TRABAJO ESPECIAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN OXÍMETRO DE PULSO

Tutor académico: Prof. Pedro Pinto

Presentado ante la ilustre
Universidad Central de Venezuela
Por el Br. Gustavo Olivero Ortiz
Para optar por el título de
Ingeniero electricista

Caracas, 2009

DEDICATORIA

A mi querida esposa, a mis padres y hermanos, por su apoyo y comprensión.

CONSTANCIA DE APROBACIÓN

Caracas, 29 de abril de 2009

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Gustavo Olivero Ortiz, titulado:

"DISEÑO E IMPLEMENTACIÓN DE UN OXÍMETRO DE PULSO"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención de electrónica, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Mercedes Arocha
Jurado

Prof. José Alonso
Jurado

Prof. Pedro Pinto
Tutor académico

Olivero O., Gustavo

DISEÑO E IMPLEMENTACIÓN DE UN OXÍMETRO DE PULSO

Tutor académico: Prof. Pedro Pinto. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Opción Electrónica y Control Industrial. 2008. 60h+anexos.

Palabras claves: Diseño de equipo, instrumentación médica, acondicionamiento de señales, intensidad lumínica.

Resumen. Se hizo el diseño y la implementación de un equipo para medir el porcentaje de saturación relativa de oxígeno en la sangre y la frecuencia de los pulsos cardiacos. El principio fundamental en que se basa la medición del equipo es la respuesta de absorción lumínica de la hemoglobina a diferentes longitudes de ondas. Se diseñó un equipo modular donde se emplearon dos emisores con longitudes de onda 660nm y 940nm junto aun fotodiodo como elemento sensor, una tarjeta analógica en la cual se realiza el acondicionamiento de las señales, una tarjeta digital con un microcontrolador donde se realiza el programa de funcionamiento del equipo, los algoritmos de cálculo y las funciones de interfaz con los usuarios. La calibración se realizó modificando la relación de las intensidades lumínicas de los emisores y tomando como referencia el módulo oxímetro de un monitor médico IntelleVue® MP20 junior de Philips.

ÍNDICE GENERAL

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

LISTA DE ACRÓNIMOS

COHb:	Carboxihemoglobina.
EEPROM:	electrically erasable programmable read only memory.
HbO2:	Hemoglobina oxigenada o oxihemoglobina.
Hb:	Hemoglobina reducida.
LCD:	liquid crystal display.
PPM:	pulsos por minuto.
PWM:	pulse wide modulated.
RAM:	random access memory.
ROM:	read only memory.
SaO2:	saturación arterial de oxigeno.
SpO2:	estimado de la saturación arterial de oxígeno.

INTRODUCCIÓN

El monitoreo de las variables físicas tales como la presión, biopotenciales, flujo, temperatura, impedancia, concentraciones químicas, luz, radiación etc. es de suma importancia para el diagnostico y monitoreo de los signos vitales, con lo cual se establece la condición clínica de los pacientes. El desarrollo de la tecnología electrónica ha generado sensores para las variables antes mencionadas y componentes electrónicos cada vez más pequeños y precisos que junto con el aumento de la capacidad de procesamiento digital ha permitido igualmente el desarrollo en el campo médico.

Las principales compañías que fabrican oxímetros de pulso comerciales son: NONIN®, NELLCOR®, MASIMO® y GE-Healthcare® división de General Electric Co.

La oxigenación en la sangre es un parámetro importante a ser monitoreado en las salas de cuidado intensivo, y en la aplicación de anestesia durante las intervenciones quirúrgicas, existen muchas técnicas para medir la oxigenación pero la oximetría de pulso es no invasiva y de rápidos resultados.

En este trabajo se presenta un equipo que mide y muestra en una pantalla gráfica: el porcentaje de saturación de oxígeno en la sangre, la frecuencia del ritmo cardiaco y la señal correspondiente a la luz roja recibida de un paciente, para lo cual se utiliza un sensor colocado en un dedo de la mano.

Se diseñó en forma modular con una tarjeta exclusivamente para el acondicionamiento analógico de las señales, una tarjeta para el tratamiento digital que incluye el microcontrolador como componente principal, y una tarjeta para la alimentación.

En el capítulo I se trató lo relacionado con la necesidad de desarrollar un equipo oxímetro de pulso para el monitoreo clínico de pacientes y se establecieron los objetivos.

En el capítulo II se trató la reseña histórica de los equipos para medir la oxigenación sanguínea por métodos ópticos, el cual se originó durante el desarrollo de la aviación en la segunda guerra mundial debido a la necesidad de evaluar las condiciones de los pilotos en vuelos de gran altitud, ya que se presentaban desmayos en los aviadores. Estos equipos tenían un error muy grande en su medición, dando origen a la investigación y desarrollo de equipos cada vez más precisos y a la utilización masiva en el campo de la medicina.

También se trató las bases teóricas que dan paso al cálculo de la oxigenación partiendo de la ley de Beer-Lambert en donde se estudia las propiedades ópticas de las diferentes hemoglobinas y determinando la curva de calibración la cual es la función que determina el valor de la oxigenación.

En el capítulo III se trató el diseño del equipo en dos partes: el bloque analógico y el bloque digital. El bloque analógico es la parte fundamental, donde se diseñó el acondicionamiento de las señales empezando por la evaluación de las características de sus componentes críticos para la correcta selección. En el bloque digital se realizó la digitalización de las señales, el procesamiento digital efectuado por los algoritmos implementados en el microprocesador, la interfaz de usuario con el teclado y pantalla gráfica LCD (liquid crystal display) y las dos rutinas más importantes junto con sus diagramas de flujo, la rutina de la función principal que da servicio a todo el programa y la rutina que realiza los algoritmos de cálculos de la saturación y la frecuencia cardiaca.

CAPÍTULO I

1.1 PLANTEAMIENTO DEL PROBLEMA.

Actualmente en Venezuela los equipos utilizados en los centros de salud son muy costosos y fabricados por pocas empresas lideres en el del desarrollo y comercialización de la instrumentación médica, por ejemplo el Nellcor OxiMax® modelo N-600™ tiene un precio de 3.695,00$ en la página web www.imdedirect.com y el Nonin® modelo Avant 9700™ con precio 1.995,00$ en www.turnermedical.com, esto se debe al alto grado de confianza que dichas empresas generan en el campo médico, pero esta situación establece una gran dependencia tecnológica.

En la medicina, y especialmente en la unidad de cuidados intensivos se requiere del monitoreo constante de las condiciones respiratorias de los pacientes, una de éstas es la cantidad de oxígeno transportada por la hemoglobina, estas mediciones deben ser rápidas y no invasivas (que no requieren catéter).

Las técnicas de medición basadas en las propiedades ópticas de la hemoglobina y realizadas con sensores no invasivos dan resultados en pocos segundos y con una precisión aceptable de aproximadamente 2%, lo que representa su mayor ventaja. Actualmente debido al avance en la tecnología electrónica, especialmente en los convertidores analógico digital, diodos emisores de luz y microcontroladores, se han desarrollado equipos muy precisos.

Es importante notar que no se desea una medida absoluta de la cantidad de oxígeno contenida en la sangre, porque para eso sería necesario una medición que implique cateterización del paciente y medición de las presiones de los gases en la sangre, lo que se requiere es un estimado de la oxigenación para establecer la calidad de la respiración y detectar cuando un paciente entra en estado de

hipoxia (deficiencia de oxígeno) cuando esta en la sala de cuidados intensivos o en la aplicación de anestesia durante una operación.

1.2 OBJETIVOS.

1.2.1 Objetivo general.

Desarrollar e implementar un sistema medidor de saturación relativa de oxígeno en la sangre (oxímetro de pulso) para el monitoreo de la respiración de los pacientes, se estudiará la posibilidad de dotar al oxímetro de conexión inalámbrica.

1.2.2 Objetivos específicos:

- Investigar sobre los medidores de oxígeno en la sangre que existen actualmente en el mercado.
- Investigar los métodos por espectroscopia de medición de saturación de oxígeno en la sangre.
- Diseñar un circuito electrónico de acondicionamiento y digitalización de señales.
- Implementar programas que procesen las señales recibidas.
- Visualización del resultado obtenido en la medición.
- Realizar las calibraciones y ajustes necesarios para determinar el valor real medido por el oxímetro.
- Realizar las pruebas del sistema de medición.

CAPÍTULO II

MARCO REFERENCIAL

2.1 RESEÑA HISTÓRICA.

El empleo de métodos ópticos para hacer análisis de oximetría en la sangre se remonta hacia la década de los años 30 cuando Carl Matthes construyó el primer aparato para la medida de la saturación de oxígeno en la oreja utilizando dos longitudes de onda. En los años 40 J. R. Squire, descubrió que la diferencia en la transmisión de la luz roja e infrarroja en la palma de la mano antes y después de comprimir fuertemente el brazo, era función de la saturación. Glen Millikan haciendo estudios sobre la hipoxia en aviadores que realizaban vuelos a gran altitud durante la segunda guerra mundial, utilizó y desarrolló el equipo inventado por Matthes, agregó sensores en el lóbulo de la oreja y en la punta de los dedos, fue quien acuñó el vocablo oxímetro. A comienzo de la década de los 70, Hewlett Packard introdujo el primer oxímetro comercial el HP47201A, este aparato utilizaba ocho longitudes de onda para diferenciar las diferentes atenuaciones lumínicas producidas por los tejidos. En 1974 Aoyagi T, Kishi M, Yamaguchi K haciendo investigaciones con oxímetro de oreja encontró artefactos (ruidos por movimientos del lugar de medición con respecto al sensor) añadidos a sus curvas debido a los flujos pulsátiles, ideó un método para eliminar estos ruidos utilizando la relación de densidad de dos longitudes de ondas, entonces fue cuando descubrió que la relación de absorción debido a las pulsaciones a diferentes longitudes de onda variaba con la saturación de oxígeno, siendo el principio fundamental para el funcionamiento de los oxímetros de pulso actuales. [1]

2.2 BASES TEÓRICAS.

Para hacer un análisis espectrofotométrico de la absorción de la luz se usa la ley de Beer-Lambert que establece que la intensidad de la luz transmitida a través de un soluto decrece exponencialmente a medida que la concentración aumenta en una solución homogénea, y que la intensidad también decrece exponencialmente a medida que la distancia recorrida por la luz en la sustancia aumenta.

$$I = Io \cdot e^{-\varepsilon \cdot c \cdot d} \qquad (1)$$

I Es la intensidad de luz transmitida, *Io* es la intensidad de luz inicial desde la fuente, ε es el coeficiente de extinción molar en l/(cm·mol), c es la concentración en mol/l y d es la distancia en cm. La ecuación (1) también se puede escribir de la siguiente forma: [2]

$$-\ln\left(\frac{I}{Io}\right) = \varepsilon \cdot c \cdot d = A \qquad (2)$$

El símbolo A es la absorbancia y es adimensional de la ley de Beer-Lambert se desprende que para una mezcla de sustancias la absorbancia total es la suma de las absorbancias individuales a una longitud de onda dada.

$$A = \varepsilon_1 o \cdot c_1 o \cdot d_1 o + \varepsilon_\downarrow i \cdot c_1 i \cdot d_1 i \qquad (3)$$

La hemoglobina es una proteína que está presente en el torrente sanguíneo, su núcleo está formado por un átomo de hierro lo que le da su color rojo característico, su función primordial es transportar el oxígeno desde los pulmones hacia todos los tejidos del cuerpo para realizar la respiración celular. Hay varios tipos de hemoglobina, las cuatro principales en un adulto son:

- Hemoglobina oxigenada (oxihemoglobina HbO2), transporta el oxígeno hacia los tejidos.
- Hemoglobina reducida (Hb), es la que ha perdido su carga de oxígeno y va de regreso a los pulmones.
- Metahemoglobina, hemoglobina que no se une con el oxígeno, proviene de una enfermedad congénita.
- Carboxihemoglobina, hemoglobina que está combinada con monóxido de carbono y no tiene capacidad de transportar oxígeno.

Las dos primeras son las llamadas funcionales y son las que se encuentran en alta concentración, las dos últimas tienen una concentración despreciable.

Para el desarrollo de un oxímetro de pulso, solo el estudio de las propiedades de la hemoglobina oxigenada y la hemoglobina reducida son importantes.

Las curvas características de absorción se obtienen de la variación del coeficiente de extinción molar en función de un rango de longitud de onda como se ve en la figura 1. Las longitudes de onda de interés son las que muestran una diferencia apreciable entre las absorciones de las hemoglobinas en estudio, usualmente se usa la longitud de onda del rojo (660nm) donde la mayor absorción ocurre por parte de la hemoglobina reducida, y la longitud del infrarrojo (940nm) donde la mayor absorción es debida a la hemoglobina oxigenada, esta diferencia es la que le da a la sangre arterial, rica en oxígeno, un color rojo brillante y a la sangre venosa un rojo oscuro.

El oxímetro de pulso basa su funcionamiento en dos hechos importantes: a) La absorción lumínica de la hemoglobina oxigenada difiere de la absorción de la hemoglobina reducida para las longitudes del rojo (660nm) y del infrarrojo (940nm) b) La absorción de la luz transmitida a través del flujo arterial con respecto al tiempo es de naturaleza pulsátil y es causada por el cambio de volumen en las arterias debido al pulso cardiaco.

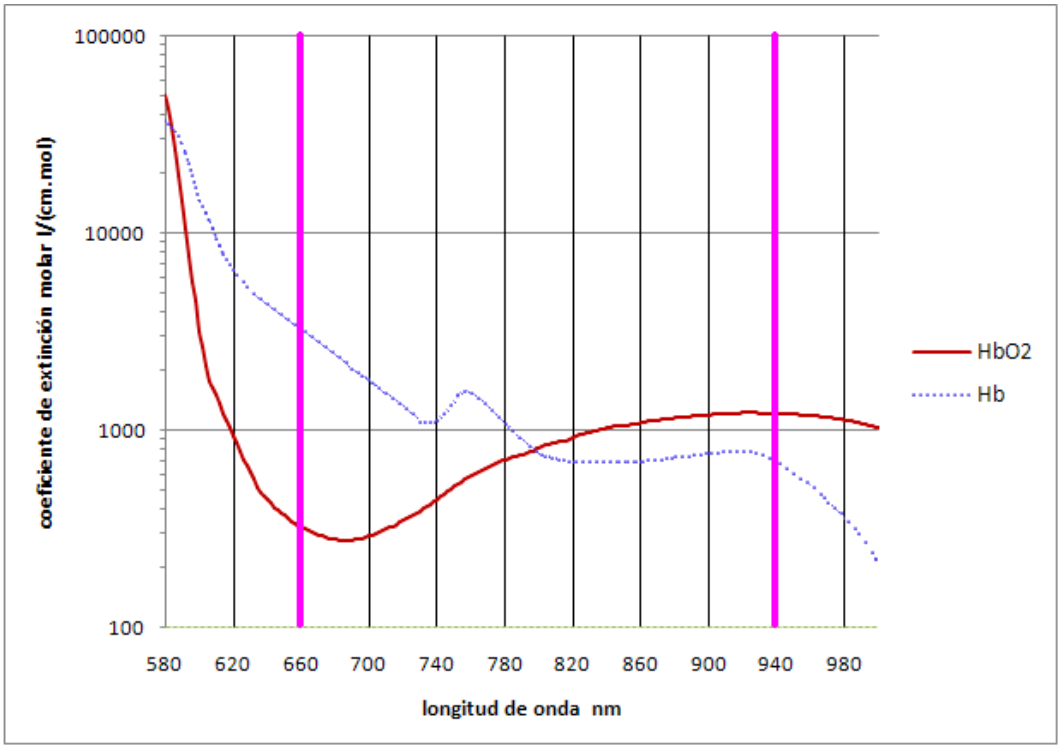

Fig. 1 Curvas absorción espectral de la hemoglobina oxigenada y reducida.

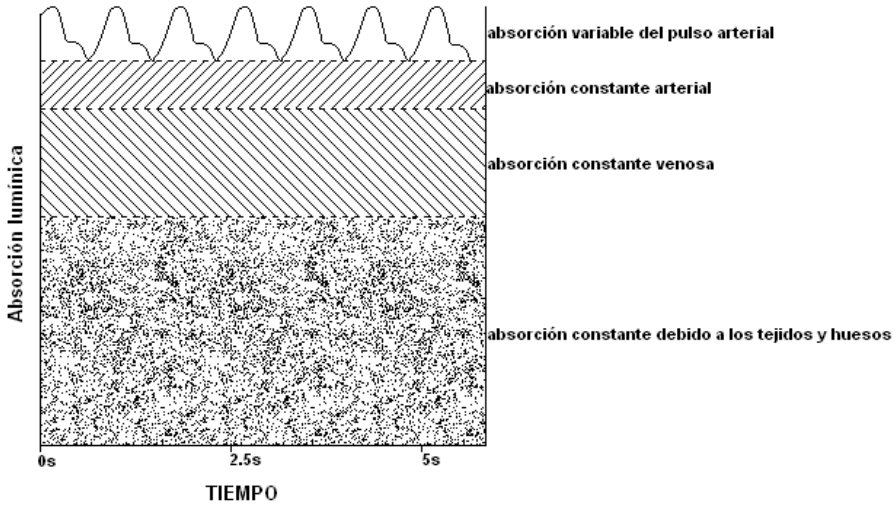

Fig. 2 Absorción de la luz en el tiempo a través de un tejido vivo.

