7 Técnica

7.1 Esquema General

El sistema consiste de tres partes:

Unidad de procesamiento
Es la encargada de procesar la información adquirida por los sensores, junto con el estado de los actuadores, para informar mediante un modelo fisiológico el estado del simulador. Al finalizar la práctica tiene la función de enviar vía inalámbrica la información recabada al servidor.

Unidad de Control
Es la encargada de manejar los sensores y actuadores instalados en el maniquí, también las comunicaciones entre el procesador central y el mando a distancia.

Servidor
Es el encargado de almacenar la información y desplegarla en una página WEB.

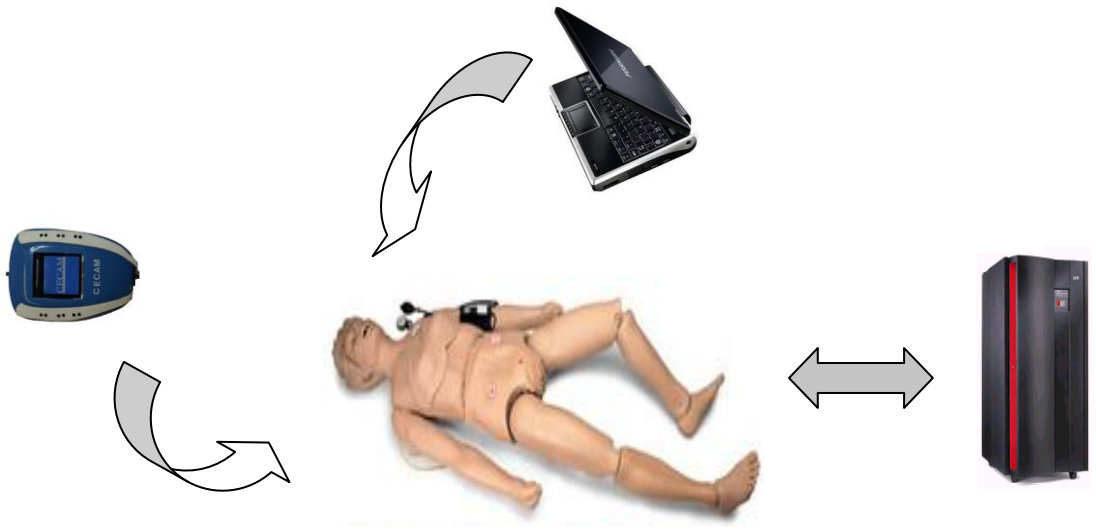


Figura 7.1 Conformación del sistema.

La unidad de control y la unidad de procesamiento están instaladas dentro del maniquí en la cavidad abdominal. El mando a distancia se encuentra conectado al maniquí mediante un cable tipo telefónico, para ser reemplazado de modo sencillo, a un lado de la cavidad abdominal se ubica también el botón de encendido y el conector para el cargado de las baterías del maniquí.

El maniquí puede operar de manera autónoma, sin conexión a la red eléctrica, durante un periodo aproximado de 2 horas 30 minutos y los actuadores pueden mantenerse encendidos un tiempo aproximado de 30 minutos.

7.2 Unidad de control

7.2.1 Esquema general

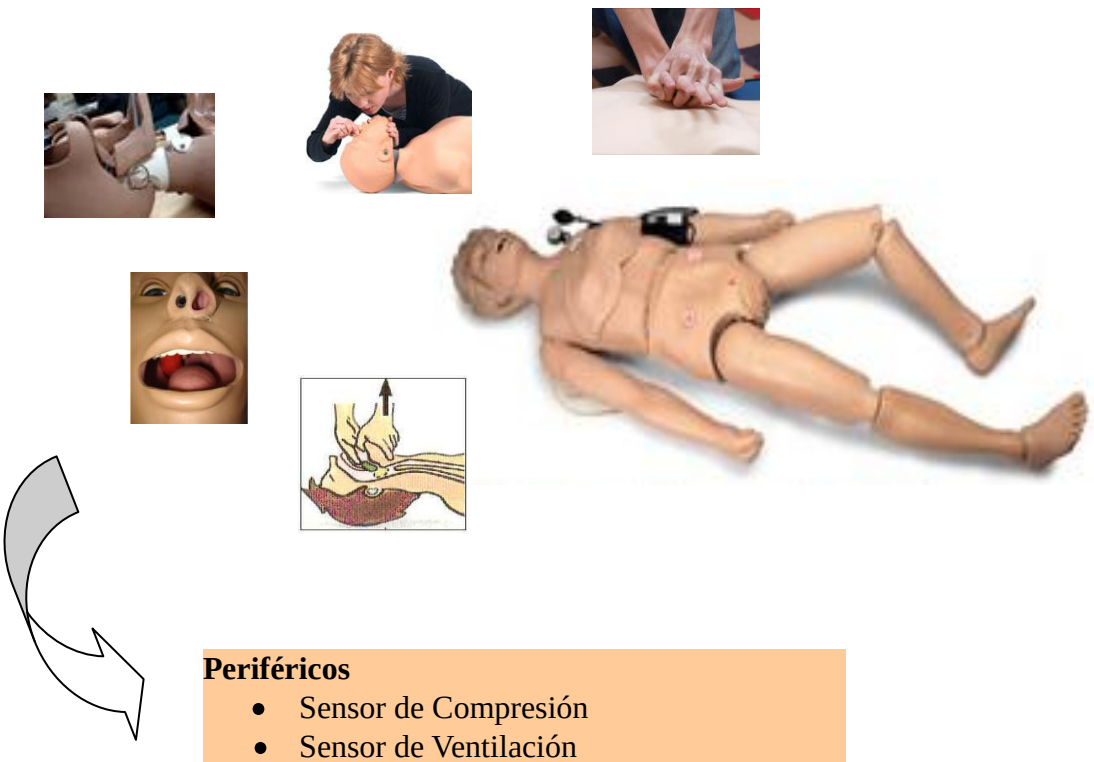
La unidad de control se compone por tres subsistemas:

Mando a distancia
Es el encargado de manejar la interfaz con el usuario, desplegar la información y recibir los comandos.
Esta compuesta por:

- Microcontrolador auxiliar
- Pantalla LCD

Tarjeta principal
Es la encargado de manejar las comunicaciones, de controlar los periféricos y del cargado de la batería.
Se divide principalmente en:

- Microcontrolador maestro
- Cargador de batería
- Amplificación de Audio



Periféricos

- Sensor de Compresión
- Sensor de Ventilación
- Sensor de Maniobra de Hiperextensión
- Sensor de Maniobra de Subluxación
- Sensor de Cuerpo Extraño en vías aéreas
- Actuador de Pulso
- Actuador de Ventilación

Figura 7.2 Composición de la unidad de control

La unidad de control se encuentra intercomunicada mediante el *bus* I^2C, el cual tiene una tasa máxima de transferencia de 400kbps. La tarjeta principal se comunica al mando a distancia mediante un cable, de un máximo de 5 metros, usando éste protocolo. Esta a su vez se comunica con la unidad de procesamiento mediante el *bus* serial universal (USB), el cual tiene una tasa de transferencia máxima a "*Full Speed*" de 12Mbps.

7.2.2 Tarjeta principal

7.2.2.1 Esquema general

La tarjeta principal contiene la mayor parte de la electrónica, incluyendo la etapa de adecuación de señal de los sensores, se encuentra colocada, también, en la cavidad abdominal, sus dimensiones son 10 x 15 cm.

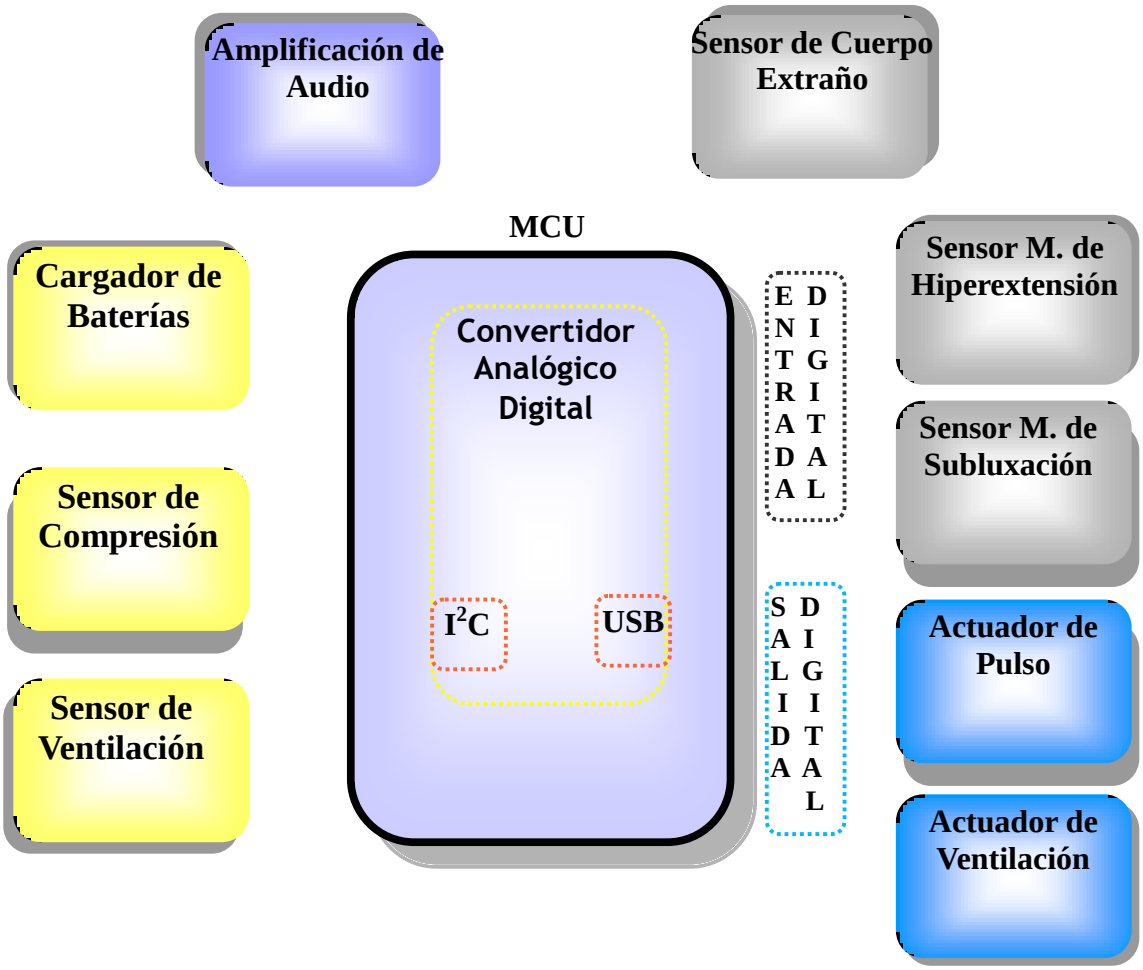


Figura 7.3 Diagrama de bloques de la tarjeta principal

7.2.2.2 Microcontrolador maestro

7.2.2.2.1 Esquema general

El microcontrolador utilizado es capaz de manejar *full speed* USB compatible con *Serial Interface Engine* (SIE), la cual es necesaria para la comunicación con el sistema de procesamiento.
Un diagrama de flujo simplificado puede observarse en la figura 7.4.

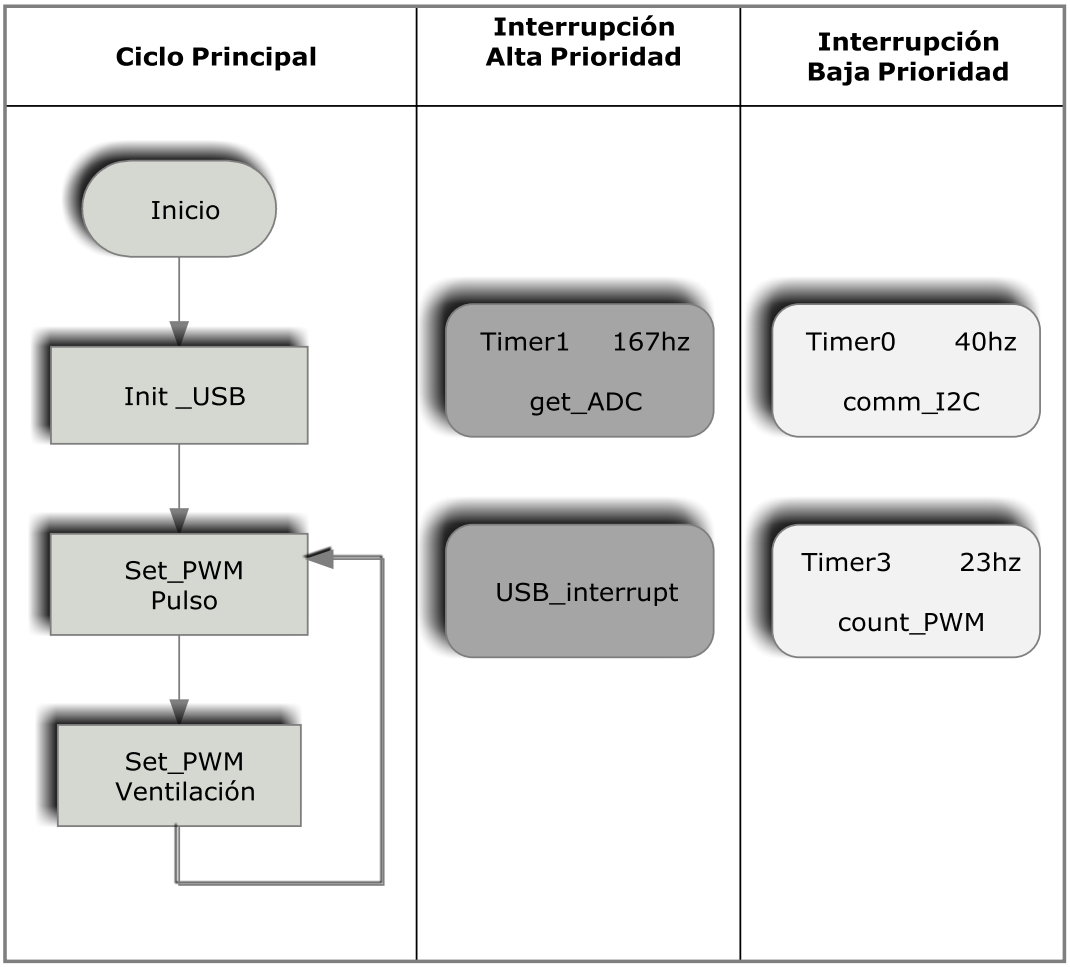


Figura 7.4 Diagrama de bloques MCU maestro

7.2.2.2.2 Ciclo principal

La función principal de este ciclo es iniciar todas las variables, con sus valores predeterminados, dando inicio al USB y entrando en un ciclo infinito de control del PWM de los actuadores.

