del sensor de Presión.
[23]

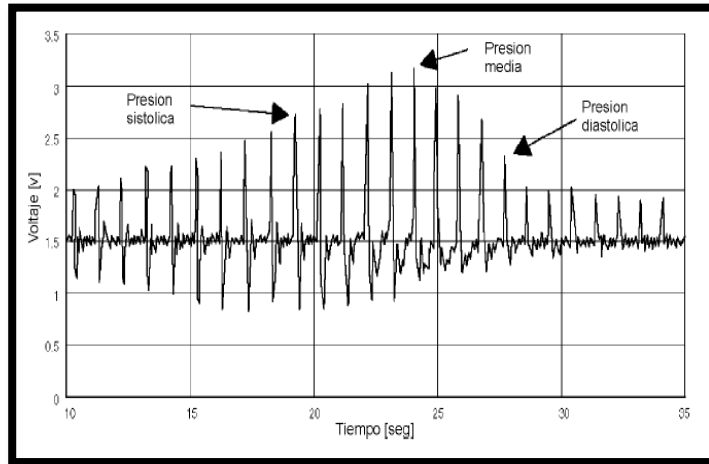


Figura 2.21: Señal de oscilación extraída de la salida del amplificador de la Figura 2.20.

[23]

2.4 Diseño y Construcción del SPO2

Para el diseño del SPO2 se partirá desde el sensor de oximetría, ya que existen pocas empresas dedicadas a la fabricación de oxímetros de pulso, una destacada es la empresa NELLCOR, el modelo del sensor de oximetría de la marca Nellcor Figura 2.22 es compatible con conectores tipo hembra DB9, donde la configuración de los pines se muestra en la Tabla 2.1.

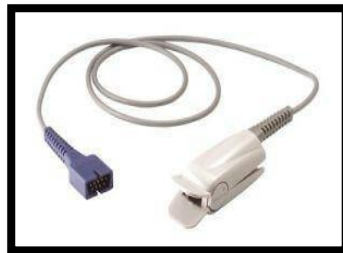


Figura 2.22: Sensor de Oximetría marca Nellcor.

Elaborado por: Gómez O, Silva I.

Tabla 2.1 Descripción de los pines del sensor Nellcor.

PIN	DESCRIPCIÓN
1	R del sensor
2	Polarización de los Leds.
3	Polarización de los Leds.
4	No existe.
5	Salida (+).
6	R sensor.
7	GND del Cable.
8	No existe.
9	Salida (-).

[24]

Los pines 1 y 6 correspondientes a R sensor, son los terminales de una resistencia interna codificada por el fabricante, la cual se utiliza por los oxímetros para determinar el tipo de sensor, ya sea desechable, reusable, para niño, o adulto. Por tanto estos dos terminales no serán utilizadas en el desarrollo del proyecto.

Los pines 2 y 3 corresponden a la polarización del led rojo e Infrarrojo, los cuales están colocados dentro del sensor, tal como se muestra en la Figura 2.23.

Los pines 5 y 9 son los terminales del fotodiodo que es el encargado de detectar la variación de la luz transmitida por los Leds.

El Pin 7 corresponde al recubrimiento del cable que al estar conectado a tierra, da un cierto grado de protección con respecto al ruido electromagnético y a la estática.

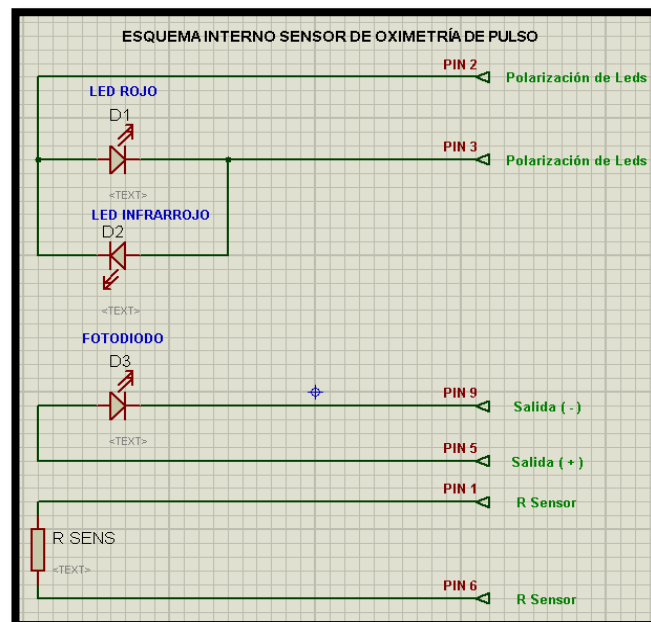


Figura 2.23: Esquema interno del sensor de Oximetría.
Elaborado por: Gómez O, Silva I.

Los diodos Leds presentan un comportamiento parecido al de un diodo rectificador sin embargo, su tensión de umbral, varía dependiendo del color del diodo esto se puede constatar en la Tabla 2.2

Tabla 2.2 Relación entre el color y la tensión de umbral de Leds.

COLOR	TENSIÓN DE UMBRAL
Infrarojo	1,3v
Rojo	1,7v

[24]

El conocimiento de esta tensión es fundamental para el diseño del circuito, normalmente se coloca en serie una resistencia que limita la intensidad que circulará por él, cuando se polariza directamente se comporta como una lámpara que emite una luz roja. Cuando se polariza inversamente no se enciende además no deja pasar la corriente. Es por eso que la marca Nellcor ha decidido colocar el led rojo inversamente con respecto al led infrarrojo, para que al conmutar la polarización se encienda uno u otro.

La intensidad luminosa con la que brilla un led se puede controlar de acuerdo a la intensidad de corriente que pasa por él. La mínima intensidad de corriente que necesita un diodo LED para que emita luz es de 4mA, y por precaución como máximo debe aplicarse 50mA. Además el sensor de saturación de oxígeno utiliza Leds diseñados especialmente para emitir su luz a una determinada longitud de onda, la misma que para los sensores Nellcor es de 662nm para el led rojo y de 905nm para el led infrarrojo (1.2.3).

La luz emitida por los Leds pasará a través del dedo y será captada por un único fotodiodo el cual es un dispositivo semiconductor de unión p-n. Hay que destacar que la corriente producida en el fotodiodo, debido a la captación de luz está en el rango de los μA , en este punto se ve la necesidad de amplificarla y convertirla en voltaje para su acondicionamiento.

A continuación se resume en la Tabla 2.3 las características del sensor que se deben de tomar en cuenta para el diseño del hardware.

Tabla 2.3: Características técnicas del sensor de oximetría

Características del sensor al tomar en cuenta para el diseño del Hardware
1 led que emite en el rango rojo a 662nm.
1 led que emite en el rango infrarrojo a 905nm.
Los dos Leds están colocados opuestamente uno del otro.
Rango de corriente de alimentación de los Leds es de 4 – 50mA.
1 fotodiodo cuyo rango de producción de corriente es de 0 - 800 μA .

[24]

2.4.1 Diseño del Hardware.

Para trabajar con la señal de corriente que entrega el sensor Nellcor se diseña una etapa intermedia de amplificación y de acondicionamiento de la señal, para lo cual es recomendable convertir la señal de corriente a una señal de voltaje, filtrar el ruido, y amplificar a niveles adecuados para su ingreso a la tarjeta Arduino. Hay que tomar en cuenta que los leds rojo e infrarrojo del sensor Nellcor que están inversamente conectados entre ellos, para lo que se necesita una fuente de 5v que alterne su polaridad periódicamente para alimentar a estos leds.

Para resolver el problema de cambio de polaridad de la fuente se decide aplicar un integrado conocido como puente H, que permite controlar e invertir la polaridad con la que se alimenta un circuito, que en nuestro caso será el encendido del led rojo o del led infrarrojo.

Para generar automáticamente dichas señales de control se implementará un generador de pulsos que tendrá 4s de tiempo en alto y 3s de tiempo en bajo en el integrado 555 (2.4.7), cuya señal además de indicar que led se enciende, debe ser ingresada a la tarjeta Arduino para la sincronización y muestreo de la señal entregada por el fotodiodo para ser demultiplexada en dos señales que representaran la absorción de cada luz, en la Figura 2.24 se visualiza el diagrama de bloques para la adquisición de la señal del SPO2.

Se requiere una señal de +5V para generar las señales de control mediante la utilización de compuertas digitales.

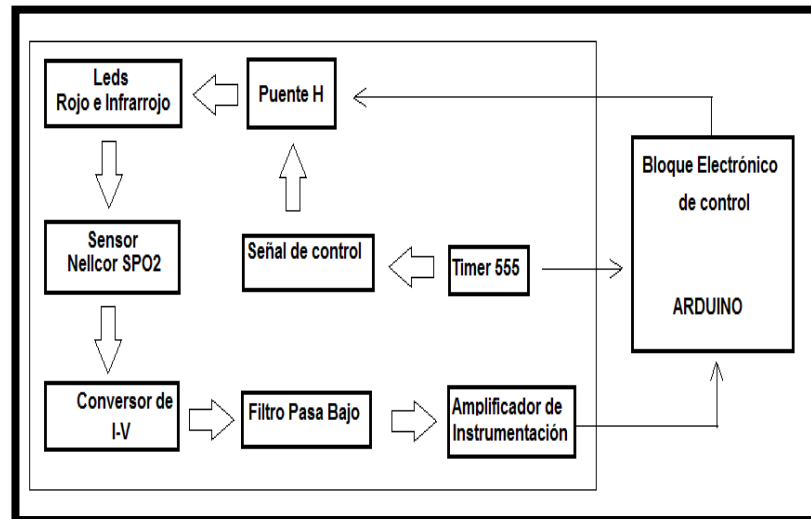


Figura 2.24: Diagrama de bloques del SPO2
Elaborado por: Gómez O, Silva I.

2.4.2 Diseño del convertor de Corriente a Voltaje.

El fotodiodo produce corriente en el rango de 0 - 800 μ A, dependiendo de la cantidad de luz incidente. Debido a que este rango es muy pequeño se procede a la amplificación y conversión a voltaje, el diagrama del convertor se puede ver en la Figura 2.25.

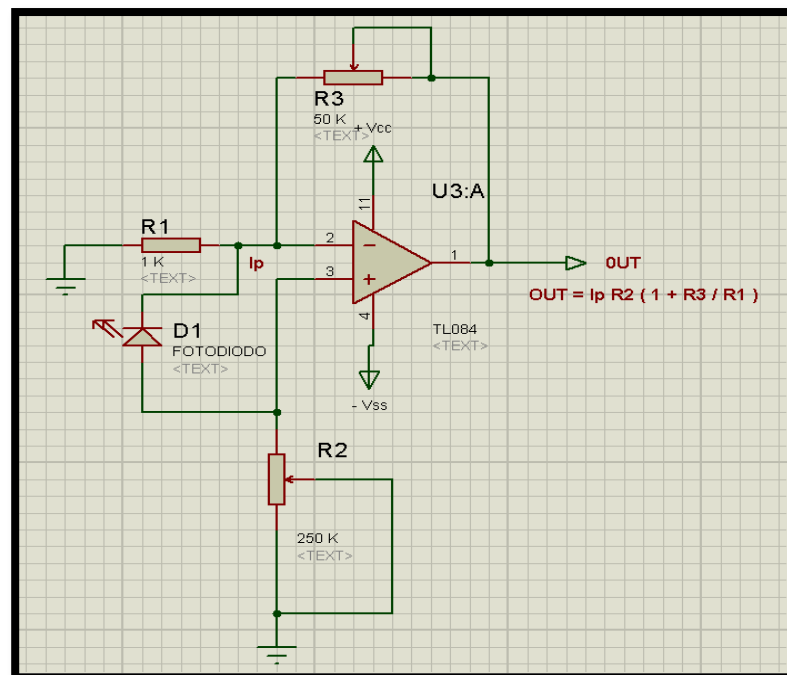


Figura 2.25: Convertor de Corriente a Voltaje diseñado
Elaborado por: Gómez O, Silva I.

Donde I_p , es la corriente inversa que se produce en el fotodiodo debido a la presencia de luz. La ganancia del conversor se ajusta mediante el potenciómetro R3 de 50K Ω , y el potenciómetro R2 de 250K Ω .

$$V_{out} = I_p \cdot R2(1 + R3/R1) \quad \text{Ecuación (2.7)}$$

Cabe resaltar que el amplificador operacional utilizado es del tipo J-FET presenta una impedancia de entrada grande, tiene un tiempo de respuesta rápido y además consume muy poca potencia, lo que favorece el acoplamiento con el fotodiodo. Ver (Anexo D).

2.4.3 Diseño del filtro

Luego de la conversión de corriente a voltaje se debe implementar un filtro que elimine cualquier tipo de ruido presente en la señal, principalmente causados por las lámparas fluorescentes en el ambiente. Además hay que considerar que la señal entregada por el sensor es de tipo continuo (CC) no cambia la dirección.

Por tanto se decidió diseñar un filtro que elimine cualquier señal AC, como un filtro pasa bajos y poder garantizar la pureza de la señal. Se decidió implementar un filtro pasa bajos de segundo orden con una frecuencia de corte de $f=3\text{Hz}$, para eliminar las señales AC que corresponden a ruidos extraños, además de establecer una ganancia del filtro de 1,5 valor que garantiza la respuesta del circuito. Los valores para las resistencias se calculan a continuación.

$$A_v = 1,5$$

$$f = 3\text{Hz}.$$

$$A_v = 1 + \frac{R_f}{R_i} \quad \text{Ecuación (2.8)}$$

$$f = \frac{1}{2\pi RC}$$

$$\text{Si } R_f = K\Omega.$$

$$\text{Si fijamos } C = 1\mu F.$$

$$R_i = \frac{50K\Omega}{1.5-1}$$

$$R = \frac{1}{2\pi(3\text{Hz})(1\mu F)}$$

$$R_i = 100K\Omega.$$

$$R = 53051.6\Omega. \approx 50K\Omega.$$

Donde A_v corresponde la ganancia del filtro, f la frecuencia de corte.

Se cambió la resistencia de $53\text{K}\Omega$, por una de $50\text{K}\Omega$, debido a su disponibilidad en el mercado, estableciéndose la frecuencia de corte en $3,18\text{Hz}$, lo cual no afecta en gran medida al diseño, el esquema del filtro pasabajos diseñado se muestra en la Figura 2.26.

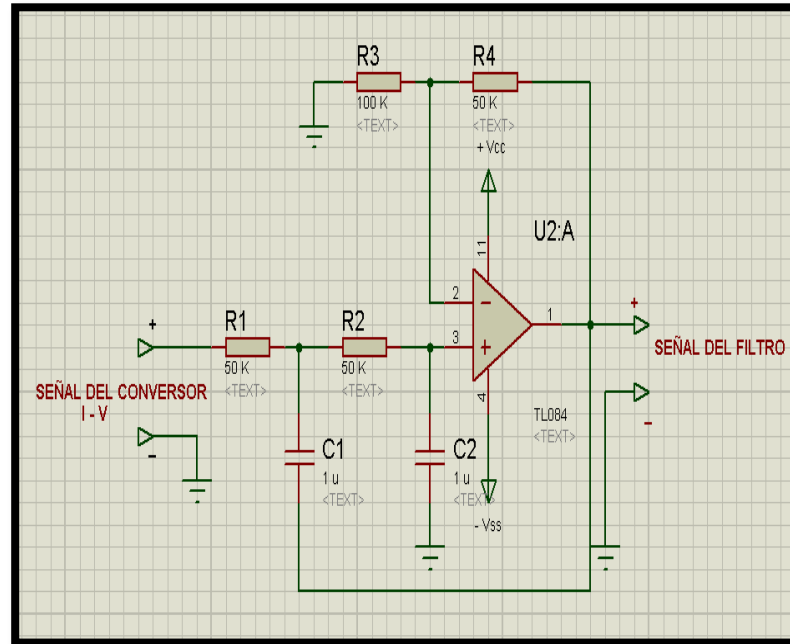


Figura 2.26: Filtro Pasabajos diseñado.
Elaborado por: Gómez O, Silva I.

2.4.4 Diseño del Amplificador final.

Luego de haber filtrado la señal se procede a una etapa de amplificación final que establezca niveles adecuados antes del ingreso al Arduino.

Dicho amplificador debe tener una Relación de Rechazo de Modo Común (RRMC) muy alta, para evitar el ingreso de ruido a la señal.

Analizando estas características se encuentra que el amplificador de instrumentación representa una buena solución para la etapa de amplificación (2.2.2).

Dicho circuito proporciona una salida con base en la diferencia entre dos entradas (multiplicadas por un factor de escala), lo que elimina componentes

de DC comunes a las entradas, es decir ruido inmerso en la señal a amplificar, lo que asegura de mejor manera el desempeño óptimo de la amplificación.

Se proporciona un potenciómetro R_p para permitir el ajuste del factor de escala del circuito, para equilibrar la diferencia en las ganancias. El voltaje de salida en esta configuración es:

$$\frac{V_0}{V_1 - V_2} = 1 + \frac{2R}{R_p} \quad \text{Ecuación (2.9)}$$

Por lo que la salida puede obtenerse a partir de:

$$V_0 = \left(1 + \frac{2R}{R_p}\right) (V_1 - V_2) = K (V_1 - V_2) \quad \text{Ecuación (2.10)}$$

Se establece $R = 10\text{K}\Omega$ y $R_p = 10\text{K}\Omega$ para lograr un rango de ganancia desde 3 con $R_p = 10\text{K}\Omega$, hasta 20 con $R_p = 1\Omega$. Con lo que se logra una variación aceptable para la posterior calibración del circuito. La Figura 2.27 muestra el esquema final del amplificador de instrumentación.

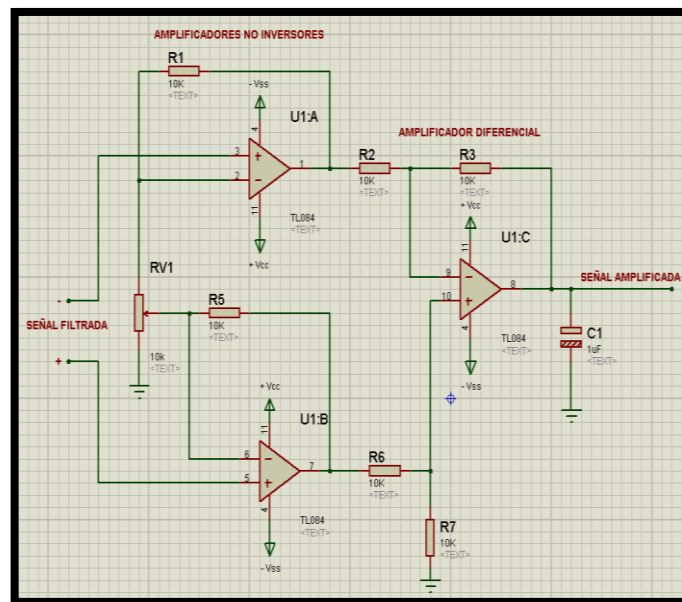


Figura 2.27: Amplificador de Instrumentación diseñado.
Elaborado por: Gómez O, Silva I.

Luego de esta etapa la señal esta lista para ingresar al Arduino. Los valores de los potenciómetros se ajustarán en la etapa de implementación.

2.4.5 Diseño del circuito de control para activación de los leds

El circuito de la Figura 2.28 constituye la configuración denominada Puente H que comúnmente sirve para controlar el sentido de giro de motores de corriente continua, pero que en este caso será utilizado para invertir la polaridad de la alimentación de los leds rojo e infrarrojo, ya que se encuentra conectado uno inverso del otro.

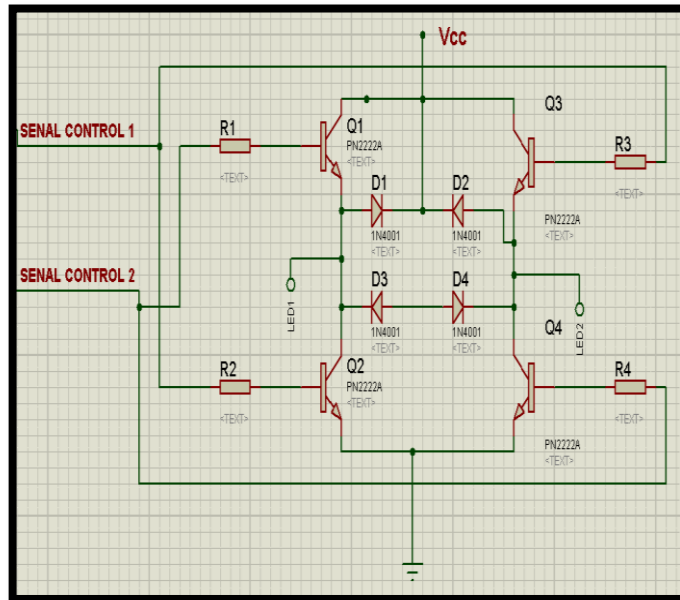


Figura 2.28: Esquema del Puente H.
Elaborado por: Gómez O, Silva I.

Aplicando un voltaje positivo en la señal de control 2, se hace conducir al transistor Q1 y Q4, obteniéndose en el punto (LED1) de la polarización de los leds el voltaje Vcc y en el punto (LED2) tierra, encendiéndose de esta manera el led rojo.

En cambio si se aplica un voltaje positivo en la señal de control 1, se activa los transistores Q2 y Q3, lo que significa que en el punto (LED1) se tiene Tierra y en el punto (LED2) se tiene Vcc, lo que enciende al led infrarrojo. Cabe aclarar que si en las señales de control 1 y 2 no hay un voltaje positivo, no se encenderá ninguno de los leds, y que por el contrario, no se deben activar simultáneamente, porque causaría un cortocircuito entre Vcc y tierra, esto se lo debe tener muy en cuenta en el diseño del circuito generador de las señales de control.

Los diodos D1, D2, D3, D4, sirven como vía de escape para corrientes parasitas que se pueden almacenar debido a la inversión de polaridad.

Para determinar los valores de las resistencias, primeramente se debe establecer el nivel de voltaje que tendrá las señales de control, el cual será de 5 Vcc, debido a que se utilizarán compuertas lógicas para generar las. Las 4 resistencias colocadas en la base de los transistores deben tener el mismo valor para asegurar que la corriente que circulará por ellos sea la misma y que no sobrepase de 50mA para evitar la destrucción de los leds del sensor. El valor que se obtuvo de los cálculos es de 1.5KΩ.

Por otro lado, la pigmentación de la piel de los pacientes presenta un problema debido a que su coloración afecta e introduce lecturas erróneas en la medición de la Saturación de Oxígeno, sin embargo los oxímetros comerciales solucionan este problema controlando la intensidad de luz con la que brillan los leds, para ello en este diseño se deciden utilizar como fuente de alimentación del puente H, una salida analógica del Arduino, para con ello variar el voltaje y por lo tanto, la corriente que circula por los leds, que es directamente proporcional a la intensidad luminosa.

En la Figura 2.29 se muestra el circuito diseñado.

