UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE UN SIMULADOR PARA REANIMACIÓN CARDIO PULMONAR

T E S I S

PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A N

FRANCISCO SUÁREZ ESCAMILLA
SINUE DANIEL GÓMEZ GARCÍA
CARLOS ARMANDO VILLAGÓMEZ HOYOS

DIRIGIDA POR

M.I. JUAN MANUEL GÓMEZ GONZÁLEZ

CIUDAD UNIVERSITARIA
MAYO 2010

Índice

Índice de Figuras

Capítulo 2 Marco conceptual

Capítulo 3 Marco Referencial

Capítulo 7 Técnica

Apéndice

Índice de Tablas

Capítulo 3 Marco Referencial

Capítulo 7 Técnica

Apéndice

1 Resumen

En este trabajo de tesis se presenta el desarrollo de un simulador para la enseñanza de la reanimación cardiopulmonar básica, elaborado en el Laboratorio de Ingeniería Biomédica de la Facultad de Ingeniería con la colaboración del Centro de Enseñanza y Certificación de Aptitudes Médicas (CECAM) a través del proyecto PAPIME PE104009, ambas instituciones dependen de la Universidad Nacional Autónoma de México.

Este documento aborda la implantación de sensores analógicos y digitales para la detección de las distintas maniobras realizadas durante la reanimación cardiopulmonar, el acondicionamiento de las señales y el procesamiento de las mismas.

También se explica el funcionamiento de los actuadores fisiológicos que son utilizados para dar mayor realismo al simulador durante su utilización.

Las interfaces con el usuario son tratadas en este documento, es decir, tanto la que es desplegada en un dispositivo portátil durante la realización de la maniobra, como la interfaz que se encuentra disponible vía internet después de la ejecución de la simulación.

2 Marco Conceptual

2.1 Proceso enseñanza – aprendizaje

El hombre tiene la necesidad de comprender, manejar y dominar el conocimiento; a partir de esto se plantea paradigmas socioculturales complejos que recaen en el proceso enseñanza–aprendizaje. Dicho proceso explota las capacidades del sujeto para aprender, llevando el desarrollo humano, social, científico y cultural a una relación dialéctica. Debemos cuestionar y mejorar los elementos del proceso docente educativo para alentar la reflexión y la postura crítica de problemas, situaciones y hechos que estimulen la investigación educativa como fuente esencial para el desarrollo social y humano.

El aprendizaje se obtiene de la interacción del hombre con sus distintas actividades, como el estudio, el trabajo, la experimentación o el simple acto de jugar; incluso puede surgir como resultado de la actividad psíquica, lo que es entendido como "autoaprendizaje".

La enseñanza es el acto didáctico del profesor (o facilitador del conocimiento) para desarrollar el aprendizaje de los estudiantes. Para comprender el aprendizaje centramos un sujeto activo (alumno) que será concebido como una entidad orientada a un objetivo de construcción de conocimiento.

El sujeto activo en interacción logra aprender a través de los actos didácticos que se producen en su relación con el objeto de aprendizaje mediante la utilización de diversos medios e instrumentos.

El resultado principal del aprendizaje lo constituyen las transformaciones dentro del sujeto, es decir, es visible por medio de las modificaciones físicas y psíquicas que el estudiante puede lograr mediante algún cambio conductual perdurable.

«A lo largo de la historia se han dado múltiples teorías del aprendizaje. Actualmente se está en proceso de nuevas formas para potencializar éste, y existe un consenso bastante amplio en considerar que el aprendizaje es un proceso constructivista, autodirigido, colaborativo y contextual»[1]

El proceso constructivista sitúa al alumno como centro de construcción y reconstrucción del conocimiento de forma activa, es decir, la aplicación de este proceso engloba su experiencia a nivel personal mejorando los procesos cognoscitivos.

La "teoría constructivista" desafía el objetivismo el cual está impregnado de la noción de que el conocimiento y la verdad existen fuera de la mente del individuo. El constructivismo considera el aprendizaje como un proceso subjetivo y sugiere que se debe experimentar el mundo para conocerlo.

El "aprendizaje autodirigido" se puede considerar un método de organización de la enseñanza y el aprendizaje donde las actividades de aprendizaje están en gran medida bajo el control de quien aprende.

Se considera "colaborativo" dado que la construcción del conocimiento siempre comienza de la interacción con otros, y tiene esta característica porque no es una repartición de tareas entre individuos sino la comprensión compartida de un problema trascendente.

[1] María Nolla Domenjó, *El proceso cognitivo y el aprendizaje profesional*, Fundació Doctor Robert. Universiat Autònoma de Barcelona. Revista Educación Médica. Volumen 9, Numero 1, Marzo 2006. p11-16. pag 12. Todas las referencias al aprendizaje profesional de este apartado fueron tomadas del mismo ejemplar.

Así como la retroalimentación social de distintos enfoques en orden de crear un conocimiento consensual con mayor probabilidad de éxito en las dinámicas sociales.

En cuanto a que el proceso constructivista dice que el aprendizaje es "contextual", la idea fundamental es que el aprendizaje se debería realizar en el contexto más parecido al que se deberá aplicar el conocimiento, es decir, lo más parecido a los contextos reales profesionales, incrementando la asimilación de este.

2.1.1 La simulación como método enseñanza – aprendizaje

La simulación, dentro de la tecnología actual, es un arma educativa que le permite al estudiante involucrarse en sus procesos de aprendizaje de manera lúdica y didáctica. La simulación de eventos conlleva al "ensayo" de escenarios intentando recrear, lo más fielmente posible, una situación como a la que en el futuro se tendrá que enfrentar el alumno.

El simulador es una herramienta útil en la enseñanza para la solución de emergencias ya que no está diseñado para que cada individuo recorra un solo camino, sino que persigue desarrollar habilidades y procesar la información que posee para buscar la solución óptima.

El aprendizaje, se concreta dependiendo el contexto en el que se vivió. Así, el conocimiento en el aula no es el mejor contexto de aprendizaje, sino alguno más cercano a la situación profesional, lo más realista posible. En este sentido, es relevante la introducción de simulaciones en las situaciones de aprendizaje para dar contextos prácticos.

2.1.2 Enseñanza médica

La enseñanza médica sigue un protocolo general: primero, el instructor da una explicación teórica dentro del aula; después, se hace un ensayo de los conocimientos en búsqueda de la práctica reflexiva, para unir la experiencia y la teoría; finalmente, el instructor ejemplifica la práctica con un paciente y en contexto real, permitiéndole al aprendiz enfocarse en detalles específicos. Este último paso sólo se puede dar ya que el alumno aprendió a dominar la técnica en sus ensayos.

Nuestro enfoque del proceso constructivista de la enseñanza médica está en la Práctica Reflexiva, figura. 2.1 donde la experiencia lleva al alumno al "conocimiento en acción" con simuladores de eventos clínicos que se aproximen a la fisiología humana real. Los profesionales, basándose en su experiencia práctica, van desarrollando habilidades que con la repetición pueden llegar a convertirse en rutinarias o automáticas. La mayoría de las veces estas habilidades se logran por decisiones intuitivas que el médico aprende a tomar.

El profesional, fuera de la teoría, se enfrenta a muchas situaciones que en la práctica son inciertas, contradictorias, complejas o únicas. A estas situaciones se les llama "sorpresa" y desencadenan un proceso de reflexión en dos tiempos. Una primera reflexión hecha sobre la marcha y la reflexión durante la acción (*reflection in action*). En la segunda reflexión el profesional intenta, en pocos segundos, plantearse el problema y elige alguna reacción que lleve a la solución.

Ya que el profesional ha pasado por la experiencia (el encuentro médico-paciente, por ejemplo), puede reflexionar con más detenimiento sobre lo sucedido (*reflection on action*). A menudo estos momentos de cavilación son informales aunque también se dan en espacios formales, como las sesiones clínicas. La reflexión sobre el acto profesional tiende

a ser aprendizaje añadido al conocimiento que el practicante obtuvo del ejercicio, o bien puede quedar por resolver, siendo aún motivo de "sorpresa". En resumen, aprendemos a partir de la experiencia, pero también hay que reflexionar sobre la práctica.

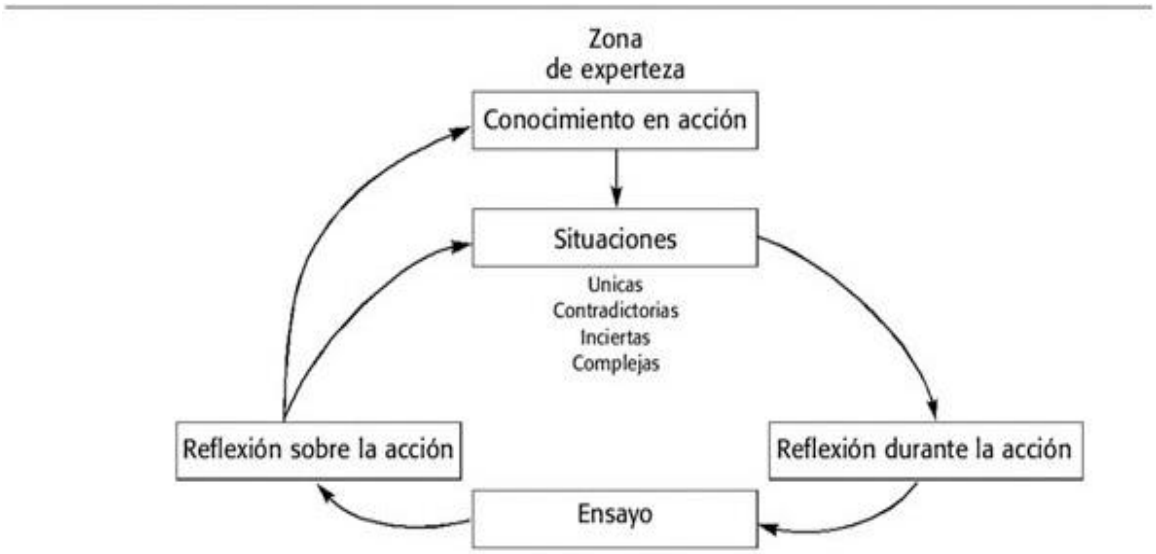


Figura 2.1. Esquema que representa el aprendizaje a partir de la práctica profesional

2.1.2.1 Investigación en la enseñanza médica

El aprendizaje en las ciencias médicas involucra el proceso de razonamiento clínico. Entre los primeros modelos propuestos está el modelo hipotético-deductivo o analítico, su sustento es que ante el paciente el profesional formula una variedad de hipótesis que serán confirmadas o desechadas posteriormente a la evidencia clínica recolectada. El problema de dicho método es que no hay diferencias tangibles entre expertos y novatos. Ambos grupos pueden generar hipótesis en poco tiempo, el único contraste se nota en que las hipótesis de los expertos son más cercanas a la conclusión final, pues explican mejor el problema y son más selectivos con la información, sin embargo, los profesionales usan menos las ciencias básicas en comparación con los novatos.

Entonces se puede inferir que el razonamiento médico, si bien se cimenta en los conceptos de las ciencias básicas, es un razonamiento no analítico (reconocimiento de patrones), más bien se sustenta en la mayor exposición a experiencias clínicas. Los expertos utilizan las experiencias anteriores y logran relacionarlos haciendo que cualquier caso en el futuro entre en una categoría particular. No obstante, cuando los profesionales se enfrentan a situaciones difíciles o ambiguas, sí recurren a explicaciones científicas básicas.

La solución de un caso específico no implica la solución de otros, por ello los docentes deben intentar dar varios ejemplos a sus alumnos a manera de crear una base amplia de datos.

Con base en lo anterior, se puede establecer que en la actualidad se buscan modelos educativos que contemplen el razonamiento clínico analítico y el reconocimiento de patrones.

2.2 La reanimación cardiopulmonar

La reanimación cardiopulmonar (RCP)[2] es una técnica de primeros auxilios que debe aplicarse a un individuo que presenta pérdida de conciencia, paro respiratorio y paro cardíaco por asfixia (a causa de estrangulación, obstrucción de la vía aérea, ahogamiento en agua, inhalación de humo, etcétera), o electrocución, inhalación de tóxicos o infarto al miocardio.

Actualmente para la reanimación cardiopulmonar se sigue el protocolo de la *American Heart Association*, *Guidelines for Cardiopulmonary Resuscitation*(AHA)[3] y la *Emergency Cardiovascular Care and the European Resuscitation Council Guidelines for Resuscitation*(ERC)[4] Estas organizaciones investigan las formas correctas de dar cuidado cardíaco en busca de reducción de decesos causados por asfixia y enfermedades cardíacas y propone un modelo para brindar asistencia médica, básica y avanzada, en cuanto a las técnicas de reanimación cardiopulmonar. Estas técnicas son el resultado de tres años y dos meses de investigación de 281 expertos sobre la RCP.

2.2.1 Reanimación cardiopulmonar básica

La RCP es una atención de primeros auxilios que se le da a personas que se encuentran en riesgo de vida, por tanto es una parte crucial del Soporte Básico de Vida (*BLS*)[5], generalmente se aplica en tanto se le puede brindar atención médica completa o intrahospitalaria al paciente, puede ser realizada por técnicos de emergencias médicas o personas adiestradas en las técnicas de *BLS*.

El *BLS* consiste en técnicas enfocadas en el ABC (por sus siglas en inglés),:

A (*airway*) Vía aéreo, mantener la vía respiratoria libre en busca de un flujo libre de gases, principalmente dióxido de carbono y oxígeno, entre los pulmones y el exterior del cuerpo.

B (*breathing*) Respiración, inflación y deflación de los pulmones.

C (*circulation*) Circulación, buscando proveer un adecuado flujo sanguíneo, especialmente a órganos críticos, así como proveer de respiración a través de la perfusión de la sangre en todo el cuerpo.

La reanimación cardiopulmonar es un procedimiento de emergencia aplicado a pacientes en paro cardiorespiratorio. Involucra intervenciones físicas buscando la circulación sanguínea por compresiones rítmicas al pecho de la víctima, para bombear manualmente la sangre a través del corazón; acto seguido hay que propiciar la ventilación, donde el rescatista exhala aire en el paciente para hacerlo llegar a los pulmones, así se oxigena la sangre. Al RCP se le considera como un procedimiento básico previo a la hospitalización y sin administración de fármacos.