El microcontrolador (MCU) cuenta con un motor de interfaz serial (SIE), el cual trabaja por separado del MCU y tiene su propio reloj, esto es para que la carga del USB no sea procesada por el MCU y se pueda responder a tiempo al *host*. Para lograr este proceso la SIE comparte 1kb de RAM con el MCU.

El acceso al microcontrolador es moderado por un sistema de semáforo, el cual indica a quién le pertenece la memoria en ese momento y cuando deja de modificarla deberá liberarla.

El encendido del módulo de USB es de la siguiente manera:

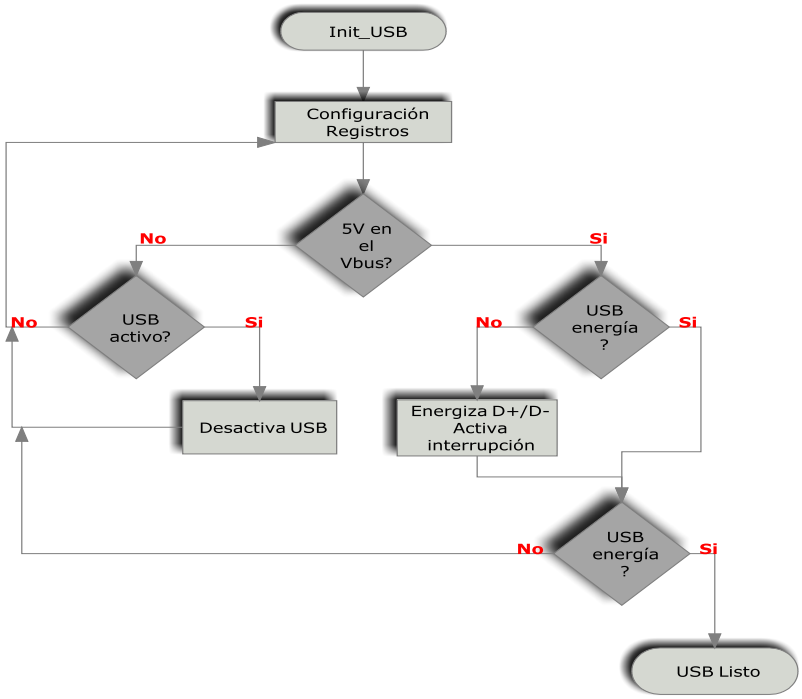


Figura 7.5 Diagrama de bloques de init-USB

Después de activar el USB se entra en un estado cíclico infinito en el cual se verifica el estado de los actuadores y se decide el valor a enviar al PWM, como se muestra en la figura 7.6.

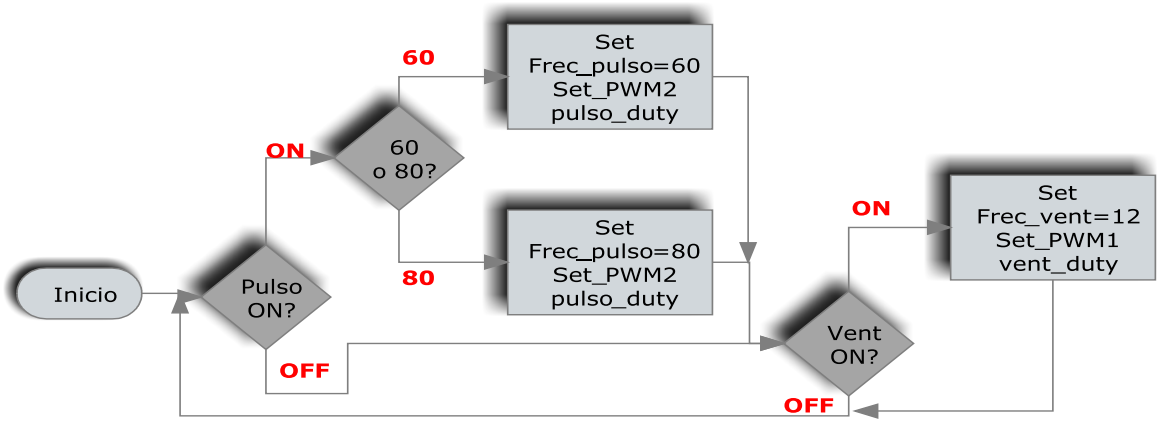


Figura 7.6 Diagrama de bloques del ciclo de los actuadores.

El pulso PWM presenta la siguiente señal de salida.

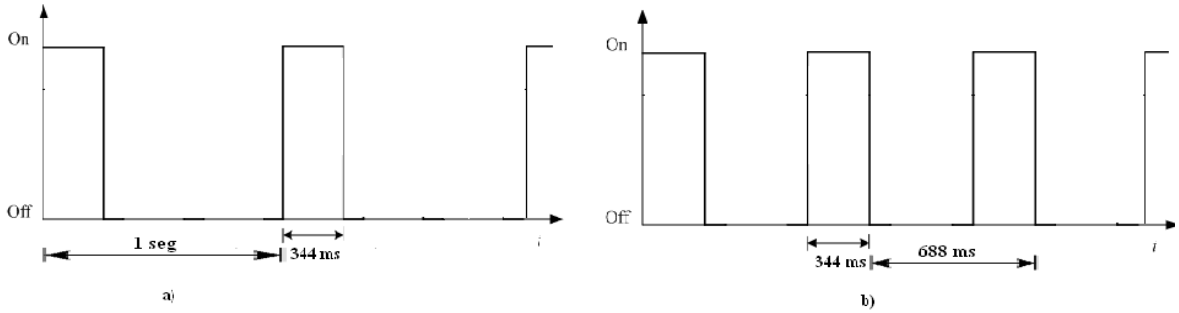


Figura 7.7.Forma de onda del PWM para actuador de pulso
a) 60 latidos por minuto; b) 80 latidos por minuto

Los ciclos de trabajo de los periodos de 42 µseg en este PWM, se encuentran al 100% o al 0%.

El PWM de la ventilación presenta la siguiente señal de salida.

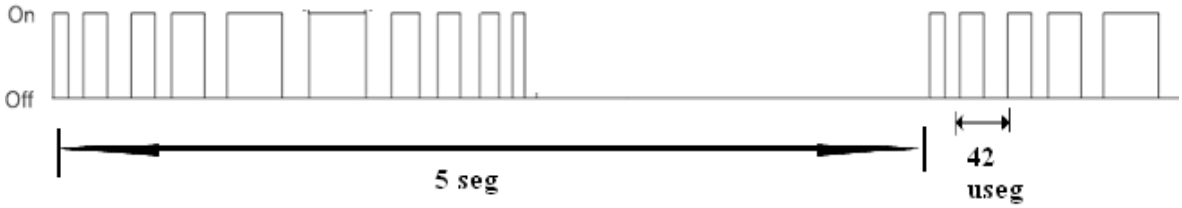


Figura 7.8. Forma de onda del PWM para actuador de ventilación.

La ventilación es controlada de manera senoidal para dar un movimiento más suave.

7.2.2.2.3 Interrupciones

Las interrupciones están divididas en dos. Las de alta prioridad deben ejecutarse lo antes posible ya que el no hacerlo provocaría mal funcionamiento de la aplicación, tienen la capacidad de detener la ejecución de las interrupciones de baja prioridad para ejecutar sus tareas en el momento solicitado.

Interrupción count_pwm – Timer3

La interrupción Timer 3 es de baja prioridad y se utiliza para controlar los tiempos de los PWM, estos controlan a su vez los actuadores de pulso y ventilación, no requiere de mucha precisión y puede ser interrumpida por otros de mayor prioridad.

Interrupción get_ADC – Timer1

La interrupción Timer1 es de alta prioridad ya que es la encargada de llevar el reloj de la práctica así como de hacer el muestreo de los sensores.

Interrupción USB_interrupt

La interrupción USB necesita la prioridad alta pues un retraso en el proceso de los *requests* del *host* podría provocar mal funcionamiento. Esta interrupción también maneja las comunicaciones entre el *endpoint* 0, el *endpoint* de control, y el *host*.

Interrupción comm_I^2C – Timer0

La interrupción *timer* "0" se encuentra dentro de las interrupciones de baja prioridad, es la encargada de mantener actualizada la información del mando a distancia, así como de recibir sus comandos (botones presionados). Algún retraso en el arribo de esta información, siendo este en el rango de milisegundos, no causa ningún funcionamiento erróneo dentro del sistema.

7.2.2.3 Cargador de Batería

7.2.2.3.1 Esquema general

Existen dos baterías en el sistema, una de ellas se encuentra en la unidad de procesamiento la cual contiene su propio cargador, la segunda es una batería de ácido sellada.

La fuente de energía es un regulador a 19V de 3.0A máximo, por lo tanto se eligió utilizar una fuente conmutada tipo *buck* para este proceso.

El cargador se encuentra a 14V, es limitado a 1A de corriente, y se encuentra activo cada vez que el regulador de 19V es conectado al sistema.

7.2.2.4 Amplificador de Audio

7.2.2.4.1 Esquema general

El audio proviene de la unidad de procesamiento, esta etapa lo amplifica. Proporciona una ganancia estática de 34dB, la salida se centra automáticamente a la mitad del voltaje de alimentación, también cuenta con protección por corto circuito.

El volumen puede ser ajustado con un potenciómetro, divisor de voltaje, en la entrada de la señal.

7.2.3 Mando a distancia

7.2.3.1. Esquema general

El mando a distancia consta de una pantalla y seis botones, se conecta con un cable telefónico al bus I^2C de la tarjeta principal así como a la batería de la unidad de procesamiento central.

Cuenta con una pantalla LCD a colores de 12 bits (4 Rojo, 4 Verde, 4 Azul), de 132 x 132 pixeles[1] y mide dos pulgadas de esquina a esquina. La interfaz está diseñada para mantener informado al usuario durante la práctica, para llevar el crononometraje de los pasos que se deben seguir durante el ejercicio y además el control del simulador.

Figura 7.9. Fotografía del mando a distancia.

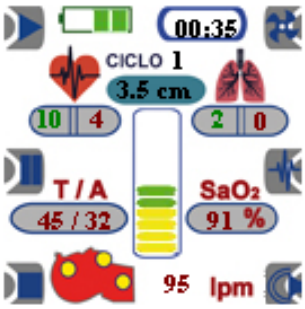


Figura 7.10. Pantalla principal.

La interfaz principal, cuya pantalla principal se muestra en la figura número 7.10, presenta los datos de la práctica en curso, las compresiones, las ventilaciones, así como las

[1] Pixel es el acrónimo del inglés *picture element, "elemento de imagen".*

variables del modelo fisiológico en tiempo real. La interfaz permite iniciar la práctica, pausarla o terminarla. También puede ingresarse a los menús de ventilación, pulso o sonido.

La interfaz de cronómetros se encuentra diseñada para tomar el tiempo de respuesta durante la práctica, tiene seis cronómetros distintos, estos cuentan a su vez con un indicador de evaluación, o sea, un marcador que cambia de color para indicar si el proceso fue completado correctamente.

Figura 7.11. Pantalla del cronómetro.

Las interfaces para controlar los actuadores son tres, muestran las opciones de cada uno de los actuadores, así como su estado.

Figura 7.12. Pantallas de actuadores.

7.2.3.2 Pantalla LCD

7.2.3.2.1 Esquema general

Para el manejo de esta pantalla se transcribieron las librerías para controlar este dispositivo, la pantalla utiliza una interfaz serial de 9 bits.

Figura 7.13. Fotografía LCD

7.2.3.2.2 Descripción

La pantalla cuenta con diez pines de acceso y se describen en la siguiente tabla.

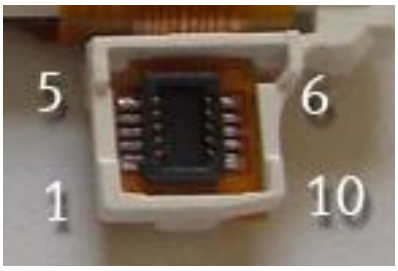


Figura 7.14. Fotografía de pines de pantalla LCD

Pin	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Función	**VVC Dital**	**Número Reset**	**SI**	**SCL**	**Número CS**	**VCC LCD**	**N.C.**	**GND**	**VLed (-)**	**VLed (+)**

Tabla 7.1 Descripción de pines de pantalla LCD

La comunicación entre el MCU y la pantalla es realizada vía una interfaz serial de 9 *bits* mediante las terminales SI, SCL, CS. Es importante señalar que esta vía de comunicación no es bidireccional, por lo tanto, no se podrán leer los *pixeles* que se encuentran en la pantalla.

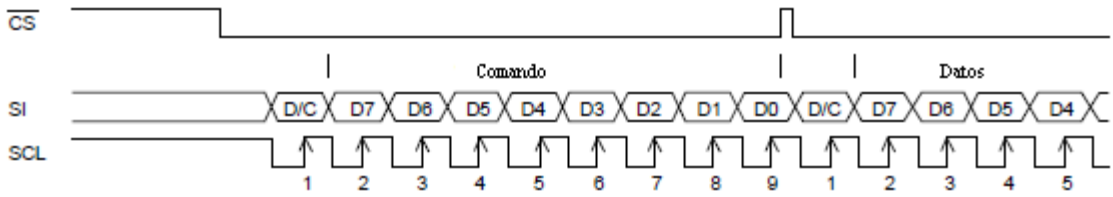


Figura 7.15. Forma de onda de comunicación con LCD

7.2.3.3 Microcontrolador Auxiliar

7.2.3.3.1 Esquema general

El MCU opera a 48MHz con un cristal de 12MHz, ya que cuenta con un PLL multiplicador. Esta es la velocidad máxima recomendada del MCU, la cual es necesaria para que el cargado de la pantalla de LCD sea lo más rápido posible.