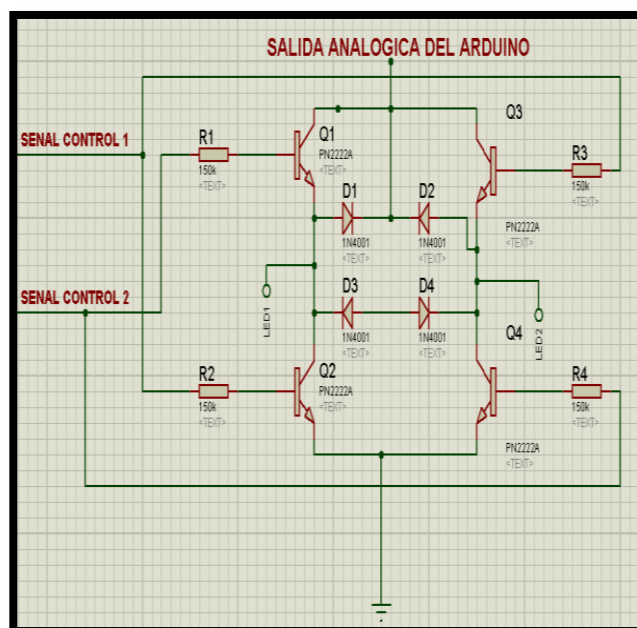


Figura 2.29: Circuito de control de activación de los leds.
Elaborado por: Gómez O, Silva I.

2.4.6 Diseño del circuito generador de las señales de control.

Es necesario recordar que las señales que controlan la inversión de polaridad del puente H nunca deben activarse simultáneamente por lo que se utilizarán compuertas lógicas cuya configuración evitará esa situación.

Primeramente la señal que sincroniza el muestreo y activación de los leds proviene tan solo de un generador de pulsos, cuya señal de salida al momento de estar en el nivel alto, equivaldrá a la activación de la señal de control 2 del puente H, que enciende el led rojo. Por el contrario, cuando se encuentre en el nivel bajo (0 V), equivaldrá a la activación de la señal de control 1 del puente H, lo que encenderá al led infrarrojo. Para lograr esto, la señal de salida del generador de pulsos ingresará a una compuerta inversora, (recuérdese que debe existir un voltaje positivo en la señal de control del puente H para su activación). Luego la señal de salida del generador de pulsos junto con su inverso, ingresarán a una compuerta NAND configurada para cumplir la siguiente tabla de verdad Tabla 2.4.

Tabla 2.4: Tabla de verdad para las señales de control.

GENERADOR DE PULSOS	INVERSO	SEÑAL CONTROL 1	SEÑAL CONTROL 2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

[24]

Esto evitará que se activen ambas señales en el caso de un mal funcionamiento o de la inserción de voltajes en la señal del generador de pulsos o en su inverso, asegurando de esta manera el correcto funcionamiento del puente H. a continuación se muestra el esquema diseñado en la Figura 2.30.

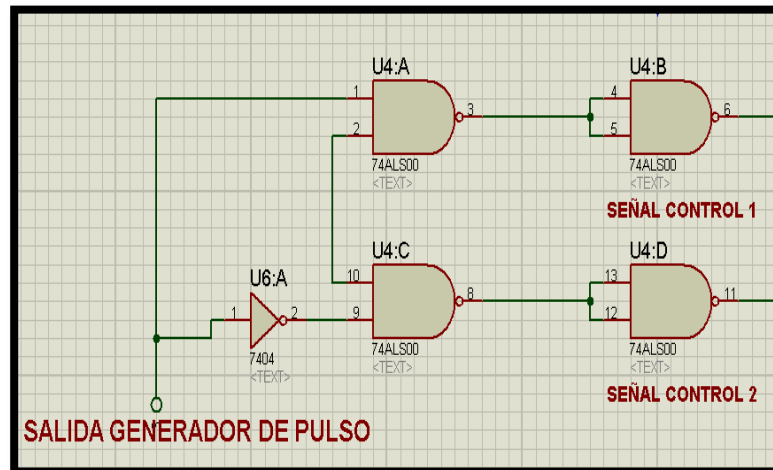


Figura 2.30: Circuito generador de señales de control para el puente H.
Elaborado por: Gómez O, Silva I.

2.4.7 Diseño del Generador de Pulsos.

Como ya se explicó, es necesaria la sincronización entre el muestreo de la señal por parte del Arduino y la activación de los leds rojo o infrarrojo en la placa diseñada, por lo que a continuación se diseña un circuito temporizador típico con la utilización de un Circuito Integrado 555, lo cual se muestra a continuación en la Figura 2.31.

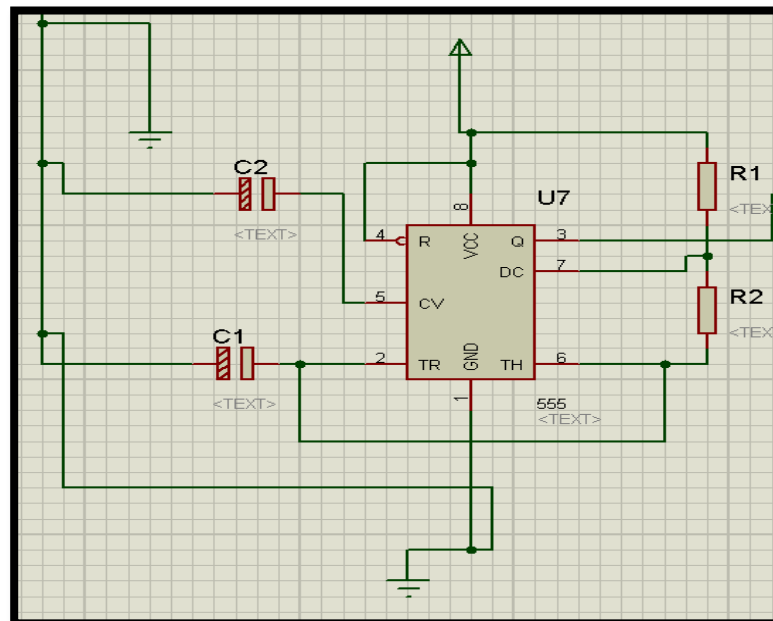


Figura 2.31: Esquema del temporizador.
Elaborado por: Gómez O, Silva I.

Se puede hacer cálculos en los intervalos de tiempo durante los cuales la salida está en alto y en bajo usando las siguientes ecuaciones:

$$T_{\text{alto}} \approx 0.7(R_1 + R_2)C_1 \quad \text{Ecuación (2.11)}$$

$$T_{\text{bajo}} \approx 0.7R_2C_1$$

$$T = \text{Periodo} = T_{\text{alto}} + T_{\text{bajo}}$$

$$f = \frac{1}{T} = \frac{1.44}{(R_1 + 2R_2)C_1}$$

C2 sirve como filtro del ruido introducido por la fuente de alimentación y cuyo valor por lo general es de 0.01uF.

Se decide establecer el tiempo (T) alto es de 4s que corresponde al tiempo en que permanecerá encendido el led rojo, y el tiempo (T) bajo en 3s que corresponde al led infrarrojo [24] , la diferencia en los tiempos se debe a que el temporizador 555 necesita que los tiempos en alto y bajo sean diferentes uno del otro, esta limitación no afecta en gran medida al diseño del sistema.

El tiempo elegido en ambos casos es lo suficientemente grande para garantizar la estabilización de la señal, debido a que si se disminuye, se introducen fluctuaciones en la señal receptada por el fotodiodo, causadas principalmente por la conmutación entre el led rojo y el infrarrojo.

Los valores de los elementos pasivos son:

Se fija en 58KΩ con lo que:

$$C_1 \approx \frac{T_{\text{bajo}}}{0.7 * R_2} \approx \frac{3s}{0.7 * 58K\Omega} \approx 73.8\mu F \approx 70\mu F$$

$$R_1 \approx \frac{T_{\text{alto}}}{0.7 * C_1} - R_2 \approx \frac{4s}{0.7 * 73.8\mu F} - 58K\Omega \approx 19429\Omega \approx 20K\Omega$$

La desviación de los valores calculados con los obtenidos en el mercado no afecta al sistema, ya que la señal del temporizador indicará

simultáneamente tanto en el Arduino como el generador de señales de control, para que conmuten sincrónicamente entre la señal roja y la señal infrarroja.

En la Figura 2.32 se muestra el esquema final del generador de pulsos diseñado.

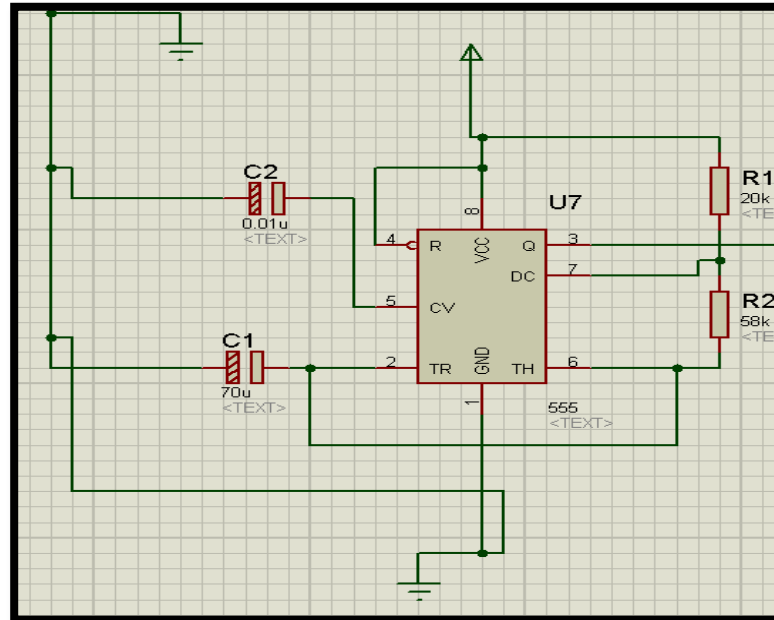


Figura 2.32: Generador de pulsos diseñado.
Elaborado por: Gómez O, Silva I.

2.5 Diseño del Espirómetro

El espirómetro diseñado es capaz de medir el flujo pulmonar de un paciente. Su principio de funcionamiento se basa en el movimiento de una turbina, que gira por efecto del aire espirado por el paciente, de tal manera de excitar a un par de fotodiodos, cuya señal va a la tarjeta Arduino el cual se encarga de procesar la información. A continuación en la Figura 2.33 se puede observar de forma general un diagrama de bloques del espirómetro.

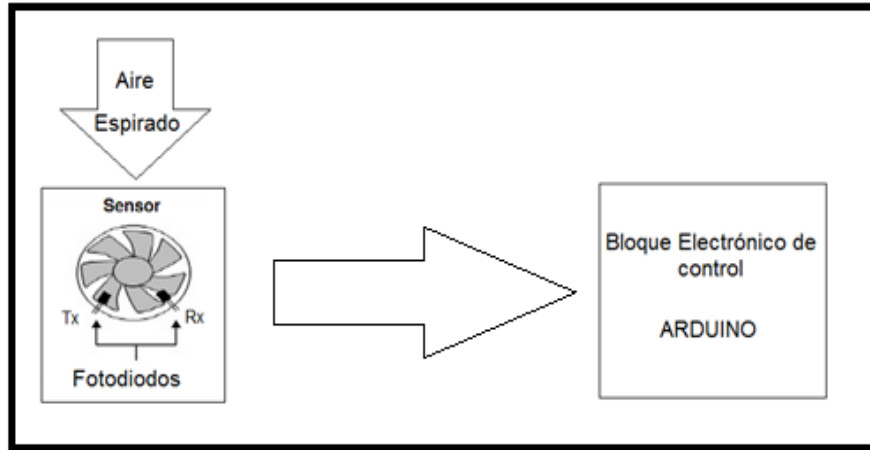


Figura 2.33: Diagrama de bloque general del Espirómetro.
Elaborado por: Gómez O, Silva I.

2.5.1 Consideraciones de Diseño

Para la elaboración del diseño, se estudiaron distintos espirómetros comerciales. Entre los estudiados se encontraban de flujo, neumotacógrafos de tipo fleisch, lilly y desechable, espirómetros de hilo caliente, de ultrasonidos y el espirómetro de turbina, obsérvese la Tabla 1.7.

Al haber estudiado todos estos, se decidió utilizar una turbina con fotodiodos, los cuales son excitados cuando esta gira, cortando el haz de luz y generando un tren de pulsos que van al Arduino.

Luego del procesamiento de los datos se envía la información a través del Bluetooth a un equipo móvil, en donde se observa los valores de la cantidad de flujo de aire que puede espirar una persona al momento de dormir.

2.5.2 Adquisición de la señal

El espirómetro diseñado consta de una mascarilla en la cual contiene una boquilla donde se colocó una pequeña hélice o turbina, cuyo movimiento es detectado por un sensor fotodiodo Figura 2.34.



Figura 2.34: El sensor fotodiodo detecta el movimiento de la turbina.
Elaborado por: Gómez O, Silva I.

2.5.3 Acondicionamiento de la señal.

Para poder medir el flujo pulmonar, la señal adquirida del sensor fotodiodo se utiliza la siguiente fórmula:

$$FP = \frac{\text{Número de Pulsos} * \text{Tiempo}}{\text{Numero de aspas de la turbina}} \quad \text{Ecuación (2.12)}$$

Donde:

- FP es el flujo pulmonar
- Numero de pulsos corresponde a la señal de salida de los sensores en el transistor.
- Tiempo es de 60 minutos en el caso del espirómetro diseñado.
- Número de aspas de la turbina son las hélices que en el espirómetro diseñado es de 2 hélices.

El circuito implementado se puede observar en la Figura 2.35.

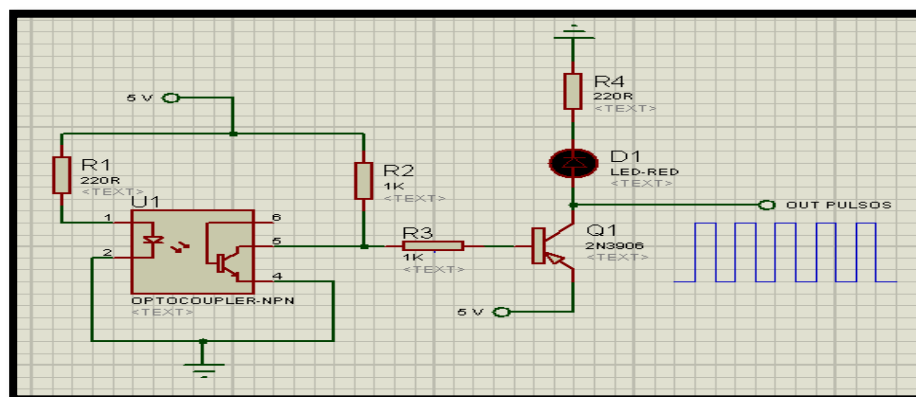


Figura 2.35: Circuito de acondicionamiento del espirómetro.
Elaborado por: Gómez O, Silva I.

2.6 Artefactos

A los artefactos se les conoce como un error en la lectura de los sensores producido por mala colocación, o movimiento de estos.

2.6.1 ECG

Los artefactos generados en el ECG se debe a:

- La mala posición del electrodo respecto a la piel produce el movimiento de este electrodo, representando una fuente potencial de error que contamina la señal del ECG.
- La incorrecta limpieza de la piel, genera una capacitancia parásita entre la dermis y el electrodo la que contamina la señal del ECG. Para una limpieza correcta de la piel es necesario afeitar el vello y limpiar con alcohol la zona donde se va a colocar el electrodo para disminuir la resistencia producida por la grasa en la dermis.
- El movimiento de los músculos o del cuerpo del paciente introduce una fuente potencial de error que contamina la señal del ECG. Para ello es necesario que el paciente se encuentre en un perfecto estado de reposo.

2.6.2 Presión Sanguínea

- La mala colocación del brazalete en el ante brazo genera datos incorrectos en la medición de presión sanguínea.
- La tensión, estrés, adrenalina o mala colocación del cuerpo para que este no este relajado no asegura datos fiables ni correctos para su respectivo análisis.

2.6.3 SPO2

- Si la colocación del sensor en el dedo es inadecuada, la cifra del SPO2 es falsamente baja, debido a que el sensor queda de lado y mide por una parte la superficie del dedo, y por otro lado, la luz que se transmite del medio ambiente. Se denomina “efecto Penumbra”.
- La causa más frecuente de lecturas inadecuadas del SPO2 es el movimiento. Afecta la habilidad de la luz para viajar de los diodos emisores de luz hacia el foto detector, el parkinsonismo, las crisis convulsivas, los temblores, originan problemas con la detección de la saturación con mediciones falsamente altas.
- Si la Luz ambiental es muy intensa o su frecuencia es similar al de los LED, ocasiona interferencia con la medición de la saturación. La luz fluorescente o la de xenón, causan lecturas bajas de SPO2, esto se evita si se cubre el sensor con un material opaco.

- El esmalte oscuro (azul, negro y verde) altera con más frecuencia las lecturas del SPO2, es necesario remover el esmalte.
- Lecturas bajas de SPO2 se pueden presentar con más frecuencia en personas con piel oscura, debido a que la piel interfiere con la absorción de las longitudes de onda. [24]

2.6.4 Espirometría

- La mala colocación de la mascarilla produce fugas de aire lo que produce una mala lectura del sensor de turbina.

2.7 Modulo Bluetooth HC-06 para la Comunicación Serial

Para transmitir los datos se utiliza comunicación serial mediante el protocolo de comunicación inalámbrica Bluetooth, el módulo que se utiliza es el HC-06 el cual funciona solo como esclavo, en la Figura 2.36 se puede observar el modulo Bluetooth HC-06.

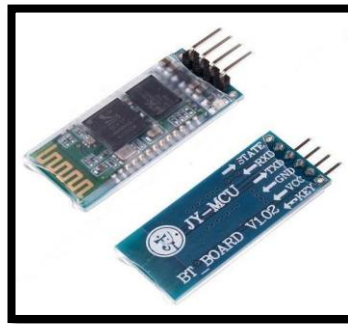


Figura 2.36: Módulo Bluetooth HC-06 [25]

Las ventajas principales del módulo Bluetooth HC-06, además de su reducido tamaño y sus excelentes características de transmisión y recepción que le brindan un alcance muy amplio, es el bajo consumo de corriente que posee tanto en funcionamiento, como en modo de espera, es decir, alimentado con energía, pero sin conexión o enlace a otro dispositivo, por ejemplo, un móvil con Sistema Operativo Android; demostrando características excepcionales en la implementación de este proyecto por su bajo consumo de energía, las características se puede observar en la Tabla 2.5

Tabla 2.5: Características del Bluetooth HC-06.

Frecuencia	2.4 GHz ISM Band
Bluetooth	V2.0+EDR
Potencia de salida de clase	Clase 2
Rango de baudios ajustable	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Default	Esclavo, 9600 Baud Rate, N, 8, 1.
Distancia	10 metros
Voltaje de funcionamiento	3.3V
Interfaz de Host	USB UART
Interfaz de audio	Ninguno
Protocolo	RFCOMM conocido también como emulador de puerto serial
Dimensiones	27x13x2.2 mm, peso: 1 gramo
Corriente	8mA

[25]

Otra característica es que la tensión de alimentación es de 3,3V y su consumo de corriente es de 8mA en transmisión/recepción activa lo transforman en un dispositivo ideal para trabajar con la tarjeta Arduino UNO. [25]. Para configurar el módulo Bluetooth vease el (Anexo J).

En la Figura 2.37 se observa la conexión de la tarjeta Arduino UNO con el módulo Bluetooth HC-06.

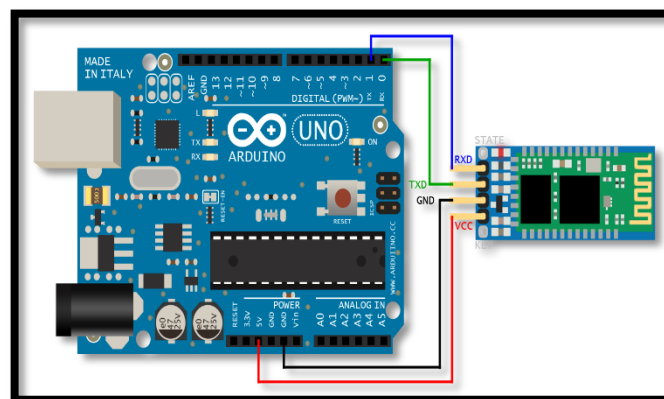


Figura 2.37: Conexión del Arduino UNO y el módulo Bluetooth HC-06

[26]

2.8 Diseño del Software.

Una vez terminado y revisado las señales de las placas del proyecto, se explica como se diseñó el software tanto en Arduino como en Android.

2.8.1 Entorno de desarrollo para Arduino.







Para empezar a programar la placa del Arduino UNO es necesario descargarse el programa de la página web de Arduino el entorno de desarrollo IDE. Se dispone de versiones para Windows y para MAC, En la Figura 2.38 se muestra el aspecto del entorno de programación, el cual está constituido por un editor de texto para escribir el código, una consola de texto, un área de mensajes, una barra de herramientas con botones para las funciones mas comunes, y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.



Figura 2.38: Entorno de desarrollo de Arduino.
[19]

Cuando se programa en el entorno de Arduino existe la posibilidad de buscar, reemplazar, cortar y pegar texto lo que facilita la programación. En el área de mensajes se puede observar la información mientras se ejecutan los programas y también muestra los errores de los programas. La barra de herramientas permite verificar el proceso de carga de la tarjeta Arduino, guardado de programas, y la monitorización serie, en la Tabla 2.6 se puede observar cada función que cumplen sus herramientas.

Tabla 2.6: Funciones comunes de la barra de herramientas del entorno de desarrollo para Arduino.

Herramientas	Función
	Verificar Chequea el código en busca de errores.
	Cargar Compila el código y lo graba en la placa E/S de Arduino.
	Nuevo Crea un nuevo sketch
	Abrir Presenta un menú de todos los programas sketch de su "sketchbook", (librería de sketch). Un click sobre uno de ellos lo abrirá en la ventana actual
	Guardar Guarda el programa Sketch
	Monitor Serial Inicia la monitorización serie

[19]

2.8.2 Desarrollo de la aplicación en Android

Para desarrollar la aplicación en Android se debe bajar el SDK de Android ya que este programa proporciona las bibliotecas API y todas la herramirntas de desarrollo para crear, probar y depurar las aplicaciones en Android, se debe tener la actualizacion del Java Development Kit o (JDK), es un software que proporciona herramientas de desarrollo para la creación de programas en Java.

Una vez que el IDE ya está cargado con el Android Developer Tools plugin y SDK se puede comenzar a desarrollar el proyecto.

a) Creación de un nuevo proyecto Android

Un proyecto Android tiene todos los archivos que contiene el código fuente de la aplicación Android, las herramientas del SDK de Android esto facilita al iniciar un nuevo proyecto con un conjunto de directorios de proyectos por defecto.

Para crear un nuevo proyecto hay que seguir los siguientes pasos:

1. Abrir el programa Eclipse seleccionar la carpeta y la dirección donde se va a guardar el proyecto como se observa en la Figura 2.39.

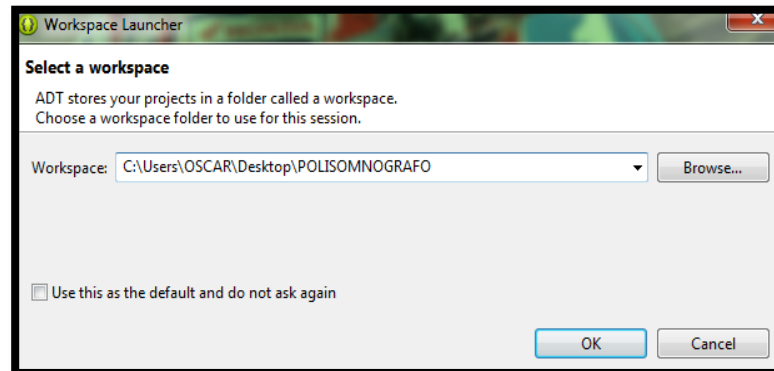



Figura 2.39: Selección de la carpeta y dirección del proyecto.
Elaborado por: Gómez O, Silva I.