De toda la luz transmitida solamente un pequeña porción es absorbida y disipada como energía térmica en la sangre, el resto se dispersa en el medio debido al choque elástico de los fotones con las partículas, en este fenómeno de dispersión la luz no se disipa sino que cambia de dirección y no llega al sensor,

aunque su efecto total es considerado como una atenuación y se le da un coeficiente de absorción equivalente a la atenuación.

Al coeficiente de absorción debido a luz disipada se le da el símbolo μa y al coeficiente de absorción debido a la dispersión, el símbolo μs' con unidades en cm^{-1}. [3]

$$\mu a = \varepsilon \cdot c \qquad (4)$$

$$\mu_{Total} = \mu a + \mu s' \qquad (5)$$

Para la detección de la intensidad lumínica transmitida se emplea un sensor en un lugar periférico del cuerpo y que ofrezca la mayor translucidez posible, normalmente se coloca el sensor en los dedos de las manos o en el lóbulo de la oreja, en los neonatos se coloca normalmente en la planta de los pies.

La señal de la absorción consta de una gran componente constante producida por la atenuación debido a los tejidos (piel, uñas, músculos, huesos, flujos constantes a través de las venas y arterias, pigmentos etc.) y una componente pulsátil pequeña debida al cambio en el volumen arterial producido por los latidos cardiacos figura 2.

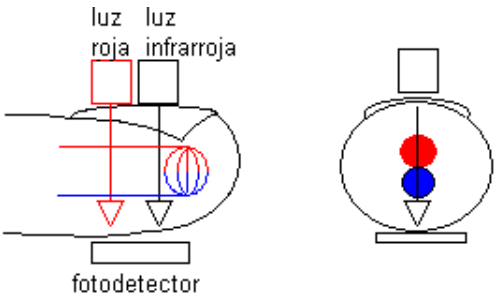

Fig. 3 Trayectoria de la luz a través del dedo.

La saturación de oxígeno está definida como el porcentaje de la concentración de oxihemoglobina contenida en la suma de las concentraciones de oxihemoglobina y hemoglobina reducida, se desprecia la presencia de otras hemoglobinas tales como la metahemoglobina y la carboxihemoglobina, por estar

en muy baja concentración y no ser funcionales, la saturación expresada de la forma anterior se llama "saturación funcional de oxigeno" y difiere de la saturación medida por un cooxímetro, equipo que mide la concentración de CO_2 y el cual mide la presencia de varias hemoglobinas por utilizar más de dos longitudes de ondas en su medición.

$$SpO2 = \frac{c_o}{c_o + c_i} \qquad (6)$$

c_o es la concentración de la oxihemoglobina, c_i es la concentración de la hemoglobina reducida, ε_o coeficiente de extinción de la oxihemoglobina y ε_i coeficiente extinción de la hemoglobina reducida.

Evaluando la ecuación (3) para dos longitudes de onda diferentes $\lambda 1$ y $\lambda 2$ y dividiendo tenemos:

$$A_{(\lambda 1)} = \varepsilon_{o(\lambda 1)} \cdot c_o \cdot d_o + \varepsilon_{i(\lambda 1)} \cdot c_i \cdot d_i \qquad (7)$$

$$A_{(\lambda 2)} = \varepsilon_{o(\lambda 2)} \cdot c_o \cdot d_o + \varepsilon_{i(\lambda 2)} \cdot c_i \cdot d_i \qquad (8)$$

$$r = \frac{\varepsilon_{o(\lambda 1)} \cdot c_o \cdot d_o + \varepsilon_{i(\lambda 1)} \cdot c_i \cdot d_i}{\varepsilon_{o(\lambda 2)} \cdot c_o \cdot d_o + \varepsilon_{i(\lambda 2)} \cdot c_i \cdot d_i} \qquad (9)$$

La concentración y la distancia no dependen de la longitud de onda y la distancia recorrida es la misma para ambas.

$$SaO2 = \frac{\varepsilon_{i(\lambda 1)} - r \cdot \varepsilon_{i(\lambda 2)}}{\left(\varepsilon_{i(\lambda 1)} - \varepsilon_{o(\lambda 1)}\right) + r \cdot \left(\varepsilon_{o(\lambda 2)} - \varepsilon_{i(\lambda 2)}\right)} \qquad \mathbf{(10)}$$

Combinando (1), (3) y luego derivando tenemos:

$$\frac{dI}{dt} = -Io \cdot d \cdot (\varepsilon_o \cdot c_o + \varepsilon_i \cdot c_i) \cdot e^{-d \cdot (\varepsilon_o \cdot c_o + \varepsilon_i \cdot c_i)} \qquad (11)$$

$$\frac{\frac{dI}{dt}}{I} = -d \cdot (\varepsilon_o \cdot c_o + \varepsilon_i \cdot c_i) \qquad (12)$$

$$r = \frac{\frac{dI_{\lambda 1}}{dt}}{\frac{I_{\lambda 1}}{\frac{dI_{\lambda 2}}{dt}}} \qquad (13)$$

La luz incidente tiene dos componentes una pulsátil mas una constante, la derivada en ecuación (13) es diferente de cero sólo para la componente pulsátil, la ecuación (13) es la razón entre las intensidades lumínicas de HbO2 y Hb que es dependiente de la saturación. Evaluando la ecuación (11) se obtiene la expresión para la saturación dada por la ley de Beer-Lambert: [4]

$$SpO2 = \frac{\varepsilon_{Hb(660nm)} - r \cdot \varepsilon_{Hb(940nm)}}{(\varepsilon_{Hb(660nm)} - \varepsilon_{HbO2(660nm)}) + r \cdot (\varepsilon_{HbO2(940nm)} - \varepsilon_{Hb(940nm)})} \cdot 100\% \quad (14)$$

Ecuaciones deducidas de las dadas en [5].

Los valores de los coeficientes de extinción molar se tomaron de los datos tabulados en [6]

$$\varepsilon_{Hb(660nm)} = 3226.56 \frac{cm^{-1}}{mol}$$

$$\varepsilon_{Hb(940nm)} = 693.44 \frac{cm^{-1}}{mol}$$

$$\varepsilon_{HbO2(660nm)} = 319.6 \frac{cm^{-1}}{mol}$$

$$\varepsilon_{HbO2(940nm)} = 1214 \frac{cm^{-1}}{mol}$$

$$SpO2 = ((3226.56 - 693.44 \cdot r)/(2906.96 + 520.56 \cdot r))(100\%) \qquad (15)$$

De la ecuación (13) se obtiene la gráfica de la saturación contra la variación de *r* Figura 4.

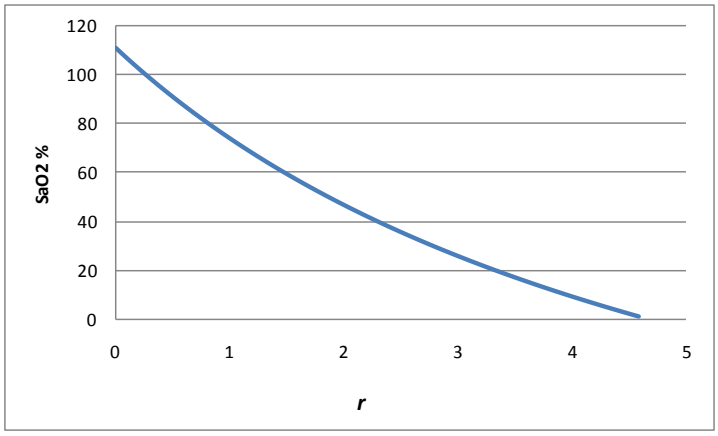

Fig. 4 Curva de la saturación en función de la razón de intensidades.

En los oxímetros de pulso comerciales no se emplea la ecuación (10) para hallar la saturación, debido a muchos factores que degradan la medición, los dos más importantes son: la no homogeneidad de la sangre y de los tejidos los cuales no siguen fielmente la ley de Beer-Lambert, y el otro fenómeno es la dispersión de la luz cuando atraviesa los tejidos. Por esa razón utilizan una aproximación empírica de la saturación de oxígeno expresada por la ecuación (16) llamada curva de calibración, donde A y B son los coeficientes de calibración, R está definida por la ecuación (17) y es una aproximación de la ecuación (13) y se obtiene midiendo las componentes pulsátiles de las señales acondicionadas recibidas en el detector y dividiendo cada una de ellas por su valor constante correspondiente. Los equipos comerciales reportan el valor SpO2 (estimado de la saturación arterial de oxígeno) como un estimado de SaO2 (saturación arterial de oxígeno),

$$SpO2 = A - B \cdot R \qquad (16)$$

$$R = \frac{\frac{ac_{HbO_2}}{dc_{HbO_2}}}{\frac{ac_{Hb}}{dc_{Hb}}} \qquad (17)$$

ac_{HbO2} componente pulsátil de la absorción debida a HbO2.

dc_{HbO2} componente constante de la absorción debida a HbO2.

ac_{Hb} componente pulsátil de la absorción debida a Hb.

dc_{Hb} componente constante de la absorción debida a Hb.

Para hallar los coeficientes de calibración el método más simple, es la realización de estudios y mediciones en un gran número de voluntarios sanos a diferentes niveles de saturación para hallar la curva de regresión lineal en los datos obtenidos. Hoy en día se utilizan diversos calibradores que simulan la saturación y se programan en los equipos complejos algoritmos de auto calibración que usan varias curvas de calibración en un mismo aparato.

El error de los oxímetros es de aproximadamente ±2% para valores entre 100% y 70% y aumenta a medida que disminuye el valor de la saturación. El valor de saturación para un individuo sano está en un rango de 99% a 90% valores por debajo de estos indican una clara condición de hipoxia en los pacientes con lo cual requiere aplicación imediata de oxígeno con mascara, por lo cual la precisión para el rango por debajo de 90% de saturación es irrelevante en una sala clínica.

2.2.1 Limitaciones del oxímetro de pulso

- Presencia de hemoglobinas no funcionales como la COHb y la metahemoglobina. Estas hemoglobinas son fuente de error debido a que sus curvas de absorción son parecidas a las funcionales; la respuesta de la COHb es semejante a la de la HbO2, y la metahemoglobina es parecida en ambas longitudes de onda por eso en alto porcentaje puede dar un valor cercano al 85%.

- Un bajo valor de perfusión (pulso débil) causa que la señal sea muy débil perdiendo precisión y dando valores erróneos.

- Ruido de artefactos, los cuales son deformaciones en las señales acondicionadas detectadas, debidas al movimiento del dedo con respecto al sensor.

- Incremento de las pulsaciones venosas debidas a obstrucciones, el oxímetro asume que el total de la componente pulsátil corresponde a flujo arterial.

CAPÍTULO III

MARCO METODOLÓGICO

El diseño del oxímetro se realizó en tres etapas: el diseño de la tarjeta que contiene la parte analógica y acondicionamiento de las señales, el diseño de la tarjeta digital, y el desarrollo del programa que fue implementado en el microcontrolador.

En el diagrama de bloques general del equipo figura 5 se tiene que la parte analógica está representada por las fuentes de corriente de los diodos emisores, con sus señales de referencias provenientes del microcontrolador. El amplificador convertidor de corriente a voltaje y la etapa de acondicionamiento la cual comienza por los retentores que reconstruyen las señales luego de la demudelación y termina en los filtros de donde salen las señales a ser introducidas en el microcontrolador. La parte digital está constituida principalmente por el microcontrolador que gobierna la pantalla gráfica, la alarma, la memoria RAM (random access memory) y recibe las líneas provenientes del teclado.

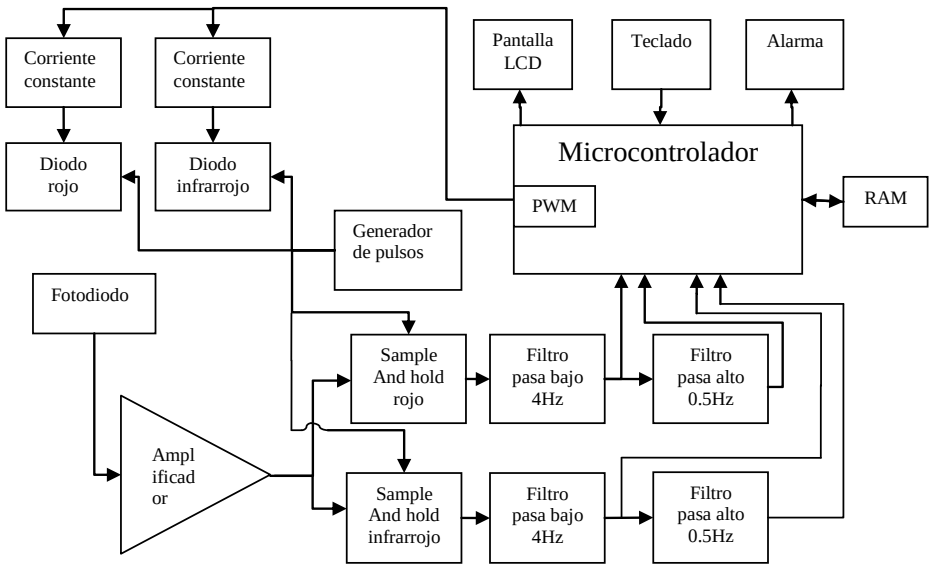

Fig. 5 Diagrama de bloques general del equipo oxímetro de pulso.

3.1 DISEÑO DE BLOQUE ANALÓGICO.

El esquema circuital de la tarjeta analógica se presenta en la figura 6.

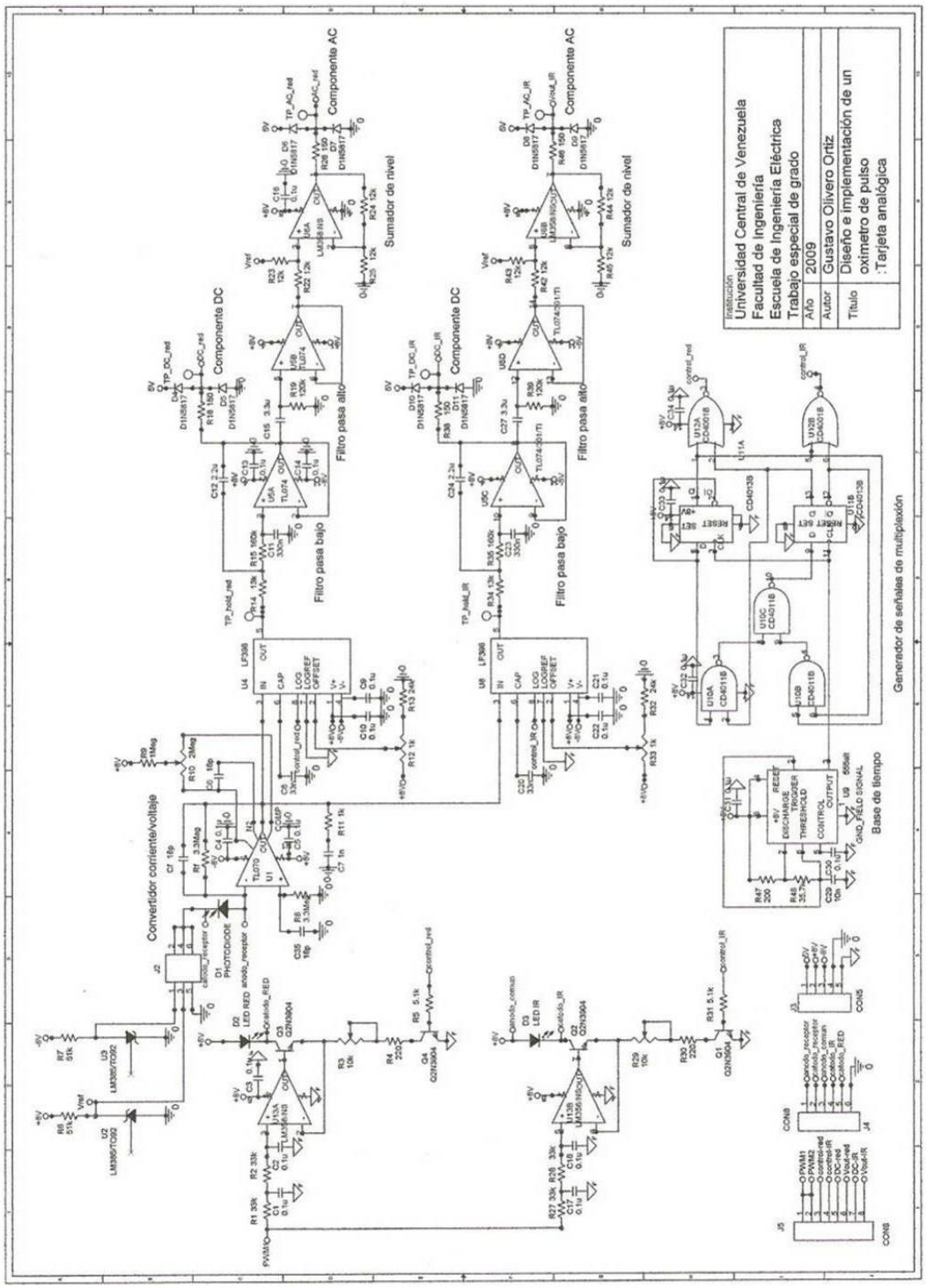

Fig. 6 Diagrama circuital de la tarjeta analógica.