[2] RCP o resucitación cardiopulmonar
[3] AHA por sus siglas en inglés
[4] ERC por sus siglas en inglés.
[5] *BLS, Basic Life Support* , por sus siglas en inglés. o soporte básico de vida

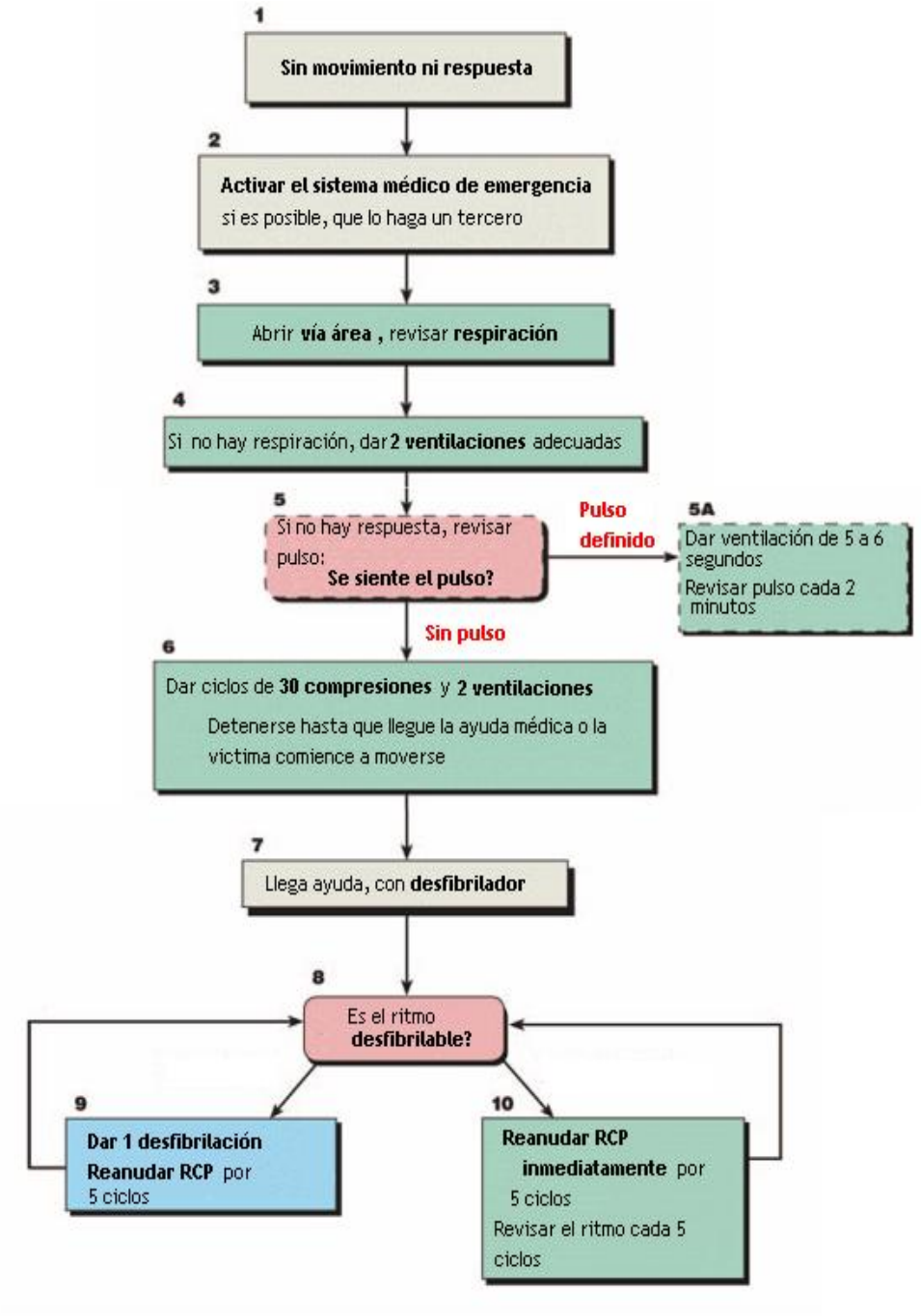


Figura 2.2. Cuadro a seguir de primeros auxilios, Basic Life Support.

El procedimiento a seguir para la reanimación cardiopulmonar y obtener un resultado óptimo, es el:

- Evaluar un área segura para la maniobra.
- Activar el sistema médico de emergencia.
- Asegurar condición de manipulación de vías aéreas, señalando la importancia de identificar probable lesión cervical. Búsqueda de cuerpos extraños en vía aérea.

- Evaluación de respiración (VES por 10 segundos)[6].
- Al no haber respiración se proporcionan dos ventilaciones de rescate. Si hay paso de aire, pero el paciente no responde, revisar pulso carotídeo y otros signos de circulación.
- En caso de que el paciente no presente circulación sanguínea se inician compresiones torácicas localizando el sitio anatómico correspondiente en relación 30 compresiones y dos ventilaciones.

Figura 2.3. Compresiones Torácicas

- Después de cinco ciclos reevaluar los primeros tres puntos (2 y 3 a la vez).
- En caso de no haber respuesta se reinicia la reanimación 30x2x5[7] ciclos.
- En ocasiones se necesita continuar con ciclos de RCP hasta la llegada del desfibrilador automático externo.

2.2.2 Reanimación cardiopulmonar avanzada

El proceso de reanimación cardiopulmonar avanzada comprende las medidas a aplicar cuando se tienen medios técnicos adecuados y personal preparado para la maniobra. Para este proceso, a diferencia de la RCP básica, se requiere total dominio técnico y práctico en lo que respecta al manejo de la vía respiratoria del paciente, la lectura e interpretación de electrocardiogramas, así como la farmacología necesaria de emergencia; por lo que este procedimiento suele efectuarse cuando el paciente ya está hospitalizado y es realizado por personal médico capacitado.

[6] VES es un acrónimo de Ver, Escuchar y Sentir, significa que para detectar la respiración uno debe ver el levantamiento del pecho, escuchar y sentir el flujo de aire.
[7] 30x2x5 significa que deben hacerse 30 compresiones y 2 ventilaciones cinco veces.

Los objetivos de la reanimación cardiopulmonar avanzada son establecer la ventilación adecuada, restablecer la actividad cardíaca, normalizar el ritmo cardíaco y estabilizar el flujo sanguíneo. Se efectuarán los siguientes pasos:

- Mantenimiento de la permeabilidad y aislamiento definitivo de la vía aérea.
- Ventilación y oxigenación.
- Masaje cardíaco.
- Empleo de fármacos y vías de administración.
- Monitorización electrocardiográfica.
- Diagnóstico y tratamiento específico de arritmias.

Los cuidados posteriores a la reanimación deben optimizar las funciones de los diversos sistemas orgánicos que pueden estar comprometidos primariamente o secundariamente a la hipoxia[8], especialmente la encefalopatía post-anóxica[9]. Por ello se debe dar atención a:

- La recuperación del paciente y las causas del paro cardiorespiratorio.
- La valoración neurológica y tratamiento específico de la encefalopatía post-anóxica.
- Control y tratamiento de los diversos órganos y sistemas.

2.3 Sistemas embebidos

«Un sistema embebido es un sistema computacional basado en una unidad de procesamiento, este puede ser un microprocesador, microcontrolador, DSP, etcétera; está diseñado para realizar una o varias funciones específicas en tiempo real»[10].

La unidad de procesamiento adjunta las aportaciones del sistema de cómputo, así se puede incluir memoria interna o externa.

El buen flujo de información adquiere gran importancia en los sistemas embebidos. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar, de cable o inalámbricas, con los componentes periféricos que pueden ser estos:

- La interfaz gráfica, suele ser una pantalla gráfica, táctil, LCD, alfanumérico u otras.
- El actuador (siendo el elemento electrónico que el sistema se encarga de controlar), puede ser un motor eléctrico, un conmutador tipo relé, o algún otro.
- El módulo de E/S, analógico y digital, suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos LED, reconocer el estado abierto cerrado de un conmutador o pulsador, y demás.
- El módulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. El tipo de oscilador es importante por varios aspectos: por la frecuencia necesaria, por la estabilidad solicitada y por el consumo de corriente requerido.

[8] La hipoxia es un trastorno en el que el cuerpo se ve privado del suministro adecuado de oxígeno.
[9] La encefalopatia post-anóxica son las afecciones del cerebro causadas por la insuficiencia de oxigenación.
[10] Embedded systems design. Heath, Steve, Newnes. 2003 p. 2

- El módulo de energía se encarga de generar los diferentes voltajes y corrientes necesarios para alimentar los diferentes circuitos del sistema embebido. Usualmente se trabaja con un rango de posibles tensiones de entrada que mediante convertidores ac/dc o dc/dc obtienen los diferentes voltajes para alimentar los diversos componentes activos del circuito. Éstos sistemas tienen comandos de ejecución *(software)*, los que se encuentran en la memoria interna del dispositivo, mismos que normalmente no se modifican.

Un sistema embebido complejo puede utilizar un sistema operativo como apoyo para la operación de sus programas, sobre todo cuando se requiere la ejecución simultánea de éstos. En una aplicación de tiempo real compleja el uso de un sistema operativo de tiempo real multitarea puede simplificar el desarrollo del *software*.

2.3.1 Sistemas operativos

«Un sistema operativo (SO) es un *software* que actúa de interfaz entre los dispositivos de *hardware* y los programas usados por el usuario para manejar una computadora»[11]. El sistema operativo es responsable de gestionar, coordinar las actividades y llevar a cabo el intercambio de los recursos, actúa como estación para las aplicaciones que se ejecutan en la máquina.

Uno de los más prominentes ejemplos de sistema operativo, es el núcleo Linux.

Linux tiene todas las prestaciones que se pueden esperar de un Unix[12] moderno y completamente desarrollado: multitarea real, memoria virtual, bibliotecas compartidas, carga de sistemas, compartimiento, manejo debido de la memoria y soporte de redes TCP/IP.

Linux corre principalmente en computadoras personales basadas en procesadores X86, usando las facilidades de proceso de la familia de procesadores 386 (segmentación TSS, etcétera) para implementar las funciones ya nombradas.

La parte central de Linux (conocida como núcleo o *kernel*), se distribuye a través de la licencia pública general GNU, lo que significa que puede ser copiado libremente, cambiado y distribuido, pero no es posible imponer restricciones adicionales a los productos obtenidos se debe dejar el código fuente disponible, de la misma forma que está disponible el código de Linux.

Estas características permiten que Linux sea un sistema operativo muy utilizado para sistemas embebidos.

[11] Estudio de un sistema operativo. Pérez, Juan Carlos; Sergio Sáez (2010). futura.disca.upv.es (ed

[12] Unix es un sistema operativo que fue el primero en ser portable, multitarea y multiusuario. Inició su desarrollo en 1969.

2.3.2 Protocolos de comunicación

Un protocolo de comunicación es un conjunto de reglas normalizadas para la representación, señalamiento, autenticidad y detección de errores, es necesario para enviar información a través de un canal de comunicación.

«El término protocolo de comunicación en informática lo utilizó por primera vez Roger Scantlebury y Keith Bartlett en *The National Physical Laboratory*, (NPL) en Inglaterra en abril de 1967, con el documento *A protocol for use in the NPL data communications network*» [13].

Es posible comparar los protocolos de comunicación con el lenguaje natural del hombre ya que contienen características similares:

- Se define un formato preciso para la validez de los mensajes, sintaxis.
- La información se fundamenta en reglas de orden, gramática.
- Consta con vocabulario para mensajes válidos intercambiables, semántica.

Existe gran número de protocolos de comunicación, algunos de los más utilizados para la interconexión entre dispositivos son el I^2C y el USB.

2.3.1.1 El bus de comunicación en serie I^2C

2.3.1.1.1 Introducción

El *bus* de comunicación en serie I^2C fue creado por Phillips para mantener una comunicación de baja velocidad entre un microcontrolador[14] (MCU) y sus periféricos[15].

El protocolo solo requiere de dos líneas, es decir, dos cables como medio físico para que todos los dispositivos conectados al *bus*[16] puedan intercambiar información.

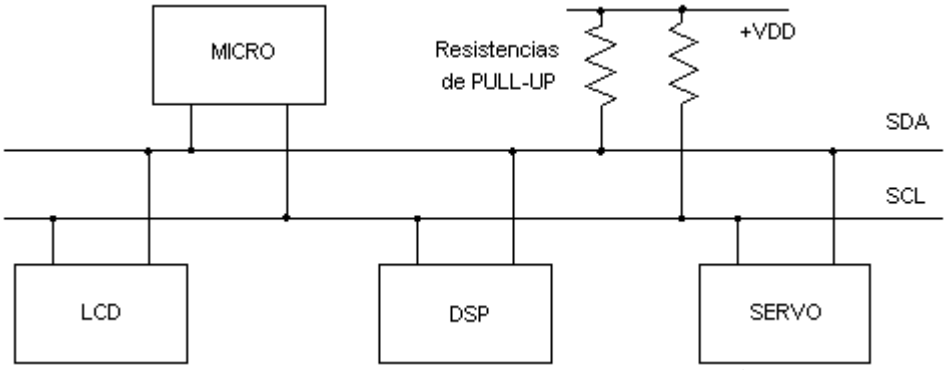


Figura 2.4. Ejemplo de una configuración I^2C
Es un bus de tipo serie bidireccional diseñado para simplificar la interconexión de dispositivos al microprocesador.

[13] Gerard H. Holzmann "*Design and Validation of Computer Protocols"* Prentice Hall 1990 pag 12

[14] Un microcontrolador (MCU por sus siglas en inglés) es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora; unidad central de procesamiento, memoria y unidades de entrada y salida.

[15] Los Periféricos son los dispositivos auxiliares e independientes conectados a la unidad central de procesamiento de una computadora.

[16] El bus es un sistema digital que transfiere los datos entre los componentes de una computadora.

Cada dispositivo cuenta con una dirección única que debe ser transmitida al inicio de cada transferencia para mantener de esta manera una relación simple de maestro[17]-esclavo[18] (*Master-Slave)* todo el tiempo.

Contiene detección de colisión y arbitraje para prevenir la perdida de información, si dos maestros tratan de iniciar una transferencia al mismo tiempo.

Existen distintas velocidades de transmisión; 100kbps para el modo estándar (*Standard*); 400kbps para el modo rápido (*fast*); 1Mbps para el modo rápido plus (*fast plus*); y 3.4Mbps para el modo alta velocidad (*high speed*). (*The I^2C bus specification,version 2.1, 2000.*)

2.3.1.2 Universal serial bus

2.3.1.2.1 Introducción

Las principales motivaciones para la creación del bus serial universal (USB) son estas:

- Facilidad de uso. La falta de flexibilidad al reconfigurar los periféricos de una computadora había sido un verdadero problema, la introducción de nuevas interfaces más flexibles y amigables habrían facilitado esta tarea, pero era necesario que los dispositivos tuvieran la capacidad de conexión y uso (*Plug & Play*[19]).
- Capacidad de expansión. Era necesario un bus que permitiera agregar más dispositivos sin desconectar otros y sin entorpecer la comunicación de los mismos.