2. Después de guardar el proyecto hay que dirigirse a la barra de herramientas y hacer clic en nuevo  como se indica en la Figura 2.40.

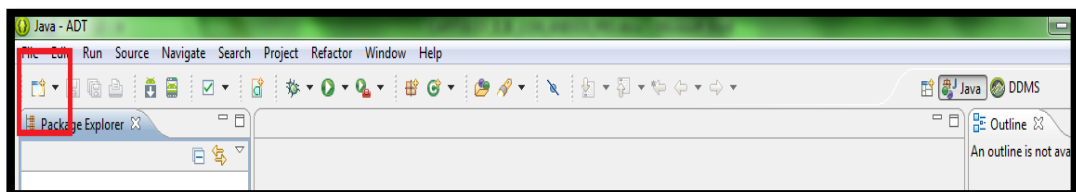


Figura 2.40: Selección de un nuevo proyecto.
Elaborado por: Gómez O, Silva I.

3. Se despliega 6 carpetas, se selecciona la carpeta "Android" y clic en siguiente como se indica en la Figura 2.41.

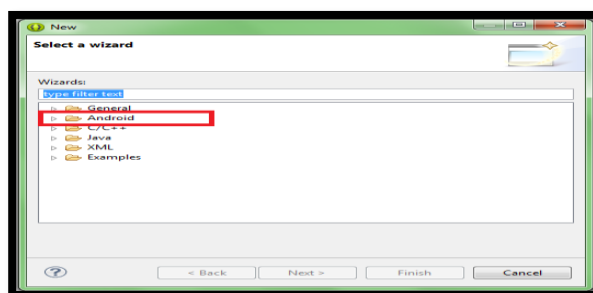


Figura 2.41: Selección de la carpeta Android.
Elaborado por: Gómez O, Silva I.

4. Se abre una ventana llamada Select a wizard en la cual se selecciona la opción “Proyecto de aplicación de Android”, y hacer clic en Siguiente, aparece una ventana como se ilustra en la Figura 2.42.

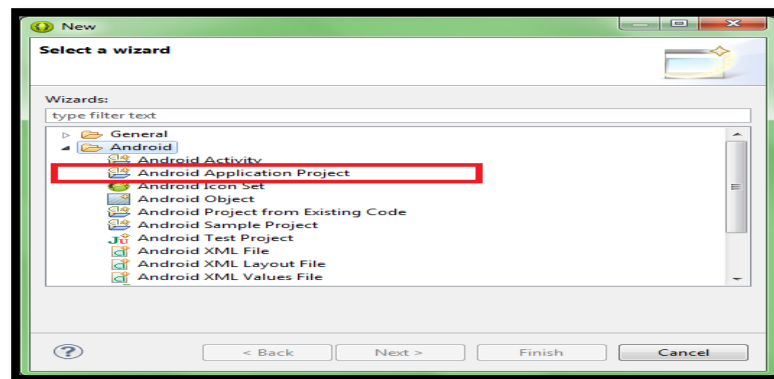


Figura 2.42: Selección de la carpeta Android.
Elaborado por: Gómez O, Silva I.

5. Al seleccionar la aplicación de Android aparece una ventana como se indica en la Figura 2.43 en donde se llena el formulario que a continuación se detalla cada una de las opciones:
- Nombre de la Aplicación: Corresponde al nombre de la aplicación que aparece a los usuarios. Para este proyecto el nombre es “Polisomnógrafo ”
 - Nombre del Proyecto: Corresponde al nombre del directorio del proyecto y el nombre visible en Eclipse
 - Nombre del paquete: Corresponde al espacio del nombre del paquete para la aplicación, éste nombre debe ser único en todos los paquetes instalados en el sistema Android.
 - SDK Mínimo requerido: Corresponde a la versión más baja de Android que admite la aplicación, indican usando el nivel de API.
 - SDK Target: Esto Indica la versión más alta de Android (también mediante el nivel de API). A medida que se disponga de nuevas versiones de Android, se debe probar la aplicación en la nueva versión y actualizar este valor para que coincida con el último nivel de API con el fin de aprovechar las nuevas características de la plataforma.

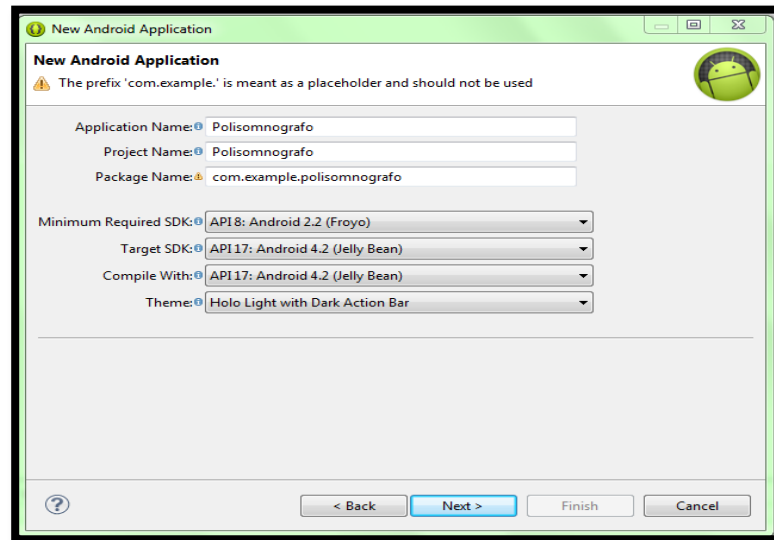


Figura 2.43: Creación de un nuevo proyecto Android.
Elaborado por: Gómez O, Silva I.

6. En la siguiente pantalla le configura el proyecto, se emplean selecciones predeterminadas como se ilustra en la Figura 2.44 y procedemos hacer clic en siguiente.

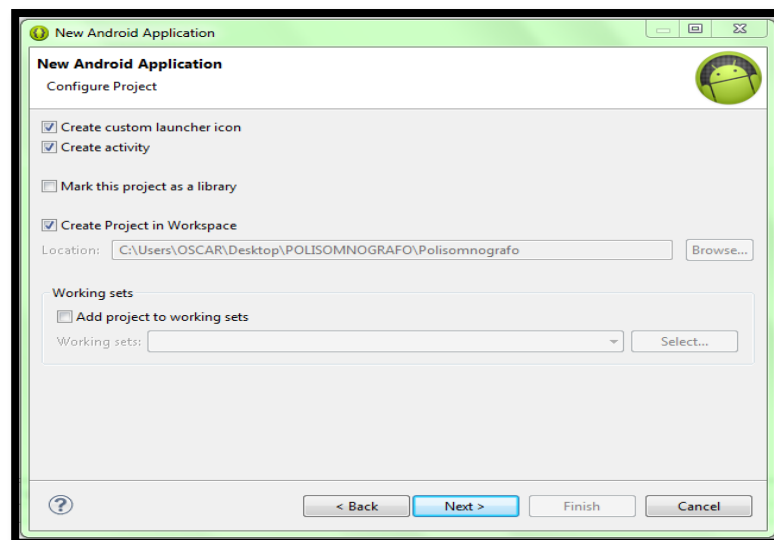


Figura 2.44: Creación de un nuevo proyecto Android.
Elaborado por: Gómez O, Silva I.

7. En la siguiente pantalla que se muestra en la Figura 2.45 es para crear un icono para el proyecto, personalizándolo de varias maneras. Antes de publicar la aplicación, asegúrese que el icono cumpla con las

especificaciones definidas en la iconografía guía de diseño y hacer clic en siguiente.

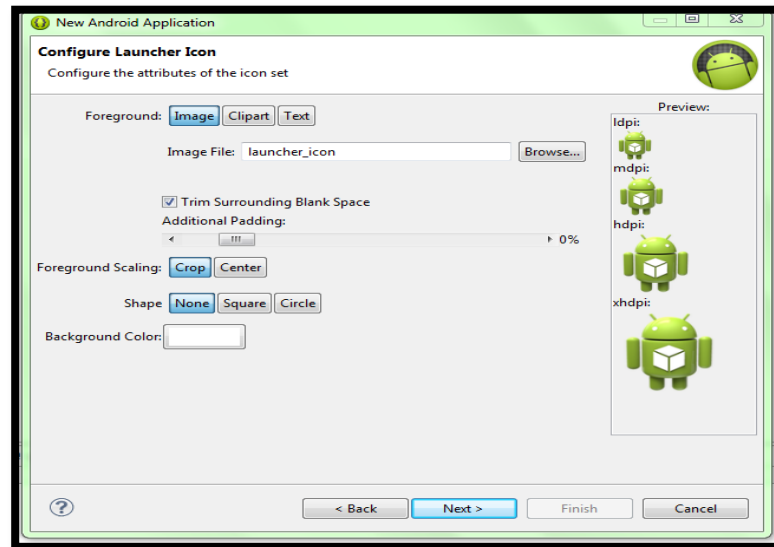


Figura 2.45: Configuración para la creación del icono para el proyecto.
Elaborado por: Gómez O, Silva I.

8. En la siguiente ventana que se muestra en la Figura 2.46 se puede seleccionar una plantilla de actividades, desde la cual se comienza a construir la aplicación; para este proyecto se seleccionó BlankActivity, a continuación dar clic en siguiente.

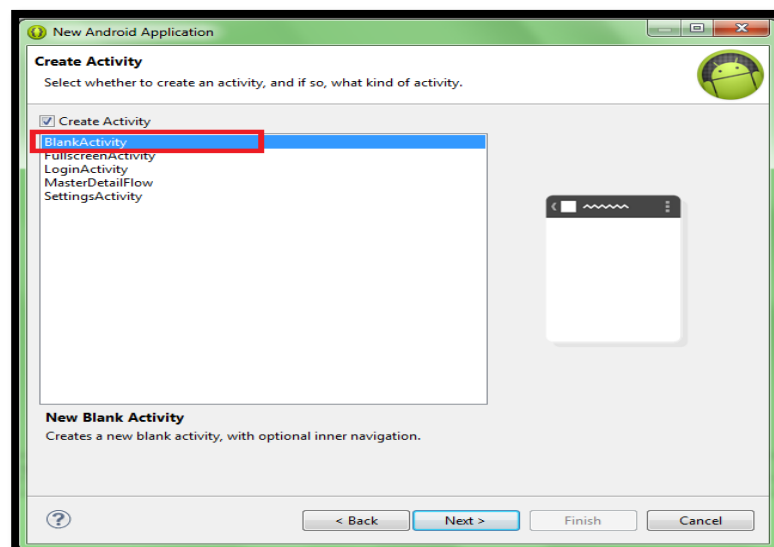


Figura 2.46: Creación de una actividad para un nuevo proyecto Android
Elaborado por: Gómez O, Silva I.

9. En la siguiente ventana que se muestra en la Figura 2.47 se configura la actividad en blanco, dejar los detalles de la actividad en su estado predeterminado y hacer clic en Finalizar.

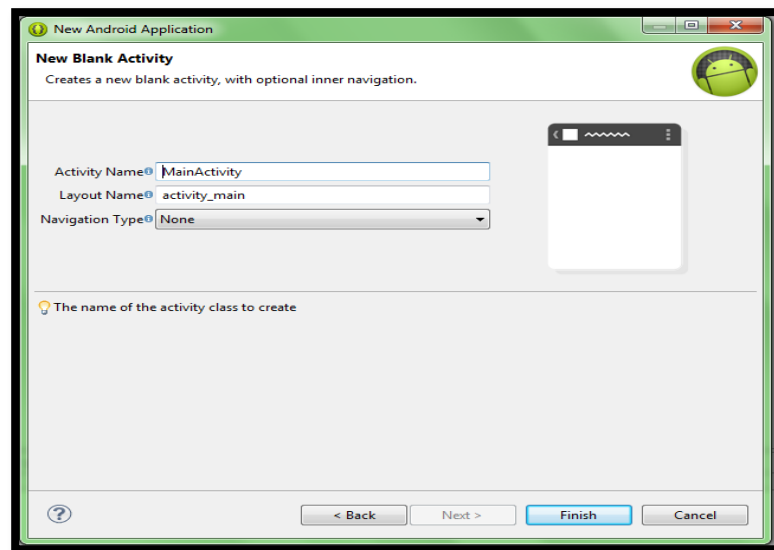


Figura 2.47: Creación de una actividad para un nuevo proyecto Android.
Elaborado por: Gómez O, Silva I.

10. Por último se observa una ventana que se despliega en donde se encuentra el proyecto ya realizado para posteriormente crear las aplicaciones que se desee hacer como se observa en la Figura 2.48.

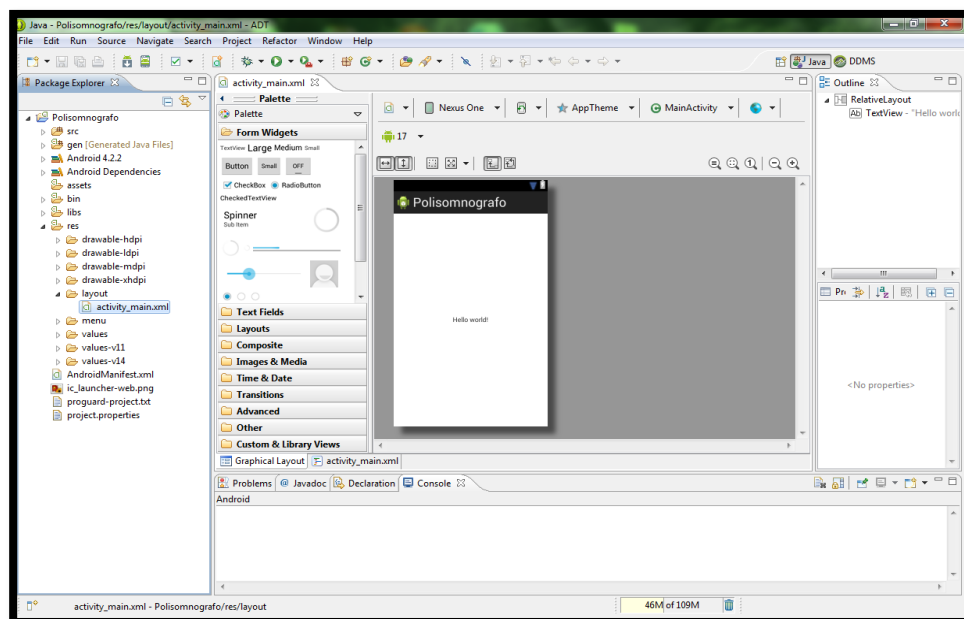


Figura 2.48: Proyecto finalizado en Android.
Elaborado por: Gómez O, Silva I.

La aplicación se encuentra terminada y se puede observar el icono de la aplicación en la Figura 2.49 y en la Figura 2.50, se muestra la ventana principal la cual contiene dos botones donde se indican las señales del ECG y los datos de la Presión sanguínea, SPO2, Flujo de aire de los pulmones.

En la Figura 2.51 y Figura 2.52 se observa los botones y sus respectivas ventanas creadas para cada una de las señales Biofísicas y Bioeléctricas donde se visualizaran los datos de las mismas.

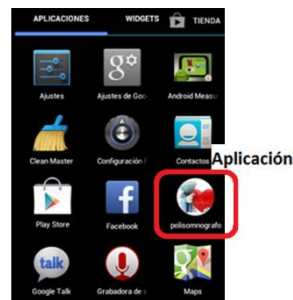


Figura 2.49: Aplicación creada para la Polisomnografía
Elaborado por: Gómez O, Silva I.



Figura 2.50: Ventana principal de la Aplicación del Polisomnógrafo
Elaborado por: Gómez O, Silva I.

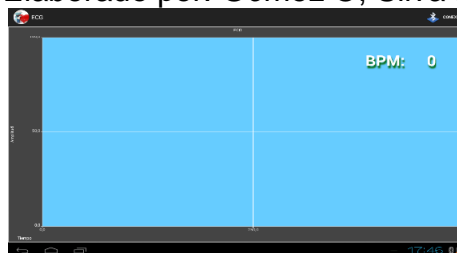


Figura 2.51: Ventana del ECG
Elaborado por: Gómez O, Silva I.



Figura 2.52: Ventana de datos para el SPO2, Presión Sanguínea y Flujo de Aire
Elaborado por: Gómez O, Silva I.

CAPÍTULO 3

3 PRUEBAS Y RESULTADOS OBTENIDOS

En este capítulo se realizó el análisis del comportamiento práctico del equipo de polisomnografía, las pruebas realizadas al equipo tanto en el ECG, SPO2, Espirometría y Presión sanguínea ayudaron a conocer los mínimos errores que se produjeron en el funcionamiento para mejorar el sistema y realizar las respectivas correcciones resultando un equipo de alto rendimiento y de precisión confiable. En tanto que los resultados de las pruebas ayudaron a la determinación de importantes conclusiones.

3.1 Pruebas del ECG

La adquisición de la señal electrocardiográfica, no presentó problema alguno gracias al adecuado diseño de los filtros analógicos (2.2.3), la presencia de ruido es mínima como se puede observar en el osciloscopio en la Figura 3.1 y en la Figura 3.2 se realiza la prueba con un voluntario Tabla 3.1 para la toma del ECG. En la Figura 3.3 se indica la señal adquirida en la Tablet con la aplicación en Android.

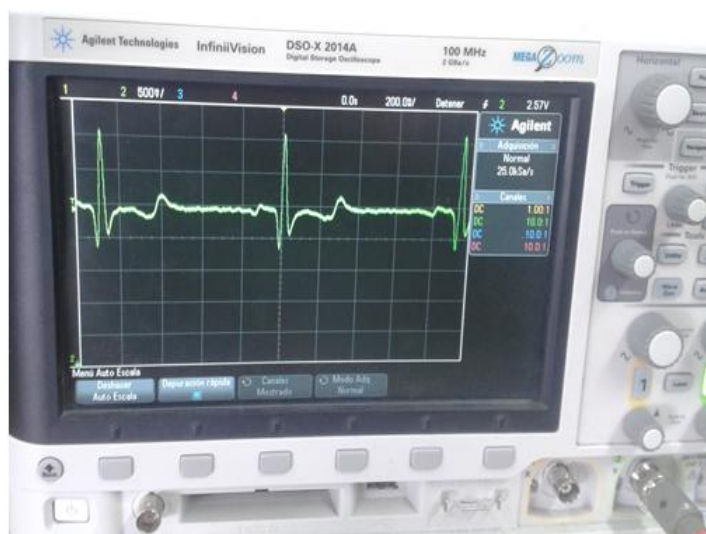


Figura 3.1: Señal obtenida del ECG implementado, mostrada en un osciloscopio.

Elaborado por: Gómez O, Silva I.

Tabla 3.1: Datos del Voluntario

VOLUNTARIO	
Nombre	Madeleine Luzuriaga
Edad	18 años
Sexo	Femenino

Elaborado por: Gómez O, Silva I.

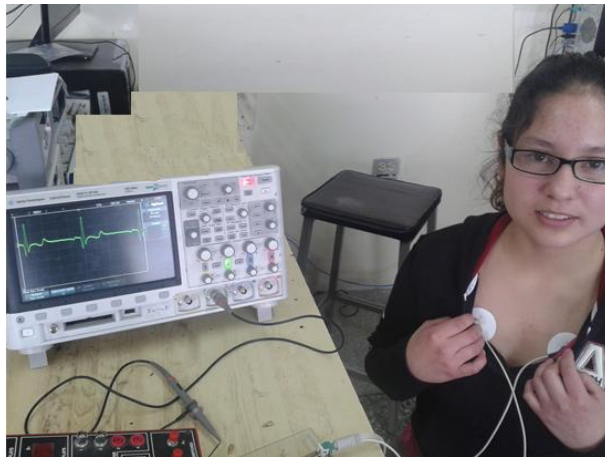


Figura 3.2: Señal del ECG de una voluntaria
Elaborado por: Gómez O, Silva I.



Figura 3.3: Señal del ECG adquirida en la Tablet
Elaborado por: Gómez O, Silva I.

Para la validación del ECG se tomó 10 muestras del ritmo cardíaco (BPM) con 10 diferentes pacientes usando un equipo patrón BMC SleepCare y el equipo de Polisomnografía, teniendo los siguientes resultados, Tabla 3.2.

Tabla 3.2: Datos del ECG

#	Edad	Sexo	BPM POLISOMNOGRAFO	BPM equipo Patrón.	Err.
1	55	F	70	73	3
2	37	F	101	108	7
3	29	F	105	109	4
4	27	F	77	79	2
5	23	F	93	92	1
6	25	M	78	79	1
7	26	M	75	73	2
8	43	M	65	68	3
9	30	M	70	71	1
10	26	M	69	72	3
PROMEDIO					2.7%

Elaborado por: Gómez O, Silva I.

3.2 Pruebas de la Presión Sanguínea

Para la adquisición de la señal de presión no se dio ningún problema, esto se debe al acondicionamiento de la señal del sensor de presión MPX5050 (2.3.5), el cual no presentó dificultad, por ser este lineal. En la Figura 3.4, se puede visualizar claramente la señal que entrega el circuito acondicionador donde cada pulso representa el choque de la sangre con las paredes de la arteria.

En la Figura 3.5 se visualiza la cooperación de un voluntario para la adquisición de la señal de su Presión Sanguínea y en la Figura 3.6 se muestra las señales codificadas en datos digitales, indicando la presión sistólica y diastólica en la plataforma Android.

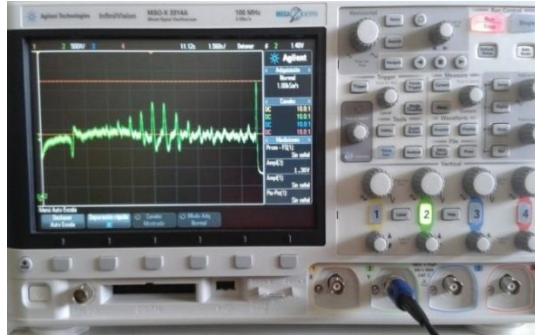


Figura 3.4: Señal obtenida del circuito de Presión Sanguínea implementado mostrada en un osciloscopio
Elaborado por: Gómez O, Silva I.

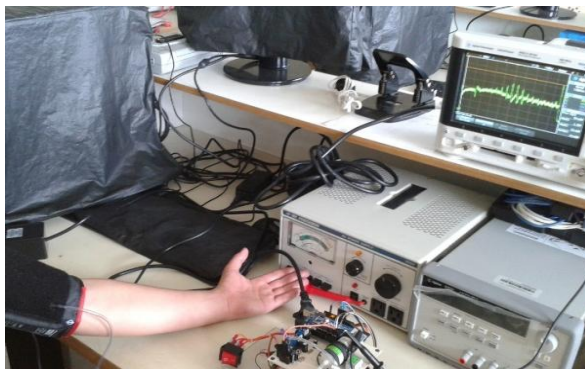


Figura 3.5: Señal de presión de un voluntario
Elaborado por: Gómez O, Silva I.



Figura 3.6: Señal de la Presión Sistólica y Diastólica adquirida en la Tablet.
Elaborado por: Gómez O, Silva I.