3.1.1 Selección de los diodos emisores y del fotodiodo sensor.

Los diodos emisores de luz tienen una densidad espectral de emisión de forma aproximada a la gausiana, con un pico de emisión en el centro de la distribución. Las longitudes de onda de los emisores se seleccionaron según las indicadas en las bases teóricas; un diodo emisor de luz roja con longitud de onda de 660nm y un diodo emisor infrarrojo con longitud de onda de 940nm, el ancho de desviación de la emisión pico debe ser lo más angosta posible.

La unidad de intensidad lumínica fotométrica (relativa a la luz visible) es la candela (cd) que es igual a un lumen por ángulo sólido lm/sr.

Se realizó una búsqueda de diodos emisores disponibles en el comercio local que tuvieran las longitudes de onda requeridas y que se dispusiera de sus hojas de datos, los resultados se recogieron en la siguiente tabla:

Tabla 1 Características de diodos emisores.

Componente	λ_{pico}	*I @ 20mA*	$\Delta\lambda$	2θ
XLMR53WDW	660nm	1790mcd	±20nm	±30º
XLMR53DH	660nm	1490mcd	±20nm	±30º
OP140A	935nm	0.4 W/cm^2	±50nm	±40º
LTE306	940nm	1.6mW/cm^2	±50nm	±40º

Se seleccionó el XLMR53WDW fabricado por Sunled® por tener la mayor potencia lumínica de emisión además de ser de alto brillo.

Se seleccionó el diodo emisor infrarrojo OP140A fabricado por Siemens® por tener un encapsulado plano el cual fue adecuado para el receptáculo del sensor a ser colocado en el dedo. La intensidad de los emisores infrarrojos es mayor que la de los diodos emisores rojos además el sensor tiene mayor sensibilidad para estas longitudes de ondas.

Los fotosensores que tienen sensibilidad para el rango de longitudes de ondas necesario, abundantes en el mercado local son los fototransistores, el único fotodiodo para luz visible disponible es el BPW34 fabricado por Siemens®.

Se seleccionó el fotodiodo BPW34 por poseer mayor linealidad, en comparación con un fototransistor y poseer mayor superficie sensible.

Tabla 2 Valores del BPW34 tomados de la hoja de datos de Siemens.

Parámetro	Valor	unidad
Área sensible	7.00	mm^2
Rango espectral	400 a 1100	nm
t_r y t_f (Vr=5V)	20	ns
Idark(E=0, Vr=2.5 curva)	500	pA
Cj (Vr=2.5 curva, E=0)	28	pF
Rsh (de la curva)	50	MΩ
NEP	$4.1x10^{-14}$	W/√Hz
Longitud de onda máx. sensibilidad	850	nm
Sensibilidad relativa a 660nm	70	%
Sensibilidad relativa a 940nm	85	%

Los fotodiodos tienen el siguiente modelo circuital aproximado:

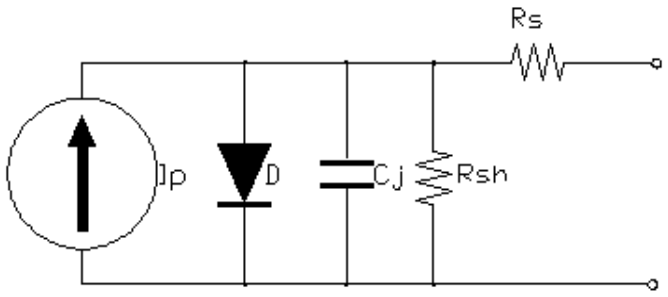

Fig. 7 Modelo circuital del fotodiodo.

Ip: fotocorriente.

Cj: capacitancia de unión.

Rsh: resistencia shunt de la unión.

Rs: resistencia serie

La capacitancia de la unión del fotodiodo se determinó con la curva dada por el fabricante Siemens [7], la cual es graficada en función de la tensión inversa aplicada al fotodiodo, en nuestro caso la tensión inversa aplicada fue de 2.5V dando como resultado un aproximado de 28pF.

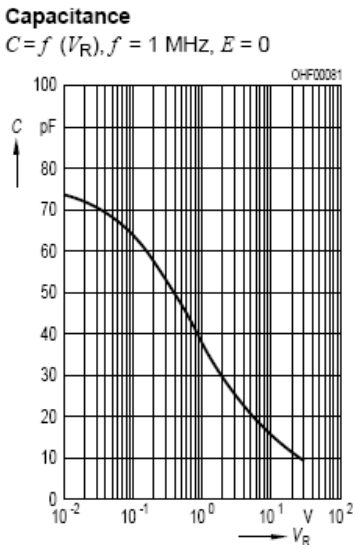

Fig. 8 Capacitancia de unión vs Tensión inversa dada en la hoja de datos.

La resistencia shunt se define como la pendiente de la curva corriente-tensión del fotodiodo en el origen, experimentalmente se obtiene aplicando 10mV y midiendo la corriente que circula. El fabricante Siemens no da este dato por lo que se obtuvo dividiendo 10mV entre 0.2nA que es el aproximado de la corriente de oscuridad en el origen.

$$Rsh \cong \frac{\mathbf{10}mv}{\mathbf{0.2}na} = 50\mathbf{M\Omega} \qquad (\mathbf{18})$$

La resistencia serie tampoco está especificada por el fabricante pero este valor suele ser de unos pocos cientos de Ω, en nuestro modelo despreciamos Rs porque este valor es mucho más pequeño que Rsh y es de importancia sólo en altas frecuencias, en este trabajo las frecuencias son muy bajas de 0.5Hz a 4Hz.

En el fotodiodo hay tres fuentes principales de ruido, dos son ruidos de disparo que dependen de la foto corriente generada por la señal y por la corriente

de oscuridad, y una que es generada por el ruido térmico de Rsh, como indica la ecuación 19.

$$i_n{}^2 = i_{ns}{}^2 + i_{nosc}{}^2 + i_{nRsh}{}^2 \quad \left[\frac{A}{\sqrt{Hz}}\right]^2 \qquad (19)$$

3.1.2 Controlador de los diodos emisores.

La intensidad lumínica de los diodos emisores se controla con una fuente de corriente constante (figura 9) para poder mantener el nivel de la señal detectada en el mayor rango posible antes de ser digitalizada, la referencia de la fuente es regulada con un PWM (pulse wide modulated), el esquema circuital es la típica fuente de corriente constante con amplificador operacional y un transistor en la salida para dar amplificación de corriente. Cada fuente tiene un transistor actuando como interruptor para controlar el apagado y encendido de los emisores según la modulación en el tiempo de las señales.

Como se dispone de un solo PWM en el microcontrolador se colocó en serie con cada uno de los diodos emisores un potenciómetro multivueltas para controlar el nivel de intensidad individual de cada diodo emisor y poder determinar con esto la relación de intensidades.

La frecuencia del PWM fue establecida en 78.12kHz para tener una resolución de 8 bits en el ciclo de trabajo. Para obtener el valor DC (direct current) se demodula la señal del PWM con un filtro pasa bajo con frecuencia de corte de 48Hz

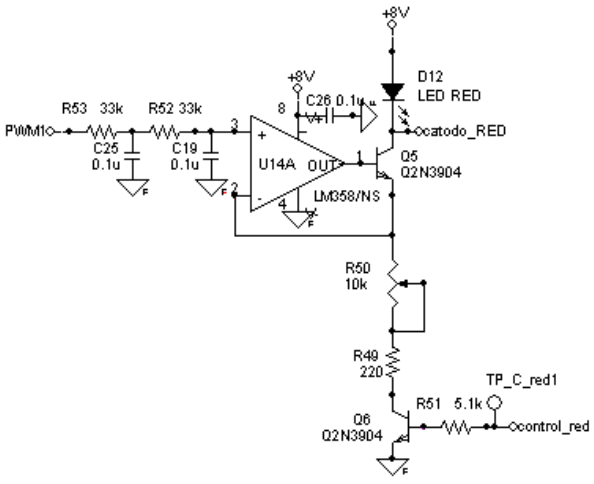

Fig. 9 Diagrama circuital de la fuente de corriente.

3.1.3 Amplificador de transimpedancia.

La señal luminosa transmitida a través del dedo incide sobre el fotodiodo y genera una corriente proporcional a la intensidad lumínica recibida. Es necesario emplear un convertidor de corriente a tensión para detectar la señal, se diseñó un amplificador de transimpedancia (figura 10) formado por un amplificador operacional y una resistencia de realimentación para realizar la conversión, el modelo para este amplificador es la ecuación (20).

$$Vout = -Is \cdot Rf \qquad (20)$$

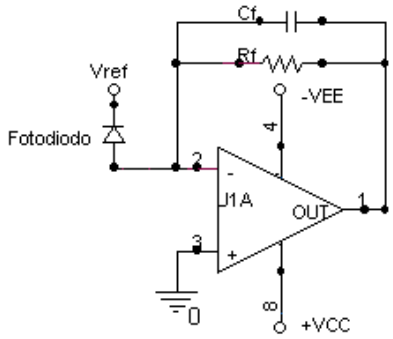

Fig. 10 Amplificador de transimpedancia.

Se usó la técnica de multiplexión en el tiempo para lograr la detección de las dos señales roja e infrarroja con un solo sensor, donde la señal lumínica proveniente de cada uno de los emisores se modula con un tren de pulso de frecuencia 500Hz y 25% de ciclo.

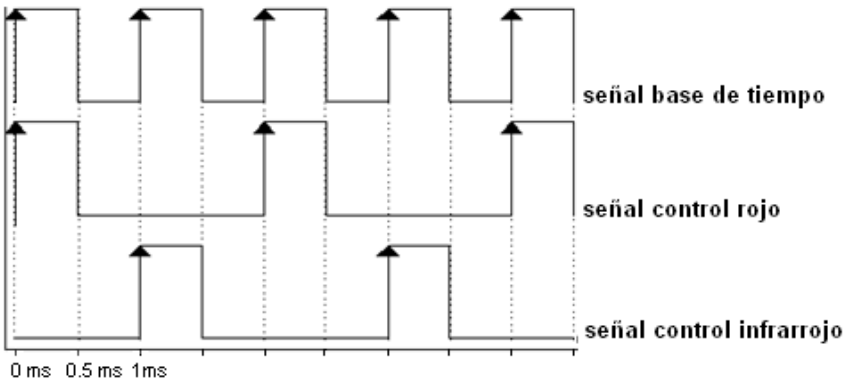

Fig. 11 Diagrama de tiempo de la multiplexión.

Se empleó un circuito astable con una frecuencia de un 1kHz utilizando un CI (circuit integrated) 555 para obtener la base de tiempo de la multiplexión, las señales de control se generaron con un circuito digital secuencial sincrónico con la base de tiempo, todo esto se hizo con la intensión de no utilizar pines del micro controlador y de hacer la tarjeta de circuitería analógica lo más independiente posible de la tarjeta digital.

Los niveles de la corriente generada por la señal lumínica son bajos en el orden de microamperios, por lo que se requiere un valor de amplificación muy alto resultando en un amplificador muy susceptible al ruido y a las corrientes de polarización en las entradas del operacional.

Se buscó en el comercio los amplificadores con bajas corriente de polarización y baja densidad espectral de corriente de ruido.

Tabla 3 Datos de operacionales de bajo ruido

Componente	*Rin* *[Ω]*	*Io* *[pA]*	*Ib* *[pA]*	$e_{n\ @\ 10Hz}$ *nv/√Hz*	$i_{n\ @\ 10Hz}$ *pA/√Hz*
LF357	10^{12}	3	30	60	≈0.06
LM11CL	10^{11}	4	17	150	-
TLC272	10^{12}	0.1	0.6	140	-
TLC2272	10^{12}	60	60	50	≈0.06
TL052	10^{12}	5	30	70	≈0.06
TL070	10^{12}	5	65	45	≈0.06
TL072	10^{12}	5	65	45	≈0.06
OP07	$80 \cdot 10^{6}$	400	1000	10.3	0.32
OP27	$6 \cdot 10^{6}$	12000	15000	3.8	1.7
OP77	$45 \cdot 10^{6}$	300	1200	10	0.032

Se seleccionó un operacional entre los que tienen entrada FET (field effect transistor) por ser los que tienen menor corriente de polarización y menor densidad espectral de corriente de ruido. Finalmente se seleccionó el TL070 fabricado por Texas instruments® porque está entre los que tienen baja densidad

espectral de tensión de ruido además posee terminales externos para la compensación y el balance del tensión de error en la salida.

Para determinar una primera aproximación de la resistencia de realimentación, se tomó el modelo del dedo conformado por las arterias, las venas y los tejidos (fundamentalmente piel, melanina y músculos) como en la figura 3 donde se supusieron valores para las distancias recorridas por la luz roja atreves de los órganos, en estos caminos se encuentra la hemoglobina oxigenada y la hemoglobina reducida, se escogió la luz roja para estos cálculos por tener el fotodiodo menor sensibilidad a ella.

Se tomó una distancia de recorrido en la piel de ocho milímetros, dos milímetros para la hemoglobina oxigenada y dos milímetros para la hemoglobina reducida.

De la ecuación (5) tenemos:

$$I = Io \cdot e^{\dagger}(-((total) \cdot d) \qquad (20)$$

Se tomo como coeficiente de absorción de la piel el valor 0.24±0.15 [mm^{-1}], para una persona de raza negra debido a la presencia de mayor cantidad de melanina, el valor fue tomado de la tabla 2.1 pág. 18 “Chapter 2 Fundamentals of Tissue Optics” [3].

Los coeficientes de extinción molar de la hemoglobina oxigenada y de la hemoglobina reducida a 660 nm es 319.60[cm^{-1}/M] y 3226.56[cm^{-1}/M] respectivamente, tomados de “Optical absorption of hemoglobin” por Scott Prahl, Oregon Medical Laser Center [6] donde también dan la formula siguiente:

$$A = \frac{e \cdot \left[\frac{\frac{1}{cm}}{\frac{moles}{litros}}\right] \cdot \left[\frac{g}{litros}\right] \cdot d[cm]}{64500\left[\frac{g}{moles}\right]} \qquad (21)$$

Donde *e* es igual a ε, *x* es la concentración de la hemoglobina con valor de 150 g/litros y *d* la distancia recorrida por la luz.

Para la oxihemoglobina tenemos:

$$A_{HbO_2} = \frac{319.60\left[\frac{\frac{1}{cm}}{\frac{moles}{litros}}\right] \cdot 150\left[\frac{g}{litros}\right] \cdot 0.20[cm]}{64500\left[\frac{g}{moles}\right]} = 0.14 \qquad (22)$$

Para la hemoglobina reducida tenemos:

$$A_{Hb} = \frac{3226.56\left[\frac{\frac{1}{cm}}{\frac{moles}{litros}}\right] \cdot 150\left[\frac{g}{litros}\right] \cdot 0.20[cm]}{64500\left[\frac{g}{moles}\right]} = 1.50 \qquad (23)$$

$\mu_{piel}=0.24mm^{-1}$

$A_{piel}=0.24mm^{-1}\cdot(8mm)=1.92$

$$I = Io \cdot (e^{-0.14} \cdot e^{-1.50} \cdot e^{-1.92}) = Io \cdot 28.43x10^{-3} \qquad (24)$$

El diodo emisor para 660nm es el XLMR53WDW el cual conduciendo una corriente de 20mA tendrá una intensidad lumínica de 1790mcd (milicandelas). [8]

I=1790mcd*28.43x10-3=50.88mcd

Como la fotocorriente en el sensor viene dada en la hoja de datos en función de la iluminancia se debió transformar la intensidad (unidad mcd) en iluminancia (unidad lux) para lo cual se calculó el ángulo sólido entre el emisor y el receptor tomando una distancia de 12mm entre ambos y una superficie de 7mm2 en el fotodiodo.

1cd=1lm/sr . (25)

Lx=lm/m^2 . (26)

cd=candela.

sr=estereorradián.

lx=lux

El ángulo sólido entre el emisor y el fotodiodo es:

$$\Omega = \frac{S}{r^2} = \frac{7mm^2}{(12mm)^2} = 0.048sr \qquad (27)$$

La sensibilidad del BPW34 a 660nm es el 70% de la sensibilidad a 850nm por eso .

La potencia lumínica es:

$$\Phi = 82.26mcd \cdot 0.048 \cdot 0.7 = 1709.57\mu lm \qquad (28)$$

La iluminancia en el fotodiodo es:

$$E = \frac{517.10\mu lm}{7\mathbf{x10}^{-6}m^2} = 244.22lx \qquad (29)$$

En la curva de la fotocorriente contra la iluminancia dada por el fabricante se halló aproximadamente una corriente de 20μA para un valor de 244.22lx, y para producir cinco voltios en la salida del amplificador de transimpedancia es necesaria una resistencia de realimentación de 250kΩ, en los cálculos anteriores no se tomó en cuenta la atenuación producida por la dispersión lumínica. Experimentalmente el valor obtenido fue de 6.8MΩ

Como la resistencia de realimentación es bastante grande para limitar el ancho de banda del circuito amplificador y evitar inestabilidad se colocó un condensador en paralelo con la resistencia de realimentación, para el cálculo del condensador se tomó en cuenta que la señal es multiplexada con pulsos rectangulares y se estableció que alcanzara nueve constantes de tiempo en el ciclo de trabajo del pulso, con nueve constantes de tiempo se comete un error aproximado de 0.012%.

$$\tau = \frac{0.5ms}{9} = 55\mu s$$

$$Cf = \frac{\tau}{Rf} = 16nF \qquad (30)$$

Cf=18*nF*

Se utilizó el circuito integrado retentor LF398 con un condensador de 0.1μF de polipropileno.