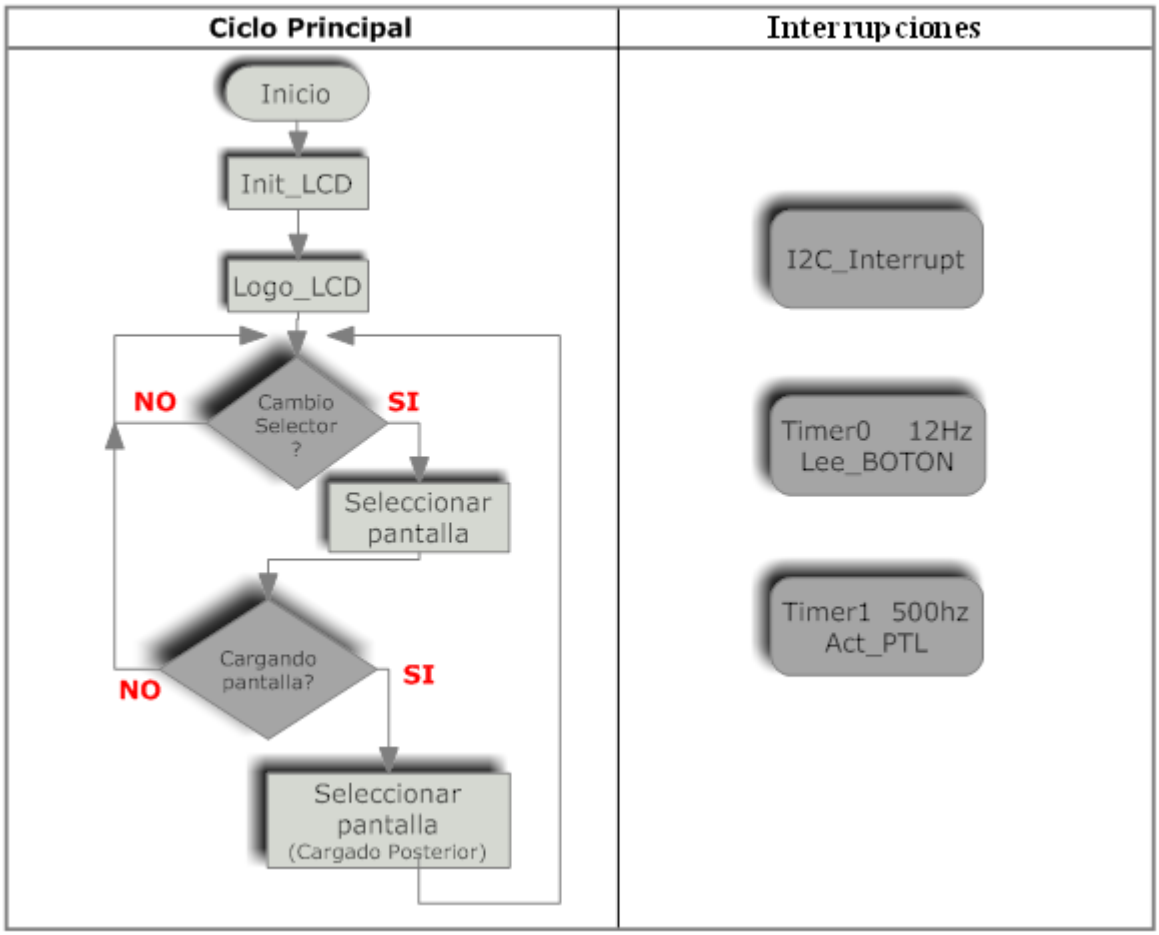


Figura 7.16. Diagrama de bloques MCU auxiliar

7.2.3.3.2 Ciclo Principal

Primeramente este ciclo se encarga de iniciar la pantalla (descrito en la *sección 7.2.3.2*) y todas las variables en sus valores predeterminados. Después carga la imagen que contienen el logotipo de inicio, dicha imagen está contenida en la memoria ROM del MCU y se encuentra en forma de mapa de *bits* en escala de grises para el uso óptimo de la memoria. Posteriormente ingresa en un ciclo infinito verificador de los posibles cambios en el selector de pantalla. Si existe un cambio solicita el mapa de *bits* mediante una bandera al MCU maestro. Si existe algún cambio del selector durante el envío del mapa de *bits* solicitado anteriormente el MCU esperará a que termine la carga del selector para enviar la nueva solicitud del mapa de *bits*.

7.2.3.3.3 ***Interrupciones***

Existen tres interrupciones en el MCU auxiliar, dos se basan en contadores y la otra en la comunicación I^2C.

Interrupción I^2C_interrupt

Esta interrupción es llamada cuando arriba un mensaje desde el maestro, el MCU debe ser configurado como esclavo, y es la encargada de manejar todas las comunicaciones con el maestro.

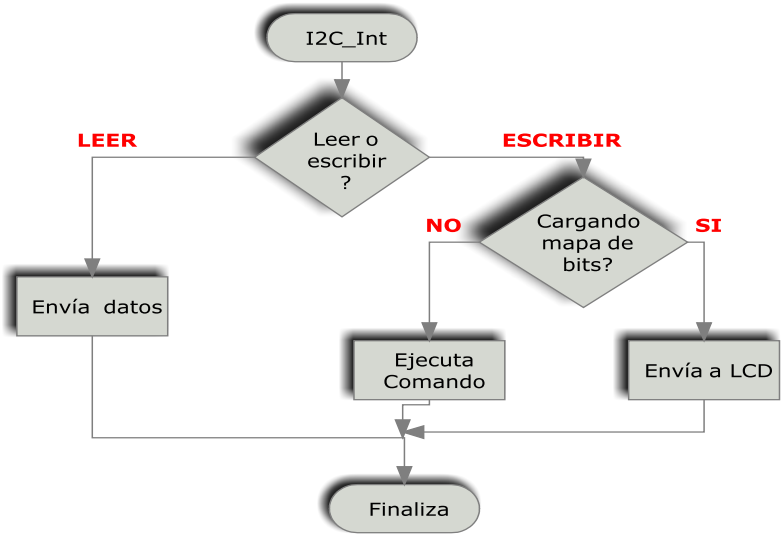


Figura 7.17. Diagrama de bloques interrupción I^2C.

Interrupción Lee_BOTON – Timer0

La función de la interrupción *Timer* "0" es monitorear si algún botón es presionado en el mando a distancia, siendo estos seis divididos en dos grupos de tres. La lectura de estos grupos se da cada 83 milisegundos, por lo tanto el botón deberá mantenerse presionado un mínimo de 166ms para ser detectado adecuadamente. Una vez que se ha detectado la activación del botón deberá ejecutarse la función del mismo. Esto es mediante la utilización de una bandera la cual indica que un botón ha sido presionado, la cual es enviada al MCU maestro en la siguiente interrupción de I^2C.

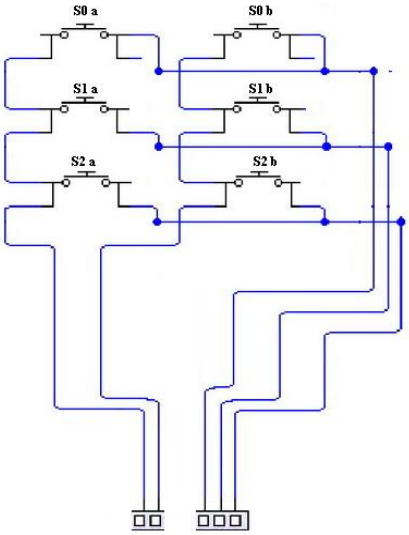


Figura 7.18. Diagrama de los botones del mando a distancia.

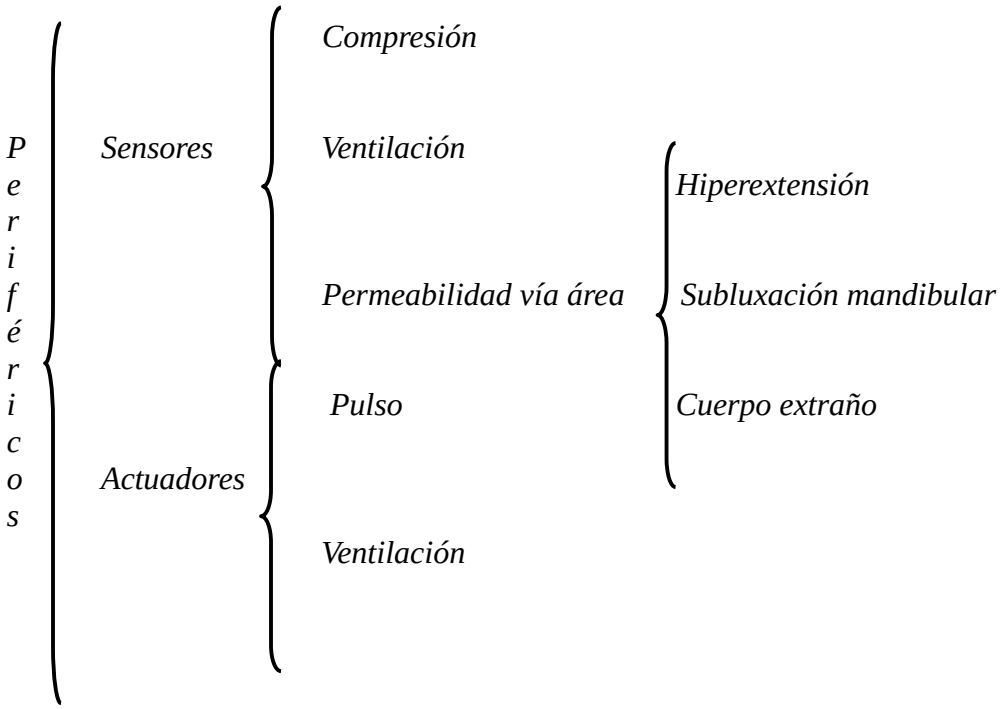
Interrupción ACT_PTL – Timer1

La interrupción *Timer* “1” es la encargada de actualizar los valores desplegados en la pantalla. La interrupción tiene una frecuencia de 500 Hz y cada dato es actualizado a la vez, siendo 18 para la pantalla principal y 7 para la pantalla de cronómetros. Los datos únicamente son actualizados si hubo algún cambio. Esto provoca que la interrupción actualice únicamente los datos que cambiaron. Si se diera un cambio total de los datos al mismo tiempo provocaría que la interrupción tardara aproximadamente 36 milisegundos en actualizar la pantalla, lo que da un margen de 14 ms antes de que llegue el próximo paquete de datos desde el maestro.

7.2.4 Periféricos

7.2.4.1 Esquema general

Los dispositivos del simulador que interactúan directamente con el usuario fueron diseñados para dar una impresión más cercana a la realidad. Estos se pueden dividir en dos partes:



Los sensores son alimentados por la batería instalada en la unidad de procesamiento central, la cual también alimenta a la tarjeta principal. Los actuadores, por su alto consumo de corriente eléctrica, tienen una batería extra que los alimenta. Estos sistemas son controlados desde el mando a distancia, pueden ser encendidos o apagados instantáneamente por el usuario.

Cada uno de estos periféricos debe ser calibrado para ajustarse a las situaciones y lograr mayor realismo.

7.2.4.2 Sensor de compresión

7.2.4.2.1 Esquema general

El sensor de compresión se basa en la ley de Boyle-Mariotte, la cual relaciona la presión con el volumen de un gas cuando se encuentra a temperatura constante. Esto es porque el sensor de compresión se compone por un elemento plástico con forma de corazón.

$$P_1V_1 = P_2V_2$$

Este corazón contiene una cavidad sellada, junto con una válvula de una sola vía, que al momento de recibir la compresión eleva su resistencia dependiendo los centímetros desplazados al aplicar el aplastamiento.

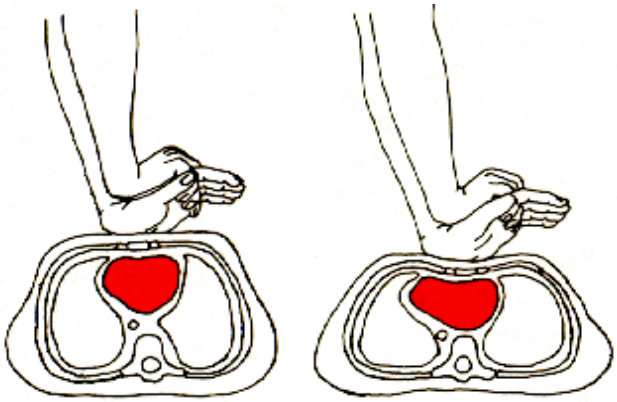


Figura 7.19. Dibujo del corte transversal del corazón durante una compresión.

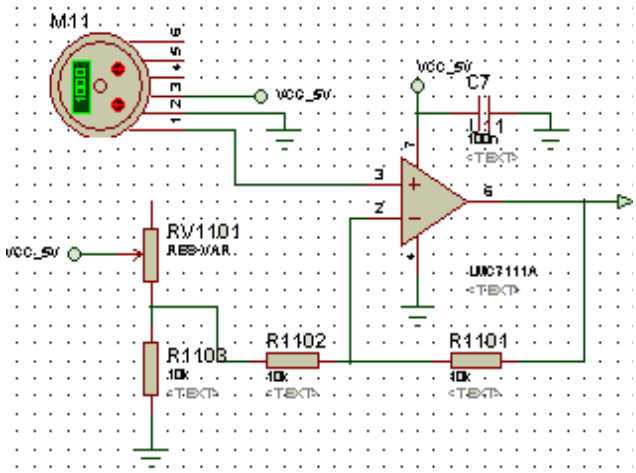
Los centímetros que se desplaza el corazón durante la compresión no tiene una relación lineal con la elevación de la presión ya que se desconoce cómo se modifica el volumen con cada centímetro de desplazamiento, por ello se utilizó el método de calibración descrito en la sección *7.2.4.2.3*.