Para la validación de la Presión Sanguínea se tomó 10 muestras de la presión sistólica y diastólica de 10 pacientes diferentes tomándoles la presión con el equipo patrón VOIC ARM BLOOD - PRESSUREMETER MODEL NO:FT-C21Y y comparándola con el equipo de Polisomnografía implementado, teniendo los siguientes resultados. Tabla 3.3

Tabla 3.3: Datos de la Presión Sanguínea

#	Edad	Sexo	Sist.	Sist. Equipo patrón	Err.	Diast.	Diast. patrón.	Err.
1	55	F	110	100	10	69	60	9
2	37	F	120	115	5	70	60	10
3	29	F	125	120	5	55	50	5
4	27	F	109	100	9	75	69	6
5	23	F	110	100	10	70	60	10
6	25	M	111	100	11	70	60	10
7	26	M	126	120	6	70	80	10
8	43	M	110	100	10	89	80	9
9	30	M	140	133	7	90	80	10
10	26	M	130	120	10	80	72	8
PROMEDIO					8.3	PROMEDIO		8.7%

Elaborado por: Gómez O, Silva I.

3.3 Pruebas del SPO2

En la adquisición de esta señal se obtuvo el único problema de la interferencia de luz del medio ambiente, el capítulo (2.6.3) explica los artefactos del SPO2. En la Figura 3.7 se muestra la señal adquirida del circuito de acondicionamiento de oximetría con sus tiempos en bajo y en alto correspondientes al LED Rojo e Infrarrojo, respectivamente. En la Figura 3.8 se observa al voluntario y su señal correspondiente al nivel de oxígeno en la sangre representado en porcentaje como se puede apreciar en la Figura 3.9

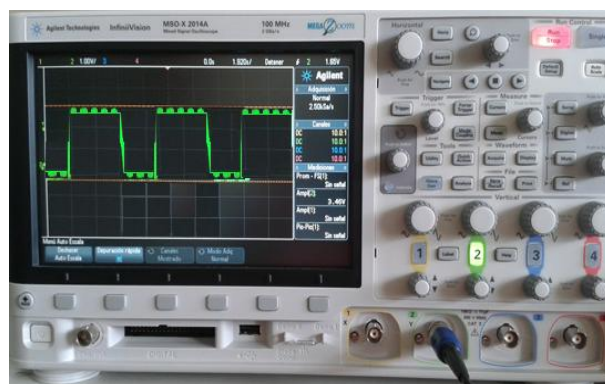


Figura 3.7: Señal obtenida del circuito de Oximetría implementado mostrada en un osciloscopio

Elaborado por: Gómez O, Silva I.

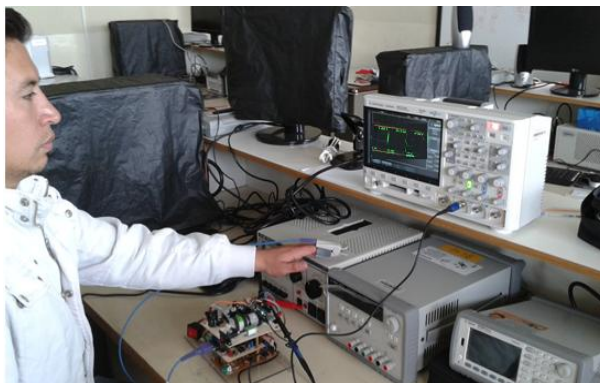


Figura 3.8: Señal del SPO2 de un voluntario.
Elaborado por: Gómez O, Silva I.



Figura 3.9: Señal del SPO2 adquirida en una Tablet
Elaborado por: Gómez O, Silva I.

Una vez implementado el Oxímetro se procede a su calibración con el equipo patrón BMC SleepCare. A continuación se toma mediciones para la validación del SPO2 comparándolas con la del equipo patrón. En la Tabla 3.4 se observa el resultado de las mediciones.

Tabla 3.4: Datos del SPO2

#	edad	sexo	SPO2 POLISOMNOGRAFO	SPO2 equipo Patrón.	Err.
1	55	F	90.35%	89.70%	0.65
2	37	F	91.12%	90.102%	1.02
3	29	F	88.64%	88.102%	0.53
4	23	F	90.03%	95.82%	5.79
5	23	F	95.00%	96.65%	1.65
6	25	M	92.34%	92.80%	0.46
7	26	M	94.50%	94.02%	0.48
8	43	M	94.20%	93.80%	0.40
9	26	M	93.76%	93.01%	0.75
10	28	M	94.22%	93.90%	0.32
PROMEDIO					1.2%

Elaborado por: Gómez O, Silva I.

3.4 Pruebas del Espirometría

Al adquirir la señal de flujo de aire no presento problema alguno gracias al adecuado diseño del sensor de turbina acoplado a la mascarilla para una mejor concentración y sin fugas de aire, se lo observa en la Figura 3.10 y en la Figura 3.11 se aprecia el tren de pulsos correspondiente a la señal captada por el sensor infrarrojo al detectar el corte del haz de luz provocado por las hélices de la turbina. En la Figura 3.12 y Figura 3.13 se observa la señal adquirida y mostrada en centímetros cúbicos en la Tablet.



Figura 3.10: Mascarilla acoplada al Sensor de turbina e infrarrojo.
Elaborado por: Gómez O, Silva I.

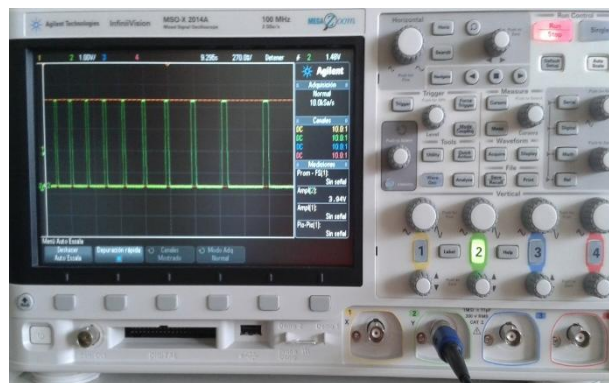


Figura 3.11: Señal obtenida en el circuito de espirometría mostrada en un osciloscopio.
Elaborado por: Gómez O, Silva I.

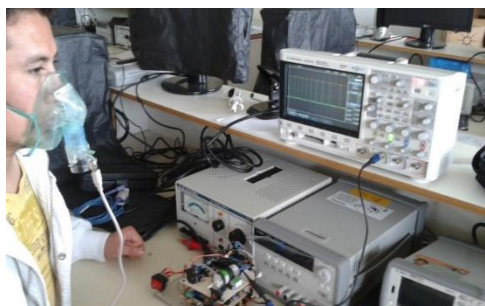


Figura 3.12: Señal del Espirómetro de un Voluntario.
Elaborado por: Gómez O, Silva I.



Figura 3.13: Señal de flujo de aire adquirida en la Tablet.
Elaborado por: Gómez O, Silva I.

Una vez implementado el espirómetro se procede a su calibración con el equipo patrón BMC SleepCare A continuación se toma mediciones para la validación del Espirómetro comparándolas con la del equipo patrón. En la Tabla 3.5 se observa el resultado de las mediciones.

Tabla 3.5: Datos del Espirómetro

#	Edad	Sexo	Espirómetro POLISOMNOGRAFO	Espirómetro equipo Patrón.	Err.
1	33		2030 cc	2034 cc	4
2	34		1997 cc	1990 cc	7
3	52		3980 cc	3983 cc	3
4	37		5700 cc	5709 cc	9
5	23		4789 cc	4784 cc	5
6	41		3890 cc	3898 cc	8
7	26		5953 cc	5955 cc	2
8	38		3945 cc	3940 cc	5
9	35		3678 cc	3679 cc	1
10	28		2987 cc	2990 cc	3
PROMEDIO					4.7%

Elaborado por: Gómez O, Silva I.

CAPÍTULO 4

4 Análisis Económico

Uno de los motivos que impulso al desarrollo de este equipo fue el de construir uno económico que pueda satisfacer las necesidades de personas con síntomas de SAHOS, y puedan revisar sus signos vitales desde los hogares.

Comparamos un equipo de características semejantes, el equipo escogido es el MONITOR DE PACIENTE MULTIPARÁMETRO CMS6000 Anexo K.

Por las características similares que posee con nuestro equipo. Las características son las siguientes:

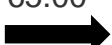
- ECG
- SPO2
- Sistemas de presión arterial no invasivo
- Espirómetro
- Temperatura



Figura 4.1: Monitor de paciente multiparámetros
[27]

El precio de este equipo en el Ecuador es de \$2895.00USD fabricado en Estados Unidos [27]. Por otro lado se coloca el detalle económico del equipo de Polisomnografía inalámbrico.

Tabla 4.1: Costo total del Polisomnógrafo inalámbrico

CANTIDAD	REFERENCIA	VALOR UNITARIO	VALOR TOTAL
9	Potenciómetros	0.65	5.85
3	Borneras de 3 Pines	0.35	1.05
6	Borneras 2 Pines	0.28	1.68
3	Amplificador operacional TL082	0.70	2.10
2	Amplificador operacional TL082	0.72	1.44
3	Circuito Integrado 555	0.65	1.95
1	Operacional 74L500	0.50	0.50
3	Operacional 74L504	0.55	1.65
1	Operacional TL074	0.89	0.89
12	Capacitor 100 μ F	0.25	3.00
2	33 μ F	0.15	0.30
2	47 μ F	0.14	0.28
18	Capacitores Electrolíticos 10 μ F	0.15	2.70
14	Transistores 2N390X	0.10	1.40
15	Diodos rectificadores 1N4001	0.10	1.50
11	CAP Electrolíticos	0.40	4.40
3	Diodos LED	0.25	0.70
15	Zócalos de 8 Pines	0.12	1.80
7	Zócalos de 14 Pines	0.16	1.12
7	Potenciómetros de Precisión 122	0.65	4.55
4	Diodos Zener	0.19	0.76
3	Regleta GM	0.69	2.07
9	Cables de Protoboard	0.20	1.80
1	Regulador de tensión 7805	0.48	0.48
1	Regulador de Tensión 7905	0.55	0.55
1	Regulador de Tensión LM1117	0.98	0.98
1	Jack DB9 Hembra + Cable	1.30	1.30
1	Jack USB	0.60	0.60
98	Resistencias	0.02	1.96
1	Molex	0.25	0.25
1	Transmisor de Infrarrojo + Cable	0.98	0.98
1	Receptor e Infrarrojo + Cable	0.78	0.78
1	Sensor de Presión MPX 5050	32.00	32.00
3	Placa PCB de Fibra de vidrio	13.72	41.16
2	Módulos Bluetooth HC-06	27.50	55.00
4	Diseño de Placa	20.00	80.00
8	Cables de Protoboard Macho hembra	0.14	1.14
1	LM331	2.19	2.19
1	LM324	0.31	0.31
1	Monitor de Presión Sanguínea	65.00	65.00
		CONTINUA	

1	Cable apantallado para ECG	45.00	45.00
150	Electrodos	0.50	75.00
1	Mascarilla	6.50	6.5
1	Tubo de Turbina	5.00	5.00
1	Cable USB Macho / Macho	3.00	3.00
1	Sensor NELLCOR	86.00	86.00
1	Galón Gel para Ultra Sonido	7.95	7.95
30	Postes	0.40	12.00
2	Arduino UNO	35.00	70.00
1	Switch doble	2.40	2.40
1	Transformador de 110Ac/24Ac	7.00	7.00
1	Caja de Proyectos	6.00	6.00
1	Cable de Poder Triple	2.50	2.50
1	Cable de Poder Doble	3.50	2.50
1	Caja de Acrílico	30.00	30.00
1	Maleta y Bordado diseñado	30.00	30.00
1	Adhesivo con nombres	30.00	30.00
1		5.00	5.00
	SUBTOTAL	0.00%	754.04
	IVA	IVA12%	90.48

SUBTOTAL 1**844.72**

1	Aplicación Android	300	300.00
60	Hora de Ingeniería	5	300.00
	SUBTOTAL	0.00	600.00
	IVA	IVA12%	72.00

SUBTOTAL 2**672.00**

1	SUBTOTAL 1	844.48	844.48
1	SUBTOTAL 2	672.00	672.00

TOTAL**1515.48**

Elaborado por: Gómez O, Silva I.

Hay que tomar en cuenta que la Producción en masa de este equipo de polisomnografía bajaría el costo del equipo en un 40% del precio normal; y realizando una comparación de los precios del Monitor de paciente multiparámetros y el equipo de Polisomnografía se puede observar que, si es más económico, esto se debe a que la aplicación de la plataforma en Android es de Bajo costo, y la utilización de la tecnología de Tablet's y celulares con

el sistema operativo en Android ha tenido una gran acogida a nivel mundial aumentando versátilmente en el diseño y creación de aplicaciones.

La comparación económica del equipo de polisomnografía y el Monitor de paciente multiparámetros es muy notable, ya que existe un ahorro de \$1379.52 dólares, sólo en el primer prototipo, si la producción fuera en masa, el ahorro del 40% sería de \$1985.71, resultando cada equipo con un precio comercial para el mercado nacional de \$909.28.

4.1 Proyectos futuros

A un futuro se puede realizar mejoras en este equipo, como la utilización de nuevas tecnologías como son los electrodos inalámbricos para la implementación del ECG, que ayudaría a la reducción de cables dando una mejor comodidad al paciente.

Se puede aumentar el número de señales biofísicas y bioeléctricas como el Electromiograma, Electroencefalograma, Temperatura corporal, Electrooculograma, etc. Para tener un equipo más completo y de mayor eficiencia.

Al tener la aplicación en un dispositivo Android con conectividad a Internet se lo puede introducir al mundo de la Telemedicina apoyando al paciente desde la comodidad de su hogar por un experto en cualquier parte del mundo.

CAPÍTULO 5

5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se diseñó e implementó el Polisomnógrafo, con transmisión de datos inalámbricos, para la medición del ECG, SPO2, ESPIRÓMETRO, PRESIÓN SANGUÍNEA.
- Se estudió e identificó claramente, cada uno de los componentes y elementos que intervienen en el diseño e implementación del Polisomnógrafo.
- El Polisomnógrafo implementado permite la medición de la señal electrocardiográfica (ECG), la presión arterial, la saturación de oxígeno en la sangre (SPO2), y el flujo de aire en los pulmones durante el sueño, con una alta velocidad de actualización de los datos.
- Las señales biofísicas, fueron acondicionadas a un rango de 0 a 5 voltios, que es el máximo voltaje que se permite a la entrada de la tarjeta Arduino UNO.
- Se implementó una aplicación móvil sencilla, amigable al usuario que satisface las necesidades tanto del ECG, Presión arterial, SPO2 y el Flujo de aire, que permita una fácil conexión entre el dispositivo móvil con sistema Android, a través del protocolo de comunicación inalámbrica Bluetooth.
- Los valores medidos difieren en un pequeño porcentaje de los valores medidos con los equipos patrones de medicina, debido principalmente a las tolerancias de los elementos pasivos utilizados.

- La visualización de la señal electrocardiográfica (ECG) se lo hizo con la ayuda de la librería Androidplot que permite graficar un vector numérico de n posiciones, de una forma sencilla.
- La comunicación entre el Polisomnógrafo y el dispositivo móvil con sistema Android fue efectiva dentro de un rango de 10 metros con línea de vista, luego del cual la comunicación se ve interrumpida.
- Se notó que la señal del ECG tomada en un paciente y otro paciente, la amplitud de la onda cambia significativamente en 0.4 - 4mA notándose la diferencia en la onda QRS de la señal electrocardiográfica.
- El aumento en el orden de los filtros Butterworth y la implementación del filtro Notch disminuye notablemente el ruido en las señales del ECG y SPO2.
- Es más fácil de usar, limpiar, desechar e incluso resulta económico el diseño del espirómetro al usar el sensor de turbina acoplado a los infrarrojos, tomando en cuenta que la velocidad del movimiento circular de las aspas es proporcional al flujo de aire que circula por ellas.
- La utilización del sensor NELLCOR es más amigable y transparente al usuario que otros sensores de otras marcas, y en el diseño es compatible con los conectores DB9 tipo hembra por ser más accesibles en el mercado.
- La medición del SPO2 varía según la pigmentación de la piel del paciente, mientras más oscura es menor porcentaje de lectura muestra la medida.
- La medición de la presión sanguínea con el equipo de polisomnografía y comparando la misma medición con un equipo patrón resulta que se encuentra dentro de los rangos permisibles.

- La calibración del ECG, SPO2, ESPIRÓMETRO, PRESIÓN SANGUÍNEA, con sus distintos equipos patrones permite dar la validación del equipo de polisomnografía.

5.2 RECOMENDACIONES

- Para una buena recepción de la señal electrocardiográfica se recomienda una buena limpieza con alcohol y si es necesario rasurarse la zona donde van colocados los electrodos con la finalidad de reducir al máximo la resistencia eléctrica causada por la suciedad y los vellos.
- Se tiene una mejor lectura del nivel de oxígeno en la sangre si el sensor NELLCOR no se expone a la luz ambiente, por el espectro electromagnético que influye en el sensor fotodiodo.
- Se requiere de un tiempo de estabilización de los circuitos aproximadamente de 30 segundos para la medición de las señales biofísicas y bioeléctricas para que estas sean más exactas.
- Observar que los cables y las mangueras del Polisomnógrafo no se enreden ya que puede provocar toma de medidas erróneas.
- Antes de iniciar la aplicación del Polisomnógrafo en el dispositivo móvil, se debe encender el Bluetooth del mismo.
- Para emparejar los módulos Bluetooth del Polisomnógrafo, con el del dispositivo móvil por primera vez, se debe ingresar una contraseña la cual en nuestro caso es 1234.
- El consumo de corriente de la bomba deja sin la suficiente energía para alimentar el equipo lo que es recomendable mínimo 2 amperios de alimentación para que alimente a todo el equipo y este funcione a su máximo rendimiento.

- El consumo de corriente de la carga da como resultado que se recaliente los reguladores de tensión, es necesario la instalación de un ventilador para mantener encendido el equipo largos lapsos de tiempo, como es la medición de datos en la polisomnografía, que consta de lapsos de 8 horas seguidas.
- Debido a las limitaciones del tamaño del cable del ECG el paciente no se puede mover cómodamente.

6 REFERENCIAS BIBLIOGRÁFICAS

- [1] H. B. R. U. S. O. U. R. GÓMEZ RESTREPO., «Polisomnografía en Psiquiatría,» de *Psiquiatría Clínica*, Bogotá, Médica Panamericana, 2008, p. 134.
- [2] A. Contreras, «Patologías del sueño,» *Revista Médica*, vol. 24, nº 3, p. 343, 2013.
- [3] D. P. C. LUIS DARÍO LARRATEGUY., «Modalidades Respiratorias,» de *Guía Esencial de Metodología en Ventilación Mecánica no Invasiva*, Madrid, Médica Panamericana, 2010, pp. 239 - 240.
- [4] R. J. L. JORGE NOGALES - GAETE ARCHIBALDO DONOSO S., «Tratado de Neurología Clínica,» de *Polisomnografía*, Santiago, Universitaria, 2005, pp. 188 - 189.
- [5] J. W. Tompkins y F. H. Neter, «Electrocardiografía.es,» Señales Clínicas, s.f. s.f. 1993. [En línea]. Available: http://www.electrocardiografia.es/bases_datos.html. [Último acceso: 5 Enero 2014].
- [6] D. DAVIS, «Interpretación del Electrocardiograma su dominio rápido y exacto,» de *Electrocardiograma*, Buenos Aires, Panamericana, 2008, pp. 1 - 3.
- [7] Hipertensión, «123rf,» Medidor de la Presión Arterial, s.f. s.f. s.f.. [En línea]. Available: <http://es.123rf.com/imagenes-de-archivo/hipertension.html>. [Último acceso: 2 Septiembre 2014].
- [8] M. J. C. L. J. M. Q. C. X. FUENTES ARDERIU., «Regulación de la Presión Sanguínea,» de *Bioquímica Clínica y Patología molecular*, Barcelona, Reverté, S.A., 1998, p. 822.
- [9] N. K. BBANERJEE, «Presión Arterial,» de *Tensión Arterial Etiología y Tratamiento*, pp. 1-4.
- [1] N. Hutchison, I. Newton y J. B. Thomas, «Wikipedia,» 16 Agosto 2014. [En línea]. Available: http://es.wikipedia.org/wiki/Espectro_visible. [Último acceso: 2 Octubre 2014].
- [1] D. M. Home, «Dolphin Medical Home,» Powered by Giraphix Studio, s.f. [En línea]. Available: <http://www.dolphinmedical.com>. [Último acceso: 2 Octubre 2014].

<http://www.dolphinmedical.co.il/index.php/61-home>. [Último acceso: 22 Febrero 2014].

[1 A. D. L. A. B. JUAN CARLOS MONTEJO, «Manual de Medicina
2] Intensiva,» de *Pulsioximetría*, España, Elsevier, 2006, p. 53.

[1 NIOSH, «Guía de Niosh sobre entrenamiento en espirometría,» 7 Marzo
3] 2007. [En línea]. Available: http://www.cdc.gov/spanish/niosh/docs/2004-154c_sp/pdfs/2004-154c.pdf. [Último acceso: 2 12 2013].

[1 J. E. Cimas Hernando y J. Pérez Fernández, «Ideap,» s.f. s.f. s.f.. [En
4] línea]. Available:
http://www.samfyc.org/index.php?option=com_docman&task=doc_download&gid=234&Itemid=5. [Último acceso: 7 Enero 2014].

[1 Basar, «Medidas de Volumen Pulmonar,» de *INTRODUCCIÓN A LA
5] BIOINGENIERÍA*, Barcelona, Marcombo S.A., 1988, p. 91.

[1 M. Andreu, de *Principios de comunicaciones móviles*, Barcelona, UPC,
6] 2003, p. 41.

[1 LaBlackBerry, «LaBlackBerry,» BlackBerry, 23 Julio 2013. [En línea].
7] Available: <http://lablackberry.com/2013/07/blackberry-soporta-bluetooth-listo-listo-para-desarrollo-de-aplicaciones-de-m2m.html>. [Último acceso: 12 Enero 2014].

[1 A. Características, «Características técnicas del Arduino Uno,» s.f. s.f. s.f..
8] [En línea]. Available:
<http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf>. [Último acceso: 12 Mayo 2014].

[1 A. Uno, «Arduino,» Atmel, s.f. s.f. s.f.. [En línea]. Available:
9] <http://arduino.cc/en/Main/arduinoBoardUno>. [Último acceso: 15 Marzo 2014].

[2 D. C. Godoy, «Unidad modular de 12 derivaciones,» Colciencias,
0] Medellín, 2008.

[2 J. M. Drake, «El amplificador de Instrumentación,» Universidad de
1] Cantabria, Cantabria, 2005.

[2 K.-7. T. HANDBOOK, BIOMEDICAL MEASUREMENT SYSTEM, SGS,
2] Ed., Taiwan R.O.C.: UKAS, 2000, pp. 5-6.