El bus USB 2.0 tiene tres tasas de transferencia; el *Low Speed* (1.5Mbps), el *Full Speed* (de 12Mbps) y el *High Speed* (480Mbps). Además de contar con dispositivos autoidentificables, distintos modos de transferencia para los distintos tipos de dispositivos a conectar, soporta hasta 127 dispositivos físicos conectados.

Por estas características ha sido adoptado con éxito por la industria tecnológica desplazando de manera contundente al puerto serial.

[17] El Maestro es el dispositivo que inicia la transferencia de información y genera una señal de reloj que permite dicha transferencia.

[18] El Esclavo es el dispositivo que debe responder a las solicitudes del maestro y sincronizarse con el reloj del mismo.

[19] Los dispositivos Plug & Play son los que son diseñados para conectarse y usarse sin ninguna configuración previa aparente para el usuario.

3 Marco referencial de la enseñanza de la reanimación cardiopulmonar

3.1 Cómo se enseña la reanimación cardiopulmonar

«La enseñanza en RCP ha generado en las últimas décadas infinidad de documentación, desde su justificación hasta, en épocas más recientes, su costo. Enseñar, aprender, formar, son aspectos muy enraizados en el colectivo sanitario, van ligados a la profesión y tienen su base en el propio origen de las actividades relacionadas con la salud y el curar»[1].

Actualmente existen cursos de capacitación en la RCP básica, instituciones como la *American Heart Association* ofrecen cursos en línea y cursos presenciales en sus centros de entrenamiento. Los alumnos presenciales reciben preparación práctica donde se les enseñan las maniobras con la ayuda de un maniquí, esto permite reforzar el conocimiento adquirido.

«No todas las metodologías docentes obtienen los mismos resultados y observamos métodos alternativos de formación con diferente suerte, desde la falta de retención de conocimientos utilizando programas de formación breves (1 hora) en enfermeras hasta buenos resultados obtenidos entre los alumnos de enseñanza secundaria»[2]

La *European Resucitation Council* señala que «independientemente del tipo de entrenamiento, es importante la preparación del alumno previa a la realización de la acción formativa»[3], es decir, el marco teórico es fundamental para la efectividad del curso.

3.1.1 Equipo utilizado en la enseñanza de la RCP

Para la enseñanza de la RCP básica se utilizan desde folletos, libros, presentaciones electrónicas, videos; hasta equipo de entrenamiento que permite una interacción más cercana a una situación real que ha de enfrentar un rescatista, como son los simuladores computacionales y simuladores físicos. En la tabla 3.1 se muestra algunos de estos equipos.

[1]Xavier Jiménez Fàbrega y Xavier Escalada Roig "Mejorar la enseñanza en la reanimación cardiopulmonar… ¿No tiene precio?", *Emergencias*, 19/6 diciembre, Madrid, 2007, p. 298

[2]*ibidem*

[3]La *European Resucitation Council* es un organismo europeo encargado de regular las guías para la reanimación cardiopulmonar

Equipo	Presentaciones	Ejemplo	Procedencia
Simulador web	Página web	Salva una vida	Madrid, España, Salud
Instructivo	Documento impreso, disco compacto, formato electrónico.	Manual de procedimientos del sistema de atención de urgencias prehospitalarias	Cruz Roja Mexicana, México.
Simulador RCP básico, adulto	Solo torso o cuerpo completo (mecánico o mecatrónico)	Maniquí Fred, adulto obeso	Quirumed, España
Simulador RCP básico, niño	Solo torso o cuerpo completo (mecánico o mecatrónico)	S151 *Five Year* CPR *and Trauma Care Simulator*	Gaumard, E.U.A.
Simulador RCP básico, neonato	Cuerpo completo (mecánico o mecatrónico)	S103 *The Susie*® y *Simon*® *Newborn* CPR *and Trauma Care Simulator*	Gaumard, E.U.A.

Tabla 3.1 Material empleado en la enseñanza de la RCP

Los fabricantes de simuladores de RCP básica ofrecen, en general, simuladores con apoyos electrónicos, los mecatrónicos, y simuladores sin estos complementos, los mecánicos que tienen como características principales:

- Simular de manera realista la elevación del pecho cuando una insuflación es provista por el usuario.
- Emular la rigidez del pecho ante una compresión del mismo.

Mientras que los simuladores mecatrónicos incorporan un sistema de retroalimentación valiéndose de sensores e indicadores (equipo electrónico auxiliar). En la figura 3.1 se muestran algunos simuladores de RCP básica.

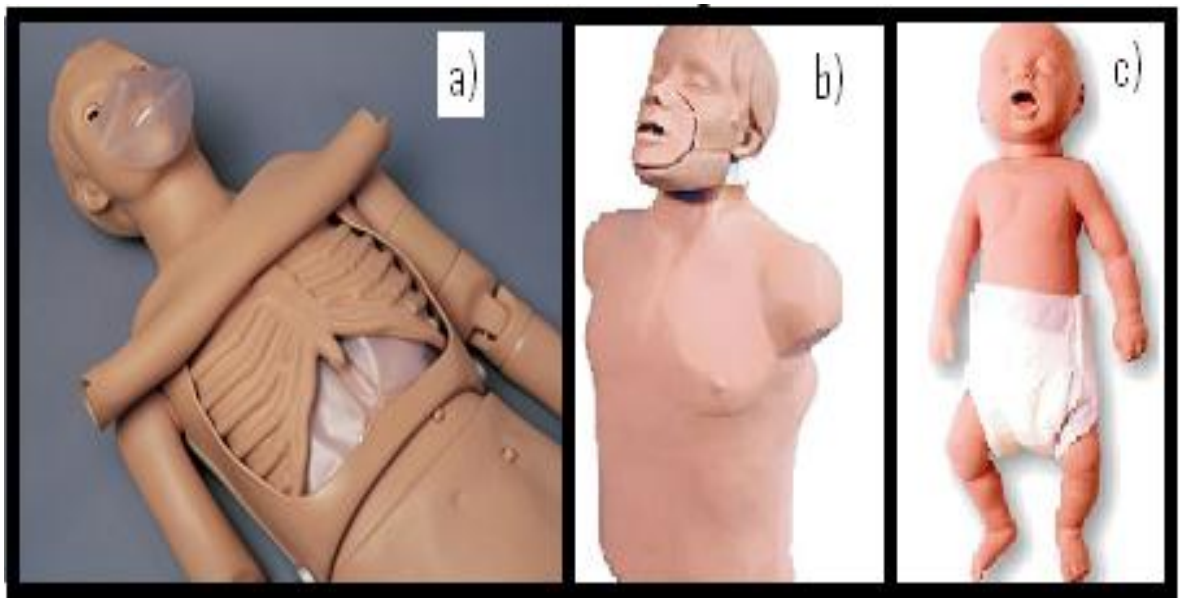


Figura 3.1 Vistas reales de simuladores de RCP básica
a) S310 CPR Simon® BLS - Full body modelo mecatrónico de cuerpo completo, b) TEC 53JT modelo mecánico de torso, c) BPP- 09 simulador mecatrónico de cuerpo completo.

3.1.1.1 Equipo electrónico auxiliar

Implantar sistemas que arrojan y despliegan información también permite medir las acciones realizadas sobre el simulador, o en su caso, indican la efectividad de una acción, de forma que la evaluación del practicante es objetiva, además permiten que haya retroalimentación con el usuario del simulador.

Actualmente el sistema de retroalimentación implementado por la mayoría de los fabricantes (en los equipos más completos o de mayor costo) indica:

- Si la posición de las manos es correcta o no (indicador luminoso)
- El ritmo adecuado de las compresiones o ventilaciones (indicadores sonoros)
- Si la insuflación es apropiada o no (indicador luminoso)
- Si la intensidad de la fuerza en las compresiones es correcta o no (indicador luminoso)
- La gráfica (impresa o formato electrónico) de las insuflaciones y compresiones aplicadas al simulador

El uso de modelos fisiológicos en simuladores de RCP básica es escasa, sin embargo, existen modelos como el *HAL S3009*, este cuenta con un programa de computadora capaz de seleccionar variables fisiológicas como saturación de oxígeno, ritmo cardíaco, presión sanguínea y ritmo respiratorio; estas variables se pueden observar en un monitor de forma virtual, pues no pueden ser observadas físicamente en el simulador.

Figura 3.2 Vistas reales de sistemas de monitoreo de los simuladores de RCP básico

a) Modelo MM1-50, b) Modelo CODE BLUE.

3.2 Dónde se realiza la enseñanza de la reanimación cardiopulmonar

El entrenamiento en RCP se lleva a cabo en escuelas, hospitales, universidades de medicina o enfermería, institutos especializados en RCP, centros de trabajo, entre otros.

Algunas instituciones llevan a cabo cursos de entrenamiento en RCP básico son:

- Fundación Cardiológica Argentina
- *American Heart Association*
- *European Resuscitation Council*
- *Conseil Français de Réanimation Cardio-pulmonaire*

En México:

- UNAM, Facultad de Medicina
- Hospital Psiquiátrico "Fray Bernardino Álvarez"
- Secretaria de Salud Pública
- Cruz Roja Mexicana

3.3 A quién se le enseña la reanimación cardiopulmonar

Las técnicas de RCP son normalmente instruidas a personal médico y paramédico, pero la mayoría de las veces, los primeros en atender emergencias son personas que se encontraban cerca, sean familiares, amigos, vecinos u otros. Por lo regular estos individuos no cuentan con instrucción, por eso cuando aparece el personal médico ya es demasiado tarde. «Los ciudadanos que han sido entrenados mejoran su capacidad para actuar de manera correcta en situaciones de emergencia y, al iniciar las maniobras adecuadas de RCP, se convierten en el primer eslabón de la cadena de supervivencia, hasta que acuden los servicios de urgencia especializados»[4].

[4] Sastre Carrera María José, *et all*, "Enseñanza de la Reanimación Cardiopulmonar básica en población general", Atención Primaria, 34-8, 2004, España, p., 409

A pesar de los adelantos científicos y técnicos de la medicina no ha mejorado el pronóstico de supervivencia durante un evento de parada cardiorespiratoria debido a que el paciente permanece mucho tiempo en este estado. La supervivencia del paciente depende de la rapidez y eficacia de las maniobras de RCP.

«La eficacia de la reanimación es directamente proporcional al entrenamiento recibido por la persona que la realiza e inversamente proporcional al tiempo transcurrido entre el momento en que se produjo la parada cardiorespiratoria y el inicio de una reanimación»[5], por lo tanto, es necesario que toda persona se encuentre familiarizada con las técnicas de RCP.

En México, la NOM-237-SSA1-2004 señala que «el manejo de la atención prehospitalaria deberá realizarse de acuerdo a los protocolos escritos, que para la naturaleza del evento tenga definidos la institución responsable de brindar la atención prehospitalaria. Los contenidos podrán diferir por cada institución, de acuerdo a la *lex artis* médica»[6], lo que indica que no existe una regulación clara acerca del protocolo que se ha de seguir en caso de que un paciente necesite de RCP.

[5] *ibidem*
[6] Diario Oficial de la Federación, jueves 15 de junio de 2006, primera sección, p. 54

4 Diagnostico de la situación o problema

«En México mueren entre 33 mil y 53 personas al año por arresto cardiaco»[1], el tratamiento inmediato elevaría la tasa de supervivencia de pacientes con paro cardiorespiratorio.

Una causa de la falta de atención inmediata a pacientes que presentan paro cardiorespiratorio es por el desconocimiento de las técnicas de reanimación cardiopulmonar básica, la enseñanza de las mismas con pacientes reales no es factible, aun estando dentro de un hospital, pues en situación de urgencia es imposible que el docente haga una exposición uniforme del problema o explique de manera detallada el procedimiento a los aprendices. Es necesario encontrar una manera de proporcionar escenarios reales para la enseñanza médica sin poner en riesgo la vida de pacientes.

En busca de una solución, se ha optado por utilizar simuladores, pero, en algunos casos, la falta de realismo desilusiona al estudiante y no logra la enseñanza adecuada.

Los simuladores necesitan ser más realistas, para ello se les debe otorgar la capacidad de emular condiciones fisiológicas similares a las que presentaría un paciente en una emergencia real.

La complejidad de las respuestas fisiológicas humanas, que además varían de individuo a individuo, han dificultado la creación de un simulador de "propósito general" para la enseñanza de la medicina, esto ha provocado la creación de simuladores para escenarios específicos, como lo es un paro cardiorespiratorio.

Los simuladores para la enseñanza de la RCP básica han sido muy bien adoptados, pero la mayoría de éstos no recrea una respuesta fisiológica, los simuladores están muy lejos de un escenario realista, de igual manera carece de retroalimentación objetiva sobre el procedimiento efectuado por el alumno o los posibles resultados en el paciente.

La retroalimentación proporcionaría al alumno una manera alternativa de aprender de sus errores, es decir, sin necesidad de exponer al paciente. Se ha observado que la retroalimentación puede mostrar mejorías en el aprendizaje hasta en un 74%[2]. Esto nos lleva a plantear los siguientes cuestionamientos:

¿Cómo otorgar mayor realismo al simulador de reanimación cardiopulmonar?

¿De qué manera se puede proporcionar una retroalimentación objetiva a los estudiantes?

[1] Enrique Asencio, «Conceptos actuales sobre la muerte súbita», Gaceta Médica de México, 141/2, mar-abr, México, 2005p. 90.

[2] Veloski J., *et all,* «Systematic review of the literature on assessment, feedback and physicians clinical performance», BEME Guide No. 7, p. 118

5 Objetivo

Objetivo General

Diseñar un simulador para la enseñanza de la reanimación cardiopulmonar básica que sea capaz de reproducir respuestas fisiológicas reales y a la vez otorgue una retroalimentación objetiva.

Objetivos Particulares

- Diseñar un simulador de pulso.
- Crear un emulador de ventilación.
- Implantar sensores de permeabilidad de la vía aérea.
- Formar una interfaz para controlar el simulador.
- Esbozar el modelado fisiológico del sistema cardiorespiratorio.
- Diseñar una unidad de procesamiento del simulador.
- Proponer el diseño de una interfaz WEB para revelar los resultados.

6 Método

Para realizar el simulador es necesario complementar un maniquí con sensores y actuadores electrónicos, estos proporcionarán la interacción con el usuario.

La interacción se caracterizará por la emulación de funciones fisiológicas, en este caso la respiración y el pulso, también se puede aparentar la permeabilidad de las vías aéreas, esto depende de las maniobras efectuadas por el usuario.

Con el simulador propuesto se podrá observar el registro de las compresiones y ventilaciones administradas por el usuario, posteriormente éstas se desplegarán en una página de internet, también contará con ayuda auditiva, para la administración de las compresiones, e incluso con una interfaz que permite controlar al simulador durante su uso.