7.2.4.2.2 Adecuación de señal

El transductor utilizado para medir la elevación de presión es piezo-resistivo y contiene una etapa de amplificación integrada. Se encuentra compensado por temperatura y tiene un máximo en presión de 50kPa, además tienen buena linealidad, adecuado para nuestra aplicación.

Figura 7.20. Imagen del transductor.

A pesar de contar con una etapa de amplificación integrada su señal debe ser adecuada para aprovechar al máximo el puerto ADC del MCU. Para este proyecto se utilizó un amplificador operacional rail-to-rail[2].



Fugura 7.21. Esquema de la adecuación de señal del sensor de compresión.

7.2.4.2.3 Calibración

Al calibrar el dispositivo se busca establecer el desplazamiento sobre el pecho del simulador, por lo tanto, se necesita una función que relacione los centímetros desplazados con la elevación de la presión aplicada en el corazón hueco ubicado en la cavidad torácica.

Los parámetros a calcular son el cambio de volumen con respecto al aumento de presión. Dado que el aire no es un gas ideal, y que existe una pérdida de masa en el momento que se cierra la válvula de una sola vía, se decidió seguir mediante estadística, de esta manera se obtendría una función polinomial que realice la conversión entre el aumento de presión y los centímetros desplazados.

Se solicitó la asistencia de nueve médicos para que realizaran cinco ciclos de reanimación cardiopulmonar, cada uno aplicó 150 compresiones, en total fueron 1350 compresiones. Del total de compresiones se analizó el desplazamiento al momento de aplicación y se comparó con los datos de aumento de presión otorgados por el sensor, esta información se ingresó al MATLAB[3], así se calculó la función polinomial que a su vez se registró en la unidad de procesamiento que posteriormente revelaría los valores del desplazamiento en tiempo real durante la aplicación.

[2] Rail-to-rail significa que puede operar en todo el largo de su alimentación, con la mínima pérdida de voltaje.

[3] MATLAB es un software matemático el cual usa representaciones matriciales para realizar sus operaciones, contiene una vasta librería de funciones matemáticas.

Figura 7.22. Médico aplicando compresión.

7.2.4.3 Sensor de ventilación

7.2.4.3.1 Esquema general

Existen distintos métodos para medir el flujo de un fluido, para este proyecto se decidió utilizar un sensor de orificio variable por su bajo costo y amplio rango de valores de flujo que puede medir.

El sensor de orificio variable se basa en la ecuación de Bernoulli en la que la presión, después de una obstrucción, será menor que la presión antes de la obstrucción, siendo que el flujo está relacionado con las dimensiones físicas y la velocidad del fluido, obteniendo la diferencia de presión es posible calcular el flujo. Finalmente se puede integrar el flujo en un intervalo de tiempo, así se obtiene la cantidad de aire insuflado.

$$P_1 + \frac{1}{2}\rho v_1^2 = P_2 + \frac{1}{2}\rho v_2^2$$

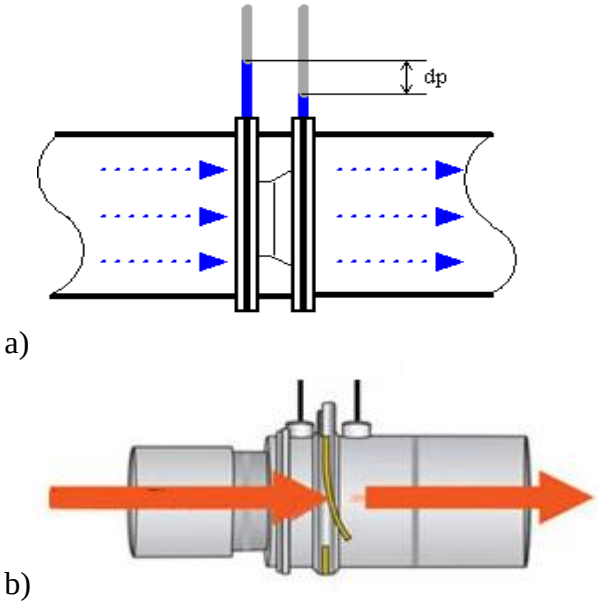


Figura 7.23. Esquema del sensor de orificio variable.
a) Esquema del cambio de presión por el flujo de aire; b) Esquema del orificio variable

La diferencia de presión es medida por el transductor (detallado en la sección 7.2.4.2.2), esta medida de presión diferencial está relacionada con la velocidad y a su vez con el flujo. En la figura 7.23 a) se observa un ejemplo del funcionamiento del sensor, mostrando la diferencia de presión generado en el mismo, y en la figura 7.23 b) se muestra el orificio variable utilizado en el sensor que tiene una ventana de plástico que permite un cambio en la resistencia al flujo.

El dispositivo de orificio variable permite hacer mediciones con mayor rango, ya que a flujos bajos la resistencia es mayor, permitiendo una mayor caída de presión. Y a flujos altos, reduce la resistencia permitiendo una menor caída de presión.

Este dispositivo es utilizado en algunas máquinas de anestesia para medir el flujo de gases que se le proporciona al paciente. La ventaja de estos sensores es el bajo costo de fabricación, frente a la desventaja de su necesidad de ser calibrados.

7.2.4.3.2 Adecuación de la señal

Antes de ser amplificada la señal se filtra mediante un dispositivo paso bajas de primer orden para eliminar el ruido de movimiento que puedan ingresar al sistema. También se cuenta con una compensación de *offset* para colocar la señal en 2.5 V cuando el valor es cero, así es posible medir flujos en dos sentidos del transductor.

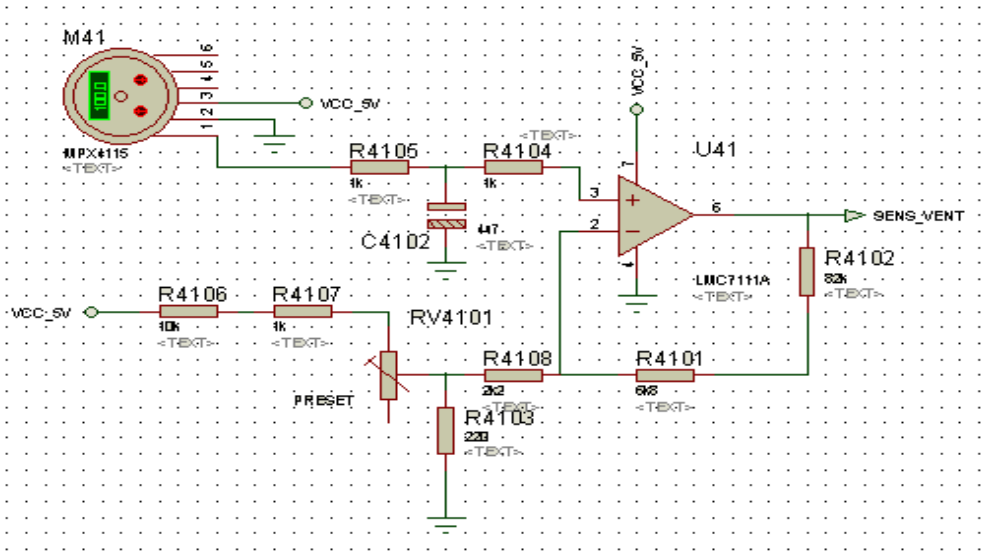


Figura 7.24. Esquema del sensor de ventilación.

7.2.4.3.3 Calibración

La calibración del sensor se realizó con un analizador de flujo de gases *VT PLUS HF* de la marca FLUKE BIOMEDICAL ya que tiene la capacidad de medir flujos y presiones bajas y altas, de esta manera ya no se necesitó de manómetros y medidores de flujo, mide hasta 21 parámetros de ventilación y puede presentarlos en la pantalla.

El sistema de calibración consta de una conexión de fuente de gas (aire médico) conectado a un regulador de flujo. Se permite ajustar el flujo requerido al sensor de flujo y al analizador de flujo de gas.

El analizador de flujo de gas otorga mayor precisión de medición, así asegura el flujo constante.

Una vez establecido un flujo constante se tomaron las medidas de voltaje en la salida del sensor. Se realizaron 3 muestreos con resoluciones similares en rangos de 0 [ml/min] ó 2.504 [V] a 107930 [ml/min] ó 3.575 [V].

Las tablas se encuentran en el apéndice C.

7.2.4.4 Sensores de permeabilidad de vía aérea

7.2.4.4.1 Esquema general

Para detectar si la vía aérea del simulador de RCP básica es permeable se acoplaron dos sensores *stringpot* y seis sensores detectores de campo magnético. Los primeros permiten detectar la maniobra de subluxación mandibular, la posición de cabeza, es decir, permite al usuario verificar si el cuello se encuentra hiperextendido, también si en la cavidad oral hay un objeto obstruyendo la vía aérea.

Los sistemas de detección de PVA (permeabilidad de vía aérea) son eficientes, los *stringpot* son de 10kΩ y están conectados a una fuente de 5V. Cada uno consume una potencia de 2.5mW, según las especificaciones del fabricante.

7.2.4.4.1-1 Sensores stringpot

El *stringpot*, también conocido como transductor cable-extensión (CETs), es un dispositivo usado para medir y detectar la posición lineal mediante un cable flexible, un potenciómetro y un resorte de espiral. Su empleo es sencillo, se monta el cuerpo del *stringpot* en una base fija y después el cable flexible se conecta al objeto en movimiento. Cuando algún objeto se mueve el transductor genera una señal eléctrica, en nuestro caso voltaje que es proporcional a la extensión del cable, este voltaje después puede ser manipulado. El *stringpot* está constituido por cuatro partes: cable de medición, carrete, resorte de espiral y un sensor de rotación.

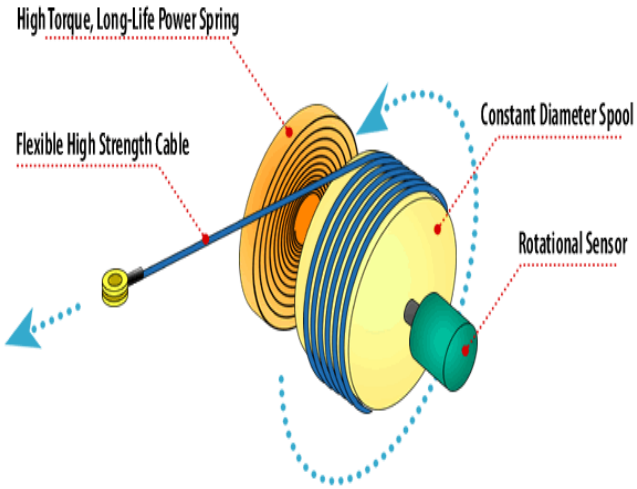


Figura 7.25. Partes de un StringPot

Para mantener la tensión del cable el resorte de espiral está acoplado al carrete, este a su vez lo está al eje de un sensor rotacional (un encoder o un potenciómetro) permitiendo que el cable del transductor se extienda libremente. El sensor de rotación genera la señal eléctrica proporcional a la extensión lineal del cable.

Se eligieron los StringPot pues con ellos es sencillo medir el movimiento y posición de objetos, además se pueden instalar en áreas estrechas con rapidez, no requieren perfecta alineación paralela, también ofrecen gran flexibilidad y un tamaño pequeño en relación a la medición, incluso cuestan menos que los de tipo varilla.

7.2.4.4.1-2 Interruptor magnético

El interruptor magnético, o “reed switch”, que se usó consta de pequeñas placas imantadas y flexibles que hacen contacto cuando están ante un campo magnético. Este dispositivo cuenta con una cubierta de vidrio que permite aislar el interruptor, así las placas no tienen contacto entre ellas a causa de objetos externos. Este interruptor es capaz de manejar hasta 0.5 A y 24 VDC.

7.2.4.4.2 Sensor de hiperextensión

7.2.4.4.2-1 Esquema general

Este sistema utiliza para su funcionamiento un sensor *stringpot* localizado en el interior del cráneo del maniquí. El extremo del cable se conecta en la parte posterior del cuello para poder detectar el movimiento del cráneo, así se obtiene un voltaje proporcional a la extensión del cable desplazado. La señal de tensión que entrega el sensor se procesa por medio de un comparador con histéresis, así se obtiene una señal digital. La señal digital es adquirida por medio de un pin de entrada/salida del microprocesador de la tarjeta principal para su posterior empleo en procesos relacionados con la permeabilidad de la vía aérea.

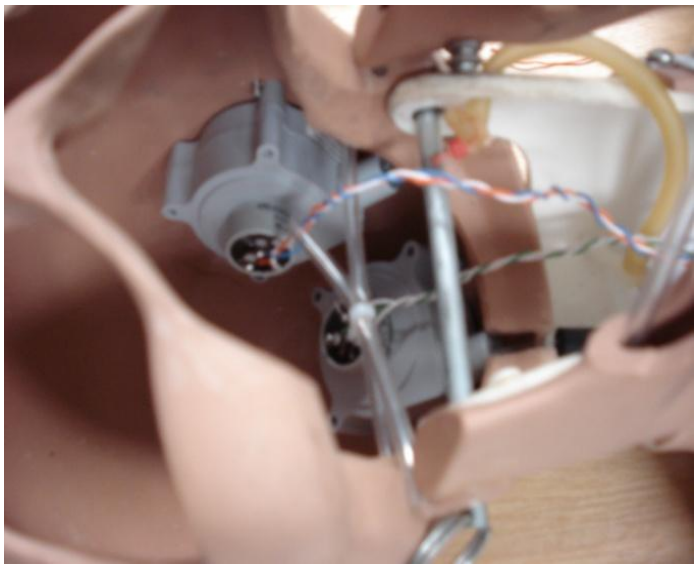


Figura 7.26. StringPots instalados en el cráneo para detectar la hiperextensión.

7.2.4.4.2-2 Acoplamiento de señal

La señal de voltaje del sensor de hiperextensión se acopla por medio de un comparador de ventana implementado con dos comparadores de voltaje. El rango se ajusta por medio de dos potenciómetros que fijan el voltaje superior e inferior de la ventana.