3.1.4 Filtrado de la señal muestreada.

La frecuencia cardiaca de un adulto sano está entre 60 ppm (pulsos por minuto) a 80 ppm y la de un neonato está en un rango de 100 ppm a 120 ppm, el ancho de banda de la señal de interés se estableció tomando una frecuencia cardiaca mínima de 30 ppm (0.5Hz) y una frecuencia cardiaca máxima de 220 ppm (3.6Hz).

Se implementó un clásico filtro pasa bajo Butterworth de segundo orden con frecuencia de corte en 4Hz en configuración Sallen Key, para eliminar el ruido y las componentes de alta frecuencia añadidas a las señales provenientes de los circuitos de muestreo y retención. Se seleccionó este filtro por tener una respuesta de ganancia plana en la banda pasante.

Las salidas de los filtros pasa bajo son introducidas al convertidor analógico digital del microcontrolador para ser tomadas como las componentes constantes de las señales fotopletismográficas. Luego estas mismas señales pasan por filtros pasa alto de primer orden y frecuencia de corte en 0.5Hz para eliminar el nivel constante de las señales y poder amplificarlas con una ganancia de 28 determinada experimentalmente, la cual produce una excursión de aproximadamente cuatro voltios pico pico para un nivel DC de cuatro voltios. Se le sumó un pedestal DC de 2.5 voltios a la señal alterna amplificada para cumplir con las especificaciones de tensión de entrada del convertidor analógico digital. Los filtros fueron diseñados con el programa de distribución gratuita Filterpro® de Texas Instrument.

Debido a que las señales de interés son de muy bajas frecuencias la tecnología de fabricación de los condensadores es de suma importancia y debe poseer la menor corriente de fuga, los condensadores utilizados son de polipropileno.

3.2 DISEÑO DEL BLOQUE DIGITAL.

En el bloque digital se realizan las funciones de digitalización de las señales analógicas, el almacenamiento de los datos en memoria RAM, la interfaz de usuario compuesta por una pantalla de visualización, un teclado de cuatro botones y una alarma, como se ve en la figura 12.

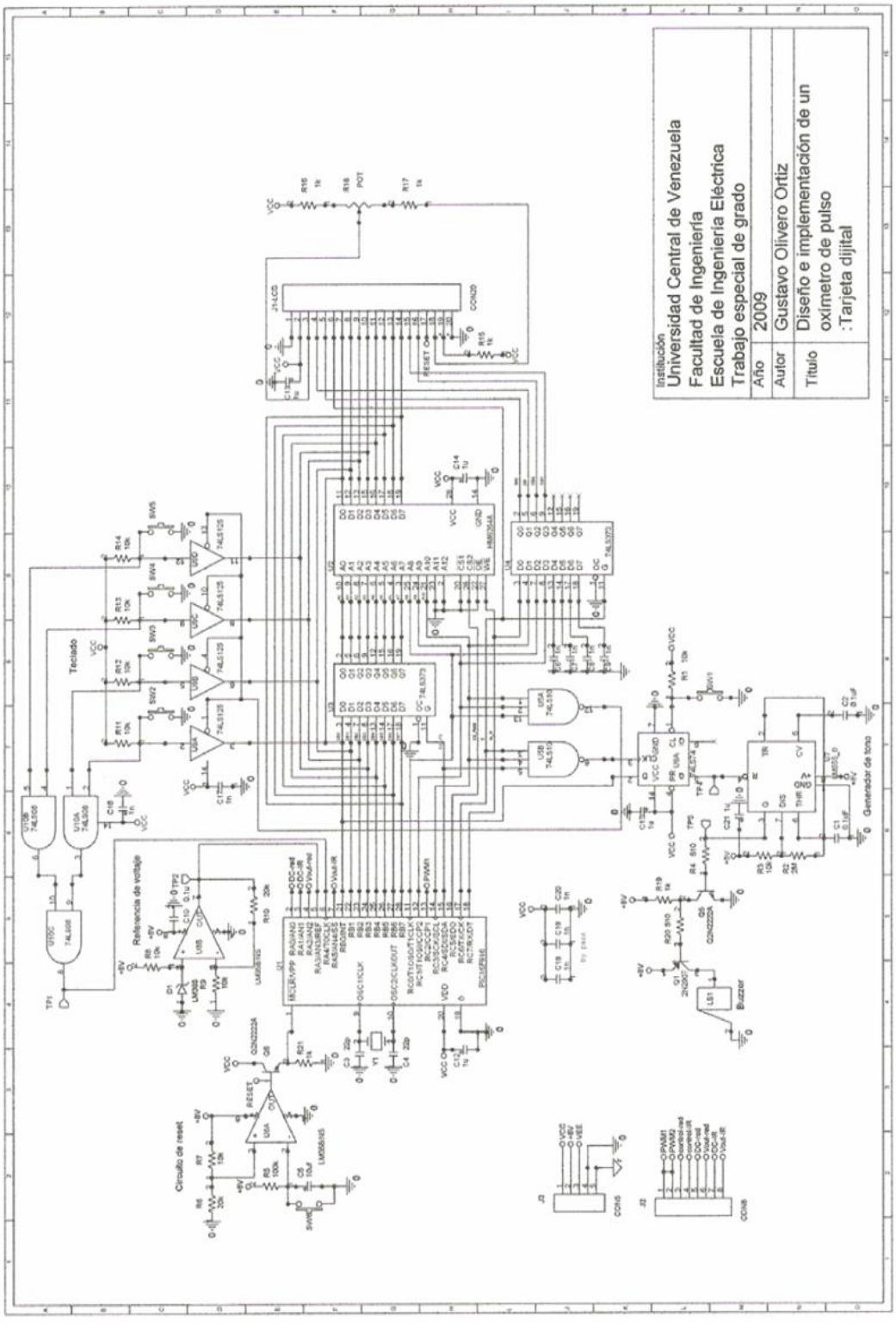

Fig. 12 Diagrama circuital de la tarjeta digital.

3.2.1 Procesamiento digital.

El microcontrolador utilizado es el PIC16F916 de gama media que tiene como características principales: 8k por 14 bits de palabras de memoria de programa, 352 bytes de memoria RAM, 24 pines de entradas/salidas, un convertidor analógico digital con 10 bits de resolución y 5 canales, 256 memoria EEPROM (electrically erasable programmable read only memory), un módulo PWM y una frecuencia máxima de reloj de 20MHz. Este microcontrolador es de fácil adquisición y cuesta aproximadamente un poco más de la mitad que uno de 40 pines de gama media, cumple con todos los requerimientos necesarios para el funcionamiento del equipo aunque su cantidad de líneas de entradas/salidas limita su desarrollo. [9]

La excursión de tensión (0.5V a 4.5V) de la componente AC de las señales roja e infrarroja se controla con el ciclo de trabajo de la señal del módulo PWM.

Dada la frecuencia de reloj del micro se configuró el convertidor A/D para tener una resolución de ocho bits y un tiempo de conversión total de aproximadamente 17μs. La frecuencia de muestro se estableció en 100Hz que es 25 mayor que la frecuencia máxima de interés. Se toman 250 muestras para cada una de las señales por periodo de análisis, lo que da un tiempo de dos segundos y medio.

La memoria RAM del microcontrolador no es suficiente para contener los 1000 datos provenientes de la digitalización por esta razón se utilizó una memoria RAM estática externa HM6264A .

Para visualizar la saturación y la frecuencia cardiaca se empleó una pantalla gráfica GDM12864D de 128x64 puntos donde aparte de los datos anteriores se muestra la señal fotopletismográfica correspondiente a la componente AC de la señal del infrarrojo, se usa esta señal porque en condiciones de alta saturación esta tiene mayor nivel que la señal proveniente del rojo. La visualización de la curva fotopletismográfica sirve como realimentación visual al usuario del equipo debido a que cuando existen artefactos los valores reportados pueden ser erróneos.

Como la pantalla no dispone de memoria ROM (read only memory) todos los elementos a ser visualizados son implementados punto por punto en la memoria de programa del micro controlador. La pantalla es un elemento de interfaz de usuario donde se muestra el menú correspondiente a las alarmas cuyos valores pueden ser configurados. La pantalla dispone de diodos que le dan luz de fondo pero consumen mucha corriente.

Se implementó un teclado con cuatro botones un botón de subida en el menú, un botón de bajada un botón de decremento de valor y un botón de incremento que también funciona como botón de actualizar valores, el menú se activa al pulsar cualquier botón.

Para generar el sonido de activación de la alarma se utilizó un circuito astable con un integrado 555 el cual genera una señal de 1kHz que controla la tensión sobre el buzzer.

3.2.2 Programa del microcontrolador.

El programa se desarrolló en lenguaje C con el compilador PCM™ de CCS®. Está dividido en: una función principal, y unas funciones secundarias que sirven como rutinas de servicio. Un gran porcentaje del programa está destinado a las rutinas de visualización de los resultados en la pantalla gráfica y del menú.

El funcionamiento del programa comienza por la ejecución de la función principal con el diagrama de flujo presentado en la figura 13 donde fundamentalmente de establece la configuración inicial de operación del equipo y un ciclo de espera donde al cumplirse las condiciones proveniente de la rutina de interrupción del temporizador uno se ejecutan las rutinas de cálculos y la visualización de los resultados en la pantalla gráfica, en caso contrario se mantiene la verificación continua del ciclo hasta que ocurra cualquiera de las interrupciones habilitadas.

La rutina de interrupción del temporizador uno (timer1) se ejecuta continuamente para establecer una base de tiempo de 10ms que es el periodo con que se muestrean las señales, cada muestra es digitalizada y guardada en la memoria RAM, se cuentan 250 periodos para obtener un intervalo de análisis.

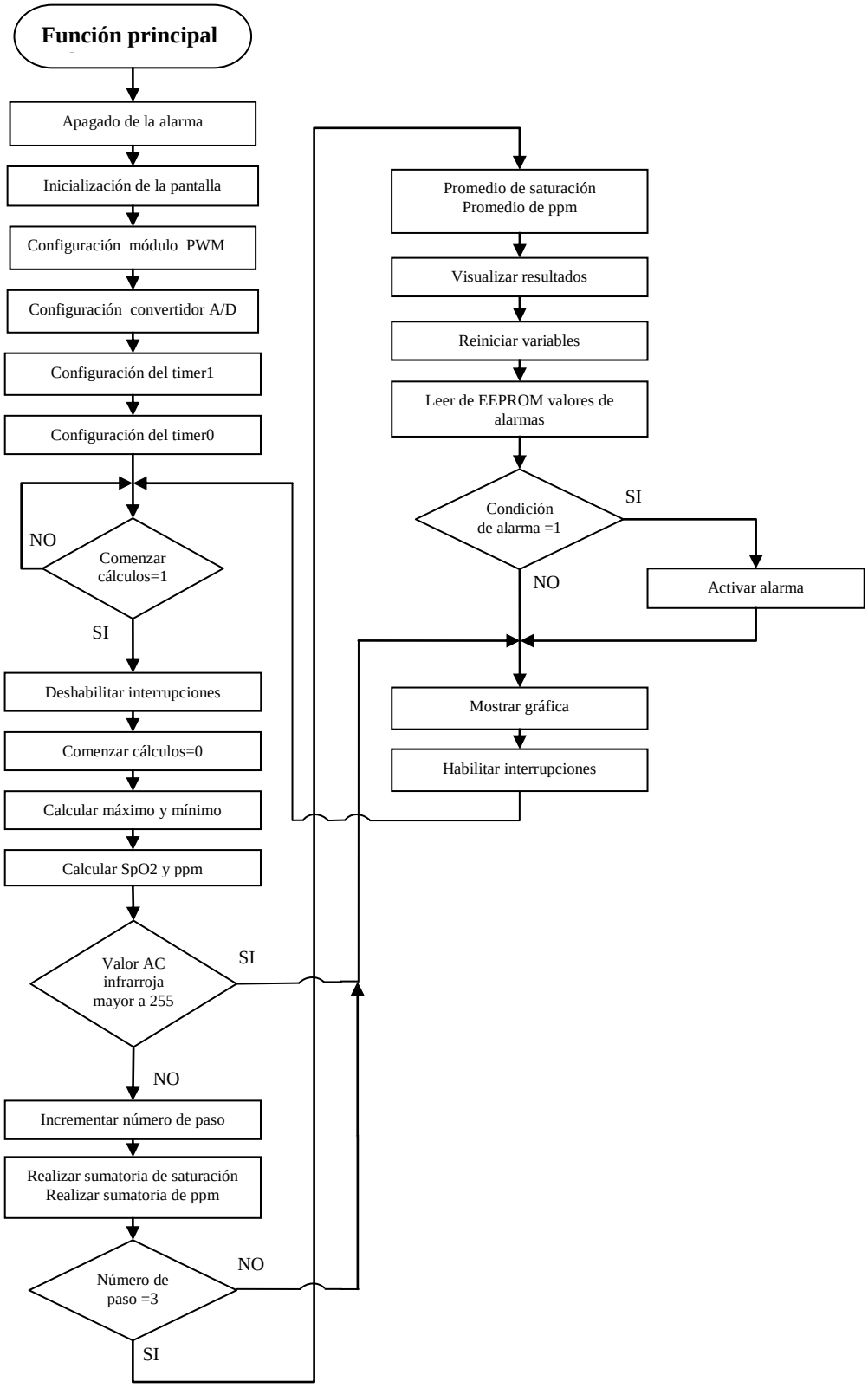

Fig 13 Diagrama de flujo de la función principal.

Cuando se enciende el equipo y se sale de la rutina del menú se lee desde la memoria EEPROM el último valor de ciclo de trabajo del PWM guardado, se realiza el análisis sobre las muestras para establecer un ciclo de trabajo que determine la mayor excursión posible del nivel de las señales.

Se configura el temporizador cero (timer0) como un contador que es incrementado cuando se detecta un pulso en la señal presente en la patilla seis del micro y la cual proviene del teclado. La rutina de servicio del temporizador cero visualiza y maneja el menú de configuración de los valores de activación de las alarmas, los cuales están definidos para el valor alto y bajo de la saturación y del ritmo cardiaco respectivamente.

En cada intervalo de análisis se toman 250 muestras a los cuales se les calcula máximo, mínimo, saturación y ritmo cardiaco, se calcula el promedio de de la saturación y del ritmo cardiaco cada tres intervalos de análisis para realizar la visualización.

3.2.3 Rutina de cálculo de la Spo2 y los ppm.

El diagrama de flujo se presenta en la figura 14, esta rutina tiene dos partes importantes: primero la determinación de los periodos de los pulsos que hay en un intervalo de análisis de 2.5 segundos con esto se determina el ritmo cardiaco y segundo el cálculo de la razón instantánea que determina la saturación a través de la ecuación (13).

Para calcular el periodo de un pulso se cuenta al número de periodos de muestreos que hay en tres cruces sucesivos de la señal a un nivel de referencia establecido. Para esta parte se usa solamente la data proveniente de la señal infrarroja debido a que en condiciones normales de saturación es la que presenta mayor componente pulsátil. Al comenzar el cálculo de los periodos de cada uno de los pulsos se detecta el momento en que el nivel de la señal está por debajo del nivel promedio (punto bajo), El nivel de referencia establecido es el valor medio entre el valor promedio de la señal y el nivel máximo absoluto de la señal (punto alto) en el intervalo de análisis.

La razón se calcula computando la derivada en cada paso de muestreo y dividiéndola por su componente DC respectiva, esto se hace tanto para la señal

roja como para la infrarroja, se realiza la sumatoria de cada valor de la razón para hallar su promedio y calcular el valor de la saturación utilizando la ecuación (15) en el intervalo de análisis.