Para dotar al simulador con las características necesarias se evaluaron distintos métodos, los elegidos son los siguientes:

- Emulación de pulso. Se encontró que la mejor manera de aparentar el pulso es mediante un sistema neumático que de manera periódica genera una sensación similar a la de las arterias dentro del cuerpo humano, por lo tanto se diseñará un dispositivo que sea capaz de producir insuflaciones de manera periódica, al menos 80 veces por minuto, y que logre producir la sensación de pulso.
- Simulación de respiración. Se decidió que el mejor modo de simular la respiración es haciendo que la cavidad torácica se levantara impulsado por un motor eléctrico, este se conectará a un fuelle para conseguir un efecto similar al proceso de exhalación e inhalación de aire, el fuelle estará conectado por un tubo a la nariz.
- Permeabilidad de las vías aéreas. Se observó que la mejor manera de calificar las maniobras realizadas por el usuario era mediante la medición de las elongaciones del maniquí, lo cual permitiría saber si la maniobra estaba en el rango correcto. También era necesario la medición de la presencia de algún objeto extraño en la cavidad bucal, así que se eligió utilizar un imán y sensores magnéticos.
- Registro de compresiones. Se eligió un sensor de presión, en forma de corazón que va en la cavidad torácica, que medirá cómo se libra de opresión un cuerpo hueco.
- Registro de ventilación. Se optó por utilizar un sensor de flujo de orificio variable ya que funciona adecuadamente en el rango de operación y tiene un bajo consumo de energía.
- Ayuda auditiva. Para la ayuda auditiva se seleccionaron bocinas de tamaño moderado que pudieran ser instaladas en la cavidad abdominal del maniquí.
- Interfaz. Se utilizará un dispositivo que cuente con una pantalla de cristal líquido a color que desplegará la información que debe ser desplegada. El dispositivo contará con varios botones que permitirán controlar el simulador, por ejemplo, el estado de los emuladores de pulso y ventilación; también desplegará información sobre los sensores instalados, como son la intensidad y la frecuencia de las compresiones proporcionadas; incluirá la información proveniente del modelo fisiológico, por ejemplo, la tensión arterial; y mostrará un cronometro del tiempo transcurrido.

Las características del simulador serán monitoreadas por una unidad de procesamiento instalado dentro del maniquí, contará con baterías instaladas de la misma manera, esta unidad de procesamiento enviará los datos recopilados durante su uso a una base de datos de manera inalámbrica. Los datos recopilados se desplegarán mediante una interfaz gráfica en un sitio WEB donde podrán ser consultados.

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.

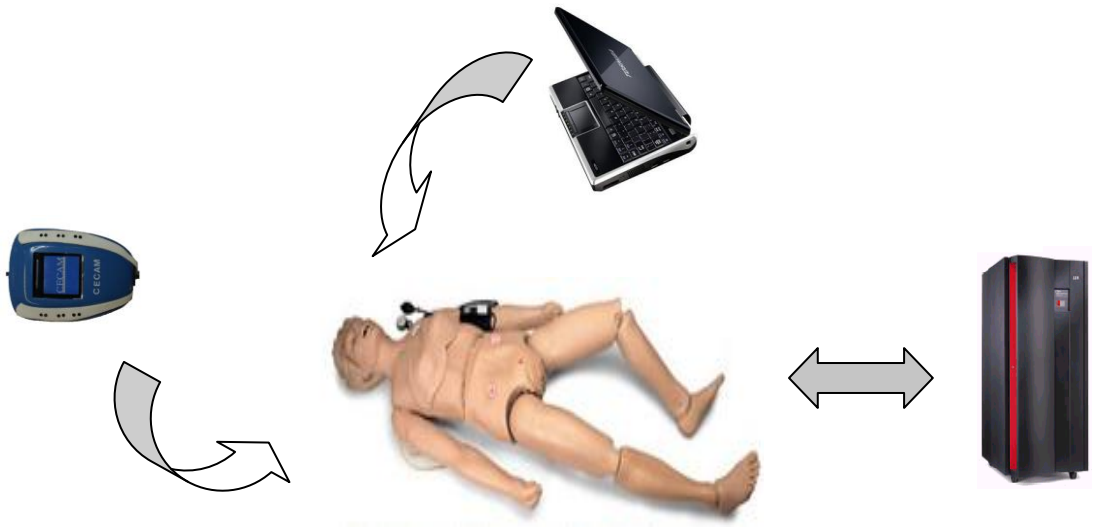


Figura 7.1 Conformación del sistema.

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

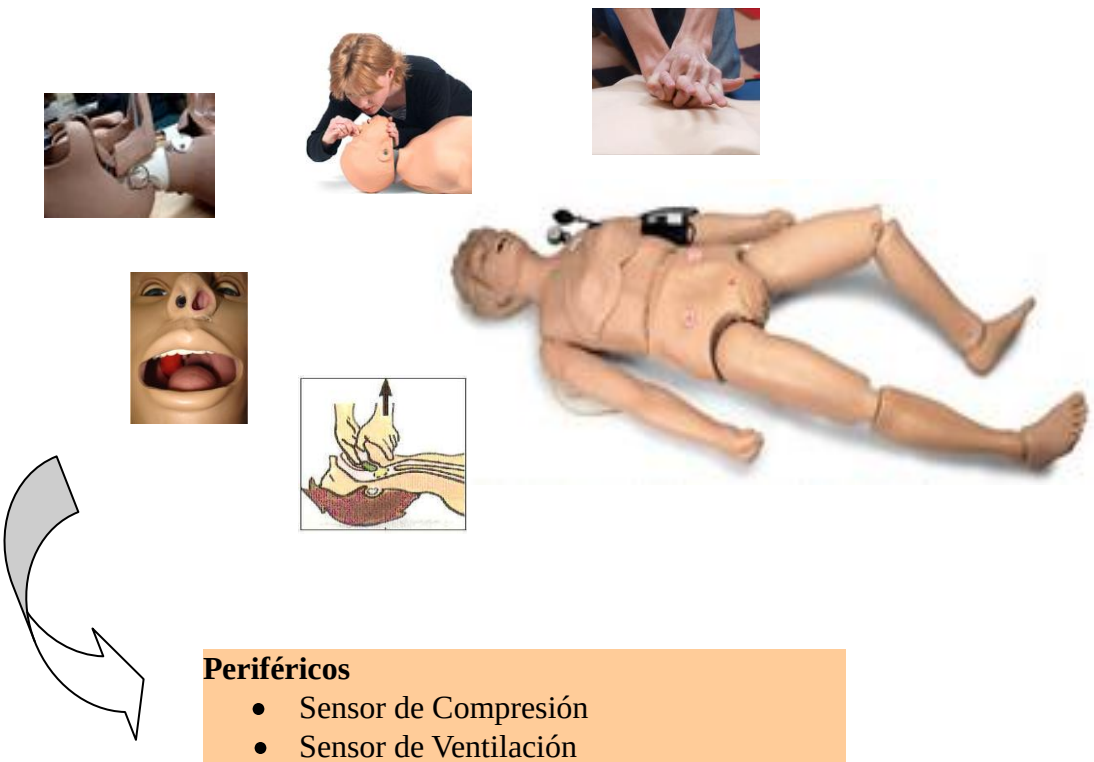
La unidad de control se compone por tres subsistemas:

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.

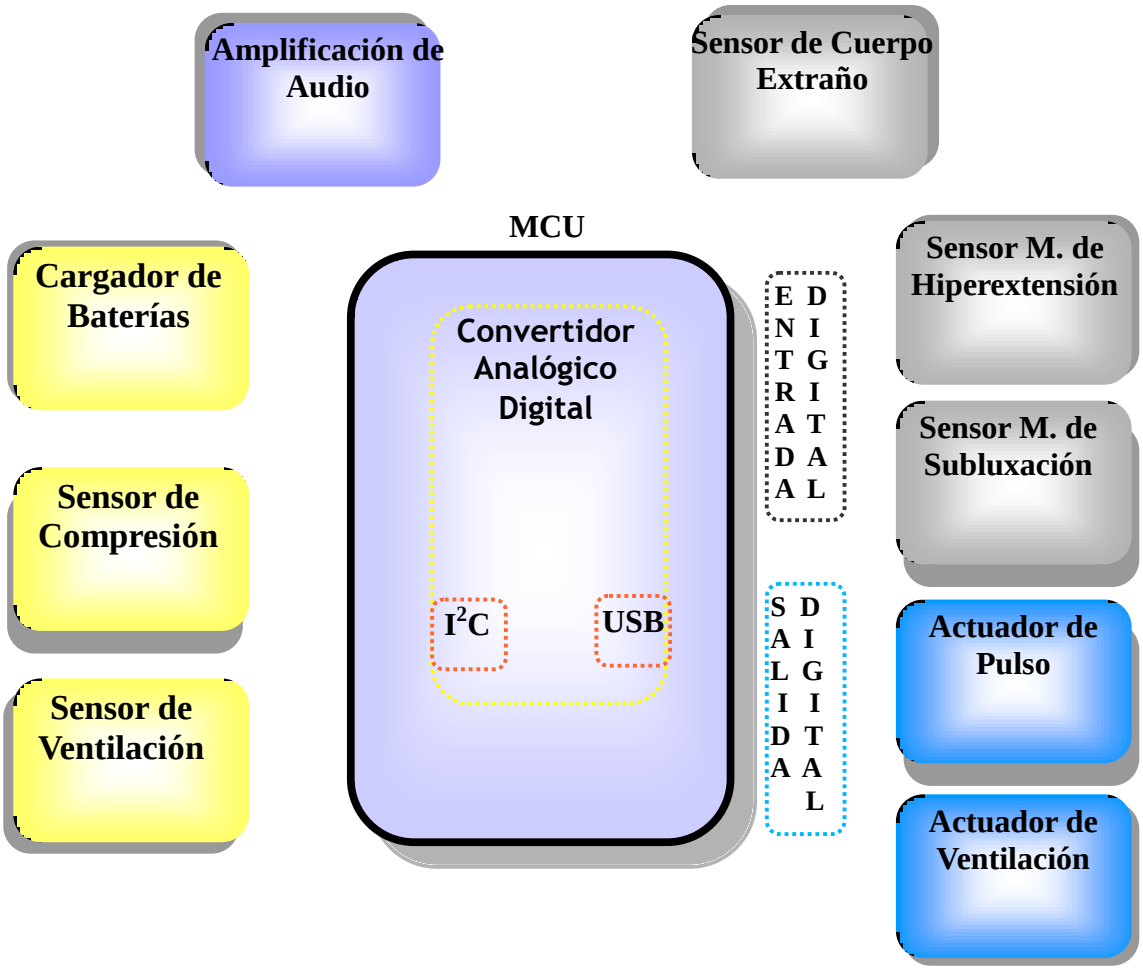


Figura 7.3 Diagrama de bloques de la tarjeta principal

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.

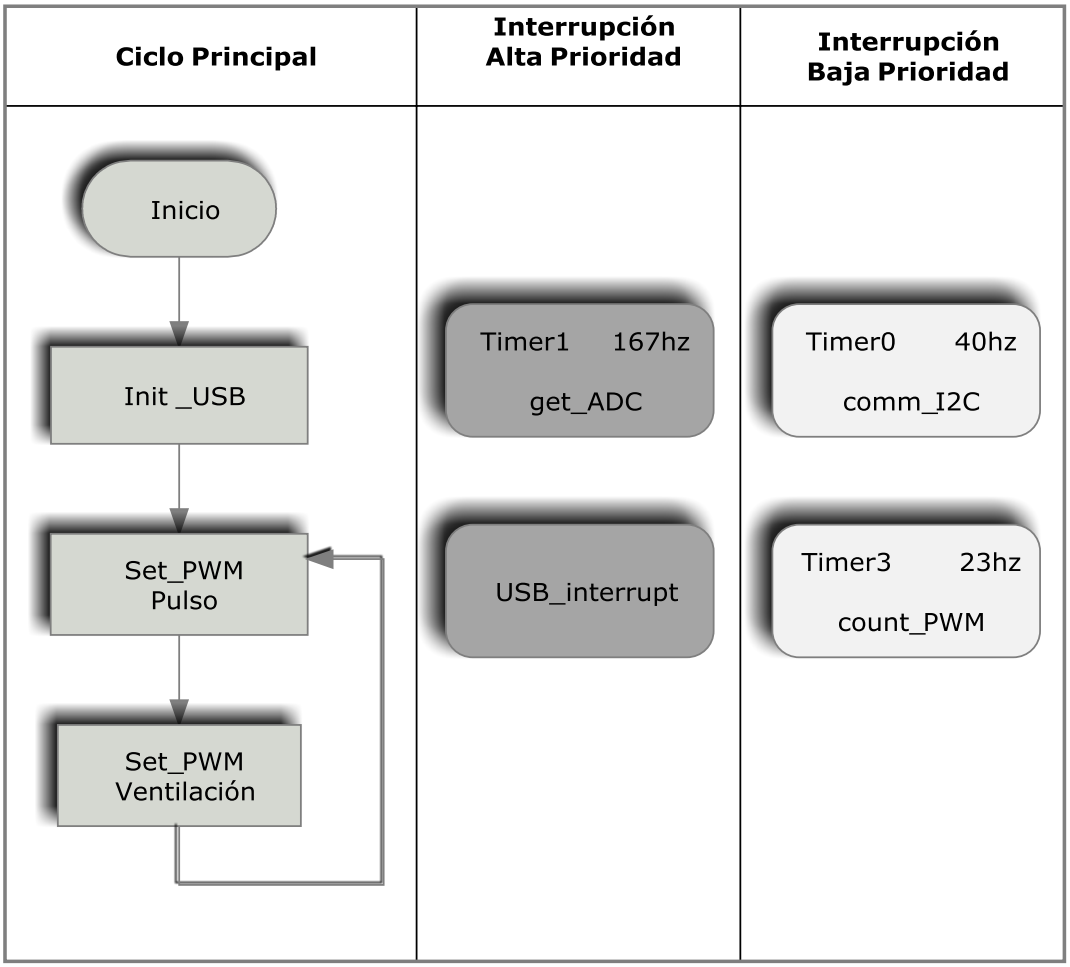


Figura 7.4 Diagrama de bloques MCU maestro

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:

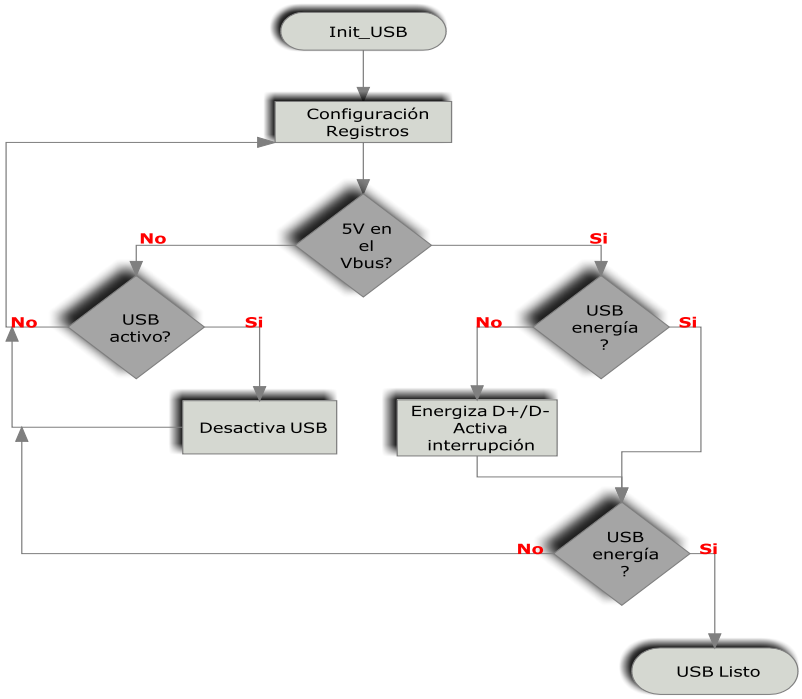


Figura 7.5 Diagrama de bloques de init-USB

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.