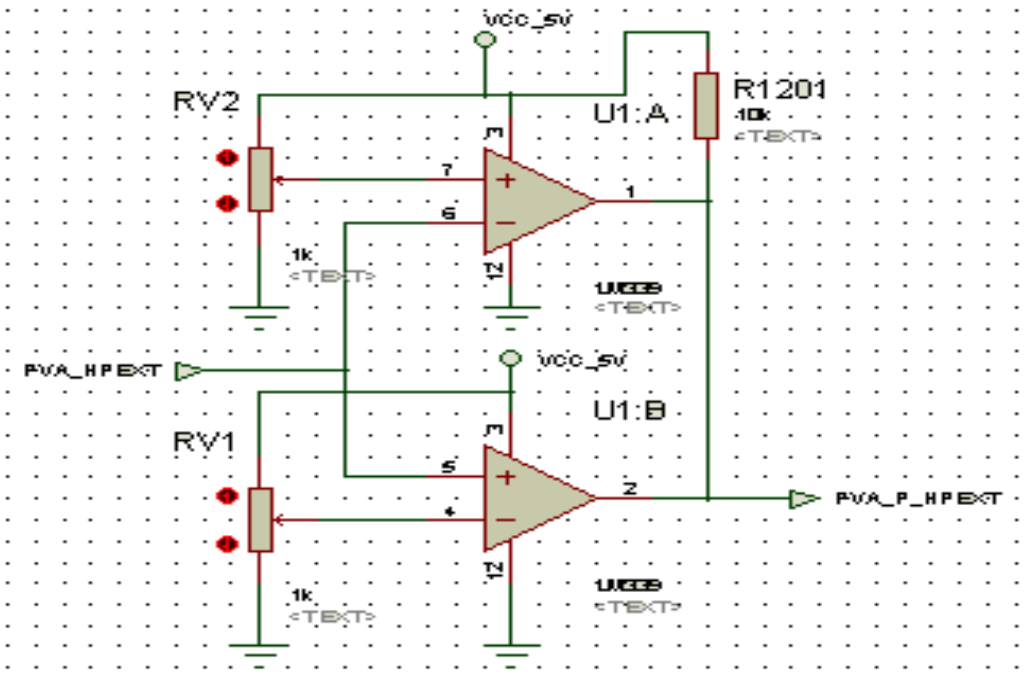


Figura 7.27. Esquema del sensor de hiperextensión.

7.2.4.4.2-3 Calibración

La calibración de este sensor se realizó con la ayuda de un médico que realizó la resucitación cardiorespiratoria de manera correcta e incorrecta, con ello se logró ajustar los potenciómetros para calibrar la ventana del comparador.

7.2.4.4.3 Sensor de subluxación

7.2.4.4.3-1 Esquema general

El sistema usa un sensor *stringpot* colocado al interior del cráneo del maniquí, el extremo del cable se conecta a la base de la mandíbula, para así poder detectar su movimiento, al conectar los extremos de este potenciómetro a una fuente de voltaje se obtiene una señal de voltaje proporcional al desplazamiento mandibular. Esta señal se procesa con un comparador de voltaje para obtener una señal digital que indica el estado de la mandíbula, posteriormente esta señal es adquirida por un pin del microprocesador para ser utilizada en procesos relacionados con la permeabilidad de la vía aérea.

7.2.4.4.3-2 Acoplamiento de señal

La señal que proviene del sensor *stringpot* está conectada a la terminal inversora del comparador de voltaje. La terminal no inversora se conectó al potenciómetro para ajustar el disparo del comparador y así la señal de referencia permite ajustar el rango de desplazamiento mandibular que se toma como subluxado.

7.2.4.4.3.3 Calibración

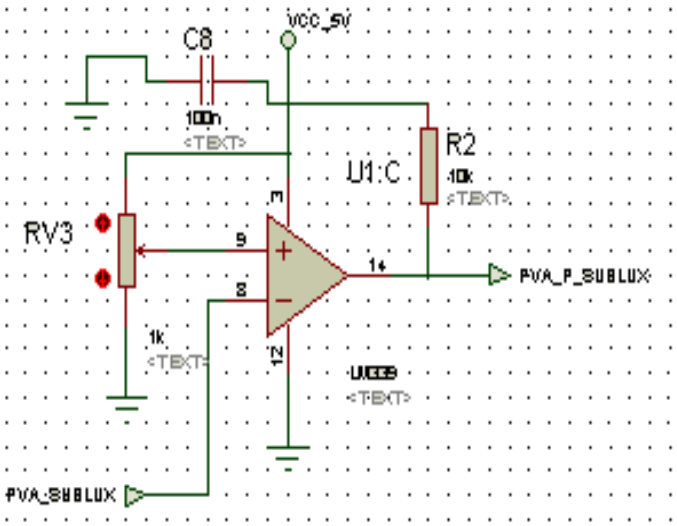


Figura 7.28. Esquema del sensor de subluxación

Se practicó la RCP sobre el maniquí de manera correcta e incorrecta la resucitación cardiopulmonar para poder ajustar el potenciómetro con ayuda de un experto.

7.2.4.4.4 Sensor objeto extraño

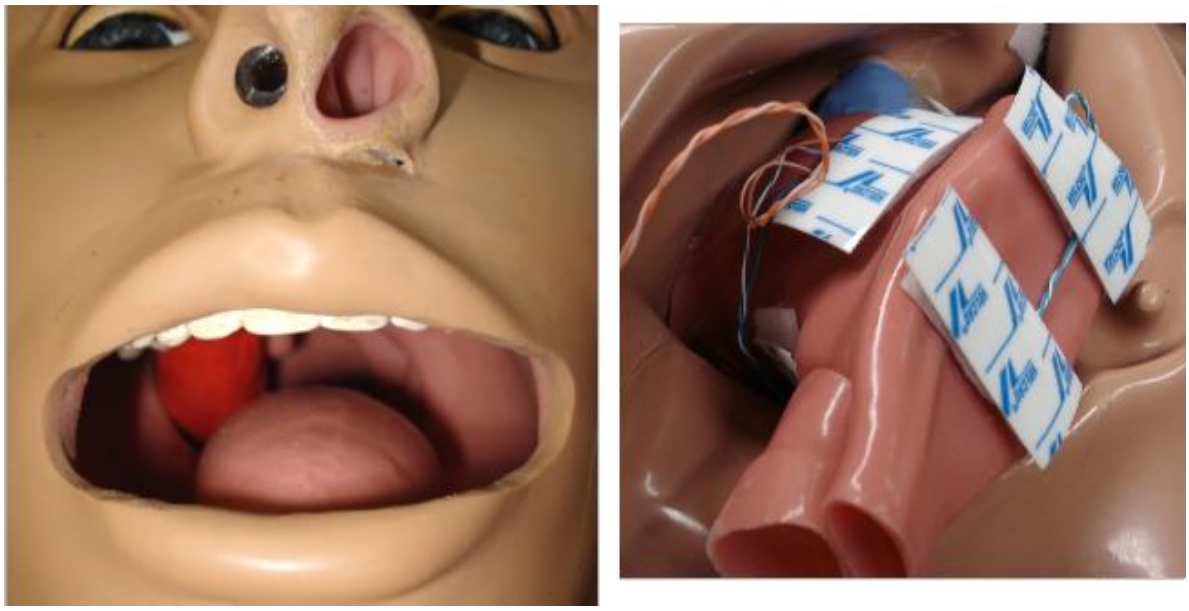
7.2.4.4.4-1 Esquema general

Para activar este sistema se requiere de un objeto extraño (imán cubierto con una resina) que entre en contacto con sensores magnéticos (normalmente abiertos y en reposo) ubicados en la garganta del simulador.

Figura 7.29. Imán permanente recubierto de resina, objeto extraño.

El imán es de 3400 a 3700 Gauss y tiene forma de ovoide alargado. La cubierta o recubrimiento del imán es un poliacrilato, formando por monómero y resina que al mezclarse lo forman, a la resina se le adicionó pintura roja para darle un aspecto más atractivo.

Los sensores magnéticos se encuentran conectados entre sí, en paralelo y en serie, con una resistencia de 10kΩ con voltaje de 5V. Cuando el imán se introduce en la boca del maniquí es suficiente un solo sensor para detectar el objeto extraño. Cuando la garganta queda bloqueada por lo menos uno de los sensores detecta una señal del microcontrolador que es interpretada a nivel de *software* como una obstrucción en vía aérea.



a) **b)**

Figura 7.30. Sensor de objeto extraño
a) Cavidad bucal con el objeto extraño b) Sensores acoplados.

7.2.4.4.4-2 Adecuación de la señal

Para acoplar la red de sensores magnéticos con el microcontrolador se utilizó la configuración *pull-up*. Cuando no hay un campo magnético suficientemente intenso los interruptores permanecen abiertos, entonces, el nodo que está conectado a los interruptores tiene un voltaje de 5V, debido a que no circula corriente por la resistencia no hay caída de voltaje. Cuando se cierra por lo menos un interruptor de la red el nodo queda directamente conectado a tierra, por lo tanto tiene un voltaje de 0V. De esta forma la red de interruptores magnéticos entrega al microcontrolador una señal digital que mantiene un "1" lógico en tanto no se obstruya la vía aérea y un "0" lógico cuando se obstruye.

7.2.4.4.4-3 Calibración

La calibración de este sensor se hace de manera visual. Si el objeto extraño se encuentra dentro de la boca debe activarse el indicador, al ser retirado debe desactivarse. No deben darse falsos positivos ni falsos negativos.

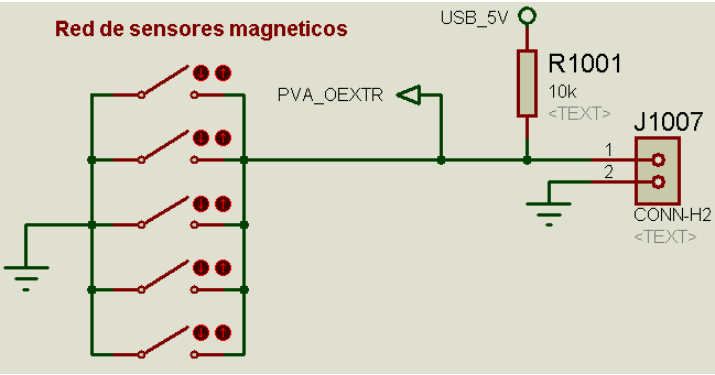


Figura 7.31. Acoplamiento de los sensores magnéticos.

7.2.4.5 Actuador de simulación de pulso

7.2.4.5.1 Esquema general

Se instaló un sistema que simula el pulso carotideo con el fin de dotar de mayor realismo las prácticas de RCP básica. Este sistema puede ser controlado, es decir, se puede prender, apagar o cambiar su velocidad a voluntad del instructor.

El motor de DC se controla por medio del control de modulación de ancho de pulso y utilizando un temporizador del microcontrolador.

Para este proyecto sólo se implementó la simulación del pulso carotideo debido a que la guía de la AHA propone tomar únicamente el pulso carotideo como signo vital.

El principal inconveniente que presenta este actuador es el sonido que produce el mecanismo de los engranes, así que fue necesario construirle una carcasa.

La carcasa está recubierta por la parte interior con aislante sonoro, mide 8.3 centímetros de profundidad, diez centímetros de largo y 6.5 de ancho.

Figura 7.32. Carcasa del actuador de pulso.

El actuador de pulso se instaló en uno de los muslos huecos del maniquí y, al igual que la carcasa, se recubrió internamente por un aislante de sonido, lo que permitió eliminar el ruido de los engranes.

Figura 7.33. Sistema de pulso instalado.

7.2.4.5.1 Etapa de Potencia

La señal de control utiliza un transistor *mosfet* como amplificador de corriente, o sea, cuando el pin CCP1 enciende el transistor se satura y el motor se activa, es un interruptor electrónico.

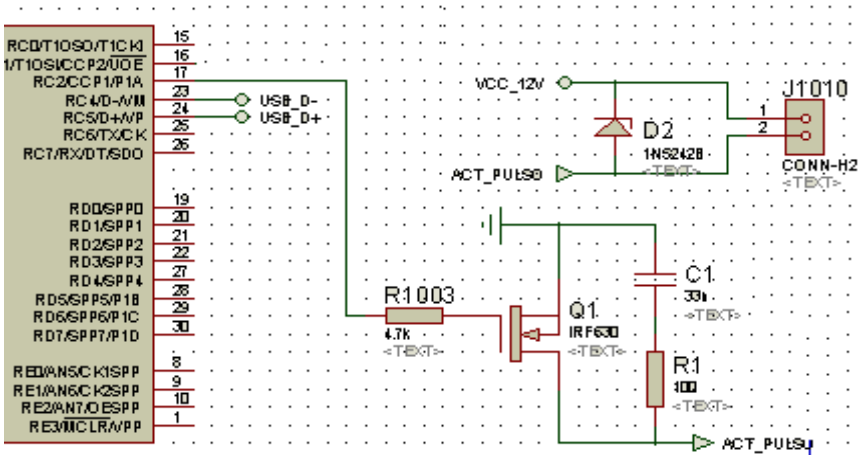


Figura 7.34. Esquema de la etapa de potencia del pulso.

7.2.4.5.3 Calibración

La calibración de este actuador se realizó con la ayuda de un médico que debería detectar el pulso del simulador en cualquiera de las arterias carótidas, exigiendo las posiciones anatómicamente correctas.

7.2.4.6 Actuador de simulación de ventilación

7.2.4.6.1 Esquema general

El objetivo principal de este actuador es permitirle al practicante de RCP ver, sentir, y escuchar la insuflación del maniquí para acercarlos a una situación real.

El funcionamiento de este sistema depende, en gran medida, del fuelle ubicado en la parte interior de la espalda del maniquí que es accionado por un motor de DC PWM. Al levantarse el fuelle este se llena de aire y al descender al aire es conducido por una manguera rígida hasta la fosa derecha de la nariz, obteniendo además un sonido semejante al de la respiración humana por efecto del fuelle.

El sistema de ventilación no interfiere con el sensor de flujo puesto que cada sistema tiene una vía neumática diferente, o sea, son circuitos distintos.

Figura 7.35. Terminal de la manguera en la cavidad nasal

7.2.4.6.2 Acoplamiento de la señal de control

La señal de control utiliza un transistor *mosfet* con la misma configuración del motor para el sistema de pulso, en otras palabras, cuando el pin CCP2 enciende el transistor se satura, entonces el motor se activa.