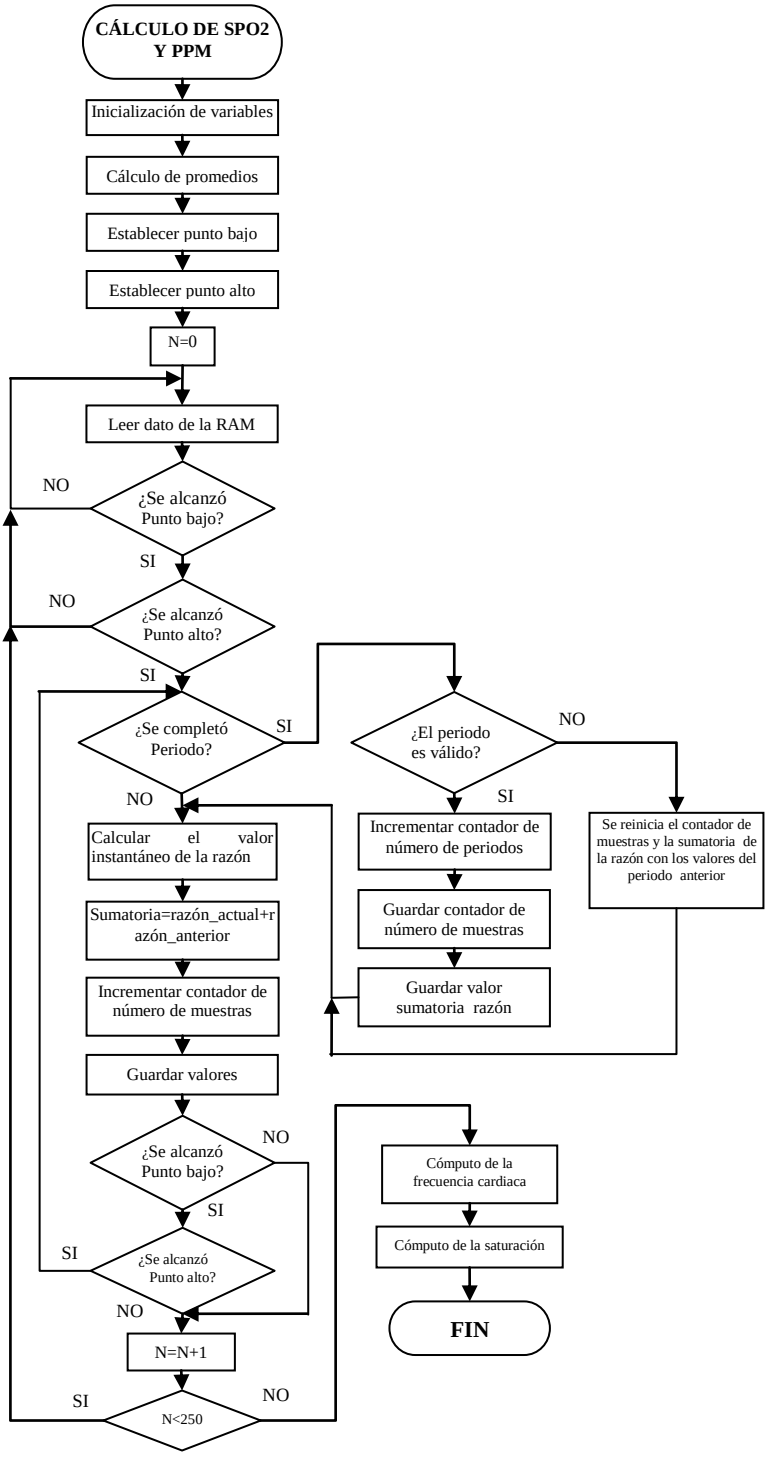

Fig. 14 Diagrama de flujo para cálculo de la saturación y el ritmo cardiaco.

3.3 RESULTADOS Y ANÁLISIS.

La calibración de la oxigenación se realizó implementando la ecuación (15) en el microcontrolador, donde al momento de la calibración la relación de intensidades se fijó variando los potenciómetros de ajuste de 10kΩ presentes en la fuente de corriente de los diodos emisores.

Como equipo de referencia se usó un monitor multiparámetros clínico modelo **IntelliVue MP20 junior** de marca Philips.

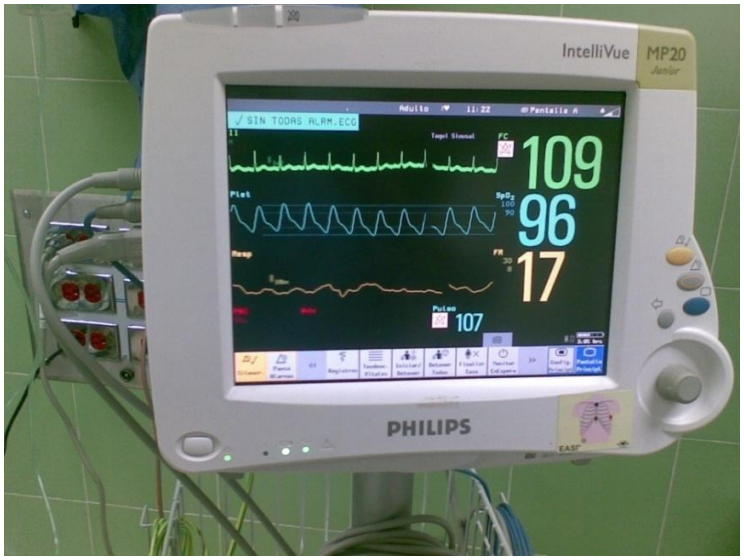

Fig. 15 Monitor con oxímetro usado como referencia.

Fig. 16 Imagen del equipo diseñado.

Se tomaron medidas en el Centro Diagnostico Integral CDI Maternidad ubicado en la Av. San Martín de Caracas, el día tres de marzo del dos mil nueve (03/03/09) a las 10:00 am al paciente Juan González, cédula de identidad 10.262.649, al cual se le colocó el sensor del equipo en el dedo índice de la mano derecha y el sensor de la referencia en el dedo índice de la mano izquierda.

En las tablas siguientes se reportan los valores medidos a las 10:00 am y a las 11:00 am, cada muestra tiene una diferencia de 10 segundos.

Tabla 4 Valores tomados el 03/03/09 a las 10:00 am con intervalos de tiempo de 10 segundos.

Tiempo (s)	Saturación equipo (%)	Frecuencia cardiaca equipo (PPM)	Saturación referencia (%)	Frecuencia cardiaca referencia (PPM)	Diferencia saturación	Diferencia frecuencia cardiaca
0	95	113	96	112	1	-1
10	94	112	95	111	1	-1
20	96	110	95	113	-1	3
30	96	110	96	114	0	4
40	94	119	95	115	1	-4
50	94	111	95	113	1	2
60	94	106	95	105	1	-1
70	94	106	95	110	1	4
80	94	114	95	107	1	-7
90	95	108	96	105	1	-3
100	95	108	96	111	1	3

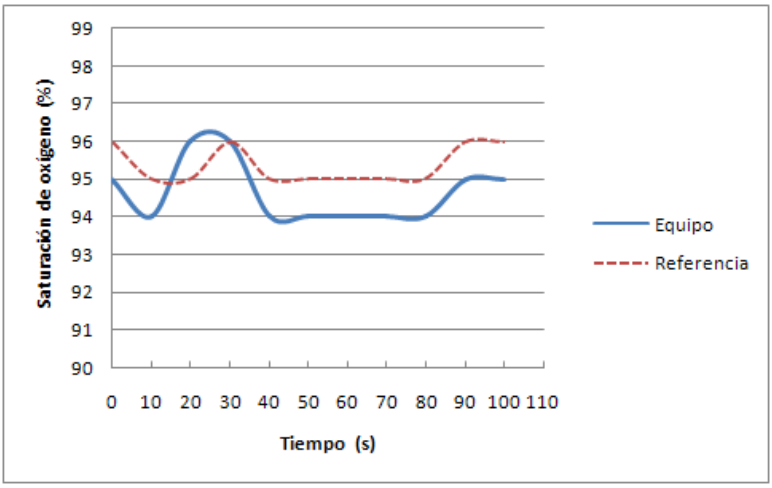

Fig. 17 Gráfica de la saturación de oxígeno datos de la tabla 4.

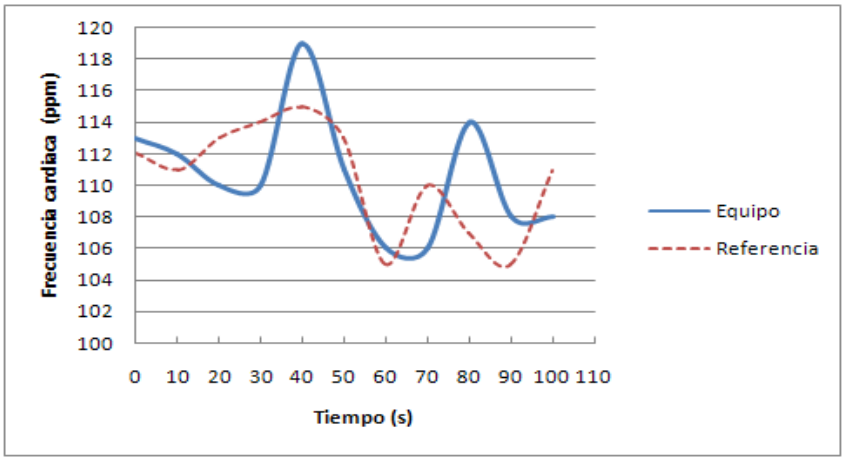

Fig. 18 Gráfica de la frecuencia cardiaca datos de la tabla 4.

Tabla 5 Valores tomados el 03/03/09 a las 11:00 am con un intervalo de tiempo de 10 segundos.

Tiempo (s)	Saturación equipo (%)	Frecuencia cardiaca equipo (PPM)	Saturación referencia (%)	Frecuencia cardiaca referencia (PPM)	Diferencia saturación	Diferencia frecuencia cardiaca
0	96	109	96	106	0	-3
10	95	110	96	111	1	1
20	95	111	96	108	1	-3
30	95	109	96	107	1	-2
40	96	106	95	109	-1	3
Tiempo	Saturación	Frecuencia	Saturación	Frecuencia	Diferencia	Diferencia

(s)	equipo (%)	cardiaca equipo (PPM)	referencia (%)	cardiaca referencia (PPM)	saturación	frecuencia cardiaca
50	96	109	96	112	0	3
60	94	113	96	108	2	-5
70	97	108	96	110	-1	2
80	96	113	95	110	-1	-3
90	95	110	95	110	0	0
100	96	107	95	110	-1	3

Continuación de la tabla 5

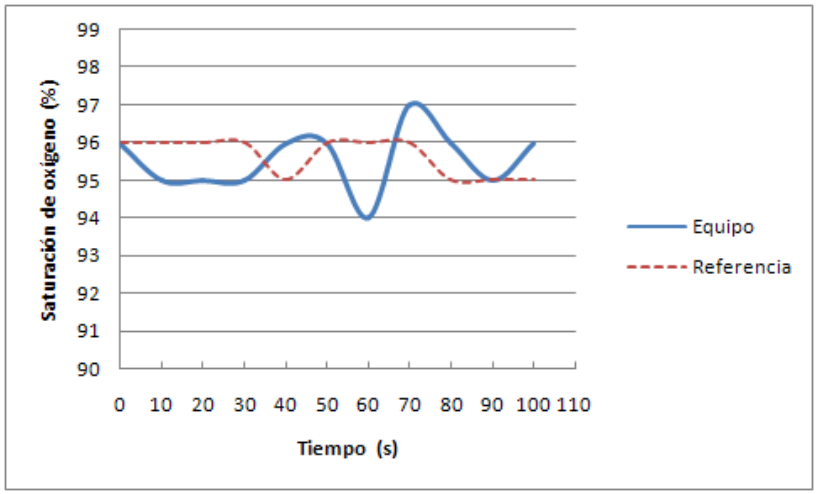

Fig. 19 Gráfica de la saturación de oxígeno datos de la tabla 5.

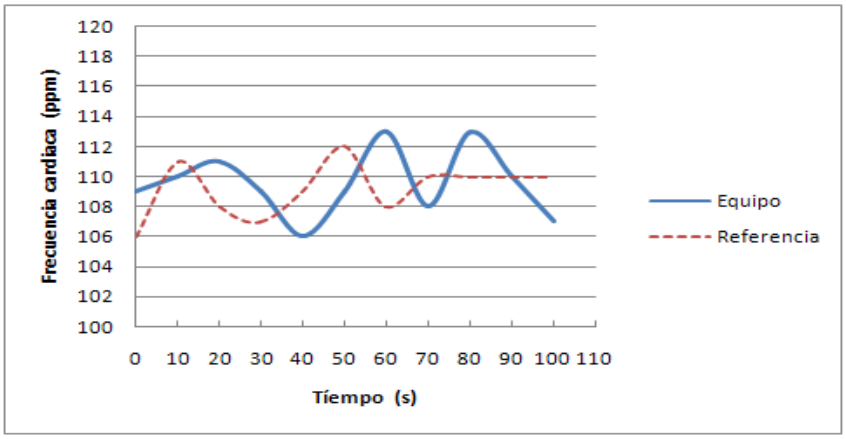

Fig. 20 Gráfica de la frecuencia cardiaca datos de la tabla 5.

La saturación medida por el equipo sigue la tendencia de la saturación medida por la referencia, lo cual demuestra que los resultados coinciden presentando un error máximo de ±3% (precisión) con respecto a la referencia en el valor de la saturación, aunque los valores medidos por en el equipo presentan una variabilidad mayor durante el tiempo de evaluación que la referencia. Muchos factores contribuyen a las diferencias, donde los más importantes son el nivel de ruido y las curvas de calibración. El equipo no está dotado con procesamiento digital de señales y la curva para determinar la saturación es la ecuación (15) la cual es no lineal.

El valor medido por el equipo para la frecuencia cardiaca tiene un error máximo ±7% (precisión).

El rango del equipo para la saturación es de 0% a 100%, y para la frecuencia cardiaca es de 30ppm a 240ppm.

La resolución para la saturación es de 1%, y para la frecuencia cardiaca es de 1ppm.

El costo aproximado del equipo es de 850 Bsf.

CONCLUSIONES

Los resultados demostraron que si es posible construir un equipo oxímetro de pulso, el equipo detectó en el intervalo de tiempo de la medición la saturación de oxigeno, con un error de ±2% con respecto al valor de una referencia.

Los niveles de ruido en la medición son elevados debido a que la señal pulsátil es extremadamente pequeña comparada con los niveles constantes detectados, esto hace necesaria una elevada ganancia, por lo tanto reducir el nivel de ruido constituyó una tarea fundamental.

El sensor y el amplificador operacional de muy bajo ruido fueron los más críticos en el diseño, en el mercado local solamente hay un único modelo de fotodiodo que abarca el espectro de la luz visible al infrarrojo el BPW34.

En Venezuela existe una gran dependencia tecnológica en el campo de los equipos médicos electrónicos, las actividades de desarrollo, comercialización y servicio técnico de estos equipos se encuentran en manos de pocas empresas, ubicadas en los países de avanzado nivel tecnológico especialmente en EEUU, Alemania y Japón. La implementación y construcción de equipos similares al del presente trabajo, representa un esfuerzo para disminuir esta dependencia, por eso es de gran importancia que se generen líneas de investigación y desarrollo que traten de subsanar este problema.

REFERENCIA BIBLIOGRÁFICA

[1] John W. Severinghaus, Takuo Aoyagi: Discovery of Pulse Oximetry, (Artículo en linea), http://www.anesthesia-analgesia.org/cgi/reprint/105/6SSuppl /S1.pdf , 2007. p. s1.

[2] Ley de Beer-Lambert, (Artículo en linea) Wikipedia http://www. en.wikipedia.org/wiki/Beer's_law

[3] Fundamentals of Tissue Optics www.medphys.ucl.ac.uk/research/borg/homepages/gbranco/Chapter%202%20Fundamentals%20of%20Tissue%20Opti...

[4] Paul D. Mannheimer, The Light–Tissue Interaction of Pulse Oximetry, (Artículo en linea) http://www.anesthesia-analgesia.org/cgi/reprint/ 105/6SSuppl/ S10. Pdf, 2007. p. s11.

[5] Hewlett-Packard Journal, A New Family of Sensors for Pulse, (Artículo en linea) http://www.hpl.hp.com/hpjournal/97feb/feb97a7.pdf

[6] Optical Absorption of Hemoglobin, (Artículo en línea) http://www.imt.liu.se/edu/courses/TBMT36/artiklar/4_Tissue_prop/hemoglobin_spectra.pdf

[7] Siemens BPW 34, (Manual técnico) http://pdf1.alldatasheet.net/datasheet-pdf/view/44653/SIEMENS/BPW34.html

[8] XLMR53WDW (Manual técnico) http://www.datasheet.digchip.com/463/463-01712-0-XLMR53WDW.pdf[9] Microchip PIC16F917, (Manual técnico) http://ww1.microchip.com/downloads /en/DeviceDoc/ 41250E.pdf

BIBLIOGRAFÍA

Horowitz, Paul. The art of electronics, 2da edición. Cambridge, Cambridge University Press, 1989.

Milá de la Roca, José. Diseño de equipos electrónicos, Caracas: Editorial MIL, 2001.

Navarro, Héctor. Instrumentación electrónica moderna, Caracas: Editorial Innovación Tecnológica-Facultad de Ingeniería Universidad Central de Venezuela, 1995.

Clinical Impact of LED Performance in Pulse Oximetry. [Manual técnico] Disponible: < www.electro.co.uk/pdfs/clinical_implications.pdf> [Consulta: enero 2008].

John W. Severinghaus, Takuo Aoyagi: Discovery of Pulse Oximetry. [Artículo] Disponible:<http://www.anesthesia-analgesia.org/cgi/reprint/105/6SSuppl/S1.pdf > [Consulta: enero 2008].

Hamamtsu, photodiode technical information. [Manual técnico] Disponible: <www.sales.hamamatsu.com/assets/applications/SSD/photodiode_technical_information.pdf > [Consulta: enero 2008].

Hewlett-Packard Journal, A New Family of Sensors for Pulse [Artículo técnico] Disponible: <www.hpl.hp.com/hpjournal/97feb/feb97a7.pdf> [Consulta: enero 2008].

Hewlett-Packard Journal, Volunteer Study for Sensor Calibration. [Artículo técnico] Disponible:<www.hpl.hp.com/hpjournal/97feb/feb97a7a2.pdf> [Consulta: enero 2008].

Microchip, Amplifying High-Impedance Sensors – Photodiode Example. [Artículo técnico] Disponible: <www1.microchip.com/downloads/en/AppNotes/ 00951a.pdf > [Consulta: enero 2008].

Paul D. Mannheimer, The Light–Tissue Interaction of Pulse Oximetry. [Artículo] Disponible: < http://www.anesthesia-analgesia.org/cgi/reprint/ 105/6SSuppl/ S10. pdf> [Consulta: enero 2008].