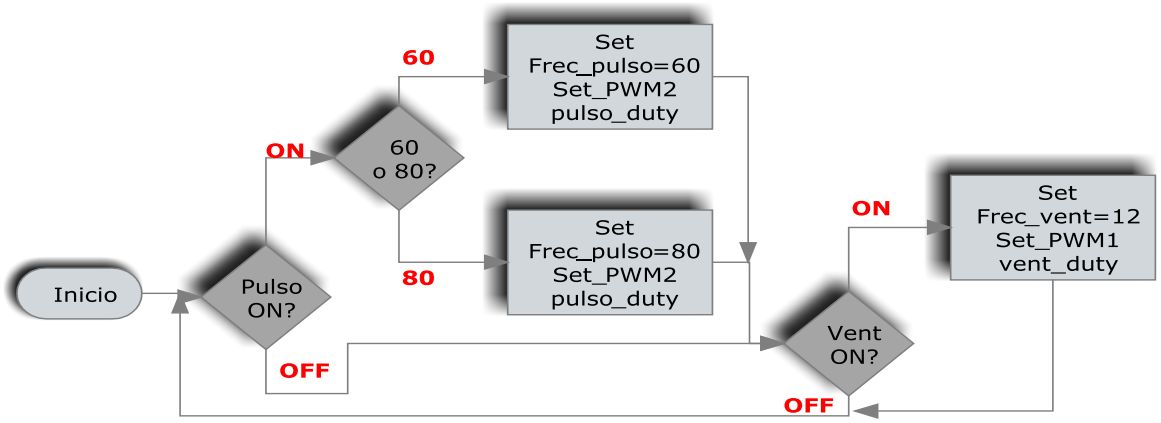


Figura 7.6 Diagrama de bloques del ciclo de los actuadores.

El pulso PWM presenta la siguiente señal de salida.

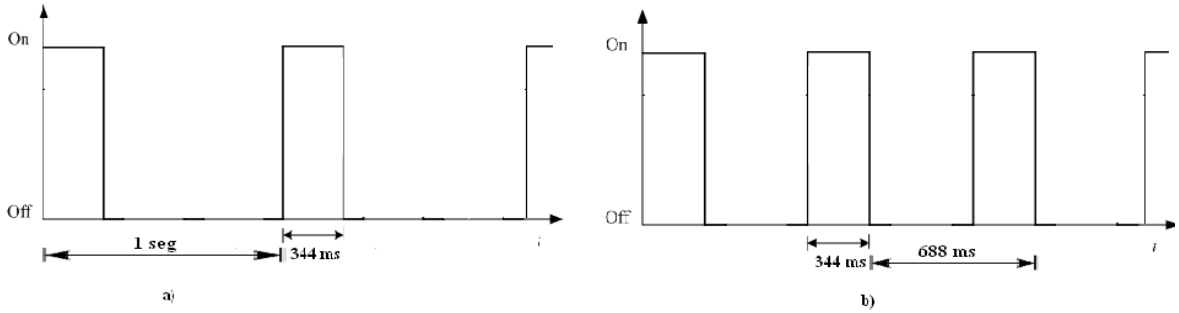


Figura 7.7.Forma de onda del PWM para actuador de pulso
a) 60 latidos por minuto; b) 80 latidos por minuto

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.

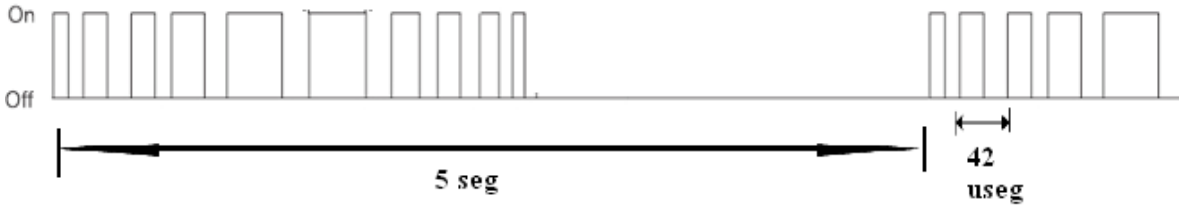


Figura 7.8. Forma de onda del PWM para actuador de ventilación.

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.

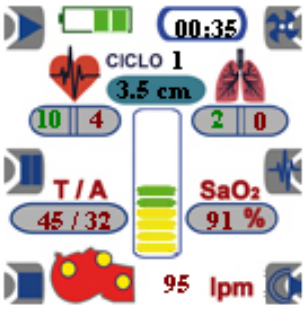


Figura 7.10. Pantalla principal.

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.

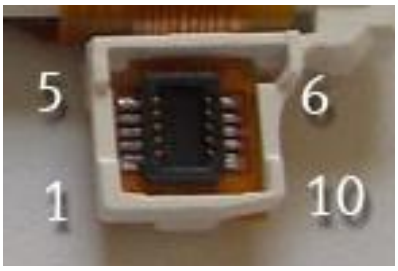


Figura 7.14. Fotografía de pines de pantalla LCD

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.

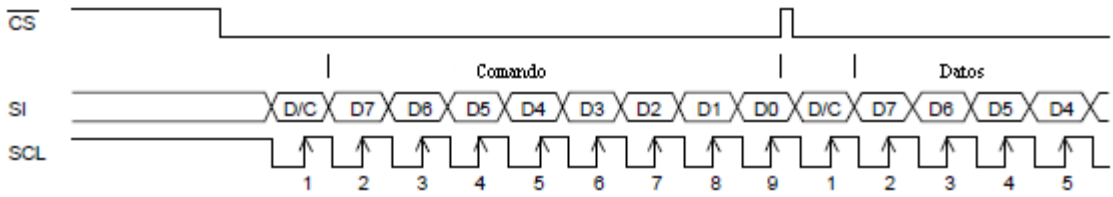


Figura 7.15. Forma de onda de comunicación con LCD

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.

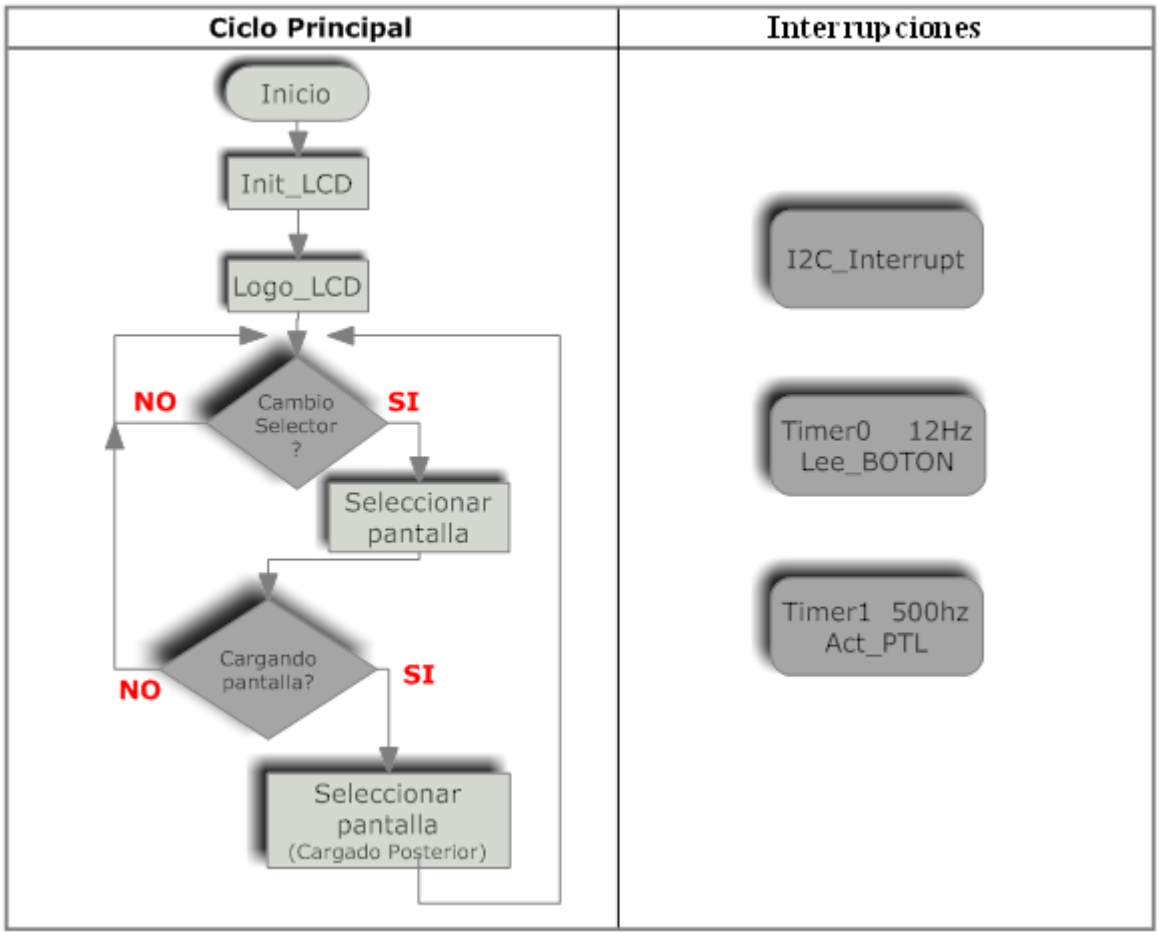


Figura 7.16. Diagrama de bloques MCU auxiliar

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.

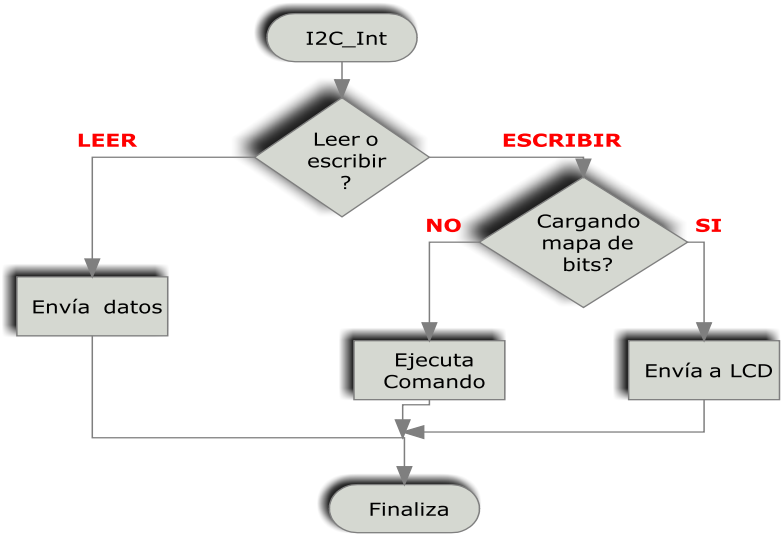


Figura 7.17. Diagrama de bloques interrupción I^2C.

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.

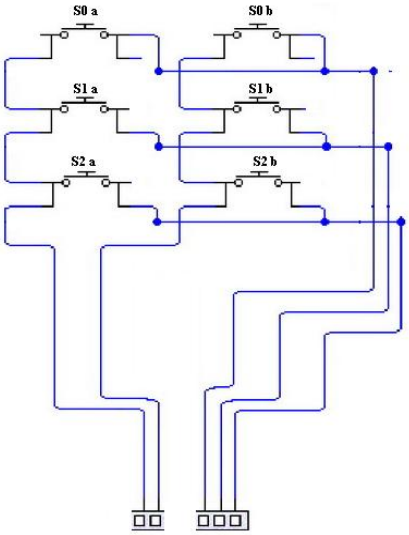


Figura 7.18. Diagrama de los botones del mando a distancia.

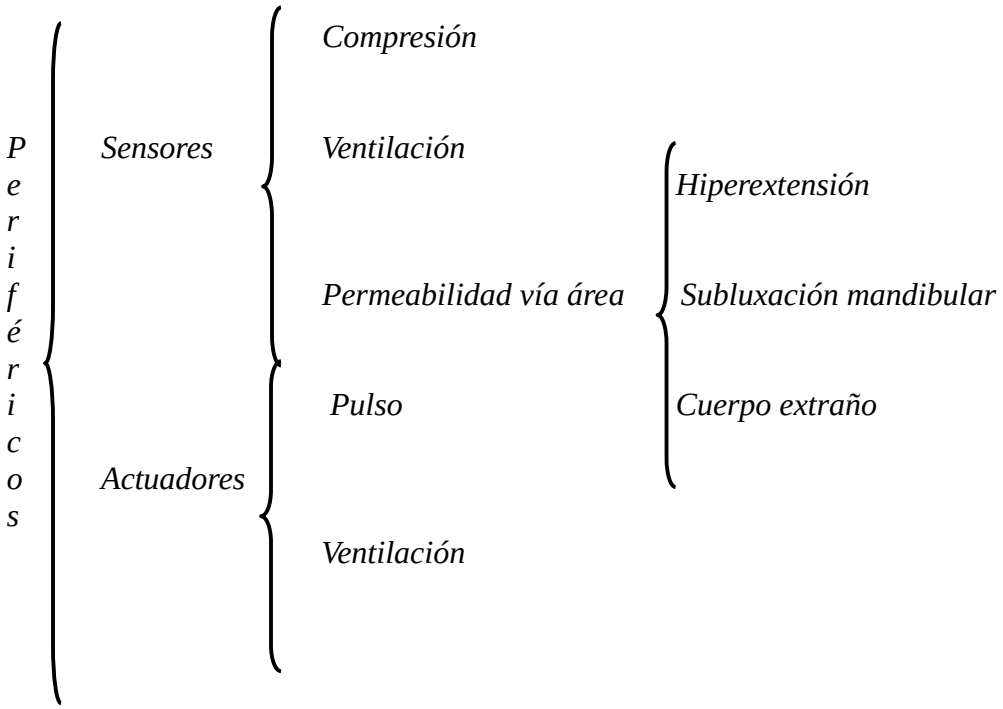
Interrupción ACT_PTL – Timer1

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.