- [2] G. Vargas, «Sensores Biomédicos,» s.f. s.f. 1997. [En línea]. Available:
3] <https://www.dspace.espol.edu.ec/bitstream/123456789/3257/1/5776.pdf>.
[Último acceso: 2 Octubre 2014].
- [2] L. F. Aguirre Valencia, «Repositorio ESPE,» s.f. s.f. 2007. [En línea].
4] Available: <http://repositorio.espe.edu.ec/bitstream/21000/179/1/T-ESPE-025087.pdf>. [Último acceso: 1 Febrero 2014].
- [2] M. B. HC-06, «ELECTGPL,» 29 Junio 2013. [En línea]. Available:
5] <http://electgpl.blogspot.com/2013/01/modulo-bluetooth-hc-06.html>.
[Último acceso: 3 Agosto 2014].
- [2] J. Ruben, «GeekFaktory,» 21 Febrero 2014. [En línea]. Available:
6] <http://www.geekfactory.mx/radio/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/>. [Último acceso: 2 Mayo 2014].
- [2] M. e. p. multiparámetros, «Mercado Libre,» 2 Agosto 2014. [En línea].
7] Available: [http://articulo.mercadolibre.com.ec/MEC-405451574-monitor-de-paciente-ecg-spo2-nibp-resp-temp-multiparametro-_JM#D\[S:HOME,L:RECOMITEM-CORE-UNO-HISTORYITEMS,V:0\]](http://articulo.mercadolibre.com.ec/MEC-405451574-monitor-de-paciente-ecg-spo2-nibp-resp-temp-multiparametro-_JM#D[S:HOME,L:RECOMITEM-CORE-UNO-HISTORYITEMS,V:0]).
[Último acceso: 26 Septiembre 2014].
- [2] T. Instruments, «Electrónicos Caldas,» 25 Julio 2007. [En línea].
8] Available: <http://www.electronicoscaldas.com/amplificadores-operacionales/39-amplificador-operacional-tl084.html>. [Último acceso: 20 agosto 2014].
- [2] L. M. T. MORERA., «Tratados de Cuidados Críticos y Emergencias,» de
9] *Pulsioximetría*, Madrid, Aran, 2002, pp. 115 - 116.
- [3] M. M. S. JOSEPH W. KANE, «Medida de presión sanguínea mediante el
0] Esfignomanómetro,» de *Física*, Barcelona, Reverté, S.A., 2007, p. 299.

ANEXOS

ÍNDICE DE ANEXOS

Anexo A.	GLOSARIO DE TÉRMINOS
Anexo B.	BIOGRAFÍAS
Anexo C	ARDUINO UNO
Anexo D	TL084 AMLIFICADOR OPERACIONAL
Anexo E	CIRCUITOS ESQUEMÁTICOS
Anexo F	DIAGRAMAS IMPRESOS
Anexo G	PLACAS TERMINADAS
Anexo H	CARACTERÍSTICAS DE LA BOMBA CLARK
Anexo I	CARACTERÍSTICAS DE LA VÁLVULA SELENOIDE
Anexo J	INTERFAZ DE CONFIGURACIÓN DE COMANDOS AT EN HC-06
Anexo K	MONITOR DE PACIENTE MULTIPARÁMETROS CMS6000
Anexo L	DIAGRAMAS DE FLUJO
Anexo M	PROGRAMACIÓN EN ECLIPSE
Anexo N	PROGRAMACIÓN ARDUINO UNO

Anexo A. GLOSARIO DE TÉRMINOS

1. Hipoventilación- Disminución de la cantidad de aire inspirado que entra por minuto en los alvéolos pulmonares para el [consumo de oxígeno](#) del individuo.
2. Pretest.- Prueba propuesta al principio de una o varias intervenciones para evaluar el nivel de capacidad o de realización de un hábito de vida.
3. Hipersomnes.- Son personas que necesitan dormir más de 70 horas semanales y no se sienten mentalmente despiertos.
4. Otorrinolaringológica.- Es parte de la medicina que se ocupa de las enfermedades del oído, nariz y garganta.
5. Hipersomnio.- Dormir demasiado sin ninguna causa obvia.
6. Parasomnías.- Es un trastorno de la conducta durante el sueño, asociadas con episodios breves de despertar.
7. Eritrocitosis de etiología.- Es el incremento de glóbulos rojos en la sangre.
8. Angina.- Inflamación de las Amígdalas y de las regiones continuas a ellas.
9. Disnea paroxística nocturna.- Es una afección que consiste en crisis de falta de aire al estar acostado.
10. NYHA.- New York Heart Association, Valora la actividad física del paciente con insuficiencia cardíaca congestiva.
11. Arritmias.- Es un trastorno del ritmo cardiaco.
12. Bariátrica.- Conjunto de procedimientos quirúrgicos usados para tratar la obesidad.
13. Comorbilidad.- La presencia de uno o más trastornos (o enfermedades) además de la enfermedad o trastorno primario.
14. Retrognatia.- Posición de la mandíbula por detrás del plano del plano de la frente.
15. Hipertrofia amigdalina.- Afección de la garganta o faringe que cursa con inflamación e infección de una amígdala.
16. CPAP.- Termino médico para describir presión continua en la vía aérea.

17. Cheyne Stokes.- Tipo de respiración que se caracteriza por cambios de ritmo en la intensidad respiratoria.
18. (SaO₂).- Oxígeno en la sangre arterial.
19. Hemoglobina.- Es una heteroproteína de la sangre de color rojo, que transporta el oxígeno desde los órganos respiratorios hasta los tejidos.
20. Vasoconstricción.- Estrechamiento de un vaso sanguíneo.
21. Hemodinámico.- Movimiento y fuerza de los vasos sanguíneos.
22. Torr.- Es una unidad de presión un Torr equivale a un milímetro de mercurio.
23. Pseudoaleatoria.- Es un número generado en un proceso que parece producir números al azar, pero no lo hace realmente.

Anexo B. BIOGRAFÍAS

1. Richard Catón.- Científico inglés. Liverpool, 1842 – 1926.
2. Hans Berger.- Psiquiatra y neurólogo alemán. Neuses, 1873 – Jena 1941.
3. Harvey.- Médico Fisiólogo inglés. Londres, 1578 – 1657.
4. Nathaniel Kleitman.- Profesor de fisiología ruso. Kishinev, 1895 – 1999.
5. Moruzzi.- Científico italiano. Campagnola Emilia, 1910 – 1982.
6. Eugene Aserinsky.- Doctor en Medicina ruso. Moscú, 1921 – 1998.
7. Dement.- Doctor en Medicina americano. Chicago, 1928.
8. Christian Guilleminault.- Neurólogo Psiquiatra francés. París, 1972.
9. Jean Baptiste Edouard Gelineau.- Médico francés, Blaye. Gironde, 1828 – 1906.
10. Charles Dickens.- Novelista inglés. Portsmouth, 1812 – 1870.
11. Peretz Lavie.- Psicólogo israelita. Petah Tikva, 1949.
12. Einthoven.- Medico holandés. Indonesia, 1860 – 1927.

Anexo C. ARDUINO UNO

El Arduino Uno es una placa electrónica basada en el microprocesador Atmega328. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 pueden utilizarse para salidas PWM), 6 entradas analógicas, un resonador cerámico 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o el poder con un adaptador o la batería AC-to-DC para empezar.

El Uno se diferencia de todas las placas anteriores en que no utiliza el chip controlador de USB a serial FTDI. En lugar de ello, cuenta con la Atmega16U2 (Atmega8U2 hasta la versión R2) programado como un convertidor de USB a serie.

Revisión 2 de la tabla Uno tiene una resistencia tirando de la línea 8U2 HWB a tierra, por lo que es más fácil de poner en modo DFU.

Revisión 3 de la placa tiene las siguientes características nuevas:

- 1.0 Pin out: SDA añadido y pines SCL que están cerca al pin AREF y otros dos nuevos pernos colocados cerca del pin de RESET, la instrucción IOREF que permiten a los escudos para adaptarse al voltaje suministrado desde la pizarra. En el futuro, los escudos serán compatibles tanto con el tablero que utiliza el AVR, que funciona con 5V y con el Arduino Debido que funciona con 3.3V. El segundo es un pin no está conectado, que se reserva para usos futuros.
- Circuito de rearme fuerte.
- Atmega 16U2 sustituir el 8U2.

"Uno" significa uno en italiano y se nombra para conmemorar el próximo lanzamiento de Arduino 1.0. El Uno y la versión 1.0 será la versión de referencia de los Arduino, moviéndose hacia adelante. El Uno es el último de una serie de placas Arduino USB y el modelo de referencia para la plataforma

Arduino; para una comparación con las versiones anteriores, consulte el índice de la placa Arduino.

Resumen:

Microcontrolador	ATmega328
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines Digitales I / O	14(6 proporcionan PWM)
Pines de entrada analógica	6
Corriente DC por Pin I / O	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria Flash	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MHz

[19]

Anexo D. TL084 AMLIFICADOR OPERACIONAL

Un amplificador operacional es un amplificador diferencial. Desde el punto de vista de una señal, tiene tres terminales: dos terminales de entrada y un terminal de salida. La Figura 0.1 muestra el símbolo que representa el Amplificador Operacional.

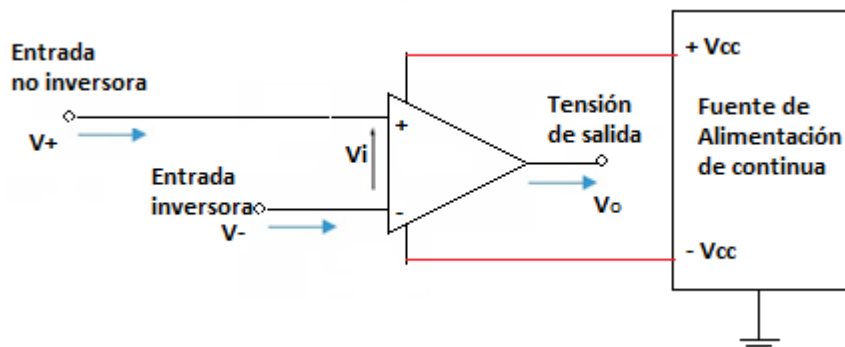
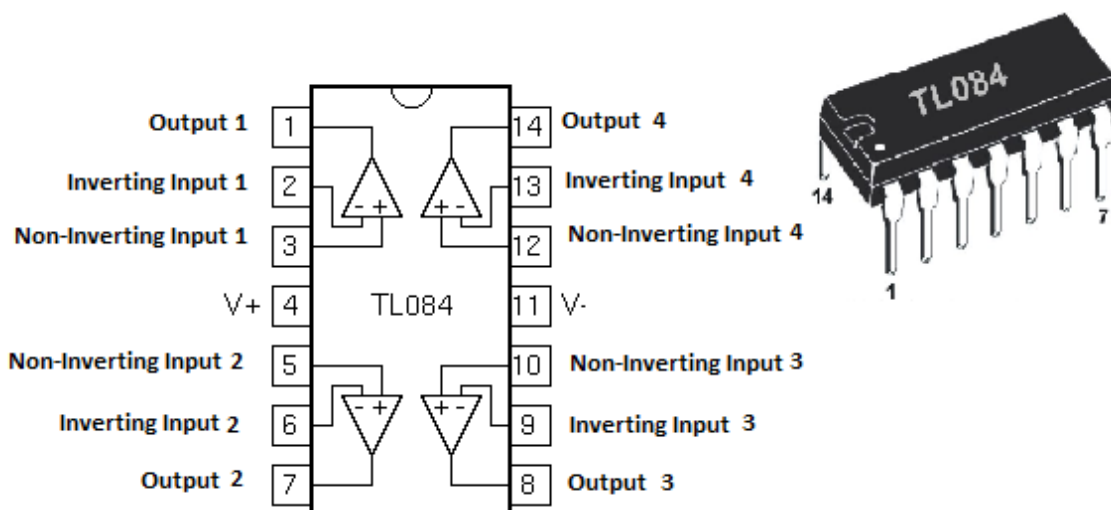


Figura 0.1 Símbolo eléctrico y terminales de un amplificador operacional [21]

En la (figura 2.13) se observa la distribución de pines del Amplificador Operacional utilizado para obtener las señales biofísicas.



[21]

La serie TL084 son amplificadores operacionales de propósito general que ofrecen bajo costo, alta velocidad, doble entrada de amplificadores

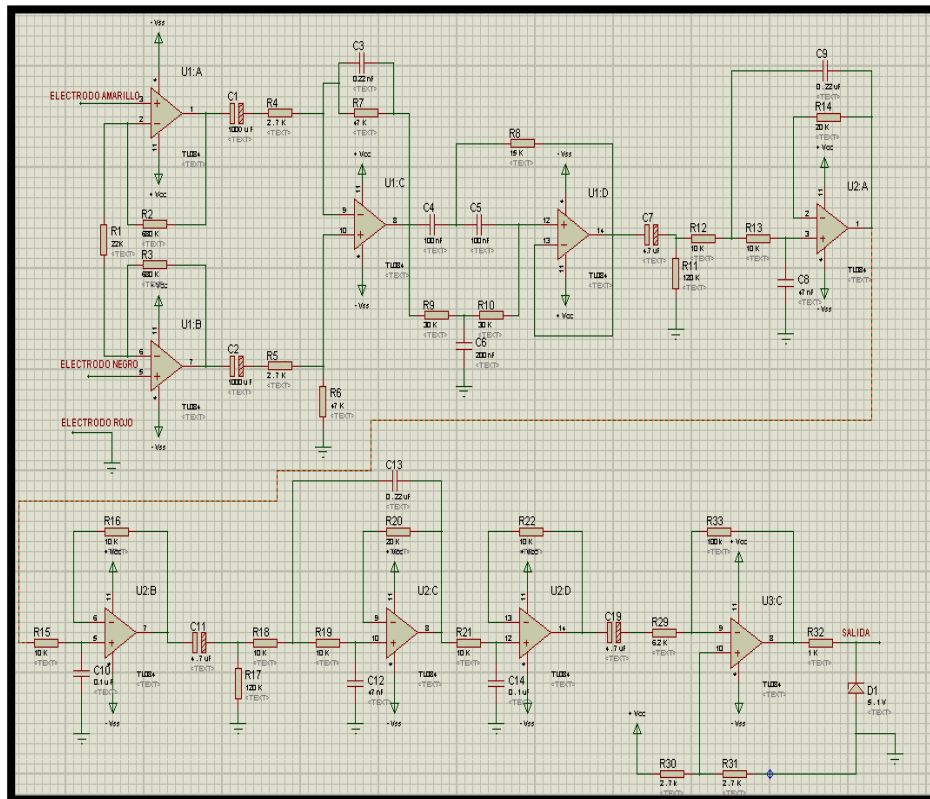
operacionales con una entrada de tensión de offset recortado internamente JFET (BI-FET II technology).

Características

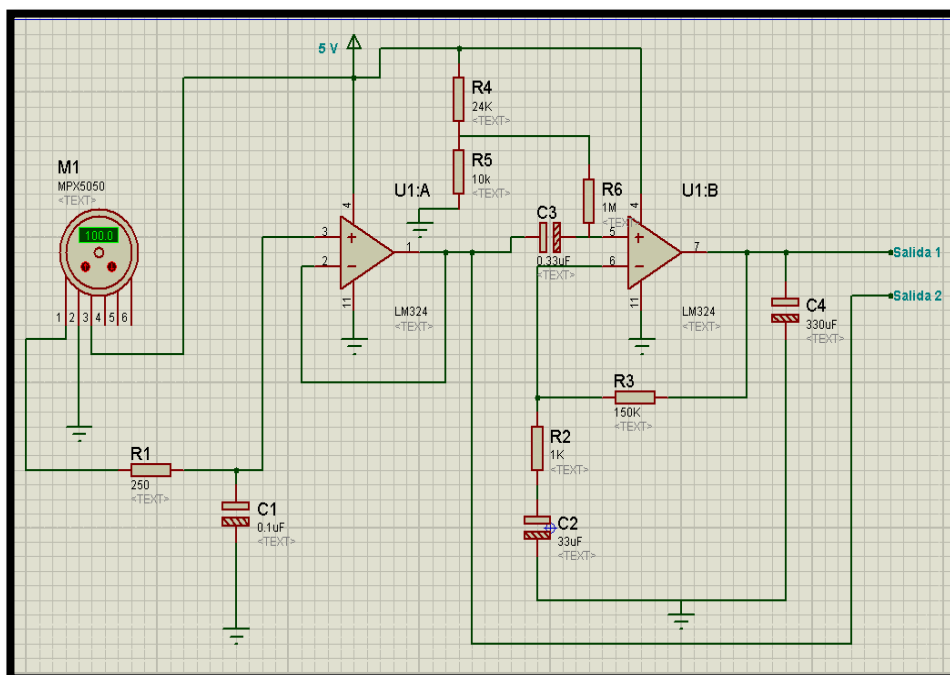
- No. de amplificadores operacionales: 4
- Voltaje de alimentación máx. ± 18 V
- Bajo consumo de potencia
- Ancho de banda típico: 3 MHz
- Alto slew rate: 13 V/ μ s típico
- Entradas a JFET con impedancias de entrada altas
- Corrientes de polarización y offset muy bajas
- Voltaje offset de entrada: 3 mV típico
- Baja distorsión armónica total típica: 0.003%
- Compensado en frecuencia internamente
- Salidas protegidas contra latch-up
- Salida protegida contra cortocircuito
- Pin compatible con el LM324
- Encapsulado: DIP 14 pines. [28]