PerkinElmer, Optoelectronics Photodiodes, Phototransistors and Infrared Emitters. [Manual técnico] Disponible: <http://www.optoelectronics.perkinelmer.com/ content/RelatedLinks/Brochures/BRO_PhotodiodesPhototransistorsAndIREDs.pd f > [Consulta: enero 2008].

Philips Medical Systems, Understanding Pulse Oximetry SpO2 Concepts. [Artículo] Disponible: <http://www.incenter.medical.philips.com/doclib/getdoc. aspx?func=ll&objid=586458&objaction=open> [Consulta: enero 2008]

Scott Prahl, Optical Absorption of Hemoglobin. [Artículo] Disponible: <http:// www .omlc.ogi.edu/spectra/hemoglobin/index.html [Consulta: enero 2008].

[ANEXO nº 1]

LISTADO DE CÓDIGO DEL PROGRAMA GRABADO EN EL MICROCONTROLADOR.

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//  Oximetro V1.1
//  Programa desarrollado para el microcontrolador PIC 16F916 por Gustavo Olivero Ortiz
//  como parte del trabajo especial de grado para optar por el título de ingeniero
//  electricista en la Facultad de Ingeniería de la Universidad Central de Venezuela.
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
#include        <16f916.h>
#device         ADC=10  *=16
#fuses NOWDT,NOPROTECT,HS
#use    delay (clock=20000000)
#define         canal_DC_red    0       //Pin 2 AN0
#define         canal_DC_ir     1       //Pin 3 AN1
#define         canal_AC_red    2       //Pin 4 AN2
#define         canal_AC_ir     4       //Pin 7 AN4
#bit    RP1=0x03.6
#bit    RP0=0x03.5
#bit    W_R=0x07.7                      //Señal de escritura "1" y lectura "0".
#bit    TOIF=0x0b.2                      //Bandera de overflow del timer0.
#bit    TMR1IF=0x0C.0                   //Bandera de overflow del timer0.
#bit    CS_RAM=0x07.2                   //Señal habilitación de la RAM.
#bit    CS_373=0x07.4                   //Señal h  abilitación del lacht de direcciones.
#bit    A8=0x07.0                       //Bit de dirección.
#bit    A9=0x07.1                       //Bit de dirección.
#bit    A10=0x07.3                      //Bit de dirección.
#bit    ADFM=0x1f.7                     //Bit de
#bit    CONVERSION=0x1f.1               //Bandera comienzo de conversión.
#byte   ADRESH=0x1e
#byte   OPTION=0x81                     //Configuración del timer0 y prescaler.
#byte   TRISA=0x85
#byte   TRISB=0x86
#byte   TRISC=0x87
#byte   PORTA=0x05
#byte   PORTB=0x06
#byte   PORTC=0x07
#byte   TMR0=0x1
//Designaciones para manejo de la pantalla LCD.
#bit    E=0x07.6                        //Señal habilitación de la LCD.
#bit    RS=0x07.3                       //Bit de data in/out de la LCD.
#bit    WR=0x07.7                       //Bit lectura escritura en la LCD.
#bit    CS1=0x07.1                      //Seleccionador de la primera mitad de la pantalla.
#bit    CS2=0x07.0                      //Seleccionador de la segunda  mitad de la
                                        //pantalla.
#bit    DB0=0x06.0
#bit    DB1=0x06.1
#bit    DB2=0x06.2
#bit    DB3=0x06.3
#bit    DB4=0x06.4
#bit    DB5=0x06.5
#bit    DB6=0x06.6
#bit    DB7=0x06.7
#list

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//Declaración de variables globales
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
```

```
signed int      cociente1;
signed long     resto=0;
short   comenzar_calculos=0;        //Bandera ("1") para salir de interrupción del timer1.
short   estableciendo_PWM=1;        //Bandera ("1") para establecer el duty cicle  del PWM.
short   guardar_CS1;
short   guardar_CS2;
short   guardar_RS;
short   strobe;
short   m;
short   m1;
short   m2;
short   m3;
int     i;
int     j;
int     n;
int     suelta_tecla=0;
int     numero_paso=0;
int     DUTY=128;                   //Variable ciclo de trabajo del PWM.
int     lineas_limpiar=8;
int     dato_anterior_pwm;
int     dato_anterior;
int     ac_red_anterior;
int     ac_ir_anterior;
int     ac_red=0;                   //Valor leido de la componente AC rojo.
int     ac_ir=0;                    //Valor leido de la componente AC ir.
int     dc_red=0;                   //Valor leido de la componente DC rojo.
int     dc_ir=0;                    //Valor leido de la componente DC ir.
int     xmax_dc_red;
int     xmax_ac_red;
int     xmax_dc_ir;
int     xmax_ac_ir;
int    xmin_ac_red=0;
int    xmin_ac_ir;
int     num_ciclos=0;
int     artefactos=0;
int     tranferir_puertoB;          //Variable para guardar valor hacia el puertoB.
int     n_intentos_pwm=0;
int     num_muestras;
int     guardar_n;
int     promedio_alto;
int     tecla_pulsada_anterior=0;
int     tecla_pulsada_uno;          //valor de la tecla pulsada primera vez.
int     tecla_pulsada_dos;          //valor de la tecla pulsada segunda vez.
int     contador_bandera=0;
int     muestra_sp_alta=99;         //alarma de  oxigenación alta
int     muestra_sp_baja=80;         //alarma de oxigenación baja.
int     muestra_ppm_baja=40;        //alarma de ritmo cardiaco bajo.
int     muestra_ppm_alta=130;       //alarma de ritmo cadiaco alto.
int     muestra_tiempo=150;
int     offset_memoria=0;           //Off-set de la dirección de memoria RAM.
int     promedio_ac_ir=0;
int     promedio_dc_ir=0;
int     promedio_dc_red=0;
float   saturacion_2=0;
float   guardar_saturacion;
float   saturacion=0;
float   numerador;
float   denominador;
float   ppm=0;                      //número de pulsos cardiacos por minutos.
float   ppm_2=0;
long    guardar_muestras;
long    suma_ac_ir=0;
long    suma_dc_ir=0;
long    suma_dc_red=0;
```

```
long    suma;
long    puntero_memoria=0;                      //Puntero hacia la direccion de momoria.
char    spo2_visual[4]="000";
char    ppm_visual[4]="000";
char    conta[4]="000";
const   int   cero88[6]={124,162,146,138,124,0};
const   int   uno88[6]={0,8,4,254,0,0};
const   int   dos88[6]={132,194,162,146,140,0};
const   int   tres88[6]={66,130,138,150,98,0};
const   int   cuatro88[6]={48,40,36,254,32,0};
const   int   cinco88[6]={78,138,138,138,114,0};
const   int   seis88[6]={120,148,146,146,96,0};
const   int   siete88[6]={2,226,18,10,6,0};
const   int   ocho88[6]={108,146,146,146,108,0};
const   int   nueve88[6]={12,146,146,82,60,0};
const   int   cero[26]={248,252,254,254,15,7,7,15,254,254,252,248,0,31,63,127
,127,240,224,224,240,127,127,63,31,0};
const   int   uno[26]={0,0,16,24,24,28,30,30,255,255,255,255,0,0,0,0,0,0,0,0
,255,255,255,255,0};
const   int   dos[26]={56,60,62,30,15,7,7,143,254,254,252,248,0,192,224,240,248
,252,254,255,239,231,227,225,224,0};
const   int   tres[26]={56,60,62,14,143,135,199,239,254,252,60,24,0,28,60,124
,112,241,225,227,247,127,127,60,24,0};
const   int   cuatro[26]={0,0,192,240,252,255,63,143,131,128,0,0,0,12,15,15,15
,15,14,14,255,255,255,14,14,0};
const   int   cinco[26]={255,255,255,255,231,119,119,247,231,231,199,135,0,25,57
,121,121,240,224,224,240,127,127,63,31,0};
const   int   seis[26]={248,252,254,254,143,199,199,207,158,158,28,24,0,31,63
,127,127,243,225,225,243,127,127,63,30,0};
const   int   siete[26]={7,7,7,7,135,199,231,247,127,63,31,7,0,192,240,252,255
,63,15,3,0,0,0,0,0,0};
const   int   ocho[26]={24,60,126,254,239,199,199,239,254,126,60,24,0,24,60,126
,127,247,227,227,247,127,126,60,24,0};
const   int   nueve[26]={120,252,254,254,207,135,135,207,254,254,252,248,0,24,56
,121,121,243,227,227,241,127,127,63,31};
const   int   sat[28]={144,64,32,144,0,32,80,80,144,0,224,80,80,224,0,0,0,0,0
,113,137,137,112,0,73,168,168,145};
const   int   pulso[28]={240,80,80,32,0,240,80,80,32,0,0,0,0,0,249,16,32,16,248
,1,0,0,0,0,0,0,0,0};
const   int   configuracion[65]={124,130,130,68,0,124,130,130,124,0,254,12,48
,192,254,0,254,18,18,2,0,2,254,2,0,124,130,146,100,0,254,128,128,254,0,254,50,82
,140,0,252,34,34,252,0,124,130,130,68,0,2,254,2,0,124,130,130,124,0,254,12,48
,192,254,255};
const   int   spo2_alta[44]={76,146,146,100,0,254,18,18,12,0,124,130,130,124,0
,132,194,162,146,140,16,16,0,252,34,34,252,0,254,128,128,128,0,2,2,254,2,2,0,252
,34,34,252,255};
const   int   spo2_baja[43]={76,146,146,100,0,254,18,18,12,0,124,130,130,124,0
,132,194,162,146,140,16,16,0,254,146,146,108,0,252,34,34,252,0,64,128,128,126,0,
252,34,34,252,255};
const   int   ppm_baja[39]={254,18,18,12,0,254,18,18,12,0,254,4,24,4,254,0,16,16
,0,254,146,146,108,0,252,34,34,252,0,64,128,128,126,0,252,34,34,252,255};
const   int   ppm_alta[40]={254,18,18,12,0,254,18,18,12,0,254,4,24,4,254,0,16,16
,0,252,34,34,252,0,254,128,128,128,0,2,2,254,2,2,0,252,34,34,252,255};
const   int   tiempo[30]={2,2,254,2,2,0,2,254,2,0,254,146,146,130,254,4,24,4,254
,0,254,18,18,12,0,124,130,130,124,255};
const   int   salvar[31]={76,146,146,100,0,252,34,34,252,0,254,128,128,128,0,14
,48,192,48,14,0,252,34,34,252,0,254,50,82,140,255};
const   int   salir[24]={76,146,146,100,0,252,34,34,252,0,254,128,128,128,0,2
,254,2,0,254,50,82,140,255};
const   int   esperar[41]={254,146,146,130,0,76,146,146,100,0,254,18,18,12,0,254
,146,146,130,0,254,50,82,140,0,252,34,34,252,0,254,50,82,140,0,0,128,0,128,0,128};
```

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//    Declaración de prototipos de funciones.
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void   digitalizacion();              // convierte a digital la 4 señales analogicas.
void   guardar_adc_ram();             // guarda los 4 valor digitalizados en la ram externa.
void   max_min();                     // busca el valor mínimo y máximo de todos los 250
                                      //valores muestreados.
void   calculo_spo2_ppm();            // calcula  la oxigenación y el ritmo cardiaco.
void   on_lcd();                      // enciende la región de la pantalla especificada por
                                      // cada controlador.
void   limpiar_lcd();                 // escribe 0 en todos los byte ram  de la pantalla.
void   escribir_lcd();                // muestra el byte de entrada en la pantalla.
void   set_page_X();                  // coloca la pagina de registros de  pantalla.
void   set_Y_addres();                // coloca la dirección del byte mostrado en pantalla.
void   set_star_line();               // coloca la posicion del puntero en la pantalla.
void   escribir_datos_lcd();          // muestra  en  la parte  superior de la pantalla el
                                      // valor de la oxigenacion
                                      //y del ritmo cardiaco.
void   mostrar_grafica();             // grafica la señal de la componente AC del rojo.
void   escribir_menu();               // muestra toda la configuración de las alarmas.
void   delay_escritura_lcd();         //  genera el diagrama de tiempo para la escritura
                                      //de datos o de instrucciones en la pantalla.
void   establecer_PWM();              //funcion que regula la intensidad luminosa de los led.

void   filtrado_pasa_bajo();

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////        Cuerpo principal del programa.                ///////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void   main()
        {
        delay_ms(1000);               //Espera para garantizar el reset en la LCD.
   setup_lcd(LCD_DISABLED);           //
        set_tris_c(0);                //Puerto C como salidas.
        set_tris_a(0xff);
        CS_RAM=0;                     //Desabilitación de la memoria RAM.
        WR=0;
        CS_373=0;
        A8=0;
        A9=0;
        A10=0;
        E=0;

// Las  instrucciones  siguientes son para garantizar
//que el buzzer este apagado cuando ocurra un reset del micro.
        set_tris_b(0xfe);
        DB0=0;
        delay_us(10);
        CS_373=1;
        A10=1;
        E=1;
        delay_us(10);
        E=0;
        CS_373=0;
        A10=0;
        set_tris_b(0xff);

//Instrucciones de inicializacion de la LCD.
        CS1=0;
        CS2=1;
```

```
        on_lcd();
        tranferir_puertoB=0;
        set_star_line();
        tranferir_puertoB=0;
        set_Y_addres();
        tranferir_puertoB=0;
        set_page_X();
        lineas_limpiar=8;
        limpiar_lcd();
        CS1=1;
        CS2=0;
        on_lcd();
        tranferir_puertoB=0;
        set_star_line();
        tranferir_puertoB=0;
        set_Y_addres();
        tranferir_puertoB=0;
        set_page_X();
        for(i=0;i<=40;++i)                          //Escribe mensaje de "esperar".
                {
                tranferir_puertoB=esperar[i];
                escribir_lcd();
                }

//Control de la luminosidad de los led con el PWM.
        DUTY=read_eeprom(0);                        //el primer duty cicle es el último guar-
                                                    //dado antes de apagarse el equipo.
        setup_timer_2(T2_DIV_BY_1,128,1);           //frecuencia del PWM en 78.12kHz.
        setup_ccp1(CCP_PWM);                        //módulo CCP1 como PWM.
        set_pwm1_duty(DUTY);                        //colocación del duty cicle en el pwm.
        delay_ms(2500);

//Configuracion del convertidor analogico digital.
        setup_adc_ports(sAN0|sAN1|sAN2|sAN4|VSS_VREF);
        setup_adc(ADC_CLOCK_DIV_32);
        ADFM=0;

//configuración del timer1 para establecer la base de tiempo del muestreo de
//las señales analogicas.
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8 );//Configura el timer1 para  frecuencia
        set_timer1(59286);                          // de  muestreo 100Hz.


//configuración del timer0 como contador de un pulso para detectar una tecla
//pulsada y entrar a la rutina de interrupcion de servicio.
        SETUP_TIMER_0(RTCC_EXT_H_TO_L|RTCC_DIV_1);
        set_timer0(255);                    //se alcanza el overflow del timer0 con
                                            //un  pulso el cual se da al pulsar una tecla.
//habilitación de las interrupciones.
        enable_interrupts(INT_TIMER0);      // habilita la interrupción del timer0.
        enable_interrupts(INT_TIMER1);      //habilita interrupcion timer1
        enable_interrupts(GLOBAL);          //desenmascara las interrupciones.
        dato_anterior_pwm=0;
        n_intentos_pwm=0;
        num_ciclos=0;
        estableciendo_PWM=1;

//el siguiente bucle infinito espera  a que se  complete el numero  total de
//muestras de las señales analogicas para actualizar la pantalla lcd a menos que
//ocurra una interrupcion por teclado.
        do
                {
                if (comenzar_calculos==1)
        {
```

```
            max_min();                          //Calcular el máximo y mínimo absoluto.
            offset_memoria=0;
            comenzar_calculos=0;                //Desabilitar la bandera de realizar cálculos.
            calculo_spo2_ppm();                 //Cálculo de saturación y del ritmo cardiaco.
                        if (artefactos<10)
                                {
                                numero_paso++;          //incrementar contador.
                                ppm_2=ppm_2+ppm;        //Sumatoria de frecuencia
                                                        //cardiaca en cada paso.
                                saturacion_2=saturacion_2+saturacion;
                                //Sumatoria de saturación en cada paso.
                                if(numero_paso==3)
                                        {
                                        ppm=ppm_2/3;    //Promedio de los tres pasos de
                                                        // cálculo.
                                        saturacion=saturacion_2/3;//Promedio de los tres
                                                                  // pasos de cálculo.
                                        sprintf(spo2_visual,"%04.2f",saturacion);
                                        sprintf(ppm_visual,"%03.2f",ppm);
                                        escribir_datos_lcd();

                                        numero_paso=0;
                                        //se reinician las variables utilizadas.
                                        saturacion_2=0;
                                        ppm_2=0;
                                        muestra_sp_alta=read_eeprom(1);
                                        //Se leen los valores de las alarmas
                                        muestra_sp_baja=read_eeprom(2);
                                        //guardadas en la memoria EEPROM.
                                        muestra_ppm_alta=read_eeprom(3);
                                        muestra_ppm_baja=read_eeprom(4);
                                        //Si los valores de la saturación y del ritmo
                                        //cardiaco estan fuera del
                                        //intervalo determinado por los valore de las
                                        //alarmas se activa el buzzer.

if((saturacion>muestra_sp_alta)||(saturacion<muestra_sp_baja)||
  (ppm>muestra_ppm_alta)||(ppm<muestra_ppm_baja))
                                                        {
                                                        set_tris_b(0xfe);
                                                        DB0=1;
                                                        delay_us(10);
                                                        CS_373=1;
                                                        A10=1;
                                                        E=1;
                                                        delay_us(10);
                                                        E=0;
                                                        CS_373=0;
                                                        A10=0;
                                                        set_tris_b(0xff);
                                                        }
                                        }
                                }
            mostrar_grafica();
            setup_timer_1(T1_INTERNAL|T1_DIV_BY_8 );
                        set_timer1(59286);          //Configura prescaler del timer1.
            enable_interrupts(INT_TIMER1);
            enable_interrupts(GLOBAL);
                    artefactos=0;
            }
            }
                    while (TRUE);
    }
```