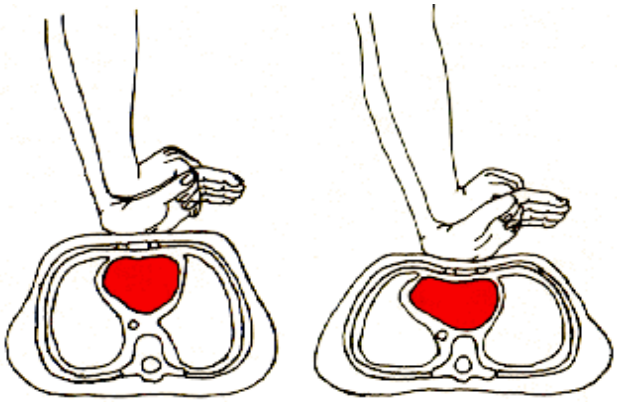


Figura 7.19. Dibujo del corte transversal del corazón durante una compresión.

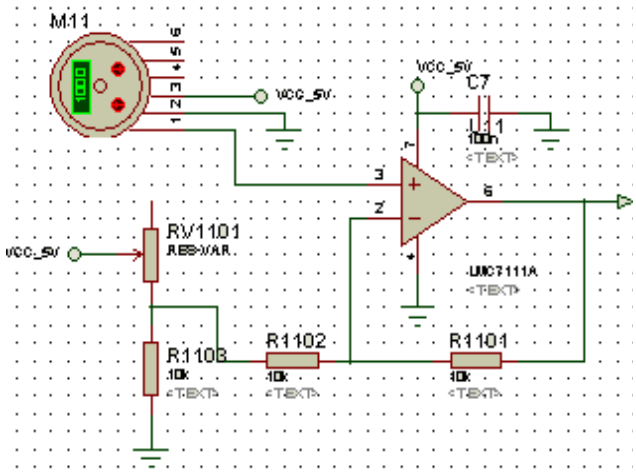
Los centímetros que se desplaza el corazón durante la compresión no tiene una relación lineal con la elevación de la presión ya que se desconoce cómo se modifica el volumen con cada centímetro de desplazamiento, por ello se utilizó el método de calibración descrito en la sección *7.2.4.2.3*.

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.

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

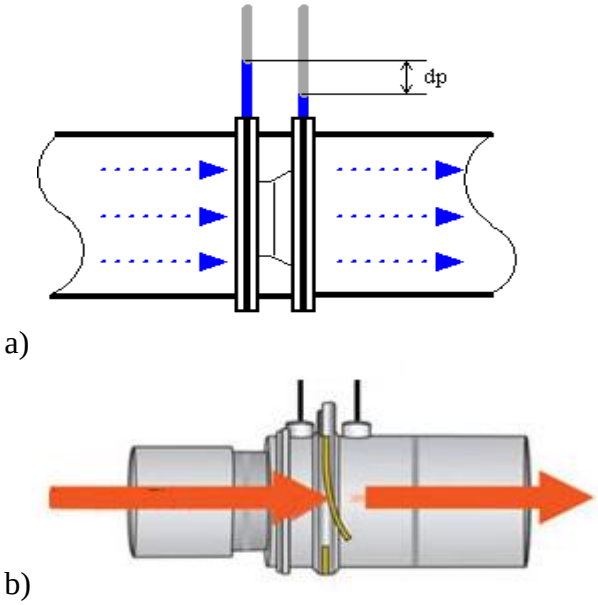


Figura 7.23. Esquema del sensor de orificio variable.
a) Esquema del cambio de presión por el flujo de aire; b) Esquema del orificio variable

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.

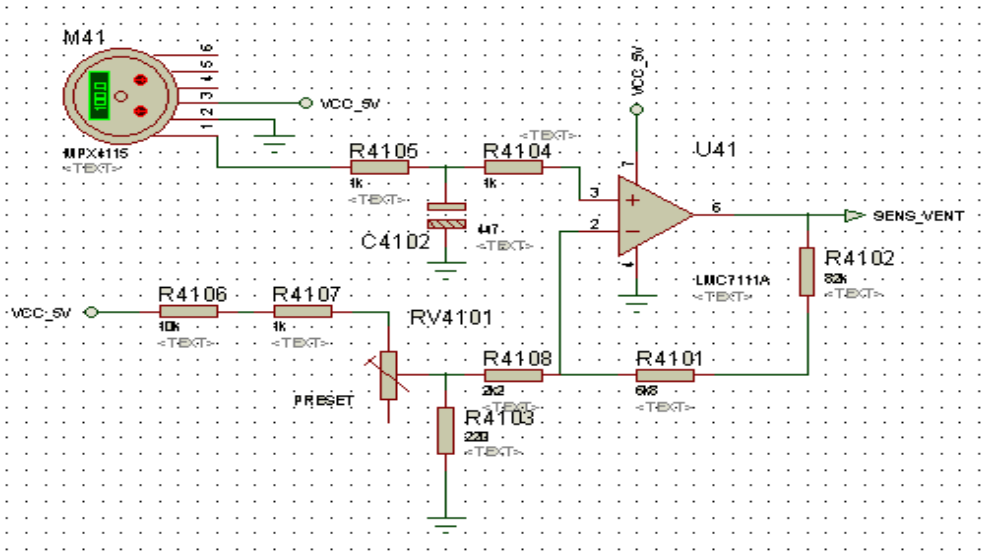


Figura 7.24. Esquema del sensor de ventilación.

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.

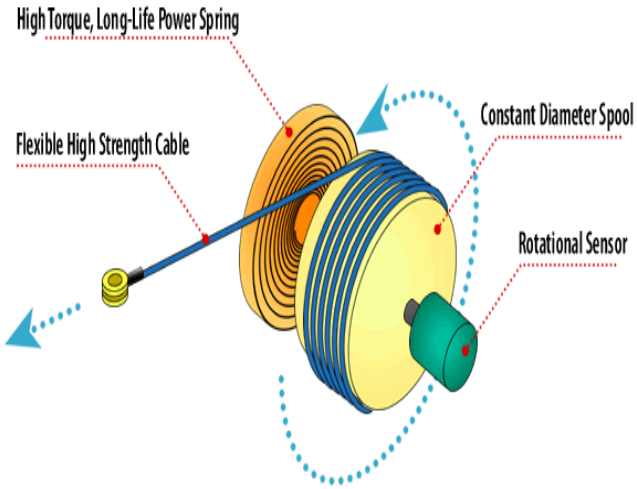


Figura 7.25. Partes de un StringPot

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.

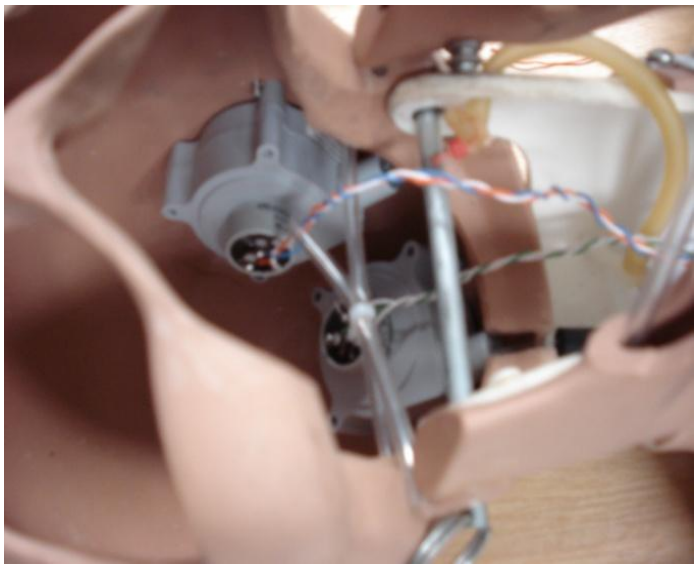


Figura 7.26. StringPots instalados en el cráneo para detectar la hiperextensión.

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.

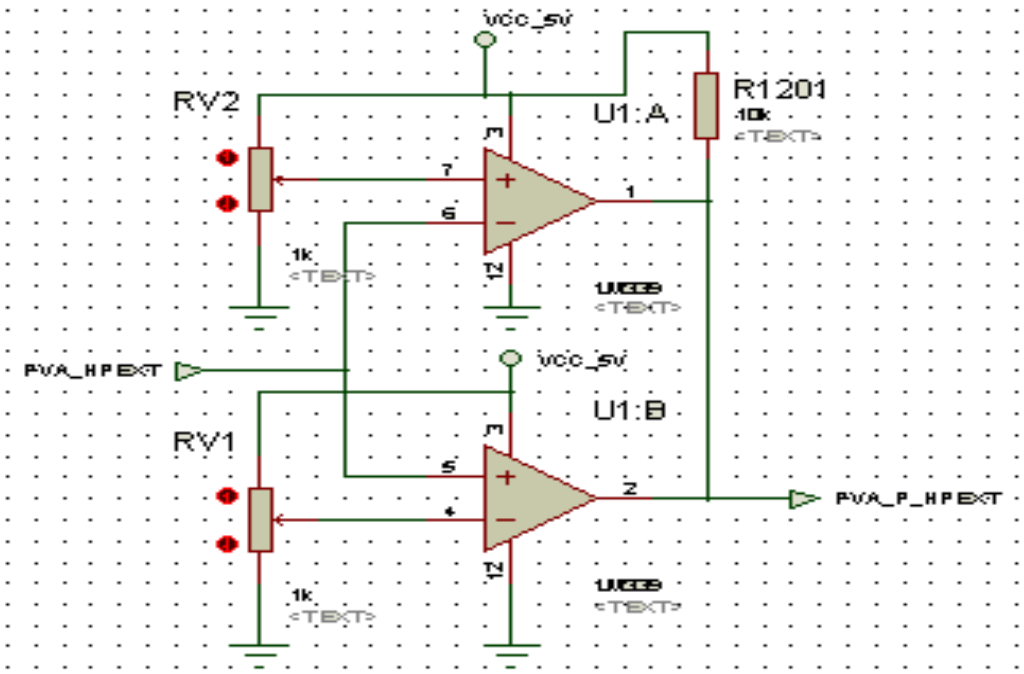


Figura 7.27. Esquema del sensor de hiperextensión.

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

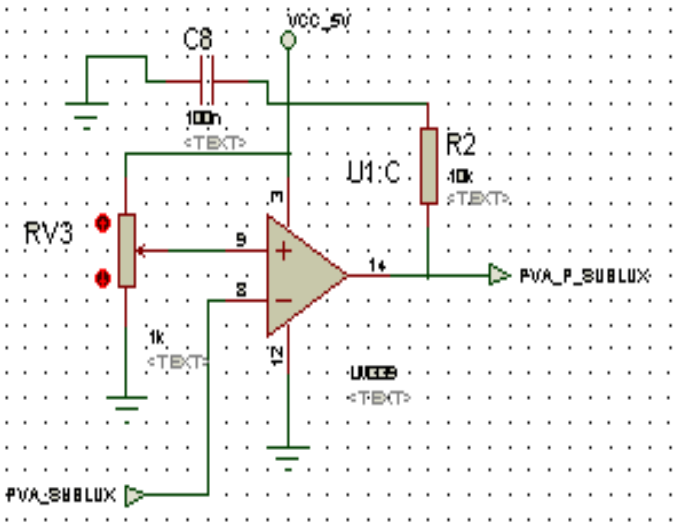


Figura 7.28. Esquema del sensor de subluxación

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

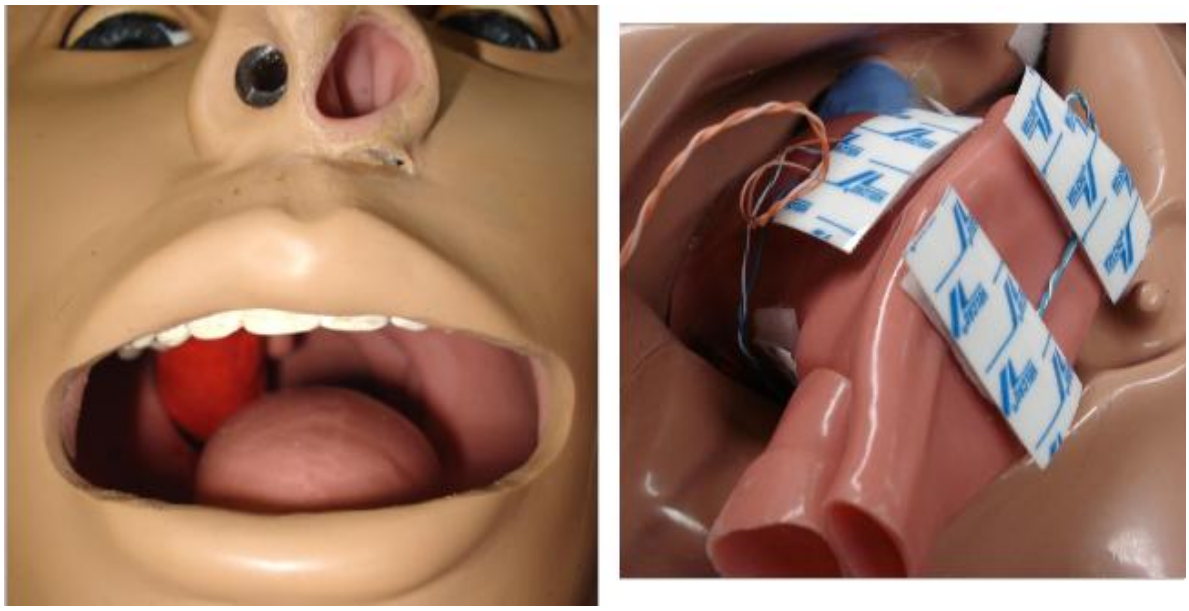
7.2.4.4.4-1 Esquema general

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.

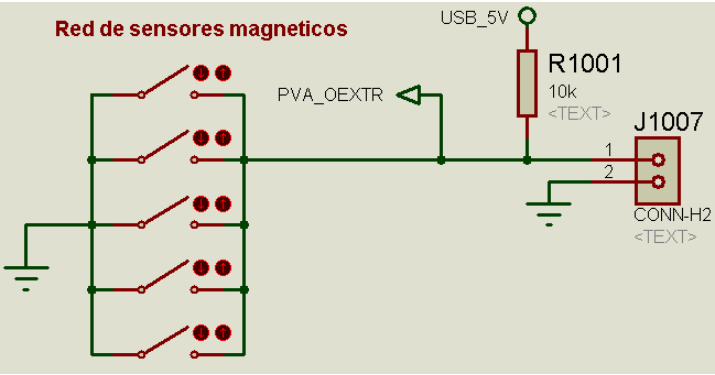


Figura 7.31. Acoplamiento de los sensores magnéticos.