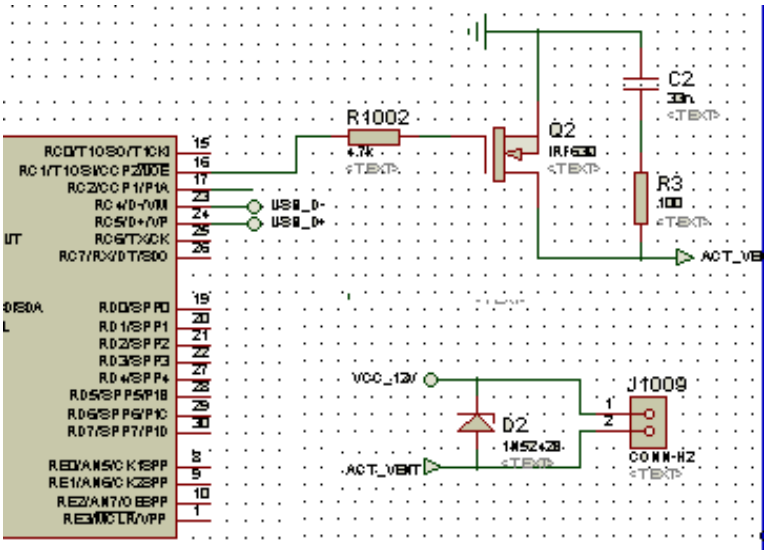


Figura 7.36. Esquema de la señal de control

7.2.4.6.3 Calibración

La calibración de este actuador se realizó con la ayuda de un médico a quien se le solicitó detectar la respiración de dos maneras distintas, observando el movimiento del pecho y escuchando y sintiendo la respiración en la nariz.

7.3 Unidad de procesamiento

7.3.1 Esquema general

La unidad de procesamiento es la encargada de analizar los datos adquiridos de los sensores así como el estado de los actuadores en tiempo real durante la aplicación. Parte de estos datos son enviados al mando a distancia para ser desplegados, como el número de compresiones realizadas, los centímetros de la compresión, también los litros insuflados, entre otros. Los datos también son almacenados en la memoria RAM del CPU, cuando la aplicación es finalizada estos se envían a una base de datos MySQL mediante la red (WiFi).

A continuación se describirán las tres capas de la unidad de procesamiento:

a) Hardware
b) Sistema operativo
c) Aplicación

7.3.2 Hardware

Se utilizó la *netbook* debido a sus dimensiones, pues estas hacen sencilla su instalación en la cavidad abdominal del simulador.

La *netbook* fue modificada para que pudiera ser encendida desde fuera por un botón externo, también puede ser alimentada desde un conector externo, incluso se utiliza la batería de la misma para alimentar la tarjeta principal.

Se eligió usar un cable CAT 5, pues también estos cables serán utilizados para transportar energía, se utilizaron dos para cada señal, excepto para el interruptor de encendido, es uno para cada lado del interruptor, como se muestran en la figura 7.36.

Las señales son obtenidas directamente de la tarjeta madre, esto se logró soldando los cables del CAT 5, en los lugares resaltados en la figura 7.37.

Figura 7.37. Configuración cable.

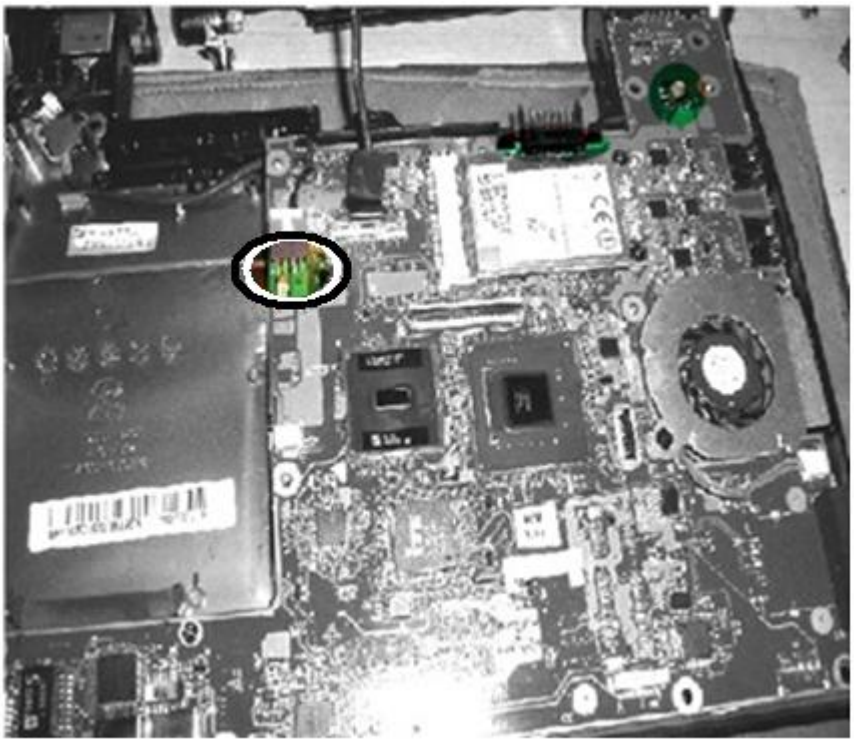


Figura 7.38. Tarjeta madre de la Netbook

7.3.3 Sistema operativo

El sistema operativo utilizado fue una distribución de Linux llamada Fedora versión once, se instaló con los aditamentos mínimos.

Una de las características utilizadas para el desarrollo de la aplicación fue el *usbmonitor* en el *debug filesystem* que contiene Linux, el cual sirve para capturar los datos que son transferidos mediante el USB. Un ejemplo de la información que se observa es el siguiente:

```
cd419d80    2772257915  S   Co:3:003:0  s  41 08 10c0 0000 0008 8 = 90211c00
00000010
cd419d80    2772258146  C   Co:3:003:0  0  8 >
cd419e40    2772261950  S   Ci:3:003:0  s  c1 00 10c0 0000 0001 1 <
cd419e40    2772262145  C   Ci:3:003:0  0  1 = 94
```

1 *Urbtag:* La primera palabra (cd419d80) es la dirección USB *Request Block* que indica dónde se localiza en la memoria del *kernel* la estructura del *request*.

2 *Timestamp:* Es el tiempo del microprocesador (2772257915), su valor en milisegundos depende de la frecuencia del reloj.

3 *Transfer Type:* Indica la dirección de la URB:

- S La información es transmitida de la PC al dispositivo
- C Es la respuesta del dispositivo
- E Significa que ocurrió un error durante la transmisión

4 *Detalles de la transferencia:* La primera letra de la palabra indica el tipo de transferencia; C para *Control*, Z para *Isochronous*, I para *Interrupt*, B para *Bulk*. La segunda letra indica la dirección; i para *input*, o para *output*. El siguiente señala el número de *bus* seguido por el identificador del dispositivo, para terminar, el número *endpoint*.

5 *URB data:* La siguiente información depende del tipo de URB, por ejemplo, las transferencias mostradas son transferencias de control, por lo tanto se muestra la letra "s" seguida por los valores del bm *RequestType*, b *Request*, w V*alues*, w *Index* y w *Length*, como se puede observar contiene ocho *bytes* de datos que mostrados después del signo "=".

Una desventaja de utilizar el *usbmon* es que al momento de ser cargado en el *debug filesystem* los URB son procesados de una manera más lenta, esto puede afectar el comportamiento de los dispositivos, por ello debe tenerse en cuenta cuando se utilice.

Otro aspecto de Fedora que también sirvió para este proyecto fue su flexibilidad para cambiar el modo en que el sistema se inicia, ya que al estar el CPU dentro de la aplicación era necesario iniciar de manera autónoma y rápida. Esta flexibilidad ya no hace necesario cargar alguna librería de la interfaz gráfica de Linux. Cada parte del inicio del sistema puede ser modificado para aumentar la velocidad de encendido.

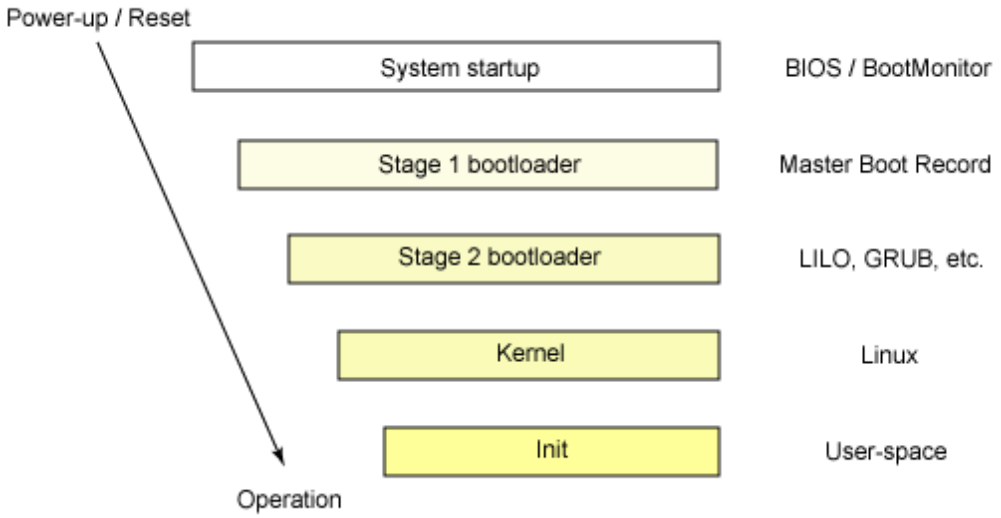


Figura 7.39. Proceso de inicio de la unidad de procesamiento.

El principio del proceso de arrancado inicia con el *Basic Input/Output System* BIOS. Ya que el programa BIOS fue cargado hace pruebas dentro del sistema, busca los periféricos y los configura, al terminar busca un dispositivo de arranque que contenga un *Master Boot Record* MBR, que es de 512 *bytes*. Este proceso dura de tres a cinco segundos, puede ser acelerado al modificar las pruebas que lleva a cabo el BIOS así como al deshabilitar la autodetección de nuevos dispositivos instalados.

La siguiente función en este proceso se divide en dos etapas, la primera es cuando se ejecuta el código en el MBR, que en el caso del GRUB *boot loader* (el utilizado en esta aplicación) redirecciona; la segunda etapa es la de arranque que no cabe en los 512 *bytes* asignados por el MBR. La finalidad de este ciclo es proporcionarle al usuario distintos sistemas operativos o *kernels* para elegir, este lapso dura desde un segundo hasta diez. Esta segunda función del proceso se puede acelerar al no cargar la parte gráfica o al tener un sistema operativo predeterminado para cargarse.

Una vez que el *kernel* es cargado se encarga de iniciar y configurar la memoria del sistema, además de otros periféricos incluyendo los discos duros. El *kernel* carga todos los controladores necesarios para la comunicación con el *hardware*, este proceso lleva de tres a cinco segundos y se puede acelerar recompilando el *kernel*, pues este normalmente contiene muchos controladores que no son utilizados por el sistema, por lo tanto pueden ser eliminados, esto no es sencillo ya que un error en estos controladores podría hacer que el sistema no arranque.

Al terminar de cargarse el *kernel* ejecuta el programa *sbin/init*.

La última etapa de arranque es efectuada por el programa *sbin/init* que ejecuta todos los *scripts* de inicio. El primer *script* que entra en función es el *etc/rc.d/rc.sysinit*, el cual inicia el *swap* así como también los sistemas de archivos, de la misma manera prepara lo necesario para la iniciación, por ejemplo lee e inicia el archivo etc/sysconfig/clock, que contiene la configuración del reloj del sistema.

El sistema después debe seleccionar el *runlevel* (se encuentra en /etc/inittab).

Runlevel	Directorio Scripts	State
0	/etc/rc.d/rc0.d/	Shutdown
1	/etc/rc.d/rc1.d/	Modo un solo usuario
2	/etc/rc.d/rc2.d/	Multiusuario sin servicios de red
3	/etc/rc.d/rc3.d/	Multiusuario solo consola
4	/etc/rc.d/rc4.d/	Reservado
5	/etc/rc.d/rc5.d/	Modo XDM X-windows GUI
6	/etc/rc.d/rc6.d/	Reboot

Tabla 7.2 Runlevels de Fedora

El *runlevel* utilizado para el desarrollo del simulador fue el "1", de un solo usuario, ya que es el que menos tareas ejecuta. Sin embargo debieron agregarse ciertos *scripts* para que el sistema funcionara correctamente. El tiempo promedio de encendido de la aplicación es de 40 segundos, pero puede variar si no se encuentra una red inalámbrica disponible para conectarse.

7.3.4 Aplicación

La aplicación fue escrita en Java. El lenguaje toma mucha de la sintaxis de C y C++, pero tiene un modelo más simple, aunque elimina herramientas de bajo nivel.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible. El objetivo es hacer que Java pueda ser portable entre distintos sistemas operativos, sin tener que modificar el código, ni tener que recompilarlo. Aunque esta característica también provoca que sus tiempos de ejecución sean más largos debido a que debe pasar por la *JAVA Virtual Machine* (JVM). Algunos procesadores ya están incluyendo el *bytecode* de Java en su *set* de instrucciones para una ejecución rápida.

La aplicación se forma por varias tramas (*Threads*) debido a que se necesita que el código sea ejecutado en paralelo para su buen funcionamiento.

Figura 7.40. Conformación de las tramas

Cada uno de estas tramas opera independientemente, pero comparte variables y funciones con los demás tramas, pueden ser creados y destruidos por su trama creador, por lo tanto, no es necesario que existan las cuatro tramas durante la ejecución de la aplicación.

7.3.4.1 Trama principal

La trama inicia con la búsqueda del dispositivo USB en los *buses* del CPU, este debe contener el descriptor correcto para ser iniciado, luego se inician las variables, después el cargado de los sonidos de la aplicación así como las imágenes del mando a distancia, ambas en la memoria RAM para su rápido acceso, entonces se inician las tramas de adquisición de datos USB y la trama de sonido, y finalmente se manda el de que el CPU se encuentra encendido y listo al MCU.