Anexo E. CIRCUITOS ESQUEMÁTICOS

Electrocardiograma

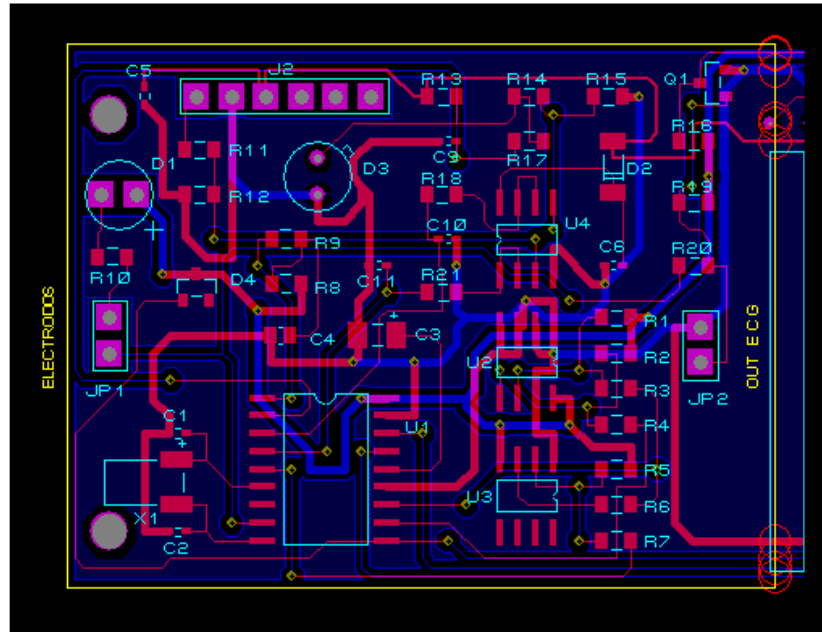


Presión Sanguínea

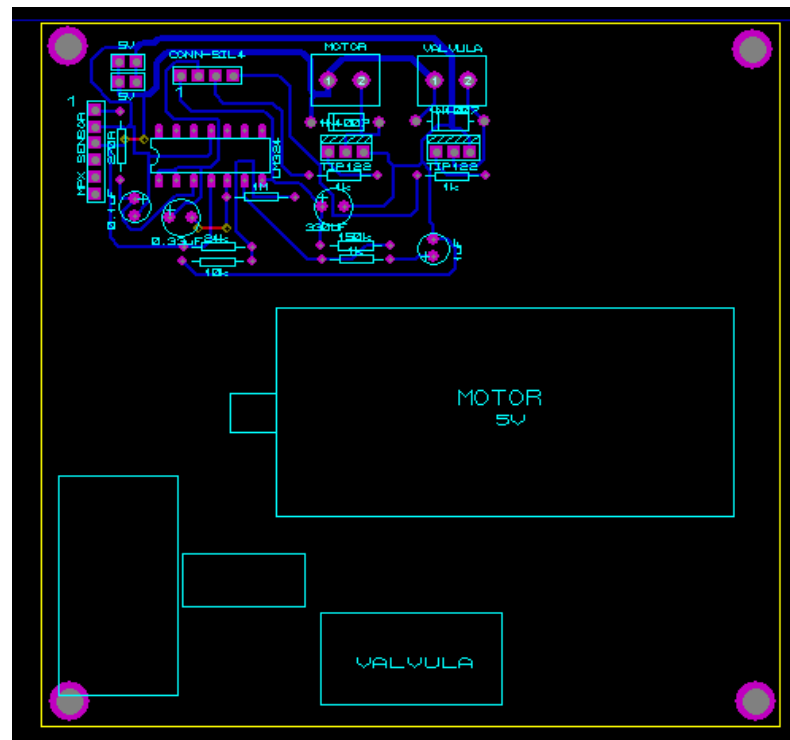


Anexo F. DIAGRAMAS IMPRESOS

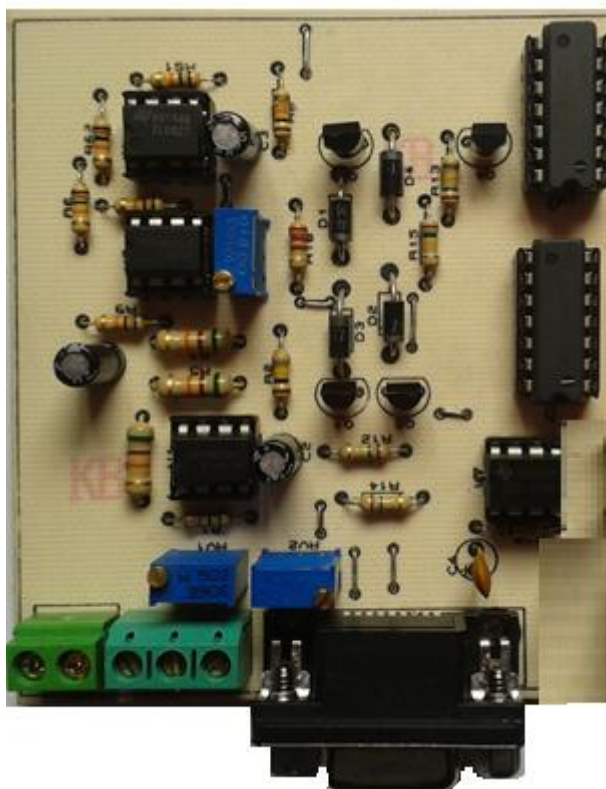
PCB ECG



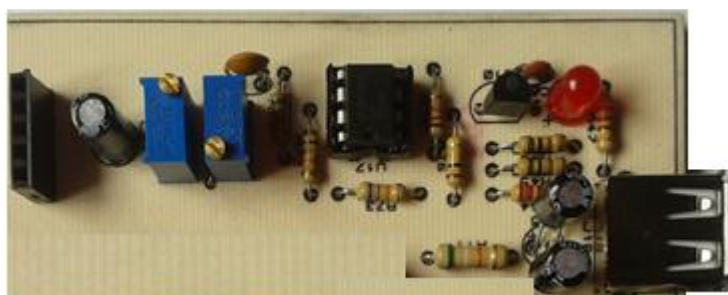
PCB Presión Sanguínea

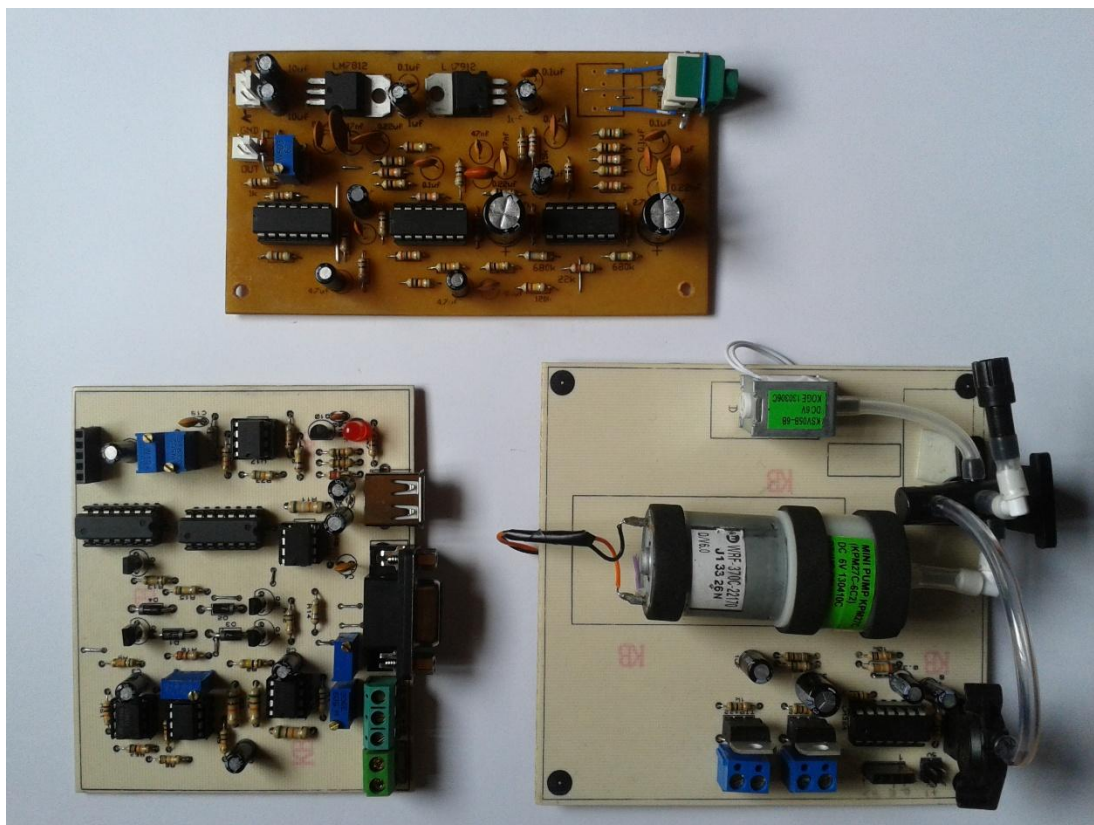
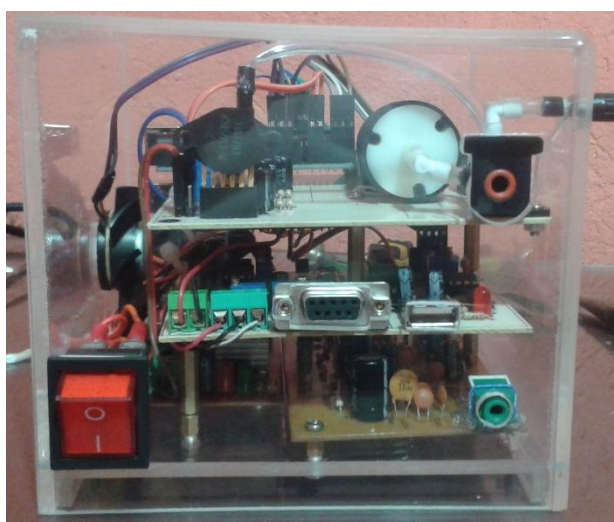


SPO2



ESPIRÓMETRO



SPO2, Espirómetro, ECG, Presión Sanguínea.**EQUIPO TERMINADO**

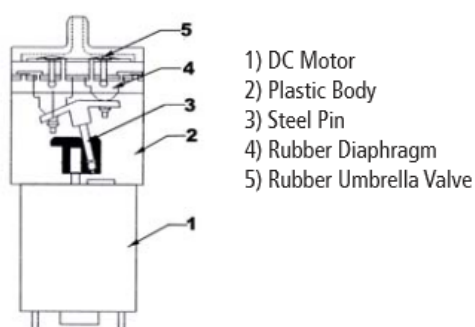
Anexo H. CARACTERÍSTICAS DE LA BOMBA CLARK

Bombas de gas miniatura KPM Round Series

DC potencia, presión de 300 mmHg (5.8 PSI)

DESCRIPCIÓN

La serie KPMR incorpora dos bombas mini-diafragmas operado por un brazo oscilante unidos a un excéntrico sobre un eje del motor. Son muy pequeñas, con longitud total de menos de 65 mm. Además de pequeño tamaño y peso ligero que ofrecen un rendimiento excelente teniendo en cuenta su coste muy bajo.



- 1) DC Motor
- 2) Plastic Body
- 3) Steel Pin
- 4) Rubber Diaphragm
- 5) Rubber Umbrella Valve

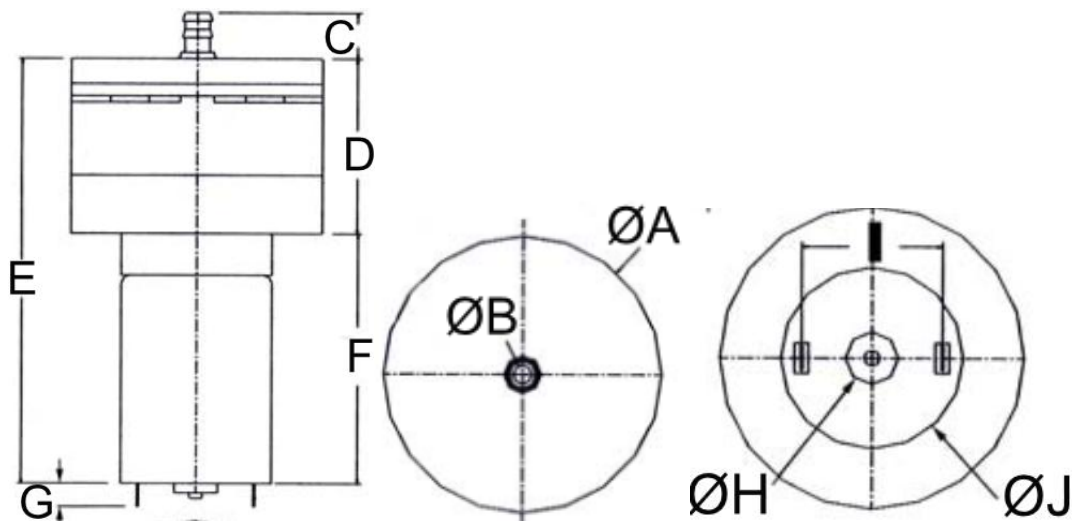


SPECIFICATIONS		
GENERAL	KPM-27C-6B3	KPM-32A-12A
Rated Voltage	6VDC	12VDC
Operating Voltage	4.0-7.0 V	4.0-13 V
Rated Current	<430 mA	<390 mA
Typ. Max Pressure	300 mmHg	300 mmHg
Typ. Max Flow (No Back Pressure)	>1.8 LPM	>3.5 LPM
Typ. Max Flow (@ 150 mmHg)	>1.2 LPM	>2.0 LPM
Typ. Startup Voltage (200 mmHg)	4V	6V
Operating Temp. Range	5-45°C	5-45°C
Operating Humidity Range	30 TO 80% RH	30 TO 80% RH
Duty Cycle	Intermittent	Intermittent
Typ Noise	<63 dB (30 cm away)	<63 dB (30 cm away)
Typ. Life	250 HRS	250 HRS
Tube barb O.D.	4 MM O.D.	4 MM O.D.

DIMENSIONS (MM)

MODEL	A	B	C	D	E	F	G	H	I	J
KPM-27C-6B3	27.0	4.2	6.3	27.0	58.0	31.0	3.5	6.5	18.3	24.2
KPM-32A-12A	32.0	4.2	9.0	31.2	50.2	19.0	3.3	-	20.3	32.0

Note: Above dimensions are for general sizing and reference purposes only. Please request specific model drawing for precise dimensions with tolerances



Anexo I. CARACTERÍSTICAS DE LA VÁLVULA SELENOIDE

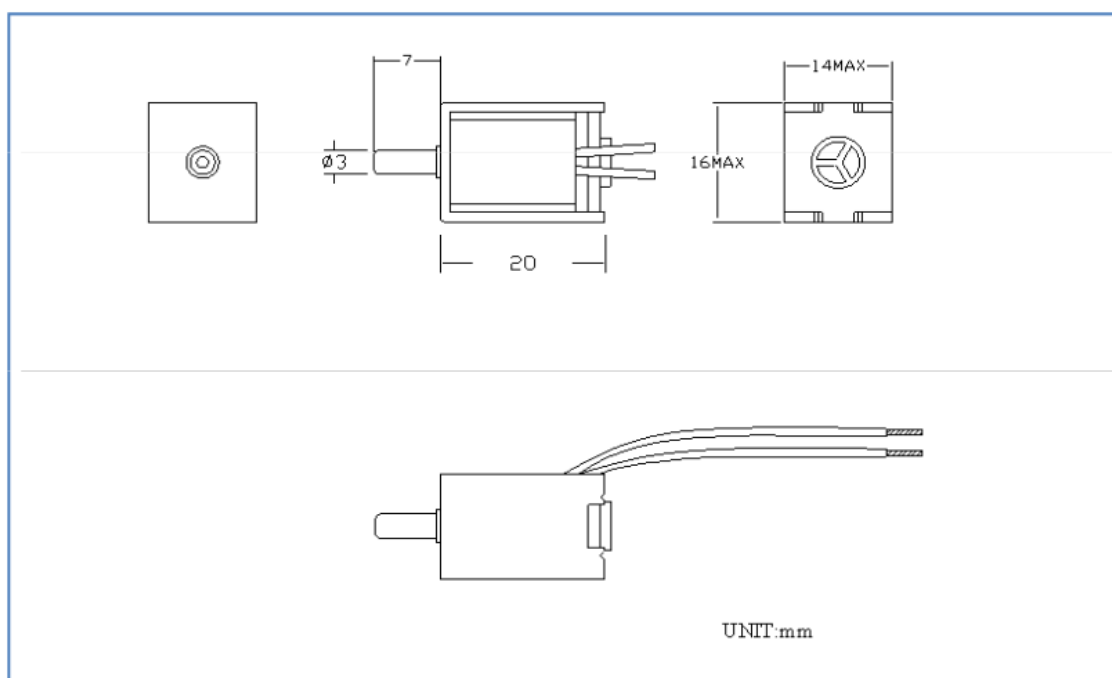
Solenoid Valve

Applications

Blood Pressure M/C, Massager, Medical Equipment, etc.

Specifications		
Rated Voltage	DC6.0V	DC12.0V
Rated Current	60mA	45mA
Exhaust Time	Max. 3.0 seconds from 300mmHg reduce to 15 mmHg at 50CC tank.	
Resistance	100Ω±10%	270Ω±10%
Leakage	Max. 3mmHg/min from 300mmHg at 100CC tank.	
Insulation level	A	
Apply For	Air	

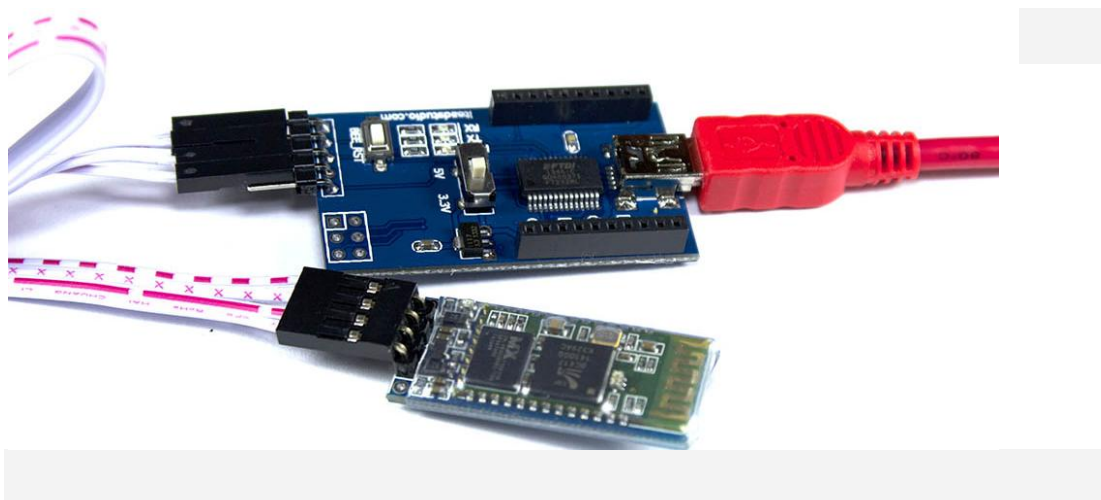
Drawing



Anexo J. INTERFAZ DE CONFIGURACIÓN DE COMANDOS AT EN HC-06

El HC-06 tiene un firmware distinto y también un funcionamiento distinto en cuanto a su modo de configuración. Para poder configurar el HC-06 es necesario que este **NO este emparejado ni siendo usado por ningún dispositivo**. De igual forma que el HC-05 es necesario conectarlo a la PC y usar un programa de terminal para darle instrucciones de configuración (Comandos AT), aunque también podemos escribir un programa de Arduino o en un microcontrolador para configurarlo.

Para conectarlo con la PC utilizamos un adaptador USB serial como se muestra en la foto:



Realizando pruebas con un módulo HC-06 y una tarjeta USB serial con FT-232RL

El módulo HC-06 acepta un set muy básico de comandos (algo raros por cierto), que permite pocas configuraciones, pero que sin duda será útil para personalizar este económico módulo y configurarlo para satisfacer las necesidades de la aplicación.

Los comandos que soporta son:

- Prueba de funcionamiento:
- **Envíar:** AT
- **Recibe:** OK
- Configurar el Baudrate:
- **Envíar:** AT+BAUD<Numero>

- El parámetro número es un caracter hexadecimal de '1' a 'c' que corresponden a los siguientes Baud Rates: 1=1200, 2=2400, 3=4800, 4=9600, 5=19200, 6=38400, 7=57600, 8=115200, 9=230400, A=460800, B=921600, C=1382400
- **Recibe:** OK<baudrate>
- Configurar el Nombre de dispositivo Bluetooth:
- **Envíar:** AT+NAME<Nombre>
- **Recibe:** OKsetname
- Configurar el código PIN de emparejamiento:
- **Envíar:** AT+PIN<pin de 4 digitos>
- **Recibe:** OK<pin de 4 digitos>
- Obtener la version del firmware:
- **Enviar:** AT+VERSION
- **Recibe:** Linvor1.8

Si ya hemos trabajado con comandos AT observaremos que los comandos estan lejos del estándar, lo más obvio es que:

- No es necesario finalizar el comando con `\r\n`, pero si es necesario ingresar los comandos con todos los caracteres seguidos sin pausas. NO hay necesidad de dar "enter" para finalizar un comando. El modulo tiene un Temporizador que hace necesario introducir el comando de una sola vez, sin pausas entre los caracteres.
- Por lo anterior, si utilizamos un emulador de terminal hay que pegarlos en leste y no escribirlos uno a uno con el teclado. También podemos usar el "monitor serial" de Arduino configurado como se muestra en la imagen más arriba en este artículo.
- Hay que tener cuidado de introducir **TODAS LAS LETRAS DEL COMANDO en MAYUSCULAS**, ya que de lo contrario, no funcionarán.
- Las respuestas no parecen respuestas estándar a comandos AT.

**Anexo K. MONITOR DE PACIENTE MULTIPARÁMETROS
CMS6000**

Monitor de paciente multiparámetros con pantalla TFT. A color de 12.1 pulgadas sensible al tacto.

Con alarmas visual y audible programables, memoria de tendencias, configuración de 4, 6 u 8 formas de ondas.

Con configuración standart de: Ecg, Resp, Spo2, Nibp, 2 X Temp, St, Arr.
Configuración opcional: ETCO2, IBP, impresora (no incluidos).
Fabricado en USA, equipo nuevo en su empaque original de fábrica.
12 meses de garantía de fábrica contra defectos de fabricación.

Certificaciones: FDA, CE.

INCLUYE LOS SIGUIENTES ACCESORIOS STANDART:

1 CABLE ECG DE 5 DERIVACIONES

1 SENSOR DE SPO2 PARA DULTO

1 BRAZAL PARA ADULTO

1 MANGUERA PARA NIBP

1 SENSOR DE TEMPRATURA PIEL

1 CABLE DE ALIMENTACION ELECTRICA

1 BATERIA RECARGABLE

1 MANUAL DE OPERACION-COPIA ELECTRONICA



[27]

Anexo L. DIAGRAMAS DE FLUJO

DIAGRAMA DE FLUJO DE LA TARJETA ARDUINO UNO

En la figura 1 se observa el diagrama de flujo para la programación de la tarjeta Arduino UNO.

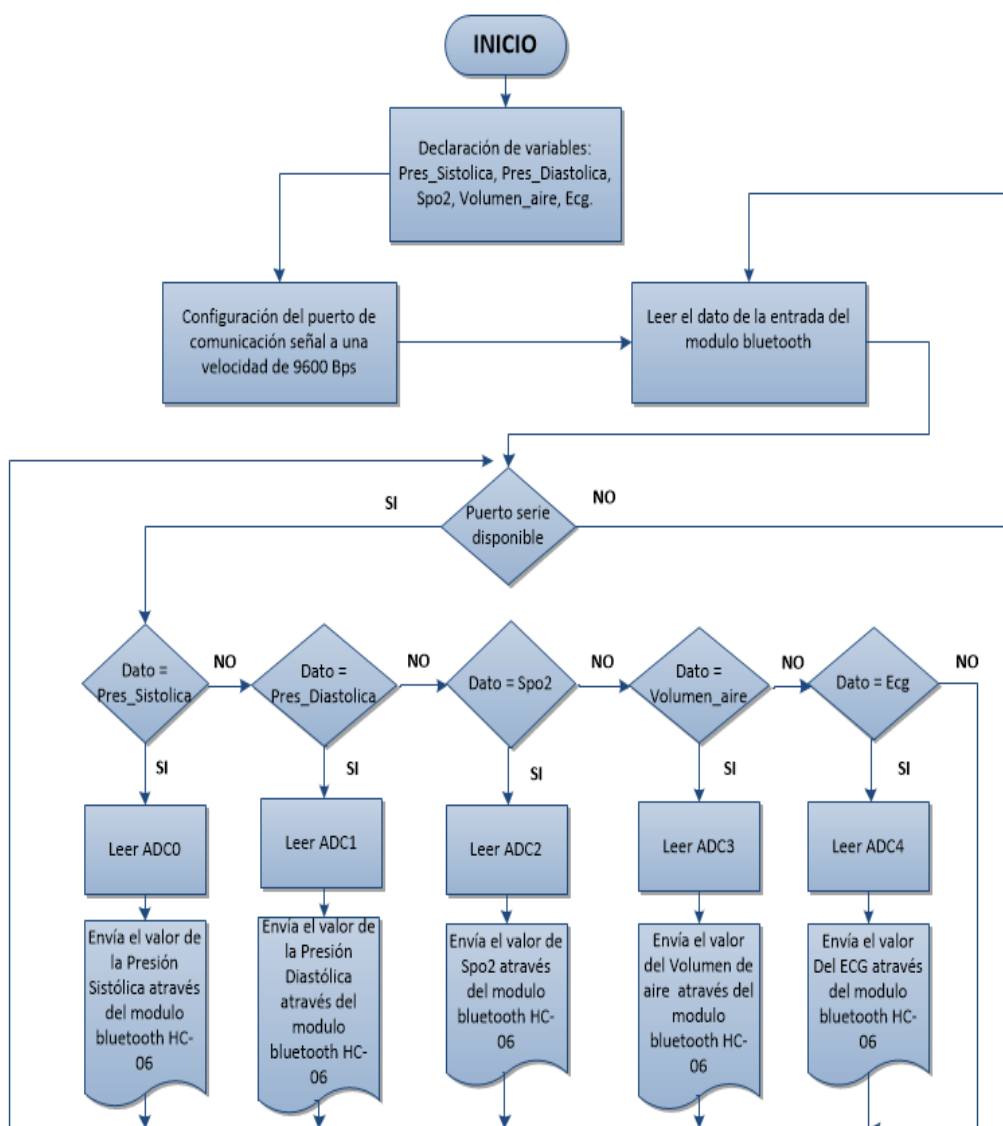


FIGURA 1: Diagrama de Flujo de la programación del Arduino UNO.

Elaborado por: Gómez O, Silva I.

Se va a explicar el funcionamiento de la programación para la tarjeta Arduino UNO, donde se asigna variables de todas las señales que se va a adquirir en la tarjeta se configura el puerto de comunicación a una velocidad de transmisión de 9600 Bps se lee el dato en la entrada del módulo Bluetooth para luego verificar si el puerto serie está disponible, si no está disponible se vuelve a leer el dato en la entrada del módulo Bluetooth cuando ya se encuentra disponible el puerto serie la tarjeta Arduino procesa todas las señales con sus respectivos cálculos y los envía por del módulo Bluetooth al dispositivo móvil donde las variables serán visualizadas . Programación en el (PROGRAMACIÓN ARDUINO UNO Anexo N).

DIAGRAMA DE FLUJO DE LA APLICACIÓN EN ANDROID

En la figura 2 se observa el diagrama de flujo de la adquisición y visualización de los datos, para la programación del dispositivo móvil bajo la plataforma Android.