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.

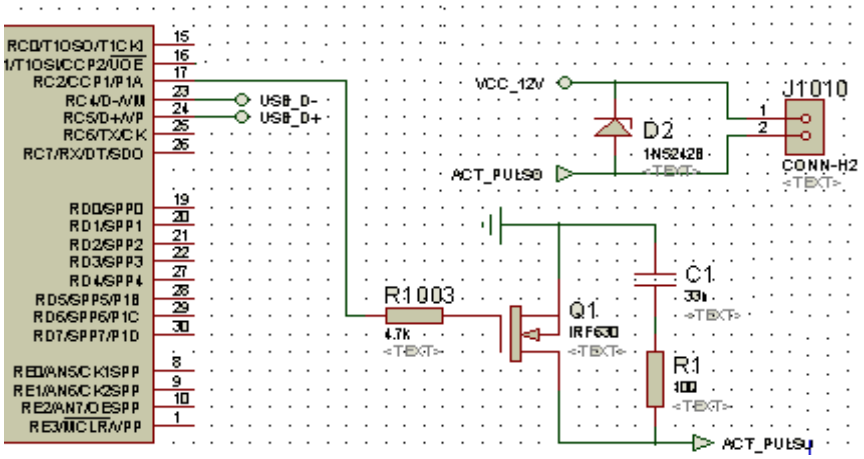


Figura 7.34. Esquema de la etapa de potencia del pulso.

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.

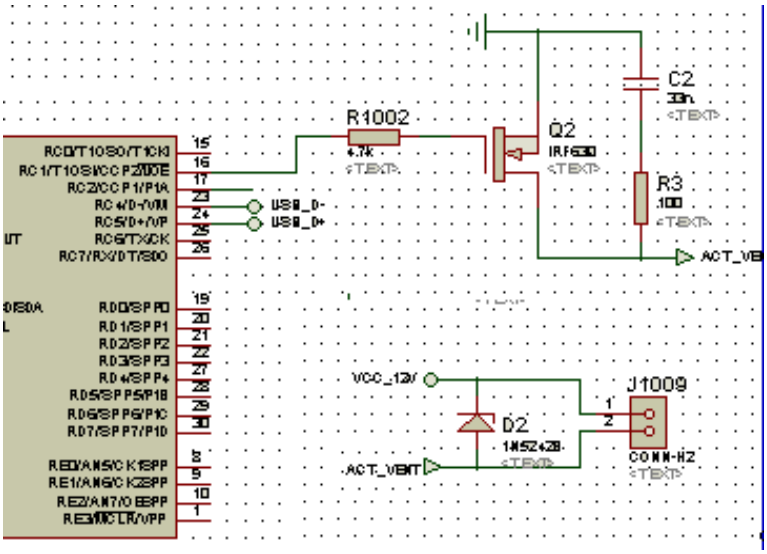


Figura 7.36. Esquema de la señal de control

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.

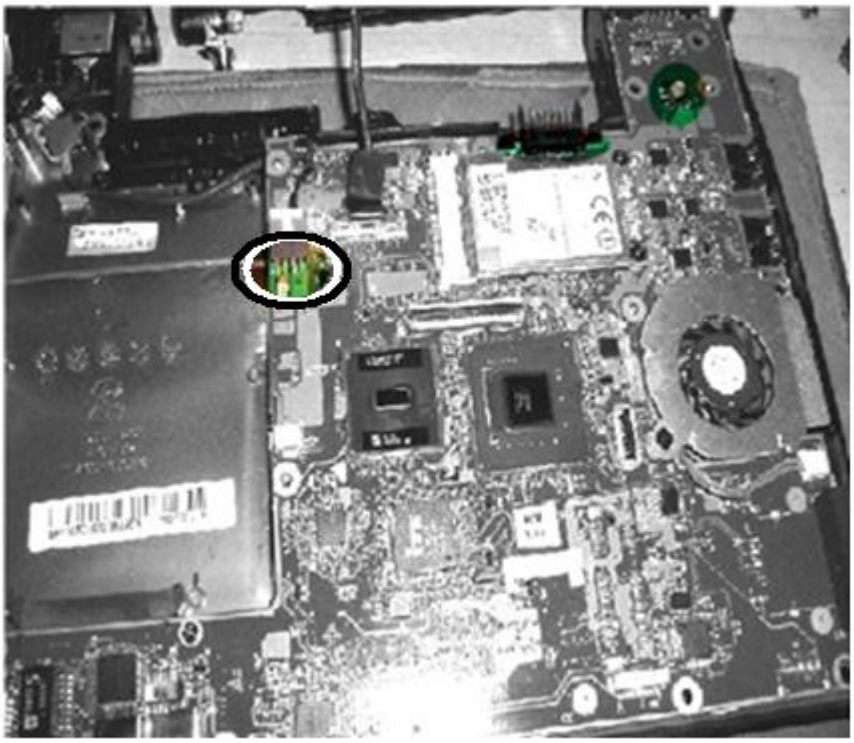


Figura 7.38. Tarjeta madre de la Netbook

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.

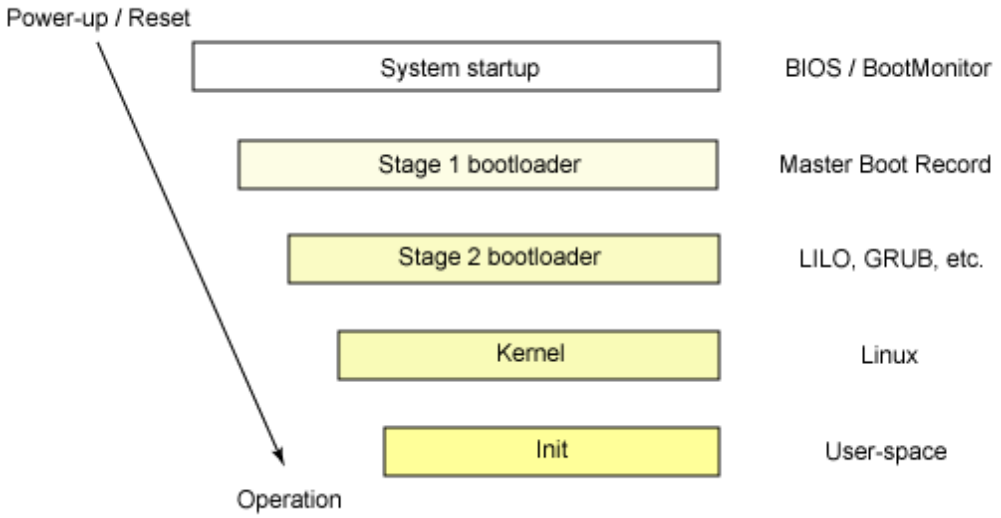


Figura 7.39. Proceso de inicio de la unidad de procesamiento.

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.

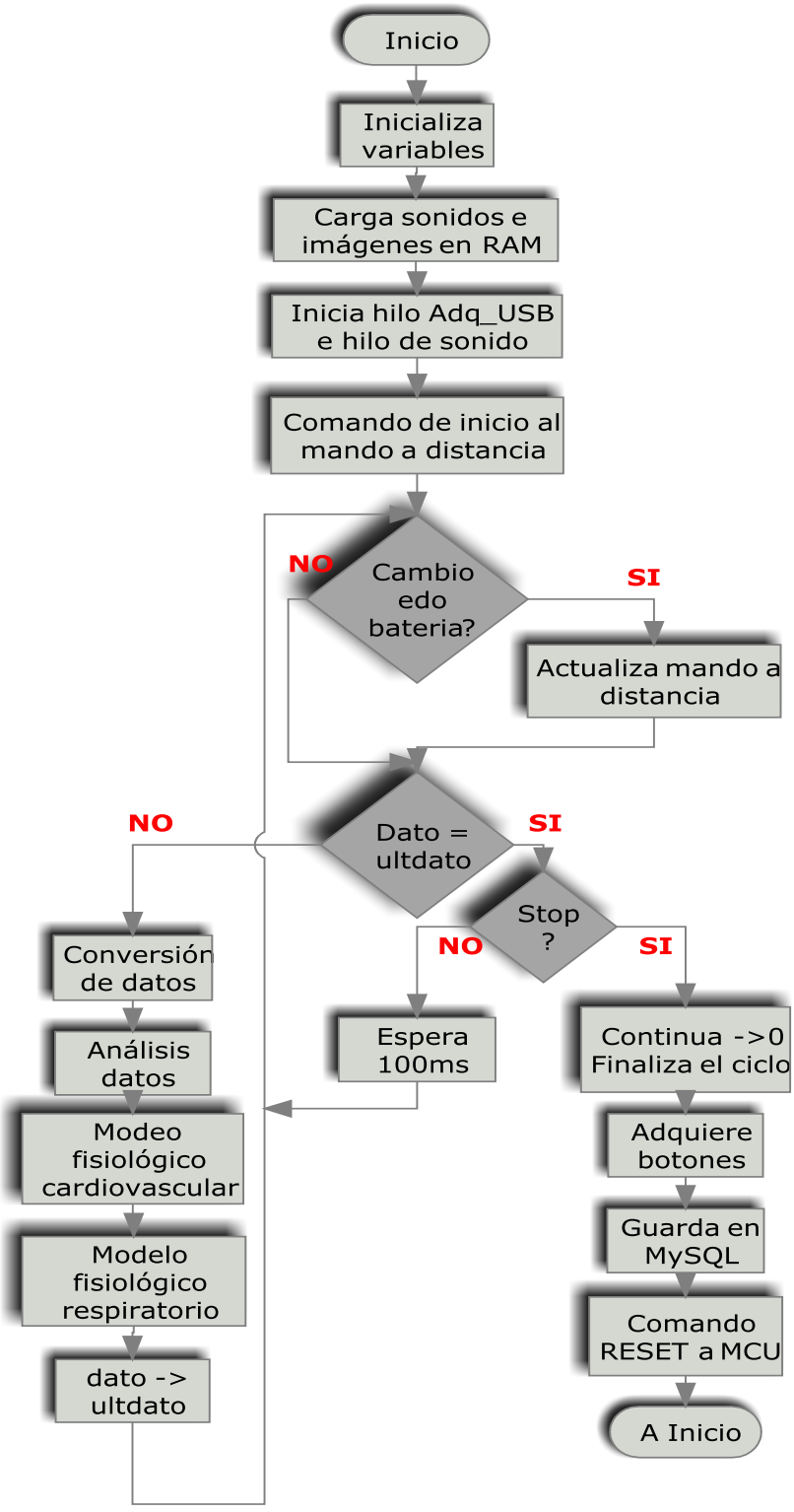


Figura 7.41. Diagrama de flujo de la trama principal.

Al terminar la aplicación se ejecutan los siguientes pasos:

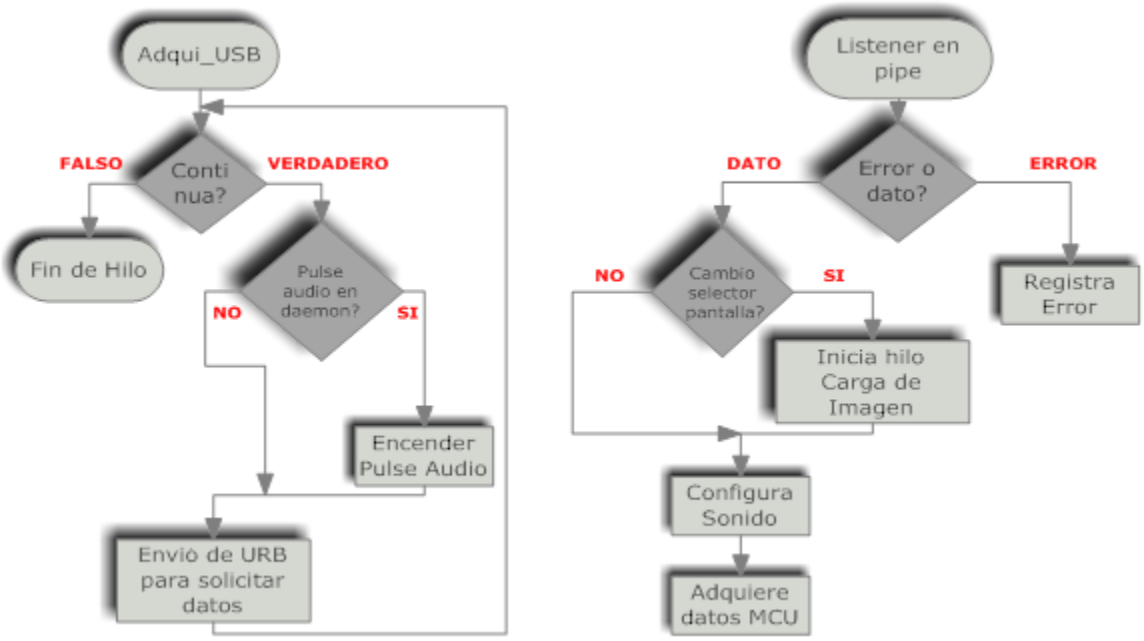
1. Se marca la variable continua como falsa, siendo esta la que controla al trama de adquisición para que este continúe o no ejecutándose; se finaliza los datos del ciclo, como son el número de compresiones, la frecuencia e intensidad promedio de las mismas en el ciclo, etcétera