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Implementación de funciones.
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
// función que toma valores de las 4 señales analógicas y los digitaliza.
//guardandolos en las respectivas variables.
////////////////////////////////////////////////////////////////////////////
void   digitalizacion()
        {
        set_tris_a(0xff);
        for (i=1;i<=4;++i)
                {
                // Sentencia suiche para establecer el canal de muestreo de los
                //cuatro disponibles.
                switch (i)
                        {

                        case 1: set_adc_channel(canal_DC_red); break;
                        case 2: set_adc_channel(canal_DC_ir);  break;
                        case 3: set_adc_channel(canal_AC_red); break;
                        case 4: set_adc_channel(canal_AC_ir);  break;
                        }
                delay_us(50);

                // Sentencia suiche para realizar la conversión de cada muestra.
                //se comienza la conversión y se lee el bit de conversión en progreso
                //hasta que el bit se vuelve cero.
                switch (i)
                        {
                        case 1:
                                {
                                CONVERSION=1;
                                do
                                        delay_us(10);
                                while (CONVERSION==1);
                                dc_red=ADRESH;
                                }break;
                        case 2:
                                {
                                CONVERSION=1;
                                do
                                        delay_us(10);
                                while (CONVERSION==1);
                                dc_ir=ADRESH;
                                }break;
                        case 3:
                                {
                                CONVERSION=1;
                                do
                                        delay_us(10);
                                while (CONVERSION==1);
                                ac_red=ADRESH;
                                }break;
                        case 4:
                                {
                                CONVERSION=1;
                                do
                                        delay_us(10);
                                while (CONVERSION==1);
                                ac_ir=ADRESH;
```

```
                                        }break;
                                    }
                                }
                            }

    ////////////////////////////////////////////////////////////////////////////////
    //función que escribe o lee una posicion de memoria en ram externa.
    ////////////////////////////////////////////////////////////////////////////////
    void   guardar_leer_adc_ram()
      {
      guardar_CS1=CS1;
      guardar_CS2=CS2;
      guardar_RS=RS;
      for (i=1;i<=4;++i)
        {
        set_tris_b(0);
        switch (i)
          {
          case 1:{
            puntero_memoria=(0+(long)offset_memoria);    //direcciones de los valores DC del
             PORTB=(int)(puntero_memoria);                                    //rojo.
            } break;
          case 2:{
            puntero_memoria=(250+(long)offset_memoria);//direcciones de los valores DC del
               PORTB=(int)(puntero_memoria);                          //infrarrojo.
            } break;
          case 3:{
            puntero_memoria=(500+(long)offset_memoria);//direcciones de los valores AC del
              PORTB=(int)(puntero_memoria);                          // rojo.
            } break;
          case 4:{
            puntero_memoria=(750+(long)offset_memoria);//direcciones de los valores AC del
              PORTB=(int)(puntero_memoria);                          //infrarrojo.
            } break;
          }
        CS_373=1;
        A8=bit_test(puntero_memoria,8);
        A9=bit_test(puntero_memoria,9);
        A10=bit_test(puntero_memoria,10);
        CS_373=0;
        if (W_R==0)
// Instrucciones para guardar las
          {
            //muestras en la memoria RAM.
          set_tris_b(0);
          switch (i)
            {
            case 1:   PORTB=dc_red;     break;
            case 2:   PORTB=dc_ir;      break;
            case 3:   PORTB=ac_red;     break;
            case 4:   PORTB=ac_ir;      break;
            }
          CS_RAM=1;
          delay_us(10);
          CS_RAM=0;
                    }
        set_tris_b(0xff);
        if (W_R==1)                     // Instrucciones para leer las muestras
          {                             //guardadas en la memoria RAM.
          CS_RAM=1;
          delay_us(10);
          switch (i)
            {
            case 1:   dc_red=PORTB;     break;
```

```
            case 2:   dc_ir=PORTB;       break;
            case 3:   ac_red=PORTB;      break;
            case 4:   ac_ir=PORTB;       break;
            }
          }
        CS_RAM=0;
        }
      set_tris_b(0xff);
      CS1=guardar_CS1;
      CS2=guardar_CS2;
      RS=guardar_RS;
      }

    ////////////////////////////////////////////////////////////////////////////////
    //Función que calcula los valores máximos y mínimos absolutos en las señales.
    ////////////////////////////////////////////////////////////////////////////////
    void   max_min()
    {
//  Las instrucciones siguientes son
            W_R=1;
//de inicialización de variables.
            suma_ac_ir=0;
            suma_dc_ir=0;
            suma_dc_red=0;
            artefactos=0;
            offset_memoria=0;
            xmax_dc_red=0;
            xmax_ac_red=0;
            xmax_dc_ir=0;
            xmax_ac_ir=0;
            xmin_ac_red=255;
            xmin_ac_ir=255;
            // Lazo para comparar el valor de la muestra anterior con el
            //valor actual y determinar el valor máximo y mínimo absoluto
            for (n=0;n<=249;++n)
                        {
                        guardar_leer_adc_ram();
                        if (xmax_dc_red<dc_red)
                                    xmax_dc_red=dc_red;
                        if (xmax_ac_red<ac_red)
                                    xmax_ac_red=ac_red;
                        if (xmax_dc_ir<dc_ir)
                                    xmax_dc_ir=dc_ir;
                        if (xmax_ac_ir<ac_ir)
                                    xmax_ac_ir=ac_ir;
                        if (xmin_ac_red>ac_red)
                                    xmin_ac_red=ac_red;
                        if (xmin_ac_ir>ac_ir)
                                    xmin_ac_ir=ac_ir;
                        offset_memoria++;
                        suma_ac_ir=suma_ac_ir+(long)ac_ir;  //Sumatoria de muestras pulsatiles
                                                            //infrarrojas.
                        suma_dc_ir=suma_dc_ir+(long)dc_ir;  //Sumatoria de muestras constantes
                                                            //infrarrojas.
                        suma_dc_red=suma_dc_red+(long)dc_red;//Sumatoria de muestras constantes
                                                            //rojas.
                        if ((ac_ir>254)||(ac_ir<5))         //Se detecta cuantas veces se llega al
                                                            //nivel
                                    artefactos++;           //límite máximo de digitalización.
                        }
            offset_memoria=0;
            set_tris_b(0xff);
      }
```

```
////////////////////////////////////////////////////////////////////////////////
//Función para clacular la saturacion relativa de oxigeno (spo2) y el
//numero de latidos por minutos(PPM)
////////////////////////////////////////////////////////////////////////////////

//  Para encontrar el periodo de cada pulso  se usan las muestras correspondientes a la
//señal infarroja, se determinan dos punto para  el cálculo: un punto bajo que es igual
//al  promedio de las muestras  y  uno alto que es  el punto  medio entre  el  valor
//promedio y el máximo absoluto de las muestras. Se realiza el conteo del número de
//muestras que hay entre dos puntos altos consecutivos.

void   calculo_spo2_ppm()
        {
                //  Las instrucciones siguientes son
        i=0;
        //de inicialización de variables.
        j=0;
        m=0;
        m1=0;
        m2=0;
        m3=0;
        W_R=1;
        strobe=0;
        num_ciclos=0;
        num_muestras=0;
        guardar_muestras=0;
        guardar_n=0;
        ac_red_anterior=0;
        ac_ir_anterior=0;
        saturacion=0;
        numerador=0;
        denominador=1;
        guardar_saturacion=0;
        max_min();
        // Cálculo de máximo y mínimo absoluto.
        promedio_ac_ir=suma_ac_ir/250;  // Se establece como punto bajo.
        promedio_dc_ir=suma_dc_ir/250;
        promedio_dc_red=suma_dc_red/250;
        ac_red_anterior=promedio_ac_ir;
        ac_ir_anterior=promedio_ac_ir;

        suma=((long)promedio_ac_ir+(long)xmax_ac_ir);// Se determina el punto alto.
        suma=suma/2;
        promedio_alto=suma;                             // Se establece como punto alto.
        for(n=0;n<=249;++n)
                {
                offset_memoria=n;                       //  Se leen los enésimos datos corres-
                guardar_leer_adc_ram();                 //pondientes al paso de cada iteración.

                if((ac_ir<promedio_ac_ir)&&(m1==0))//  Se espera que la señal pase por
                        {                               //el punto bajo
                        m1=1;
                        }
                if((ac_ir>promedio_alto)&&(m1==1)) // Comienza el conteo de las muestras.
                        {                               //para cálcular el periodo.
                        if((m3==1))                     // Se detecta si se cumplió
                                {                       //con un periodo.

                                if(((num_muestras-guardar_muestras)>25)&&
                                        ((num_muestras-guardar_muestras)<200))
                                        //  Si el periodo calculado no
                                        {
```

```
                                        num_ciclos++;
                                        //Se incrementa contador de número
                                        //de periodos
                                        guardar_muestras=num_muestras;
                                        //Se guarda el contador de muestras.
                                        guardar_saturacion=saturacion;
                                        //Se guarda la sumatoria correspondiente
                                        }           //a la razón.
                                else
                                        {
                                        //  Si no se cumplen las condiciones
                                        //anteriores se desechan las
                                        num_muestras=guardar_muestras;
                                        saturacion=guardar_saturacion;
                        }
                                //actualizaciones y se reinicia con
                                //los valores anteriore.
                                m2=0;
                                m3=0;
                                }
        //   En cada iteración se  calcula  el valor instantaneo  de  la  razón que nos
        //determinará la saturación y con estos valores se va realizando una sumatoria.

                        if(ac_ir!=ac_ir_anterior)       //  Condición para que el denominador
                                {                       //sea diferente de cero.
                                resto=(ac_red-ac_red_anterior);
                                //  Cálculo del numerador de la razón.
                                numerador=(float)promedio_dc_ir*(float)resto;
                                resto=(ac_ir-ac_ir_anterior);
                                //Cálculo del denominador de la razón.
                                denominador=(float)promedio_dc_red*(float)resto;
                                }

                        numerador=numerador/denominador;  //  Cómputo de la razón.
                        saturacion=saturacion+numerador;      //  sumatoria de la razón.
                        num_muestras++;             //  Conteo de los tiempos de muestreo
                                                    //para hallar el periodo.
                        ac_red_anterior=ac_red;     //Se guardan los valores actuales de
                                                    //los datos de la componente AC
                        ac_ir_anterior=ac_ir;       //para ser usadas en el próximo paso
                                                    //de análisis.
                        if (ac_ir<promedio_ac_ir)   //  Se detecta si la señal es menor al
                                m2=1;               //punto bajo.

                        if ((ac_ir>promedio_alto)&&(m2==1))
                        //Se detecta si la  señal subió el punto alto.
                                {
                                m3=1;
                                guardar_n=n;
                                //Se guarda el valor del contador de /uestras.
                                }
                        }
                }
//  Cómputo del promedio de la razón de las componentes.
saturacion=guardar_saturacion/((float)(guardar_muestras));
//  Cómputo del la saturación dependiente de la razón.
saturacion=((saturacion*722.8-3226.5)/(saturacion*(-49.28)-2906.9))*100;
//  Cómputo del promedio del número de periodos de muestreo.
numerador=(float)guardar_muestras/(float)num_ciclos;
// Cómputo de la     frecuencia cardiaca en pulsos por minutos.
ppm=(float)60/(numerador*0.01);
if(saturacion>100)
        saturacion=100;
if(saturacion<0)
```

```
                saturacion=0;

  sprintf(spo2_visual,"%04.2f",saturacion);
  sprintf(ppm_visual,"%03.2f",ppm);
  set_tris_b(0xff);
  }

////////////////////////////////////////////////////////////////////////////////
//Función que regula la intensidad luminosa en los led atraves del nivel DC
//del PWM.
////////////////////////////////////////////////////////////////////////////////
void  establecer_PWM()
      {
      max_min();
      if((xmax_ac_red<230)&&(xmax_ac_ir<230)&&(xmin_ac_red>25)&&(xmin_ac_ir>25))
            {

            DUTY=DUTY+2;
            set_pwm1_duty(DUTY);
            }
      if((xmax_ac_red>250)||(xmax_ac_ir>250)||(xmin_ac_red<15)||(xmin_ac_ir<15))
            {
            DUTY=DUTY-2;
            set_pwm1_duty(DUTY);
            }
      if(dato_anterior_pwm==DUTY)
            num_ciclos++;
      if((DUTY>245)||(DUTY<10)||(n_intentos_pwm>64)||(num_ciclos>2))
            {
            estableciendo_PWM=0;
            dato_anterior_pwm=read_eeprom(0);
            if(dato_anterior_pwm!=DUTY)
                  write_eeprom(0,DUTY);
            }
   delay_ms(2500);
      mostrar_grafica();
      n_intentos_pwm++;
      dato_anterior_pwm=DUTY;
      }

////////////////////////////////////////////////////////////////////////////////
//funcion para encender  cualquiera delas dos mitades de la LCD..
////////////////////////////////////////////////////////////////////////////////

void  on_lcd()
  {
  WR=0;
  RS=0;
  set_tris_b(0);
  PORTB=0x3f;
  E=1;
  delay_us(50);
  E=0;
  delay_us(50);
  set_tris_b(0xff);
  }

////////////////////////////////////////////////////////////////////////////////
//Escribe en la LCD el dato presente en el puerto B del microcontrolador.
////////////////////////////////////////////////////////////////////////////////
void  escribir_lcd()
  {
  WR=0;
  RS=1;
```

```
  set_tris_b(0);
  PORTB=tranferir_puertoB;
  E=1;
  delay_us(50);
  E=0;
  delay_us(50);
  set_tris_b(0xff);
  }

////////////////////////////////////////////////////////////////////////////////
//Establece la fila de la pantalla  que fue previamente escojida.
////////////////////////////////////////////////////////////////////////////////
void   set_page_X()
  {
  WR=0;
  RS=0;
  set_tris_b(0);
  PORTB=tranferir_puertoB;
  DB7=1;
  DB6=0;
  DB5=1;
  DB4=1;
  DB3=1;
  E=1;
  delay_us(50);
  E=0;
  delay_us(50);
  set_tris_b(0xff);
  }

////////////////////////////////////////////////////////////////////////////////
//Habilita la columna especificada previamente.
////////////////////////////////////////////////////////////////////////////////
void   set_Y_addres()
  {
  WR=0;
  RS=0;
  set_tris_b(0);
  PORTB=tranferir_puertoB;
  DB7=0;
  DB6=1;
  E=1;
  delay_us(50);
  E=0;
  delay_us(50);
  set_tris_b(0xff);
  }

////////////////////////////////////////////////////////////////////////////////
//Actualiza el puntero hacia los registros de la pantalla.
////////////////////////////////////////////////////////////////////////////////
void set_star_line()
  {
  RS=0;
  WR=0;
  set_tris_b(0);
  PORTB=tranferir_puertoB;
  DB7=1;
  DB6=1;
  E=1;
  delay_us(50);
  E=0;
  delay_us(50);
  set_tris_b(0xff);
```

```
  }

////////////////////////////////////////////////////////////////////////////////
//Borra todos los registros  de la pantalla.
////////////////////////////////////////////////////////////////////////////////
void   limpiar_lcd()
  {
  for (n=(8-lineas_limpiar);n<=7;++n)
    {
    for (i=0;i<=127;++i)
      {
        if (i==0)
          {
          CS1=1;
          CS2=0;
          on_lcd();
          tranferir_puertoB=i;
          set_Y_addres();
          tranferir_puertoB=n;
          set_page_X();
          }
        if (i==64)
          {
          CS1=0;
          CS2=1;
          on_lcd();
          tranferir_puertoB=0;
          set_Y_addres();
          tranferir_puertoB=n;
          set_page_X();
          }
        tranferir_puertoB=0;
        escribir_lcd();
      }
    }
  }

////////////////////////////////////////////////////////////////////////////////
//Función que grafica la señal de la componente ac infrarroja.
////////////////////////////////////////////////////////////////////////////////
void   mostrar_grafica()
        {
        CS1=0;
        CS2=0;
        E=1;
        delay_us(10);
        E=0;
        delay_us(10);
        A10=0;
        delay_us(10);
        lineas_limpiar=6;
        limpiar_lcd();
        offset_memoria=0;
        for(n=0;n<=124;++n)
                {
        W_R=1;
        guardar_leer_adc_ram();
        numerador=ac_ir/48;
                cociente1=numerador;
                dato_anterior=48*cociente1;
        dato_anterior=ac_ir-dato_anterior;
        numerador=dato_anterior/6;
                resto=numerador;
        if(n==0)
```

```
                {
CS1=1;
CS2=0;
on_lcd();
}
        if(n==64)
                {
CS1=0;
CS2=1;
on_lcd();
                }
        if(n<64)
                {
                tranferir_puertoB=n;
set_Y_addres();
}
                else
        {
        tranferir_puertoB=n-64;
        set_Y_addres();
        }
        tranferir_puertoB=7-cociente1;
        set_page_X();
        j=tranferir_puertoB+1;
        if   (resto==0)
                tranferir_puertoB=0x80;//0x80//
        if   (resto==1)
                tranferir_puertoB=0xC0;//0x40//
        if   (resto==2)
                tranferir_puertoB=0xE0;//0x20//
        if   (resto==3)
                tranferir_puertoB=0xF0;//0x10//
        if   (resto==4)
                tranferir_puertoB=0xF8;//0x08//
        if   (resto==5)
                tranferir_puertoB=0xFC;//0x04//
        if   (resto==6)
                tranferir_puertoB=0xFE;//0x02//
        if   (resto==7)
                tranferir_puertoB=0xFF;//0x01//
        escribir_lcd();
        for(i=j;i<=7;++i)
                {
                if(n<64)
                        {
        tranferir_puertoB=n;
        set_Y_addres();
        }
                        else
                {
                tranferir_puertoB=n-64;
                set_Y_addres();
                        }
                tranferir_puertoB=i;
set_page_X();
tranferir_puertoB=0xFF;
escribir_lcd();
                }
        offset_memoria=offset_memoria+2;
        }
offset_memoria=0;
}
```

//

//Función que escribe en la pate superior de la pantalla los resultados de los cálculos.