Entonces sigue una serie de procesos que se repiten durante toda la ejecución de la aplicación. Lo primero que se debe hacer es verificar si el estado de la batería, mostrado en el mando, es correcto o ha cambiado. Después hay una revisión encargada de detectar nueva información al interior del MCU disponible, la trama de adquisición de datos USB es el encargado de actualizar estos datos. En caso de la falta de datos nuevos implicaría que la aplicación está pausada o ha terminado, si hay nuevos datos la trama principal los procesa de esta manera:

1. Convierte los datos RAW de los sensores en valores físicos mediante los datos de calibración que estos mismos le proporcionan.
2. Analiza los datos de calibración para encontrar compresiones o ventilaciones.
3. Utiliza las compresiones encontradas para generar las presiones y flujos sanguíneos.
4. Utiliza las ventilaciones para calcular el intercambio de gases en pulmones y sangre.
5. Finalmente marca los datos como procesados.

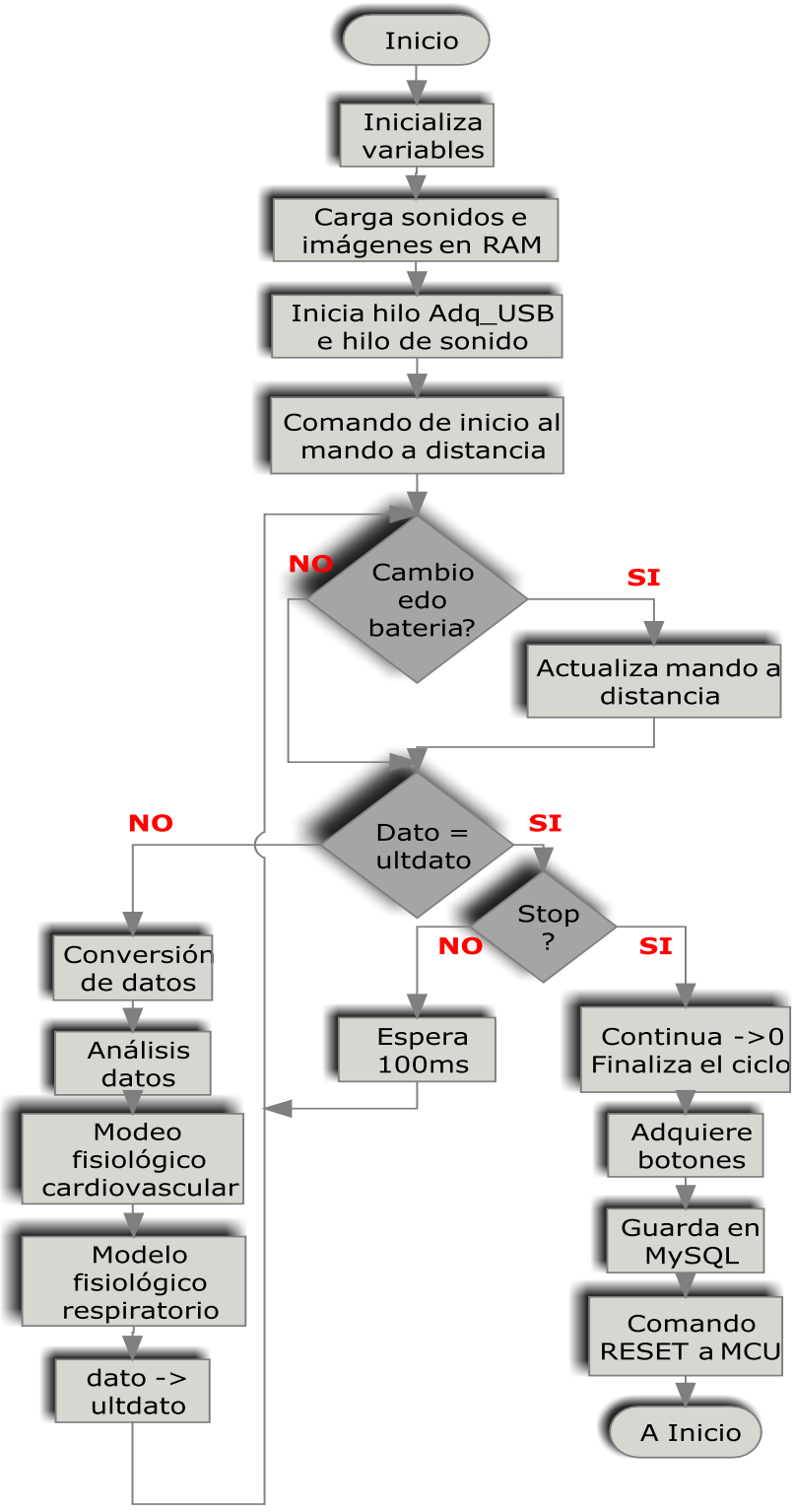


Figura 7.41. Diagrama de flujo de la trama principal.

Al terminar la aplicación se ejecutan los siguientes pasos:

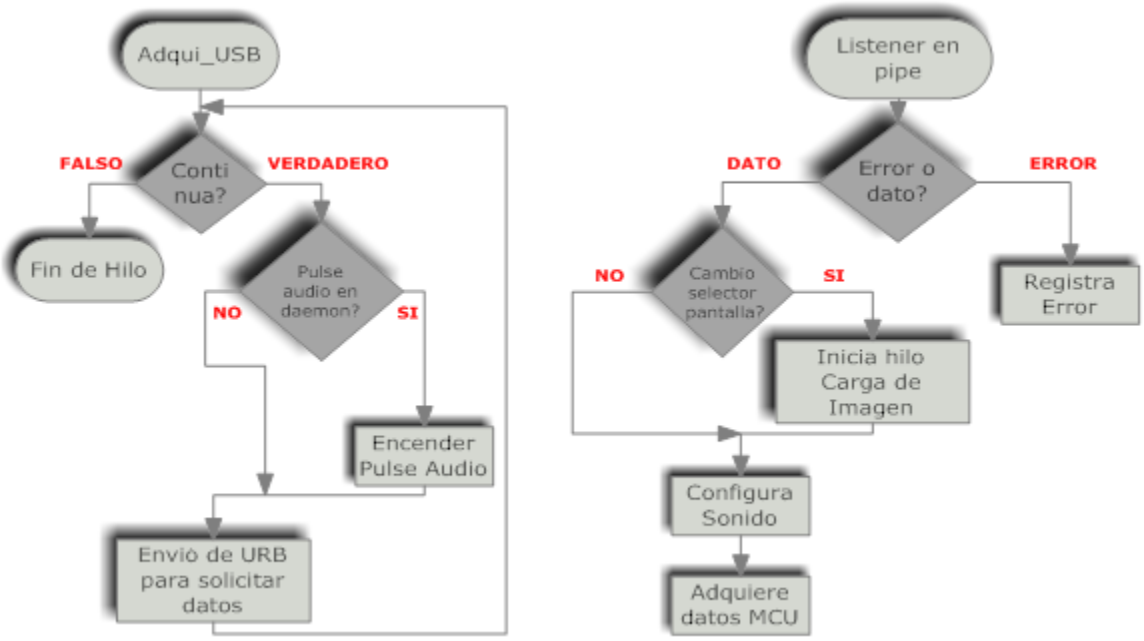
1. Se marca la variable continua como falsa, siendo esta la que controla al trama de adquisición para que este continúe o no ejecutándose; se finaliza los datos del ciclo, como son el número de compresiones, la frecuencia e intensidad promedio de las mismas en el ciclo, etcétera

2. Se adquieren los cronómetros de la práctica, los cuales se encuentran almacenados en el MCU.

3. Se envían todos los datos al servidor MySQL para almacenarlos ya que posteriormente serán procesados en el servidor.

4. Se envía el comando *reset* al MCU, el cual reinicia los datos del mismo.

7.3.4.1 Trama Adquisición USB



a) **b)**

Figura 7.42. Diagramas de flujo del trama de adquisición USB
a) Ciclo principal b) El listener

La figura 7.41 muestra el diagrama de flujo del trama de adquisición USB, el cual es cargado en el inicio del trama principal, al iniciarse el trama activa un *listener*[4], cuando llega un dato del USB es el encargado de procesarlo.

El trama sólo se detiene cuando la variable continua es falsa, también es el encargado de revisar que el proceso de PulseAudio[5] se mantenga corriendo, ya que si este se detiene no podría escucharse ningún sonido. Finalmente se encarga de colocar los URB para que el sistema operativo se encargue de mandarlos mediante el *bus*.

[4] Listener es el nombre en inglés que se le da a una función que se encuentra a la espera de un evento, y es activada cuando este evento sucede.

[5] PulseAudio es un servidor de sonido multiplataforma, es utilizado en los sistemas Linux.

Cuando un dato que había sido solicitado por una URB llega desde el *bus* activa el *listener*, el cual también es capaz de capturar los errores durante la transmisión.

Lo primero que se verifica en los datos es si el selector de pantalla fue modificado, de haber sido cambiado el trama carga otro trama que tiene la tarea de enviar la imagen al MCU maestro para que este los vuelva a enviar al mando a distancia.

De manera consecutiva al envío, el trama configura el sonido, así se mantiene alerta por si es necesario hacer algún cambio, por ejemplo, si durante una compresión no es adecuado el tono del sonido bien se puede manipular. Y, finalmente el trama convierte los datos del MCU que vienen en paquetes de ocho *bits* sin signo en enteros de 32 bits con signo.

La razón por la que el proceso de cargar imágenes debe ejecutarse de manera consecutiva al proceso de adquisición es que este último no puede detenerse cuando hay un cambio de pantalla.

7.3.4.3 Trama cargado de imagen

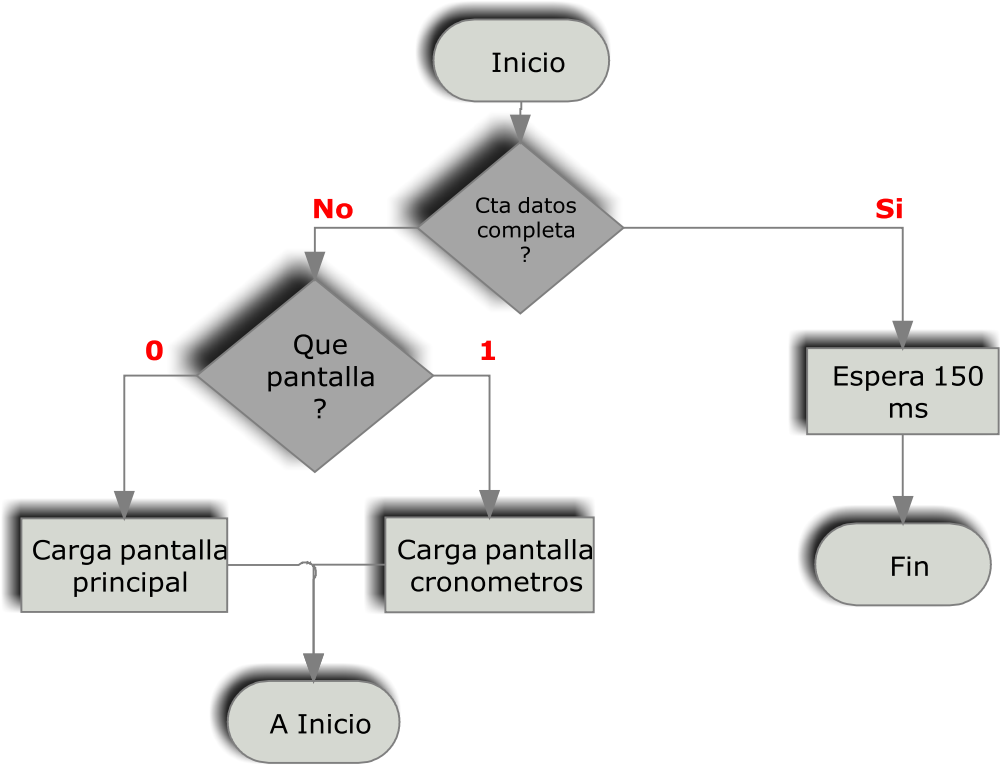


Figura 7.43. Diagrama de flujo del trama de cargado de imágenes

El trama es iniciado cuando se detecta el cambio de selector de pantalla en el mando a distancia, se inicia un ciclo *for* donde se mandan todos los pixeles en el formato especificado en la sección 7.2.3.2. Estos pixeles son enviados en paquetes de 32 *bytes* vía USB, teniendo en cuenta que son 132x132 pixeles resultan 17424 pixeles, cada uno de doce *bits*, dando un total de 25136 *bytes*. El ciclo debe ser corrido 817 veces para poder transferir por completo la imagen, la cual debe ser extraída de la memoria RAM previamente cargada

en el trama principal.

Al finalizar debe esperar 150 milisegundos para permitir que el mando a distancia determine que ha terminado de cargar la pantalla y retire la bandera. Si no existiera este retraso la carga de pantalla podría entrar en un *loop* infinito.