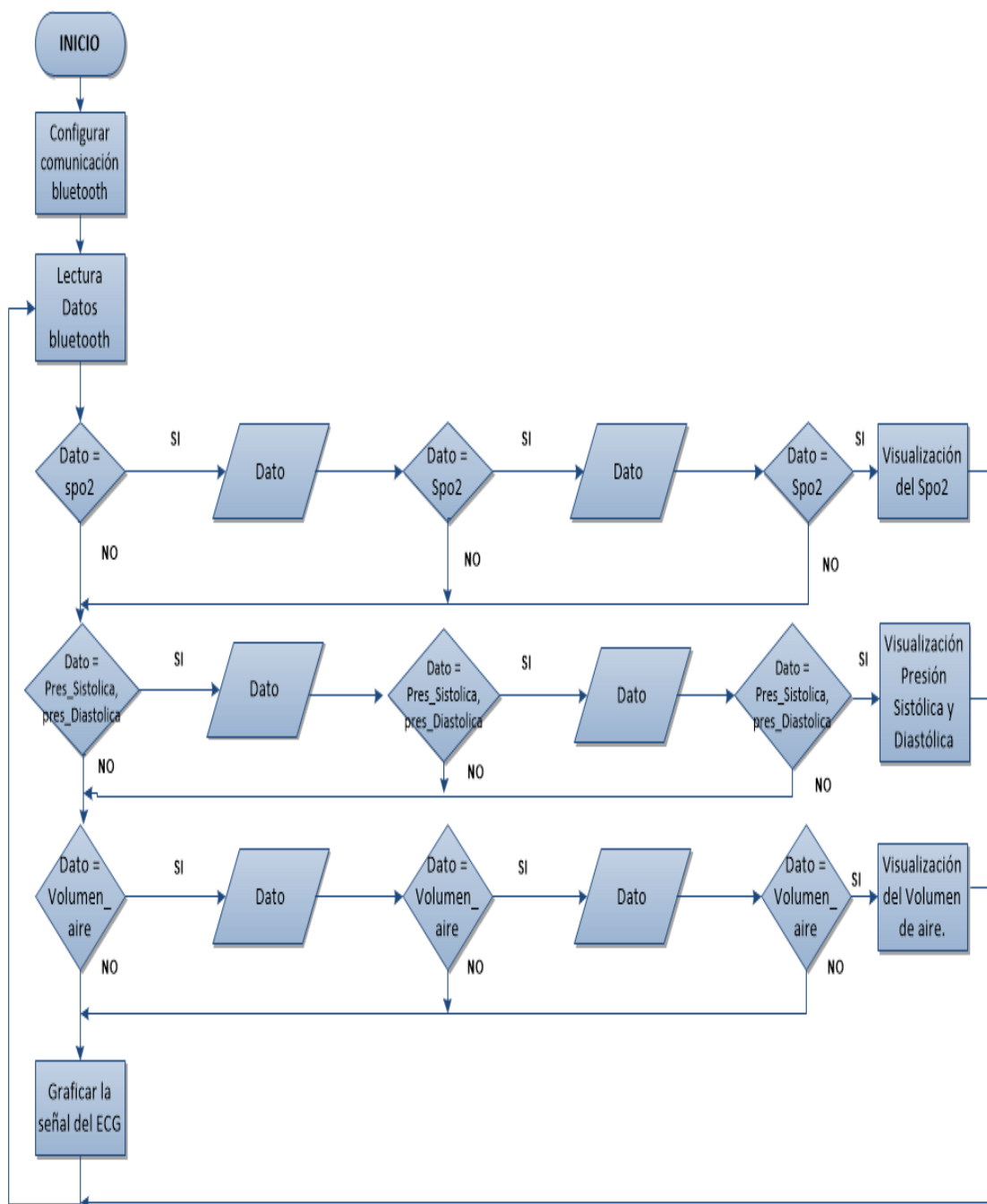


FIGURA 2: Diagrama de Flujo del dispositivo móvil bajo la plataforma Android.

Elaborado por: Gómez O, Silva I.

El diagrama de flujo de la programación para el dispositivo móvil para la obtención y visualización de los datos, en primer lugar toca configurar la

comunicación Bluetooth y luego leer los datos de cada variable ya asignada en la tarjeta Arduino con su valor correspondiente para luego ser visualizada en el dispositivo móvil. Programación en el (Anexo M).

En la figura 3 se observa el diagrama de flujo general en cómo está constituido la aplicación del dispositivo móvil bajo la plataforma Android.

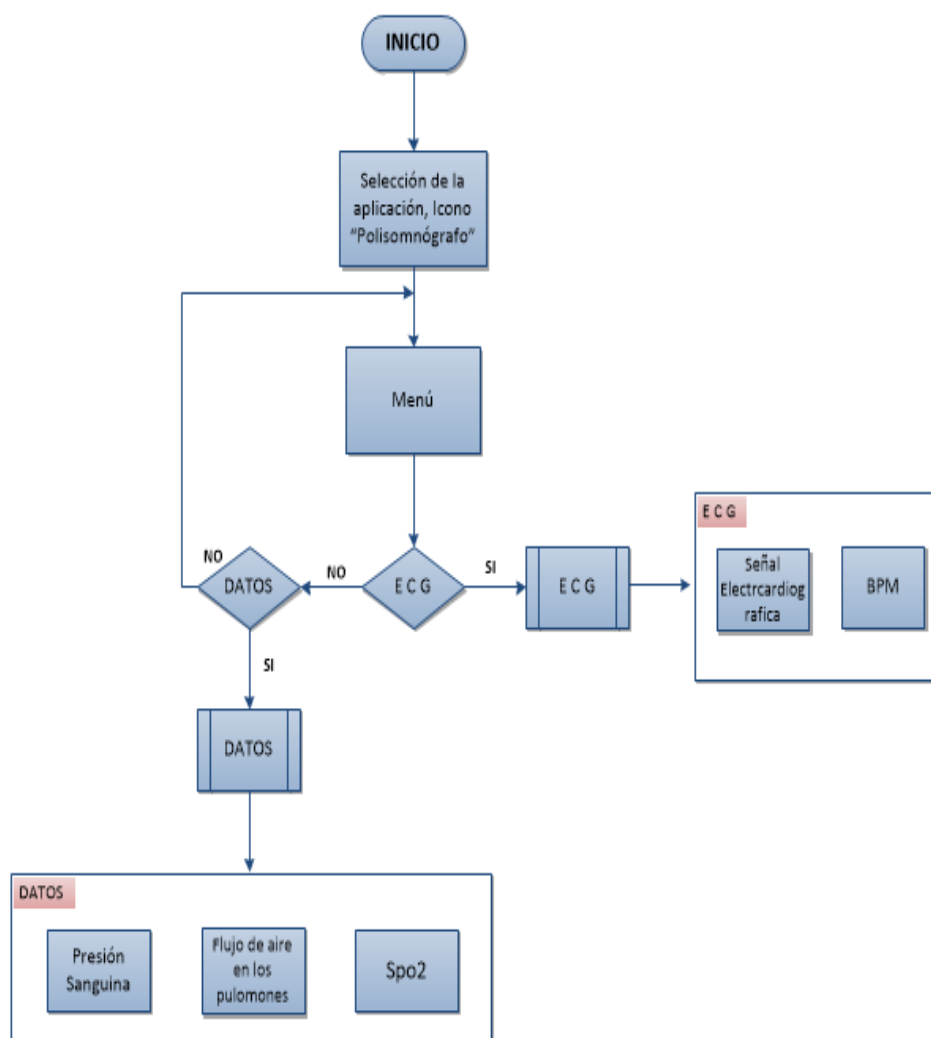


Figura 3: Diagrama de Flujo de la Activity Principal.

Elaborado por: Gómez O, Silva I.

El diagrama de flujo general en cómo está constituido la aplicación en el dispositivo móvil se explica el funcionamiento, se pulsa el icono llamado Polisomnógrafo y se despliega un menú en donde se encuentra dos botones llamados “Datos” y “Ecg”, al pulsar el botón Datos se despliega una ventana en donde se visualiza los valores del Spo2, la presión sanguínea y el Flujo de aire de los pulmones, al pulsar el botón Ecg se despliega una ventana en donde se visualiza la señal electrocardiográfica y los pulsos por minutos (Bpm).

**Anexo M. PROGRAMACIÓN EN ECLIPSE
CLASE ECG**

```
package com.example.polisomnografo;
```

```
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Set;  
import java.util.UUID;  
  
import com.androidplot.ui.SizeLayoutType;  
import com.androidplot.ui.SizeMetrics;  
import com.androidplot.xy.BoundaryMode;  
import com.androidplot.xy.LineAndPointFormatter;  
import com.androidplot.xy.SimpleXYSeries;  
import com.androidplot.xy.XYPlot;  
import com.androidplot.xy.XYSeries;  
import com.androidplot.xy.XYStepMode;  
  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.Message;  
  
import android.os.PowerManager;  
import android.annotation.SuppressLint;  
import android.app.Activity;  
import android.bluetooth.BluetoothAdapter;  
import android.bluetooth.BluetoothDevice;
```



```
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;

import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

public class Principal extends Activity {

    BluetoothSocket mmSocket;
    BluetoothDevice mmDevice;
    OutputStream mmOutputStream;
    InputStream mmInputStream;
    Thread workerThread;
    byte[] readBuffer;
    int readBufferPosition;
    int counter;
    // Debugging
    private static final String TAG = "ecg";
    private static final boolean D = true;
    // Intent request codes
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int
REQUEST_ENABLE_BT_AND_CONNECT_DEVICE = 2;
    private static final int REQUEST_ENABLE_BT = 3;
    private static final int REQUEST_FILE = 5;
    private String mConnectedDeviceName = null;
    private SimpleXYSeries mDataSeries = new SimpleXYSeries("ECG");
```

```

private BTcom btCom;
private BluetoothAdapter mBluetoothAdapter;

private DataBuffer datBuf;
    public SimpleXYSeries series;
    volatile boolean stopWorker;

protected PowerManager.WakeLock mWakeLock;

private double mCount = 0;
private int mplotmemory=600;
private int max_samples = 500; // Default size
    private XYPlot plot;

////////////////////////////////////
// variables del sistema
TextView bpm;
ArrayList <Double> Vector = new ArrayList <Double>();
public Double datox=0.0, datoy, ecg;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.principal);

////////////////////////////////////
/// layouts

    bpm=(TextView)findViewById(R.id.BPM);

////////////////////////////////////

```

```

////////////////////////////////////
//configuracion de pantalla para graficar
//
//
plot=(XYPlot)findViewById(R.id.grafica);
//
plot.setDomainStep(XYStepMode.INCREMENT_BY_VAL,
250);
//
plot.setRangeStep(XYStepMode.INCREMENT_BY_VAL, 50);
//

plot.getGraphWidget().getGridBackgroundPaint().setColor(Color.r
gb(102,204,255));
//

plot.getGraphWidget().getDomainGridLinePaint().setColor(Color.
WHITE);
//

plot.getGraphWidget().getRangeGridLinePaint().setColor(Color.W
HITE);

plot.getLegendWidget().setSize(new SizeMetrics(40,
SizeLayoutType.ABSOLUTE, 600, SizeLayoutType.ABSOLUTE));

plot.setRangeBoundaries(0,100, BoundaryMode.FIXED);
//
plot.setDomainBoundaries(0, 500, BoundaryMode.FIXED);
//
plot.getGraphWidget().setSize(new SizeMetrics(
//
50, SizeLayoutType.FILL,
//
50, SizeLayoutType.FILL));
//

```

```
////////////////////////////////////
```

```

        if (btCom == null) setupCom();
        try {
            findBT();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        PowerManager pm = (PowerManager)
getSystemService(Context.POWER_SERVICE);
        this.mWakeLock =
pm.newWakeLock(PowerManager.SCREEN_BRIGHT_WAKE_LOCK,
"My Tag");
        this.mWakeLock.acquire();
        datBuf = new DataBuffer(this, mHandler);

    }

```

```

        @Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.principal, menu);
    return true;
}

```

```

        @Override
public void onStart() {

```

```

    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");
}

```

@Override

```

public synchronized void onResume() {
    super.onResume();
    /*try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }*/
    if(D) Log.e(TAG, "+ ON RESUME +");

```

// Performing this check in onResume() covers the case in which BT was

// not enabled during onStart(), so we were paused to enable it...

// onResume() will be called when ACTION_REQUEST_ENABLE activity returns.

```

    if (btCom != null) {

```

// Only if the state is STATE_NONE, do we know that we haven't started already

```

        if (btCom.getState() == BTcom.STATE_NONE) {

```

// Start the Bluetooth com services

```

            btCom.start();

```

```

        }

```

```

    }

```

```

}

```

@Override

```

public synchronized void onPause() {
    super.onPause();

```

```
    if(D) Log.e(TAG, "- ON PAUSE -");  
}
```

```
@Override  
public void onStop() {  
    super.onStop();  
    if(D) Log.e(TAG, "-- ON STOP --");  
}
```

```
@Override  
public void onDestroy() {  
    super.onDestroy();  
    // Stop the Bluetooth chat services  
    if (btCom != null) btCom.stop();  
    if(D) Log.e(TAG, "--- ON DESTROY ---");  
}
```

```
    public void setupCom(){  
        btCom = new BTcom(this, mHandler);  
        return;  
    }
```

```
@SuppressWarnings("NewApi")  
private final Handler mHandler = new Handler() {  
    @Override  
    public void handleMessage(Message msg) {  
        switch (msg.what) {  
  
            case BTcom.MESSAGE_STATE_CHANGE:  
                if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);  
                switch (msg.arg1) {  
                    case BTcom.STATE_CONNECTED:
```

```
// setStatus(R.string.connected);

    break;
    case BTcom.STATE_CONNECTING:
        Toast.makeText(getApplicationContext(), "Conectando... ",
Toast.LENGTH_SHORT).show();
        break;
    case BTcom.STATE_LISTEN:
    case BTcom.STATE_NONE:
        // setStatus(R.string.not_connected);
        break;
    }
    break;

    case BTcom.MESSAGE_WRITE:
        break;

    case BTcom.ECG:
        //System.out.println((String)msg.obj);
        try{

            datBuf.add((String) msg.obj,true);

        }
        catch(Exception e){}
        break;

    case BTcom.SPO2:
```

```
try{

    Double puls=Double.parseDouble((String)msg.obj);
    //System.out.println(puls);
    bpm.setText("");
    bpm.setText(puls.toString());

}
catch(Exception e){}
break;

case BTcom.MESSAGE_DEVICE_NAME:
    // save the connected device's name
    mConnectedDeviceName =
msg.getData().getString(BTcom.DEVICE_NAME);
    Toast.makeText(getApplicationContext(), "Conectado a
Polisomnografo ", Toast.LENGTH_SHORT).show();
    break;

case BTcom.MESSAGE_TOAST:
    Toast.makeText(getApplicationContext(),
msg.getData().getString(BTcom.TOAST),
        Toast.LENGTH_SHORT).show();
    break;

case DataBuffer.MESSAGE_NEW_VALUE:

    datox = (Double)msg.obj;
    //System.out.println(datox);
    UpdatePlot(datox);
```



```

        // graficar(datox);
        break;

    }
}
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//graficar

private void UpdatePlot(Double vr1) {
    //mDataPlot.centerOnDomainOrigin((max_samples/2));

    if(mCount>=max_samples)
    {
        mCount=0;

        Vector.clear();
        plot.clear();

    }
    if(true)
    {
        Vector.add((double) mCount);
        Vector.add(vr1);

    }

    if (Vector.size() > 2 * mplotmemory)
    {
        Vector.remove(0);
        Vector.remove(0);
        plot.clear();
    }
}

```



```

else {
    // User did not enable Bluetooth or an error
    occurred
    Log.d(TAG, "BT not enabled");
    Toast.makeText(this, R.string.bt_not_enabled_leaving,
    Toast.LENGTH_SHORT).show();
    finish();
    break;
}
case REQUEST_CONNECT_DEVICE:
    if (resultCode == RESULT_OK){
        if (btCom == null) setupCom();
        connectDevice(data, true);
    }
    break;
}
return;
}

void findBT() throws IOException {
    //Toast.makeText(getApplicationContext(), "Conectando..."
, Toast.LENGTH_LONG).show();
    mBluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null) {

    }

    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBluetooth = new Intent(

BluetoothAdapter.ACTION_REQUEST_ENABLE);

```

```

        startActivityForResult(enableBluetooth, 0);
    }

    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter
        .getBondedDevices();
    if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
            System.out.println(device.getName());
            if
(device.getName().equals("ECG_POLISOMNOGRAFO")) {
                mmDevice = device;

                break;
            }
        }
    }
    // Attempt to connect to the device
    btCom.connect(mmDevice, true);

}

private void connectDevice(Intent data, boolean secure) {
    // MAC address
    String address = data.getExtras()

        .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
    // BluetoothDevice object
    BluetoothDevice device =
mBluetoothAdapter.getRemoteDevice(address);
    // Attempt to connect to the device
    btCom.connect(device, secure);
    return;
}

```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent serverIntent = null;
    // Handle item selection
    switch (item.getItemId()) {

        case R.id.scan_connect:
            if (btCom != null){
                btCom.stop();
            }
            try {
                findBT();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return true;

        case R.id.disconnect:
            if (btCom != null){
                btCom.stop();
            }

            return true;

        default:
            return super.onOptionsItemSelected(item);
    }
}
```

CLASE BTCOM

```
/*  
/*  
* Copyright (C) 2009 The Android Open Source Project  
*  
* Licensed under the Apache License, Version 2.0 (the "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either  
express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*  
* Modified by Joe Smallman April 2012 to make appropriate for serial  
* communication with Arduino device  
* Copyright (C) 2009 The Android Open Source Project  
*  
* Licensed under the Apache License, Version 2.0 (the "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either  
express or implied.
```

* See the License for the specific language governing permissions and
* limitations under the License.

*

* Modified by Joe Smallman April 2012 to make appropriate for serial
* communication with Arduino device

*

*/

```
package com.example.polisomnografo;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.util.UUID;

import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

@SuppressLint("NewApi")
public class BTcom {

    // Debugging
    private static final String TAG = "BTcom";
    private static final boolean D = true;
```

```
// Name for the SDP record when creating server socket
//private static final String NAME = "BTcom";

// Unique UUID for this application
private static final UUID MY_UUID = UUID.fromString("00001101-0000-
1000-8000-00805f9b34fb"); //Standard SerialPortService ID

private final BluetoothAdapter mAdapter;
    private Context mContext;
    private Handler mHandler;
    private ConnectThread mConnectThread;
private ConnectedThread mConnectedThread;

    // Message types sent to the main activity
public static final int MESSAGE_STATE_CHANGE = 1;
public static final int MESSAGE_READ = 2;
public static final int ECG=8;
public static final int DIAST=6;
public static final int SIST=10;
public static final int FLUJO=12;
public static final int SPO2=14;
public static final int MESSAGE_WRITE = 3;
public static final int MESSAGE_DEVICE_NAME = 4;
public static final int MESSAGE_TOAST = 5;

// Key names sent to the main activity
public static final String DEVICE_NAME = "device_name";
public static final String TOAST = "toast";

    // Constants that indicate the current connection state
public static final int STATE_NONE = 0;    // we're doing nothing
public static final int STATE_LISTEN = 1;  // now listening for
incoming connections
```



```

public static final int STATE_CONNECTING = 2; // now initiating an
outgoing connection
public static final int STATE_CONNECTED = 3; // now connected to a
remote device
public static final int STATE_RECONNECT = 4; // reconnecting to
remote device
private int mState;

public BTcom(Context context, Handler handler){
    mAdapter = BluetoothAdapter.getDefaultAdapter();
    mContext = context;
    mHandler = handler;
    mState = STATE_NONE;
}

private synchronized void setState(int state) {
if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
mState = state;

// Give the new state to the Handler so the UI Activity can update
mHandler.obtainMessage(MESSAGE_STATE_CHANGE, state, -
1).sendToTarget();
}

public synchronized int getState() {
return mState;
}

/**
 * Start the chat service. Specifically start AcceptThread to begin a
 * session in listening (server) mode. Called by the Activity
onResume() */
public synchronized void start() {
    if (D) Log.d(TAG, "start");

```

```
// Cancel any thread attempting to make a connection
    if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

    setState(STATE_LISTEN);

}
/**
 * Stop all threads
 */
public synchronized void stop() {
    if (D) Log.d(TAG, "stop");

    if (mConnectThread != null) {
        mConnectThread.cancel();
        mConnectThread = null;
    }

    if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
    }

    setState(STATE_NONE);
}
/**
 * Start the ConnectThread to initiate a connection to a remote device.
 * @param device The BluetoothDevice to connect
```

```

* @param secure Socket Security type - Secure (true) , Insecure
(false)
*/
public synchronized void connect(BluetoothDevice device, boolean
secure) {
    if (D) Log.d(TAG, "connect to: " + device);

    // Cancel any thread attempting to make a connection
    if (mState == STATE_CONNECTING) {
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}
    }

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

    // Start the thread to connect with the given device
    mConnectThread = new ConnectThread(device, secure);
    mConnectThread.start();
    setState(STATE_CONNECTING);
}
/**
* Start the ConnectedThread to begin managing a Bluetooth
connection
* @param socket The BluetoothSocket on which the connection was
made
* @param device The BluetoothDevice that has been connected
*/
public synchronized void connected(BluetoothSocket socket,
BluetoothDevice
device, final String socketType) {
    if (D) Log.d(TAG, "connected, Socket Type:" + socketType);

```

```

    // Cancel the thread that completed the connection
    if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

    // Start the thread to manage the connection and perform
transmissions
    mConnectedThread = new ConnectedThread(socket, socketType);
    mConnectedThread.start();

    // Send the name of the connected device back to the UI Activity
    Message msg =
mHandler.obtainMessage(MESSAGE_DEVICE_NAME);
    Bundle bundle = new Bundle();
    bundle.putString(DEVICE_NAME, device.getName());
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    setState(STATE_CONNECTED);
}
/**
 * Indicate that the connection attempt failed and notify the UI
Activity.
 */
private void connectionFailed() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(TOAST, "No se pudo conectar !!");

```

```
msg.setData(bundle);
mHandler.sendMessage(msg);

// Start the service over to restart listening mode
BTcom.this.start();
}

/**
 * Indicate that the connection was lost and notify the UI Activity.
 */
private void connectionLost() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(TOAST, "Conexion perdida");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    // Start the service over to restart listening mode
    BTcom.this.start();
}

/**
 * Write to the ConnectedThread in an unsynchronized manner
 * @param out The string to write
 * @see ConnectedThread#write(byte[])
 */
public void write(String out) {
    // Create temporary object
    ConnectedThread r;
    // Synchronize a copy of the ConnectedThread
    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mConnectedThread;
```

```

    }
    // Perform the write unsynchronized
    r.write(out);
}

/**
 * This thread runs while attempting to make an outgoing connection
 * with a device. It runs straight through; the connection either
 * succeeds or fails.
 */
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    private String mSocketType;

    public ConnectThread(BluetoothDevice device, boolean secure) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            tmp = device.createRfcommSocketToServiceRecord(
                MY_UUID);
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "create() failed",
e);
        }
        mmSocket = tmp;
    }

    public void run() {

```

```
    Log.i(TAG, "BEGIN mConnectThread SocketType:" +
mSocketType);
    setName("ConnectThread" + mSocketType);

    // Always cancel discovery because it will slow down a
connection
    mAdapter.cancelDiscovery();
    // Make a connection to the BluetoothSocket
    try {
        // This is a blocking call and will only return on a
        // successful connection or an exception
        mmSocket.connect();
    } catch (IOException e) {
        // Close the socket
        try {
            mmSocket.close();
        } catch (IOException e2) {
            Log.e(TAG, "unable to close() " + mSocketType +
                " socket during connection failure", e2);
        }
        connectionFailed();
        return;
    }

    // Reset the ConnectThread because we're done
synchronized (BTcom.this) {
        mConnectThread = null;
    }

    // Start the connected thread
    connected(mmSocket, mmDevice, mSocketType);
}
```

```

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "close() of connect " + mSocketType + " socket
failed", e);
    }
}
}
}

```

```

/**

```

```

* This thread runs during a connection with a remote device.

```

```

* It handles all incoming and outgoing transmissions.

```

```

*/

```

```

private class ConnectedThread extends Thread {

```

```

    private final BluetoothSocket mmSocket;

```

```

    private final InputStream mmInStream;

```

```

    private final OutputStream mmOutStream;

```

```

    public ConnectedThread(BluetoothSocket socket, String
socketType) {

```

```

        Log.d(TAG, "create ConnectedThread: " + socketType);

```

```

        mmSocket = socket;

```

```

        InputStream tmpIn = null;

```

```

        OutputStream tmpOut = null;

```

```

        // Get the BluetoothSocket input and output streams

```

```

        try {

```

```

            tmpIn = socket.getInputStream();

```

```

            tmpOut = socket.getOutputStream();

```

```

        } catch (IOException e) {

```



```
        Log.e(TAG, "temp sockets not created", e);
    }

    mmInStream = tmpIn;
    mmOutStream = tmpOut;
}

public void run() {
    Log.i(TAG, "BEGIN mConnectedThread");

    char delimiter = '\n';
    byte[] readBuffer = new byte[1024];
    byte[] p_dias=new byte[1024];
    byte[] p_sis=new byte[1024];
    byte[] vecg=new byte[1024];
    byte[] spo_2=new byte[1024];
    byte[] a=new byte[4095];
    byte[] p=new byte[1024];

    int y=0;
    int readBufferPosition = 0;