```
////////////////////////////////////////////////////////////////////////////////
void   escribir_datos_lcd()
         {
         lineas_limpiar=8;
         limpiar_lcd();
         for(j=0;j<=1;++j)
                     {
         CS1=1;
         CS2=0;
         on_lcd();
         tranferir_puertoB=0;
         set_page_X();
         tranferir_puertoB=0;
         set_Y_addres();
         if(j==1)
                                    {
         CS1=0;
         CS2=1;
         on_lcd();
         tranferir_puertoB=0;
         set_Y_addres();
         tranferir_puertoB=0;
         set_page_X();
         }
       for (i=0;i<=2;++i)
        {
        if(j==0)
          dato_anterior=spo2_visual[i];
        else
          dato_anterior=ppm_visual[i];
        for (n=0;n<=25;++n)
          {
          if (dato_anterior=='.')
            {
            n=28;
            i=10;
            CS1=0;
            CS2=0;
            on_lcd();
            }
          if ((n==13)&&(i==0))
            {
            tranferir_puertoB=1;
            set_page_X();
            tranferir_puertoB=0;
            set_Y_addres();
            }
          if ((n==13)&&(i==1))
            {
            tranferir_puertoB=1;
            set_page_X();
            tranferir_puertoB=13;
            set_Y_addres();
            }
          if ((n==13)&&(i==2))
            {
            tranferir_puertoB=1;
            set_page_X();
            tranferir_puertoB=26;
            set_Y_addres();
            }
          switch (dato_anterior)
            {
            case '1': tranferir_puertoB=uno[n];   break;
```

```
            case '2': tranferir_puertoB=dos[n];   break;
            case '3': tranferir_puertoB=tres[n];   break;
            case '4': tranferir_puertoB=cuatro[n];   break;
            case '5': tranferir_puertoB=cinco[n];   break;
            case '6': tranferir_puertoB=seis[n];   break;
            case '7': tranferir_puertoB=siete[n];   break;
            case '8': tranferir_puertoB=ocho[n];   break;
            case '9': tranferir_puertoB=nueve[n];   break;
            case '0': tranferir_puertoB=cero[n];   break;
            }
          escribir_lcd();
          if (n==25)
            {
            tranferir_puertoB=0;
            set_page_X();
            }
          }
        }
      }
    for (j=0;j<=1;++j)
      {
      for (n=0;n<=27;++n)
        {
        if(n==0)
          {
          CS1=1;
          CS2=0;
          on_lcd();
          if(j==1)
            {
            CS1=0;
            CS2=1;
            on_lcd();
            }
          tranferir_puertoB=0;
          set_page_X();
          if (((saturacion<(float)100)&&(j==0))||((ppm<(float)100)&&(j==1)))
            tranferir_puertoB=26;
          else
            tranferir_puertoB=42;
          set_Y_addres();
          }
        tranferir_puertoB=sat[n];
        if(j==1)
          tranferir_puertoB=pulso[n];
        escribir_lcd();
        if (n==13)
          {
          tranferir_puertoB=1;
          set_page_X();
          if(((saturacion<100)&&(j==0))||((ppm<100)&&(j==1)))
            tranferir_puertoB=26;
          else
            tranferir_puertoB=42;
          set_Y_addres();
          }
        }
      }
    }

////////////////////////////////////////////////////////////////////////////////
// funcion que muestra el menu y sus valores en la pantalla.
////////////////////////////////////////////////////////////////////////////////
void escribir_menu()
```

```
{
for(n=0;n<=7;n++)
  {
  CS1=1;
  CS2=0;
  on_lcd();
  tranferir_puertoB=n;
  set_page_X();
  tranferir_puertoB=46;
  set_Y_addres();
  if(n==1)
    sprintf(conta,"%u",muestra_sp_alta);
  if(n==2)
    sprintf(conta,"%u",muestra_sp_baja);
  if(n==3)
    sprintf(conta,"%u",muestra_ppm_alta);
  if(n==4)
    sprintf(conta,"%u",muestra_ppm_baja);
  if(n==5)
    sprintf(conta,"%u",muestra_tiempo);
  if(n==6)
    {
    conta[0]='\x00';
    conta[1]='\x00';
    conta[2]='\x00';
    }
  if(n==7)
    {
    conta[0]='\x00';
    conta[1]='\x00';
    conta[2]='\x00';
    }
  for (i=0;i<=2;i++)
    {
    dato_anterior=conta[i];
    for(j=0;j<=5;j++)
      {
      switch (dato_anterior)
        {
        case '1': tranferir_puertoB=uno88[j];      break;
        case '2': tranferir_puertoB=dos88[j];      break;
        case '3': tranferir_puertoB=tres88[j];     break;
        case '4': tranferir_puertoB=cuatro88[j];   break;
        case '5': tranferir_puertoB=cinco88[j];    break;
        case '6': tranferir_puertoB=seis88[j];     break;
        case '7': tranferir_puertoB=siete88[j];    break;
        case '8': tranferir_puertoB=ocho88[j];     break;
        case '9': tranferir_puertoB=nueve88[j];    break;
        case '0': tranferir_puertoB=cero88[j];     break;
        case '\x00': tranferir_puertoB=0;          break;
        }
      if((n==1)&&(contador_bandera==0))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==2)&&(contador_bandera==1))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==3)&&(contador_bandera==2))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==4)&&(contador_bandera==3))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==5)&&(contador_bandera==4))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==6)&&(contador_bandera==5))
        tranferir_puertoB=~tranferir_puertoB;
      if((n==7)&&(contador_bandera==6))
```

```
            tranferir_puertoB=~tranferir_puertoB;
          if   ((n>0)&&(n<=7))
          escribir_lcd();
          }
        }
      tranferir_puertoB=0;
      set_Y_addres();
      for(i=0;i<=64;i++)
        {
        if(n==0)
          tranferir_puertoB=configuracion[i];
        if(n==1)
          tranferir_puertoB=spo2_alta[i];
        if(n==2)
          tranferir_puertoB=spo2_baja[i];
        if(n==3)
          tranferir_puertoB=ppm_alta[i];
        if(n==4)
          tranferir_puertoB=ppm_baja[i];
        if(n==5)
          tranferir_puertoB=tiempo[i];
        if(n==6)
          tranferir_puertoB=salvar[i];
        if(n==7)
          tranferir_puertoB=salir[i];
        if(tranferir_puertoB!=255)
          escribir_lcd();
        else
          i=65;
        }
      }
    set_tris_b(0xff);
    }
////////////////////////////////////////////////////////////////////////////////
// Rutina que le da servicio a la interrupcion del timer0 la cual es activada
//cuando se pulsa un boton del teclado.
////////////////////////////////////////////////////////////////////////////////
#INT_TIMER0
void rutinatimer0()
          {
          disable_interrupts(GLOBAL);
          disable_interrupts(INT_TIMER0);
          disable_interrupts(INT_TIMER1);
          TOIF=0;
          TMR1IF=0;
          CS_RAM=0;
          CS_373=0;
          lineas_limpiar=8;
          limpiar_lcd();                    //  Borra toda la pantalla.
          //En las instrucciones siguientes se leen los  valores guardados de
          //las alarmas.
          muestra_sp_alta=read_eeprom(1);
          muestra_sp_baja=read_eeprom(2);
          muestra_ppm_alta=read_eeprom(3);
          muestra_ppm_baja=read_eeprom(4);
          muestra_tiempo=read_eeprom(5);
          do                        // Lazo que se ejecuta hasta que se escoje la
                    {               //opción salir del menú.
                    suelta_tecla=0;
                    tecla_pulsada_dos=0;
                    set_tris_b(0);          //Se escribe cero en el puerto b.
                    portb=0;
                    set_tris_b(0xff);       //se coloca el puerto b como entrada.
```

```
CS1=1;              //Instrucciones para habilitar la salida
CS2=1;              //del buffer del teclado.
E=1;
delay_us(5);
E=0;
delay_us(5);
A10=1;
delay_us(5);

do
                    // Lazo donde se lee el teclado y se espera a
        {           //que se suelte la tecla pulsada.
        delay_ms(50);   //Retardo para descartar rebotes de los botones.
        tecla_pulsada_uno=portb;
        tecla_pulsada_uno=(tecla_pulsada_uno<<4);
        if(tecla_pulsada_uno!=0xf0)
                {
                tecla_pulsada_dos=tecla_pulsada_uno;
                tecla_pulsada_anterior=0xff;
                }
        else
        suelta_tecla++;
        }
while(suelta_tecla<=3);

suelta_tecla=0;
CS1=0;              // En estas instrucciones se desactiva la
CS2=0;              //salida del buffer del teclado.
E=1;
delay_us(5);
E=0;
delay_us(5);
A10=0;
delay_us(5);
if (tecla_pulsada_anterior==0xff)
        {
// En esta sentencia suiche se ejecuta la acción correspondiente a la tecla
//o sucesión de teclas oprimidas.
        switch(tecla_pulsada_dos)
                {
                case 0xe0:      // Si desplaza hacia arriba el cursor que
                        {       //indica la acción a ejecutarse, si se llega
                                //al tope se regresa a la linea inferior.
                        contador_bandera++;

        if(contador_bandera==7)
                        contador_bandera=0;
                        }break;
                case 0xd0:      //  Se decrementa el valor de la alarma
                                //que indique el cursor.
                        {

                        if(contador_bandera==0)
                                {
                                muestra_sp_alta--;
                                if(muestra_sp_alta==0xff)
                                muestra_sp_alta=100;
                                }
                        if (contador_bandera==1)
                                {
                                muestra_sp_baja--;
                                if(muestra_sp_baja==0xff)
                                muestra_sp_baja=100;
                                }
```

```
                        if(contador_bandera==2)
                                {
                                muestra_ppm_alta--;
                                if(muestra_ppm_alta==0xff)
                                muestra_ppm_alta=220;
                                }
                                if(contador_bandera==3)
                                {
                                muestra_ppm_baja--;
                                if(muestra_ppm_baja==0xff)
                                muestra_ppm_baja=220;
                                }
                                if(contador_bandera==4)
                                muestra_tiempo--;
                                }break;
                case 0xb0:      // Si desplaza hacia abajo el cursor que
                        {       //indica  la acción a ejecutarse.
                        contador_bandera--;
                        if(contador_bandera==0xff)
                                contador_bandera=6;
                        }break;
                case 0x70:      //  Se incrementa el valor de la alarma
        {                       // que  indique el cursor.
        if(contador_bandera==0)
{
muestra_sp_alta++;
if(muestra_sp_alta==101)
muestra_sp_alta=0;
}
if(contador_bandera==1)
{
muestra_sp_baja++;
if(muestra_sp_baja==101)
muestra_sp_baja=0;
}
if(contador_bandera==2)
{
muestra_ppm_alta++;
if(muestra_ppm_alta==221)
muestra_ppm_alta=0;
}
if(contador_bandera==3)
{
muestra_ppm_baja++;
if(muestra_ppm_baja==221)
muestra_ppm_baja=0;
}
if(contador_bandera==4)
muestra_tiempo++;
                        }break;
                }
        tecla_pulsada_anterior=0;
        escribir_menu();    // Se muestra el menú en la pantalla.
        for (n=0;n<=7;++n)// Lazo para limpiar la segunda parte de la
                {       //pantalla.

CS1=0;
CS2=1;
on_lcd();
tranferir_puertoB=0;
set_star_line();
tranferir_puertoB=0;
set_Y_addres();
tranferir_puertoB=n;
```

```
                                        set_page_X();
                                        for (i=0;i<=63;++i)
                                                {
                                tranferir_puertoB=0;
                                escribir_lcd();
                                                }
                                        }
        if((contador_bandera==5)&&(tecla_pulsada_dos==0x70))
                {
                                        write_eeprom(1,muestra_sp_alta);
                                        // Si se selecciona la opción guardar
                                        // valores estos se escriben en la memoria EEPROM.
                                        write_eeprom(2,muestra_sp_baja);
                                        write_eeprom(3,muestra_ppm_alta);
                                        write_eeprom(4,muestra_ppm_baja);
                                        write_eeprom(5,muestra_tiempo);
                }
                                }
                }while ((contador_bandera!=6)||(tecla_pulsada_dos!=0x70));

        lineas_limpiar=8;
        limpiar_lcd();                          //  Se borra la pantalla gráfica.
        CS1=1;                                  //se selecciona la primera mitad de la pantalla
        CS2=0;                                  //para escribir el mensaje esperar.
        on_lcd();
        tranferir_puertoB=0;
        set_Y_addres();
        tranferir_puertoB=0;
        set_page_X();
        for(i=0;i<=40;i++)
                {
                tranferir_puertoB=esperar[i];
                escribir_lcd();
                }
        numero_paso=0;
        saturacion_2=0;
        ppm_2=0;
        dato_anterior_pwm=0;
        n_intentos_pwm=0;
        num_ciclos=0;
        estableciendo_PWM=1;
        contador_bandera=0;
        offset_memoria=0;
        set_timer0(255);
        TOIF=0;
        TMR1IF=0;
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
        set_timer1(59286);                              //6250 ciclos para determinar una
        enable_interrupts(INT_TIMER1);                  //frecuencia de muestreo de 100Hz.
        enable_interrupts(INT_TIMER0);
        enable_interrupts(GLOBAL);
        }
////////////////////////////////////////////////////////////////////////////////
// Rutina que da servicio a la interrupcion del timer1 la cual fija la base
//de tiempo de 10ms para el muestreo de las señales analógicas ademas cuenta
//250 muestras antes de activar las condiciones para la realización de los
//cálculos.
////////////////////////////////////////////////////////////////////////////////
#INT_TIMER1
void rutinatimer1()
  {
  set_timer1(59286);                                    // El timer se incrementa 6250 veces
  TMR1IF=0;                                             //antes de llegar a el overflow.
  W_R=0;
```

```
digitalizacion();                       //Rutina de digitalización.
guardar_leer_adc_ram();                 //Guarda las muestras en la memoria RAM.
offset_memoria++;
if ((offset_memoria>249)&&(estableciendo_PWM==0))
  {                                     //Este bloque se ejecuta dsespues de establecer
  offset_memoria=0;                     //el PWM desactiva la rutina de interrupción
  comenzar_calculos=1;                  //una vez que se alcanzó 250 muestras.
  disable_interrupts(INT_TIMER1);
  setup_timer_1(T1_DISABLED );
  }
if((offset_memoria>249)&&(estableciendo_PWM==1))
  {                                     //Este bloque se ejecuta al encenderse el equipo

  offset_memoria=0;                     //para establecer el ciclo de trabajo del PWM.
  disable_interrupts(INT_TIMER1);
  setup_timer_1 ( T1_DISABLED );
  establecer_PWM();
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8 );
       set_timer1(59286);
  enable_interrupts(INT_TIMER1);
  }
}
```

[ANEXO nº 2]

LISTADO DE COMPONENTES

Tarjeta analógica:

U1: CI. TL070.

U2,U3: CI. LM385/2.5.

U4,U8: CI. LF398.

U5: CI. TL074.

U6,U13: CI. LM358.

U9: CI. LM555.

U10: CI. CD4011.

U11: CI. CD4013.

U12: CI. CD4001.

Q1,Q2,Q3,Q4,Q5: 2N3904.

D1: BPW34.

D2: XLMR53WDW.

D3 : OP140A.

D4,D5,D6,D7,D8,D9,D10,D11: 1N5817.

R1 33 kΩ±5%.

R2 33 kΩ±5%.

R3 Potenciómetro 10kΩ.

R4 220 Ω±5%.

R5 5.1 kΩ±5%.

R6 51 kΩ±5%.

R7 51 kΩ±5%.

R8 6.81 MΩ±1%.

Tarjeta digital :

U1: PIC16F916.

U2: HM6264A.

U3,U4: 74LS373.

U5: 74LS10.

U6: 74LS74.

U7:	LM555.
U8:	LM358.
U9:	74LS125.
U10:	74LS08.
Q1,Q5:	2N22222.
J1-LCD:	GMD12864
LS1:	Buzzer.