7.3.4.4 Trama de sonido

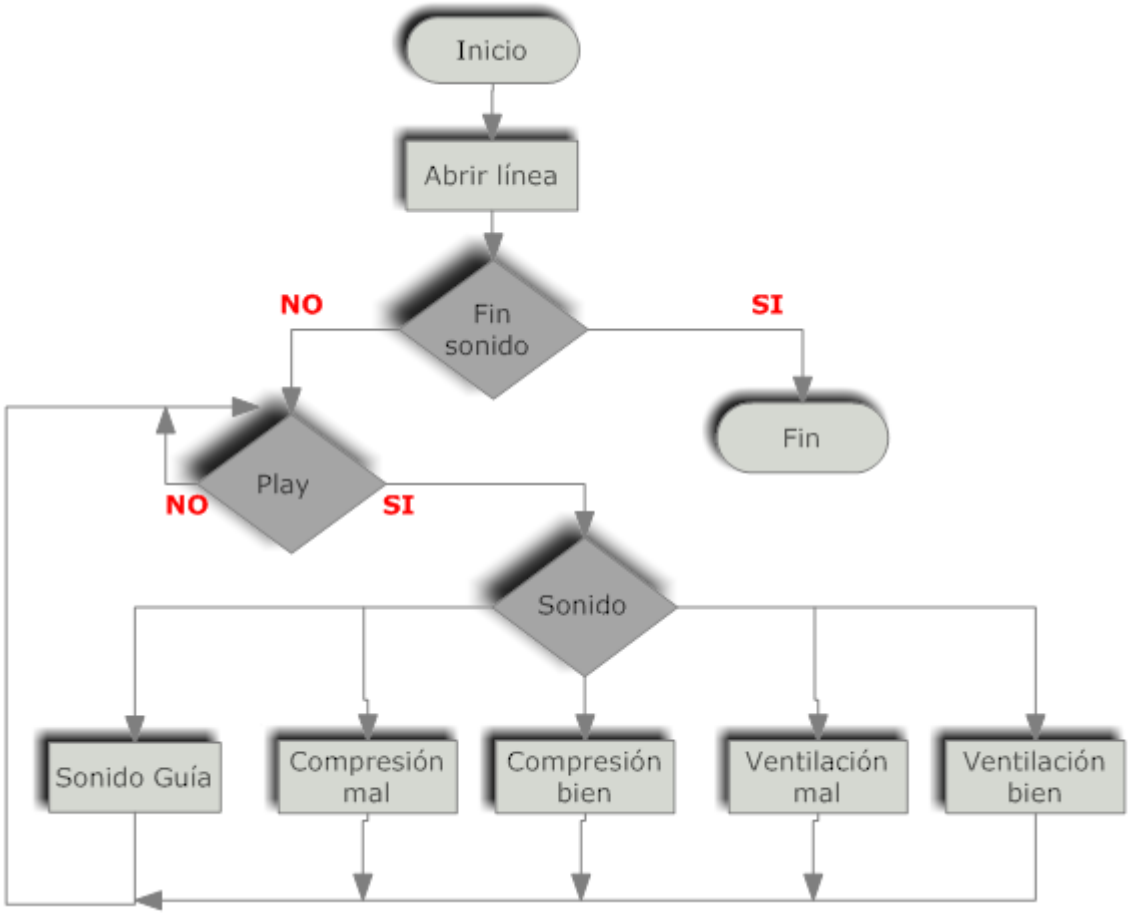


Figura 7.44. Diagrama de flujo de la trama de sonido.

El trama de sonido es iniciado cuando el mando a distancia le cambia la bandera.

Cuando el sonido es pausado el trama entra en un *loop* infinito, pero al estar activado es necesario que mantenga el flujo de datos con el *buffer* del mezclador de sonido del CPU. Este flujo de información debe ser lo suficientemente grande para que el *buffer* de sonido no se atrase y exista una condición de *underflow*, pero también lo suficientemente pequeño para que cuando deba existir un cambio en el sonido se realice sin mucho retraso. Por ejemplo, si se está escuchando el sonido guía y se realiza una compresión efectiva, es recomendable que el sonido cambie instantáneamente, pero esto no es posible ya que el *buffer* del mezclador tiene la información del sonido guía, entonces deberá esperar a que termine este *buffer* para tocar el nuevo sonido. Por estas razones el sonido debe correr en trama en paralelo, así opera correctamente.

7.4 Servidor

7.4.1 Esquema general

El servidor es el encargado de recibir los datos de la aplicación, almacenarlos en su base y servir como plataforma para verlos.

Para los fines de este proyecto el *hardware* de la aplicación y el sistema operativo del servidor son indistintos, pero ambos deben contar con una base de datos MySQL y con un servidor web capaz de soportar PHP y *Java Applets*. Se recomienda que se cuente con una IP fija para su fácil localización mediante la *World Wide Web*.

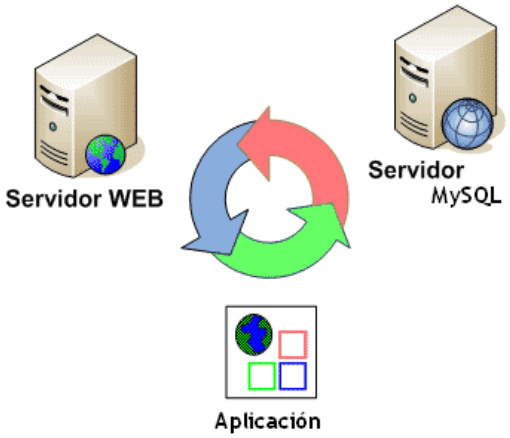


Figura 7.45. Diagrama funcionamiento servidor.

El servidor puede estar o no localizado en la intranet de la aplicación, sólo es necesario administrar los permisos de escritura y reescritura de la base de datos, dependiendo de la localización del servidor.

7.4.2 Base de datos

La base de datos fue realizada en MySQL que es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL AB, MySQL es un *software* de código abierto licenciado bajo la GPL de la GNU.

El lenguaje de programación que utiliza MySQL es *Structured Query Language* (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

En las últimas versiones se pueden destacar las siguientes características principales:

- El principal objetivo de MySQL es velocidad y robustez.
- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, es decir, puede trabajar en distintas plataformas y sistemas operativos.
- Cada base de datos cuenta con tres archivos, uno de estructura, otro de datos y el de índice; soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso gracias a su implementación multitrama.
- Sistema flexible de contraseñas (*passwords*) y gestión de usuarios, con muy buen nivel de seguridad en los datos.

- El servidor soporta mensajes de error en distintos idiomas

Una ventaja es su velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento, otra es su bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema. También soporta gran variedad de sistemas operativos por lo tanto es posible su implementación prácticamente en cualquier equipo.

La base de datos está formada por distintas tablas:

- Tabla General. Guarda los datos de todas las ejecuciones, se utiliza para hacer estadísticas de las mismas.
- Tabla de compresiones. Contiene la información de las compresiones, es única para cada ejecución.
- Tabla de ventilaciones. Almacena la información de las ventilaciones, es única para cada ejecución.
- Tabla de modelo fisiológico. Es donde está la información del modelo fisiológico, es utilizada en la visualización de la aplicación y es única para cada ejecución.

La descripción de las tablas es la siguiente:

Columna	Tipo	Tamaño (bytes)	Descripción
Id	**INT**	**4**	**Identificador único de la ejecución de la aplicación.**
Fecha	**DATE**	**3**	**Fecha de la ejecución.**
Hora	**TIME**	**3**	**Hora de la ejecución.**
Grupo	**SMALLINT**	**2**	**Grupo al cual se pertenece (definido por usuario).**
Equipo	**SMALLINT**	**2**	**Equipo al cual se pertenece (definido por usuario).**
Id_E	**SMALLINT**	**2**	**Estudio al cual se pertenece (definido por usuario).**
Instructor	**CHAR**	**50**	**Instructor a cargo (definido por usuario).**
Pasante	**CHAR**	**50**	**Pasante a cargo (definido por usuario).**
Visible	**BIT**	**1 bit**	**Información visible en la web**
T_Total	**SMALLINT**	**2**	**Tiempo total de ejecución (en segundos).**

Tabla 7.3 Tabla general - base de datos - Parte 1

La siguiente tabla se repite en cada ciclo y el signo # indica el número de ciclo.

Columna	Tipo	Tamaño (bytes)	Descripción
C_#_TV	SMALLINT	2	Tiempo de las ventilaciones en el ciclo.(en segundos)
C_#_NV_E	TINYINT	1	Número de ventilaciones efectivas
C_#_NV_NE	TINYINT	1	Número de ventilaciones no efectivas
C_#_NC	SMALLINT	2	Tiempo de las compresiones en el ciclo.(en segundos)
C_#_NC_E	SMALLINT	2	Número de compresiones efectivas
C_#_NC_NE	SMALLINT	2	Número de compresiones no efectivas
C_#_FC	TINYINT	1	Frecuencia promedio de compresiones en el ciclo
C_#_IC	FLOAT	4	Intensidad promedio de compresiones en el ciclo

Tabla 7.4 Tabla General - base de datos - Parte 2

La siguiente tabla se repite para cada cronómetro, el signo # indica el número del cronómetro.

Columna	Tipo	Tamaño (bytes)	Descripción
TB_#	SMALLINT	2	Tiempo del cronómetro # (en segundos)
TB_#_E	BIT	1 bit	Segundo click en cronómetro #

Tabla 7.5 Tabla General - base de datos - Parte 3

Columna	Tipo	Tamaño (bytes)	Descripción
tCompr	SMALLINT	2	Tiempo de la compresión (en segundos)
iCompr	FLOAT	4	Intensidad de la compresión (en centímetros)

Tabla 7.6 Tabla Compresiones - base de datos

Columna	Tipo	Tamaño (bytes)	Descripción
tVent	**SMALLINT**	**2**	**Tiempo de la ventilación (en segundos)**
iVent	**FLOAT**	**4**	**Intensidad de la ventilación (en litros)**
Gasto	**FLOAT**	**4**	**Gasto cardíaco en cada ciclo (en litros/min)**

Tabla 7.7 Tabla Ventilaciones - base de datos

Columna	Tipo	Tamaño (bytes)	Descripción
Paor	**SMALLINT**	**2**	**Presión en la arteria aórtica (mmHG) contra tiempo**
PaO2	**SMALLINT**	**2**	**Presión parcial oxígeno arterial (mmHG) contra tiempo**
PaCO2	**SMALLINT**	**2**	**Presión parcial CO2 arterial (mmHG) contra tiempo**
pH	**FLOAT**	**4**	**PH contra tiempo**
SaO2	**TINYINT**	**1**	**Saturación de oxígeno contra tiempo**

Tabla 7.8 Modelo Fisiológico - base de datos

La cantidad de *bytes* utilizados para cada práctica varía dependiendo del tiempo de duración, para una práctica de tres minutos es de aproximadamente 341 *kbytes*.

7.4.3 Aplicación

La aplicación es una *Java Applet*, su objetivo es informar sobre los datos recabados durante la aplicación, así como los datos teóricos de la reanimación cardiopulmonar. Como se muestra en la figura 7.45.

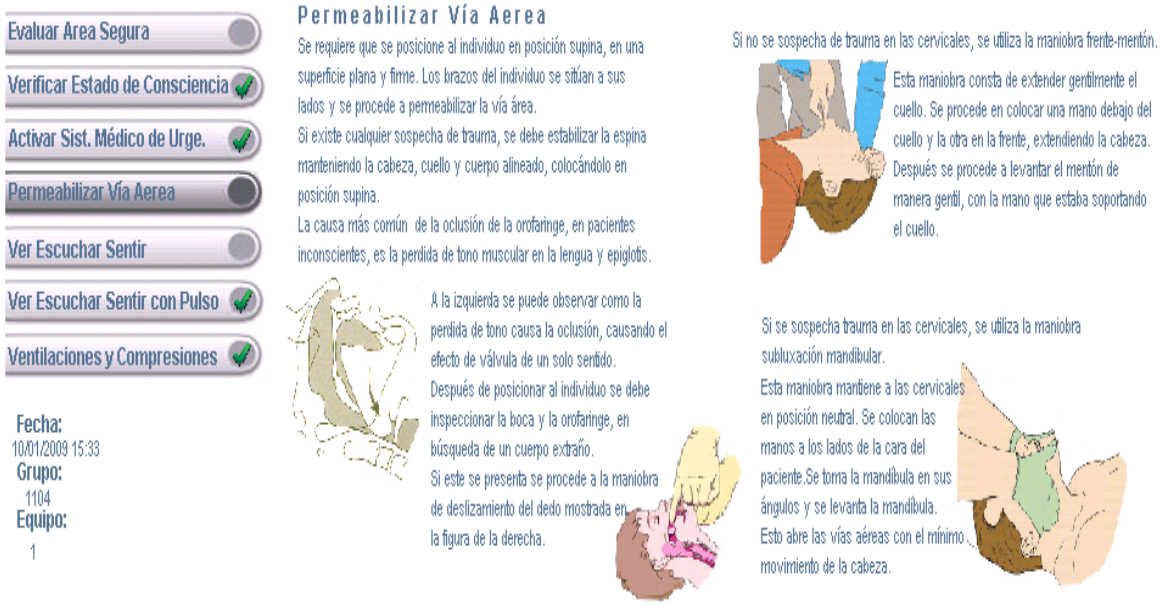


Figura 7.46. Parte de la interfaz con la información teórica.

Cada uno de los pasos del protocolo tiene la información teórica, con esto se pretende dar ayuda visual y teórica de la reanimación cardiopulmonar.

La aplicación desarrollada también cuenta con una línea de tiempo que representa las acciones realizadas por el usuario durante la aplicación,

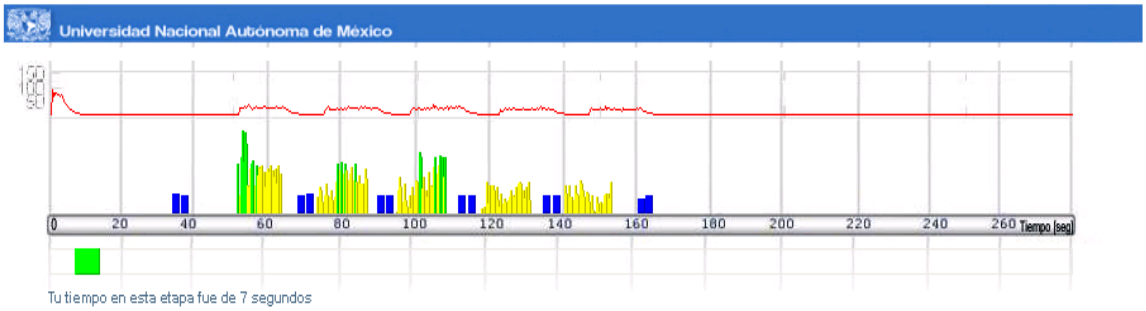


Figura 7.47. Parte de la interfaz con la línea de tiempo.

Las líneas verdes corresponden a las compresiones realizadas y las azules a las ventilaciones.

La interfaz además de mostrar el resumen recopilado durante la práctica, el número de compresiones por ciclo y la frecuencia de las compresiones por ciclo, también muestra una ampliación de los datos de la línea de tiempo señalada por un puntero con los valores

del modelo fisiológico en tal punto.

Figura 7.48 Parte de la interfaz que muestra las compresiones y el modelo fisiológico.

Cada usuario podrá observar sus resultados así como una guía teórica de los actos que debieron seguirse para ayudar al aprendizaje de la reanimación cardiopulmonar básica.

La página donde el usuario selecciona la ejecución que desea observar no ha sido implementada, únicamente se puede seleccionar por su número de Id.