    // Keep listening to the InputStream while connected
    while (true) {
        try {
            // Read from the InputStream
            int bytesAvailable = mmInStream.available();
            if(bytesAvailable > 0)
            {
                byte[] packetBytes = new byte[bytesAvailable];
                mmInStream.read(packetBytes);

                for (int i = 0; i < bytesAvailable; i++)
```

```

{
    byte b = packetBytes[i];
    if (b == delimiter)
    {

        // diastolica
        if(readBuffer[0]=='d')
        {
            for(int x=1; x < bytesAvailable;
x++)
            {
                y=x-1;
                if(readBuffer[x]=='&')
                {
                    String pd = new
String(p_dias, Charset.forName("ASCII")).trim();
                    mHandler.obtainMessage(DIAST,
pd).sendToTarget();

                    readBufferPosition = 0;
                }
                else
                {
                    p_dias[y]=readBuffer[x];
                }

            }
            readBufferPosition = 0;
        }

        //ecg
        if(readBuffer[0]=='e')

```

```

    {
        for(int x=1; x < bytesAvailable;
x++)
        {
            y=x-1;
            if(readBuffer[x]=='&')
            {
                String p1 = new
String(vecg, Charset.forName("ASCII")).trim();
                mHandler.obtainMessage(ECG,
p1).sendToTarget();

                readBufferPosition = 0;
            }
            else{
                vecg[y]=readBuffer[x];
            }

        }
        readBufferPosition = 0;
    }

//sistolica
if(readBuffer[0]=='s')
{
    for(int x=1; x < bytesAvailable;
x++)
    {
        y=x-1;
        if(readBuffer[x]=='&')
        {
            String ps = new
String(p_sis, Charset.forName("ASCII")).trim();

```

ps).sendToTarget());

```
mHandler.obtainMessage(SIST,
```

```
readBufferPosition = 0;
```

```
}
```

```
else
```

```
{
```

```
p_sis[y]=readBuffer[x];
```

```
}
```

```
}
```

```
readBufferPosition = 0;
```

```
}
```

```
//SPO2
```

```
if(readBuffer[0]=='o')
```

```
{
```

```
for(int x=1; x < bytesAvailable;
```

x++)

```
{
```

```
y=x-1;
```

```
if(readBuffer[x]=='&')
```

```
{
```

```
String _spo2 = new
```

```
String(spo_2, Charset.forName("ASCII")).trim();
```

```
mHandler.obtainMessage(SPO2,
```

```
_spo2).sendToTarget();
```

```
readBufferPosition = 0;
```

```
}
```

```
else
```

```
{
```

```
spo_2[y]=readBuffer[x];
```

```
}
```

```

    }

    readBufferPosition = 0;
}

//flujo
if(readBuffer[0]=='f')
{
    for(int x=1; x < bytesAvailable;
x++)
    {
        y=x-1;
        if(readBuffer[x]=='&')
        {
            String fl = new
String(a, Charset.forName("ASCII")).trim();
            mHandler.obtainMessage(FLUJO,
fl).sendToTarget();

            readBufferPosition = 0;
        }
        else
        {
            a[y]=readBuffer[x];
        }
    }
    readBufferPosition = 0;
}

if(readBuffer[0]!='e' || readBuffer[0]!='f' ||
readBuffer[0]!='d' || readBuffer[0]!='s' || readBuffer[0]!='o')
{

```

```
String s = new String(readBuffer,  
Charset.forName("ASCII").trim());  
  
mHandler.obtainMessage(MESSAGE_READ, s).sendToTarget();  
    readBufferPosition = 0;  
    y=0;  
    }  
  
    else  
    {  
  
        readBufferPosition = 0;  
        y=0;  
    }  
  
    }  
    else  
    {  
        readBuffer[readBufferPosition++] = b;  
    }  
    }  
    }  
  
} catch (IOException e) {  
    Log.e(TAG, "disconnected", e);  
    connectionLost();  
    // Start the service over to restart listening mode  
    BTcom.this.start();  
    break;  
}  
}
```



```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.Charset;
import java.text.DecimalFormat;
import java.util.Arrays;
import java.util.Set;
import java.util.UUID;

import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.PowerManager;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class Datos extends Activity {
    TextView spo2_, diast, sisto,t_flujo;
    BluetoothSocket mmSocket;
    BluetoothDevice mmDevice;
    OutputStream mmOutputStream;
    InputStream mmInputStream;
```



```

Thread workerThread;
byte[] readBuffer;
int readBufferPosition;
int counter;
// Debugging
private static final String TAG = "datos";
private static final boolean D = true;
// Intent request codes
private static final int REQUEST_CONNECT_DEVICE = 1;
private static final int REQUEST_ENABLE_BT_AND_CONNECT_DEVICE
= 2;
private static final int REQUEST_ENABLE_BT = 3;
private static final int REQUEST_FILE = 5;
private String mConnectedDeviceName = null;
private BTcom btCom;
private BluetoothAdapter mBluetoothAdapter;
String data, _diast="0", _sist="0", _aire="0", _oxigeno="0";
int aux=0;
Double suma=0.0;
protected PowerManager.WakeLock mWakeLock;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.datos);
    spo2_=(TextView)findViewById(R.id.SPO2);
    diast=(TextView)findViewById(R.id.diast);
    sisto=(TextView)findViewById(R.id.sist);
    t_flujo=(TextView)findViewById(R.id.txtflujo);
    if (btCom == null) setupCom();
    try {
        findBT();
    } catch (IOException e) {
        // TODO Auto-generated catch block

```

```
        e.printStackTrace();
    }

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.principal, menu);
    return true;
}

@Override
public void onStart() {

    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");
}

@Override
public synchronized void onResume() {
    super.onResume();
    /*try {
        Thread.sleep(100);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }*/
    if(D) Log.e(TAG, "+ ON RESUME +");

    // Performing this check in onResume() covers the case in which
    BT was
```

```
// not enabled during onStart(), so we were paused to enable it...
// onResume() will be called when ACTION_REQUEST_ENABLE
activity returns.
    if (btCom != null) {
        // Only if the state is STATE_NONE, do we know that we haven't
started already
        if (btCom.getState() == BTcom.STATE_NONE) {
            // Start the Bluetooth com services
            btCom.start();
        }
    }
}

@Override
public synchronized void onPause() {
    super.onPause();
    if(D) Log.e(TAG, "- ON PAUSE -");
}

@Override
public void onStop() {
    super.onStop();
    if(D) Log.e(TAG, "-- ON STOP --");
}

@Override
public void onDestroy() {
    super.onDestroy();
    // Stop the Bluetooth chat services
    if (btCom != null) btCom.stop();
    if(D) Log.e(TAG, "--- ON DESTROY ---");
}
```

```

public void setupCom(){
    btCom = new BTcom(this, mHandler);
    return;
}

```

```

@SuppressLint("NewApi")
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {

            case BTcom.MESSAGE_STATE_CHANGE:
                if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
                switch (msg.arg1) {
                    case BTcom.STATE_CONNECTED:
                        // setStatus(R.string.connected);

                        break;
                    case BTcom.STATE_CONNECTING:
                        Toast.makeText(getApplicationContext(), "Conectando... ",
Toast.LENGTH_SHORT).show();
                        break;
                    case BTcom.STATE_LISTEN:
                    case BTcom.STATE_NONE:
                        // setStatus(R.string.not_connected);
                        break;
                }
                break;

            case BTcom.DIAST:

```

```
try{
    Double diastolica = Double.parseDouble((String)msg.obj);

    diast.setText(diastolica.toString()+"mmHg");}
catch(Exception e){}
break;
case BTcom.SIST:
    try{
        Double sistolica = Double.parseDouble((String)msg.obj);

        sisto.setText(sistolica.toString()+"mmHg");}
        catch(Exception e){}
        break;
case BTcom.FLUJO:

    try{
        Double dato = Double.parseDouble((String)msg.obj);

        DecimalFormat df = new DecimalFormat("0.00");
        t_flujo.setText(df.format(dato));
        dato=0.0;
    }
    catch(Exception e){}
    break;
case BTcom.SPO2:
    try{
        Double oxigeno = Double.parseDouble((String)msg.obj);
        DecimalFormat df = new DecimalFormat("0.00");

        if(oxigeno<=0.1)
        {
            spo2_.setTextSize(40);
            spo2_.setText("Revise el sensor");
```

```
    }  
    else  
    {  
        if(aux==5)  
        {  
            suma=suma/5;  
            spo2_.setTextSize(80);  
            spo2_.setText(df.format(suma)+"%");  
            suma=0.0;  
            aux=0;  
        }  
        else  
        {  
            suma+=oxigeno;  
            aux++;  
        }  
    }  
}  
}  
}  
  
    catch(Exception e){}  
    break;  
  
    case BTcom.MESSAGE_DEVICE_NAME:  
        // save the connected device's name  
        mConnectedDeviceName =  
msg.getData().getString(BTcom.DEVICE_NAME);  
        Toast.makeText(getApplicationContext(), "Conectado a  
Polisomnografo", Toast.LENGTH_SHORT).show();  
        break;  
  
    case BTcom.MESSAGE_TOAST:  
        Toast.makeText(getApplicationContext(),  
msg.getData().getString(BTcom.TOAST),
```

```

        Toast.LENGTH_SHORT).show();
    break;

    }
}
};

protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    switch(requestCode){
        case REQUEST_ENABLE_BT_AND_CONNECT_DEVICE:
            if (resultCode == RESULT_OK) {
                Toast.makeText(this, R.string.bt_enabled,
Toast.LENGTH_SHORT).show();
                Intent serverIntent = new Intent(this,
DeviceListActivity.class);
                startActivityForResult(serverIntent,
REQUEST_CONNECT_DEVICE);
                break;
            }
            else {
                // User did not enable Bluetooth or an error
occurred
                Log.d(TAG, "BT not enabled");
                Toast.makeText(this, R.string.bt_not_enabled_leaving,
Toast.LENGTH_SHORT).show();
                finish();
                break;
            }
        case REQUEST_CONNECT_DEVICE:
            if (resultCode == RESULT_OK){

```

```

        if (btCom == null) setupCom();
        //connectDevice(data, true);
    }
    break;

}
return;
}

// private void connectDevice(Intent data, boolean secure) {
void findBT() throws IOException {
    //Toast.makeText(getApplicationContext(), "Conectando..."
, Toast.LENGTH_LONG).show();
    mBluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null) {

}

    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBluetooth = new Intent(

BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBluetooth, 0);
    }

    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter
        .getBondedDevices();
    if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
            System.out.println(device.getName());
            if
(device.getName().equals("DATOS_POLISOMNOGRAFO")) {

```



```
        mmDevice = device;

        break;
    }
}

// Attempt to connect to the device
btCom.connect(mmDevice, true);

}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent serverIntent = null;
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.scan_connect:
            if (btCom != null){
                btCom.stop();
            }
            try {
                findBT();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return true;
        case R.id.disconnect:
            if (btCom != null){
                btCom.stop();
            }
            return true;
    }
}
```

```

        default:
            return super.onOptionsItemSelected(item);
    }
}
}
}

```

ACTIVITY ECG

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/Logo"
    tools:context=".Polisomnografo" >

    <Button
        android:id="@+id/ecg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/datos"
        android:layout_marginLeft="79dp"
        android:layout_toRightOf="@+id/datos"
        android:background="@drawable/bt_ecg"
    />

    <TextView
        android:id="@+id/txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/TextView01"
        android:layout_alignBottom="@+id/TextView01"
        android:layout_toLeftOf="@+id/ecg"
        android:shadowColor="#000000"
        android:shadowDx="10"
        android:shadowDy="0"
        android:shadowRadius="7"
        android:text="DATOS"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#ffffff"
        android:textSize="20pt"
        android:textStyle="bold" />

    <Button
        android:id="@+id/datos"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

```

android:layout_alignParentBottom="true"
android:layout_alignParentLeft="true"
android:layout_marginBottom="193dp"
android:layout_marginLeft="397dp"
android:background="@drawable/bt_dato" />

```

```
<TextView
```

```

    android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/ecg"
    android:layout_alignRight="@+id/ecg"
    android:shadowColor="#000000"
    android:shadowDx="10"
    android:shadowDy="0"
    android:shadowRadius="7"
    android:text="ECG"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#ffffff"
    android:textSize="20pt"
    android:textStyle="bold" />

```

```
</RelativeLayout>
```

ACTIVITY DATOS

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```

    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/AbsoluteLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/enf"
    tools:context=".Datos" >

```

```
<TextView
```

```

    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="876dp"
    android:layout_y="73dp"
    android:text="SP02:"
    android:shadowColor="#0000ff"
    android:shadowDx="3"
    android:shadowRadius="5"
    android:textSize="30pt"
    android:textAppearance="?android:attr/textAppearanceLarge"
    />

```

```
<TextView
```

```

    android:id="@+id/diast"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="115dp"
    android:layout_y="374dp"
    android:shadowColor="#0000ff"

```

```

android:shadowDx="10"
android:shadowDy="10"
android:shadowRadius="20"
android:text="0.0"
android:textAppearance="?android:attr/textAppearanceLarge"
android:textColor="#ffffff"
android:textSize="40pt" />

```

<TextView

```

android:id="@+id/textView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="128dp"
android:layout_y="292dp"
android:shadowColor="#0000ff"
android:shadowDx="3"
android:shadowRadius="5"
android:text="DIASTÓLICA:"
android:textAppearance="?android:attr/textAppearanceLarge"
android:textSize="30pt" />

```

<TextView

```

android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="179dp"
android:layout_y="22dp"
android:shadowColor="#0000ff"
android:shadowDx="3"
android:shadowRadius="5"
android:textSize="30pt"
android:text="SISTÓLICA:"
android:textAppearance="?android:attr/textAppearanceLarge"
/>

```

<TextView

```

android:id="@+id/sist"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="121dp"
android:layout_y="107dp"
android:shadowColor="#0000ff"
android:shadowDx="10"
android:shadowDy="10"
android:shadowRadius="20"
android:text="0.0"
android:textAppearance="?android:attr/textAppearanceLarge"
android:textColor="#ffffff"
android:textSize="40pt" />

```

<TextView

```

android:id="@+id/textView4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="828dp"
android:layout_y="376dp"
android:text="FLUJO DE AIRE:"
android:textAppearance="?android:attr/textAppearanceLarge"

```

```

    android:shadowColor="#0000ff"
    android:shadowDx="3"
    android:shadowRadius="5"
    android:textSize="25pt" />

```

```
<TextView
```

```

    android:id="@+id/SP02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="889dp"
    android:layout_y="155dp"
    android:shadowColor="#0000ff"
    android:shadowDx="10"
    android:shadowDy="10"
    android:shadowRadius="20"
    android:text="0.0"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#ffffff"
    android:textSize="40pt" />

```

```
<TextView
```

```

    android:id="@+id/txtflujo"
    android:layout_width="252dp"
    android:layout_height="wrap_content"
    android:layout_x="839dp"
    android:layout_y="437dp"
    android:shadowColor="#0000ff"
    android:shadowDx="10"
    android:shadowDy="10"
    android:shadowRadius="20"
    android:text="0"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#ffffff"
    android:textSize="30pt" />

```

```
<TextView
```

```

    android:id="@+id/TextView01"
    android:layout_width="446dp"
    android:layout_height="214dp"
    android:layout_x="767dp"
    android:layout_y="551dp"
    android:shadowColor="#0000ff"
    android:shadowDx="10"
    android:shadowDy="10"
    android:shadowRadius="20"
    android:text="cm3/min"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#ffffff"
    android:textSize="40pt" />

```

```
</AbsoluteLayout>
```

PROGRAMACION ARDUINO

Anexo N. PROGRAMACIÓN ARDUINO UNO

```

#include <SoftwareSerial.h>
SoftwareSerial BT(6, 5); // RX, TX
int vcc=7;
int brazaletes, pres;
long tiempo_in, tiempo_fin;
double volt;
float sist, dist;
int p_max=100;
int in_presion=0; // entradas analogicas
int bomba=9, valvula=8; // puertos digitales
double v, spo2;
double sp,mayor_h=600, menor_h=500, mayor_l=200, menor_l=200;
int val;
long last=0;
int stat=LOW;
int stat2;
int contar=0, cont=0;
int inala, exala;
boolean ok=false;
int sens=512; // this value indicates the limit reading between dark and
light,
    // it has to be tested as it may change according on the
    // distance the leds are placed.
int nPalas=2; // the number of blades of the propeller
double rpm;
int milisegundos=500; // the time it takes each reading
long inicio, fin;
double flujo;
int cont_f=0, bpm;
boolean listo=false;
long tiempo_pres_ini, tiempo_pres_fin;

```

```
int led;
void setup() {
  pinMode(vcc,OUTPUT);
  digitalWrite(vcc,HIGH);
  BT.begin(9600);

  Serial.begin(9600);

  pinMode(bomba,OUTPUT);
  pinMode(valvula,OUTPUT);
  tiempo_pres_ini=millis();
}
// 2 horas = 7200000
void loop() {
  digitalWrite(vcc,HIGH);
  presion();
  while(1)
  {

    BT.flush();
    espirometro();
    /* if(flujo>=1000)
    {
      flujo*=10;
    }*/
    BT.print("f");
    BT.print(flujo);
    BT.println("&");

    if((tiempo_pres_fin - tiempo_pres_ini)>= 360000)//tiempo de la presion
    {
      presion();
    }
  }
}
```

```
//delay(100);  
v_oxigeno();  
ok=false;  
//delay(100);  
contar=0;
```

```
BT.print("d");  
BT.print(dist);  
BT.println("&");  
BT.print("s");  
BT.print(sist);  
BT.println("&");  
BT.print("o");  
BT.print(spo2);  
BT.println("&");
```

```
Serial.print("d");  
Serial.print(dist);  
Serial.println("&");  
Serial.print("s");  
Serial.print(sist);  
Serial.println("&");  
Serial.print("o");  
Serial.print(spo2);  
Serial.println("&");  
Serial.print("f");  
Serial.print(fluj);  
Serial.println("&");  
fluj=0;  
contar=0;  
rpm=0;
```



```
    tiempo_pres_fin=millis();
}
}
void presion()
{
    tiempo_pres_ini=millis();
    brazalete=analogRead(0);
    pres=map(brazalete,0,1024,0,375);
    while(pres<200)
    {
        digitalWrite(bomba,HIGH);
        digitalWrite(valvula,HIGH);
        brazalete=analogRead(in_presion);
        pres=map(brazalete,0,1024,0,375);
        //Serial.println(pres);
        delay(100);
    }
    tiempo_in=micros();
    while(pres>50)
    {

        digitalWrite(bomba,LOW);
        digitalWrite(valvula,HIGH);
        brazalete=analogRead(1);
        pres=map(brazalete,0,1024,0,300);
        if(pres<190)
        {

            if(pres>p_max)
            {
                p_max=pres;
            }
        }
    }
}
```

```
}  
digitalWrite(valvula,LOW);  
  
dist=0.45*p_max;  
sist=p_max/1.6;  
  
}  
  
void v_oxigeno(){  
  
sp=analogRead(4);  
if(sp>=936 && menor_l==200)  
{  
  //Serial.println("0");  
  spo2=0;  
  mayor_h=600;  
  menor_h=500;  
  mayor_l=200;  
  menor_l=200;  
}  
else  
{  
  if(sp>422)  
  {  
    if(sp>mayor_h)  
    {  
      mayor_h=sp;  
    }  
  
    if(sp<menor_h)  
    {
```

```
    menor_h=sp;
  }

}
if(sp<400)
{
  if(sp>mayor_l)
  {
    mayor_l=sp;
  }

  if(sp<menor_l)
  {
    menor_l=sp;
  }

}
volt=(mayor_h/menor_h)/(mayor_l/menor_l);
spo2=(-7)*volt)+98;
mayor_h=600;
menor_h=500;
mayor_l=200;
menor_l=200;
}

}

void espirometro(){

  inicio=millis();
  while(!ok)
  {
```

```

val=analogRead(5);
  if(val<sens)
  {
    stat=LOW;
  }
else
{
  stat=HIGH;
}
// digitalWrite(13,stat); //as iR light is invisible for us, the led on pin 13
//indicate the state of the circuit.

if(stat2!=stat){ //counts when the state change, thats from (dark to
light) or
//from (light to dark), remmember that IR light is invisible
for us.
  contar++;
  stat2=stat;
}
if(millis()-last>=milisegundos){

  rpm=((double)contar/2)/2.0*60000.0/(milisegundos);
  flujo+=rpm;

  contar=0;

  last=millis();
  fin=millis();
}
if((fin-inicio)>= 1000)
{
  flujo=flujo;
  inicio=millis();
}

```

```
    ok=true;
  }

}

}

#include <SoftwareSerial.h>

SoftwareSerial BT(6, 5); // RX, TX
int vcc=7, gnd=6;
double ecg, sens=60;
double pulso=0;
int brazalete, pres;
long t_i, t_f;
long last=0;
int stat=LOW;
int stat2;
int contar=0, cont=0;

int nPalas=1; // the number of blades of the propeller
double rpm;
int milisegundos=7000; // the time it takes each reading
long inicio, fin;
int flujo, cont_f=0, bpm;
boolean ok=false;
long tiempo_pres_ini, tiempo_pres_fin;
int led;
void setup() {
  pinMode(vcc,OUTPUT);
  pinMode(gnd,OUTPUT);
  pinMode(13,OUTPUT);
```

```
digitalWrite(vcc,HIGH);
digitalWrite(gnd,LOW);
Serial.begin(9600);
BT.begin(9600);
}
```

```
void loop() {
  inicio=millis();
  while(1)
  {
    digitalWrite(vcc,HIGH);
    digitalWrite(gnd,LOW);

    ecg=analogRead(0);
    ecg=(ecg*100)/1023;
    BT.print("e");
    BT.print(ecg);
    BT.println("&");

    if(ecg<sens)
    {
      stat=LOW;
    }
    else
    {
      stat=HIGH;
    }
    digitalWrite(13,stat);
    //Serial.println(stat);
    if(stat2!=stat){
      contar++;
      stat2=stat;
    }
  }
}
```

```
if(millis()-last>=milisegundos){  
  rpm=(contar/nPalas)/2*60000.0/(milisegundos);  
  BT.print("o");  
  BT.print(rpm);  
  BT.println("&");  
  Serial.println(rpm);  
  contar=0;  
  
  last=millis();  
}  
//BT.flush();  
}  
}
```

CERTIFICACIÓN

Latacunga, Octubre 2014

ELABORADO POR:

OSCAR GÓMEZ

C.C: 0503358624

ISAAC SILVA

C.C: 0502611569

APROBADO POR:

ING. NANCY GUERRÓN

**DIRECTORA DE LA CARRERA DE INGENIERÍA EN ELECTRÓNICA E
INSTRUMENTACIÓN**

CERTIFICADO POR:

DR. RODRIGO VACA

**SECRETARIO ACADÉMICO
UNIDAD DE ADMISIÓN Y REGISTRO**