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

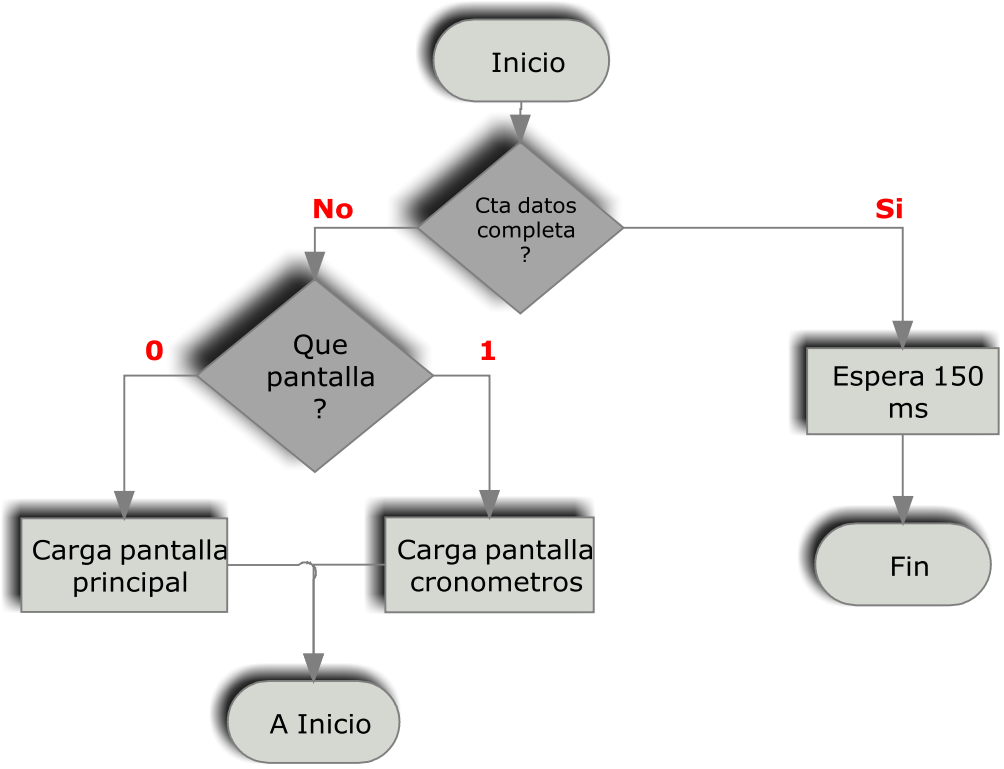


Figura 7.43. Diagrama de flujo del trama de cargado de imágenes

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

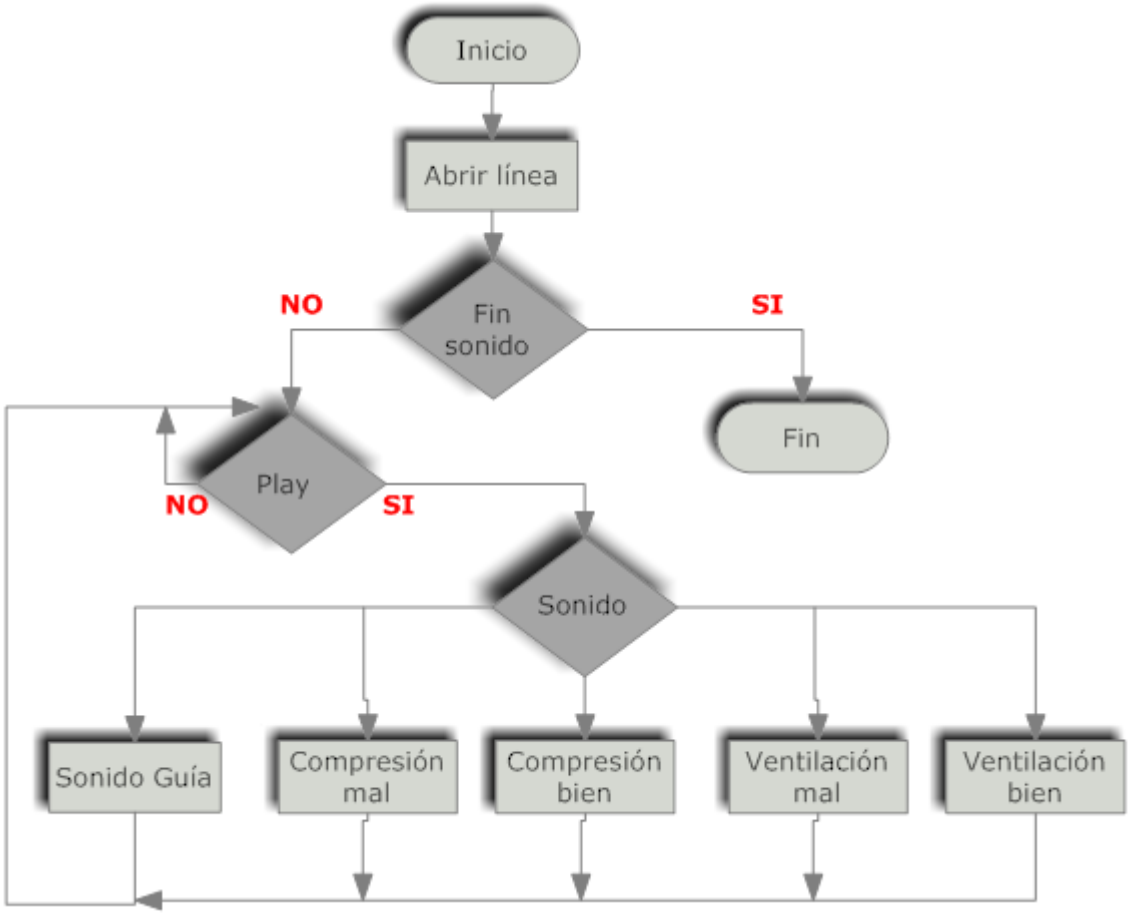


Figura 7.44. Diagrama de flujo de la trama de sonido.

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*.

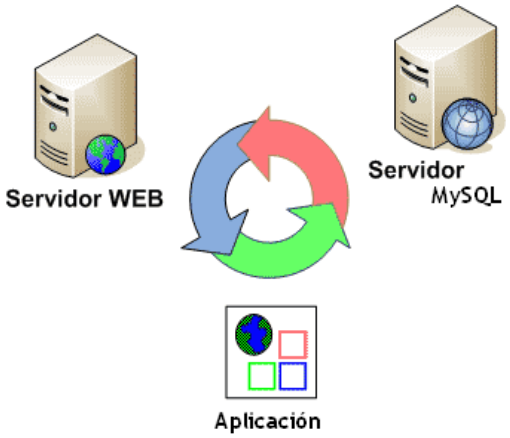


Figura 7.45. Diagrama funcionamiento servidor.

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.

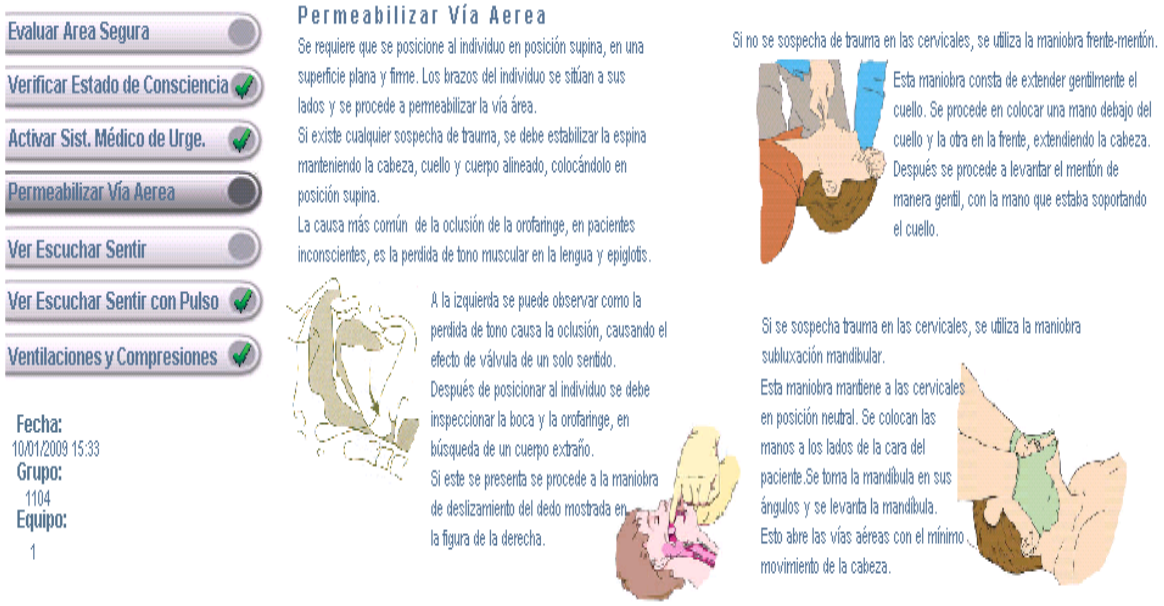


Figura 7.46. Parte de la interfaz con la información teórica.

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,

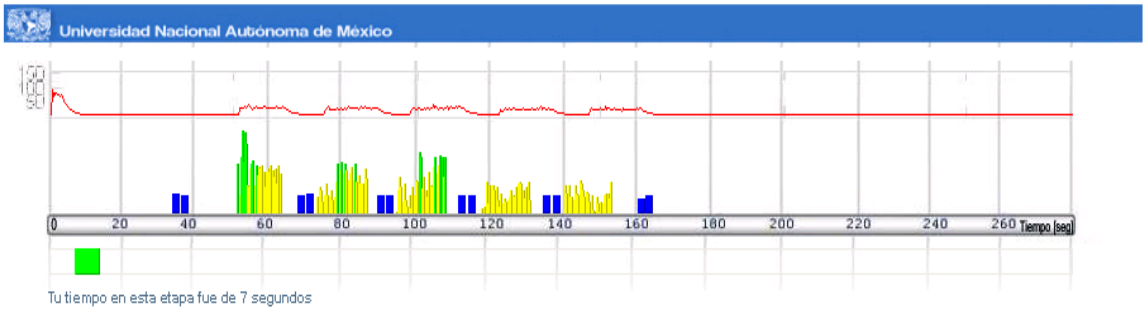


Figura 7.47. Parte de la interfaz con la línea de tiempo.

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.

8 Conclusiones

El trabajo de instrumentación del simulador, el diseño electrónico y su modelo fisiológico; completan el primer prototipo de simuladores para la enseñanza de la reanimación cardiopulmonar y establece una línea de investigación y desarrollo tecnológico del Laboratorio de Ingeniería Biomédica, en colaboración con el Centro de Enseñanza y Certificación de Aptitudes Médicas (CECAM) de la Universidad Nacional Autónoma de México.

Se acoplaron sensores que permiten detectar las maniobras de Permeabilidad de Vía Aérea (PVA), como son: hiperextensión, subluxación y presencia de cuerpo extraño en la vía aérea. Así como sensores que permiten medir las compresiones y ventilaciones aplicadas al maniquí.

Se creó un actuador de pulso carotideo y un actuador de ventilación que permite al practicante ver, escuchar y sentir la "respiración".

Se diseñó y construyó un mando a distancia alambrico que permite al practicante controlar los actuadores de sonido, pulso, y ventilación. El mando también permite monitorear las maniobras de PVA. En él es posible visualizar la intensidad de las compresiones y ventilaciones, además de la tensión arterial y la saturación de oxígeno en sangre (variables entregadas por el modelo fisiológico), el estado de las baterías, un temporizador y 5 cronómetros que permiten conocer el tiempo en que se realiza el protocolo de rescate. También cuenta con funciones que permiten iniciar, pausar o terminar el accionamiento del maniquí, es decir, se inicia, pausa o termina el funcionamiento del temporizador, actuadores y adquisición de los sensores.

Se implementó una unidad que procesa los datos enviados por el mando a distancia y los sensores permitiendo el control de las funciones del maniquí, el envio de información al servidor y el mando a distancia. Mediante la incorporación de un modelo cardiorespiratorio, esta unidad transfiere variables como la tensión arterial y la saturación de oxígeno en sangre para ser desplegadas en las interfaces del mando remoto y la página web.

Se diseñó una página web donde se muestran los resultados de la práctica de Reanimación Cardio Pulmonar (RCP) y la información teórica esencial sobre esta práctica, es decir, se muestra información teórica de la RCP en información sobre la práctica realizada en el simulador como son: las gráficas de las compresiones y ventilaciones, número de compresiones por ciclo y la frecuencia, tensión arterial y la saturación de oxígeno en sangre, PH sanguíneo, entre otras variables.

9 Apéndice

A Cyclic redundancy check

La base de la revisión por redundancia cíclica (CRC por sus siglas en inglés), se encuentra en la división de enteros, la cual se encuentra formada por un dividendo, un divisor y un residuo. El proceso de transmisión de un mensaje con CRC consta de un mensaje (dividendo) y la CRC (residuo), por lo tanto la transmisión puede ser verificada si se vuelve a calcular la división del mensaje y los residuos coinciden. Otra manera de verificar el mensaje, siendo ésta la más común, es restarle el residuo al mensaje y verificar si el resultado el residuo es cero.

El concepto de división puede ser también aplicado a los polinomios, ya que el CRC trata al mensaje como un polinomio, por ejemplo, el mensaje 11001001 daría como polinomio el $x^7+x^6+x^3+1$, pero para que la CRC pueda ser posible el transmisor y el receptor deben estar en concordancia con el divisor que se va a utilizar para el mensaje.

La elección de este divisor no es fácil ya que dependiendo de las características de este dependerá la calidad de detección de los errores de transmisión. Una particularidad del divisor es que el residuo de la división será siempre menor al orden del divisor, por lo tanto, el número de *bits* a utilizar para la CRC será el orden de su polinomio divisor.

Para el protocolo USB existen dos divisores:

- El CRC5, utilizado para los paquetes tipo *token*[1], consta de 5 bits, siendo su polinomio x^5+x^2+1.
- El CRC16, que es utilizado para los paquetes de datos, consta de 16 bits, siendo su polinomio el $x^{16}+x^{15}+x^2+1$.

Un problema del CRC es que al enfrentar un mensaje que contenga únicamente "ceros" éste sería tomado como correcto. El problema se supera aplicando las siguientes medidas:

- El registro es cargado con "unos" antes de iniciar la operación, esto es similar a sumar una constante al dividendo. Al no aplicar esta instrucción los primeros "ceros" no serían protegidos por el algoritmo.
- El residuo es invertido antes de ser anexado al mensaje, esto es similar a sumar una constante al residuo. Sin emplear esta medida el algoritmo no protegería los últimos "ceros" del mensaje.

  Estas medidas deben ser proporcionadas al transmisor y al divisor para ser efectivas.

[1] Token: tipo de paquete en el protocolo USB (véase la sección 2.3.1.2)

B Datos calibración sensor ventilación

Flujo [ml/min]	Voltaje [V]	Flujo [ml/min]	Voltaje [V]	Flujo [ml/min]	Voltaje [V]
0	2.504	0	2.503	320	2.524
1243	2.539	1450	2.593	1300	2.533
1465	2.542	1807	2.597	1620	2.539
1640	2.547	2323	2.606	1810	2.543
1852	2.55	3356	2.623	2090	2.546
1945	2.551	4093	2.633	2310	2.55
2131	2.552	4362	2.636	2480	2.552
2391	2.557	5031	2.646	2780	2.561
2619	2.56	5270	2.647	3500	2.561
2804	2.561	5867	2.657	3150	2.564
3120	2.565	7007	2.675	3380	2.564
3520	2.569	7443	2.684	3740	2.57
2817	2.574	7923	2.686	4130	2.576
4050	2.579	8629	2.7	4370	2.581
4715	2.587	9368	2.71	4600	2.584
5182	2.594	14211	2.726	4840	2.591
5745	2.6	15535	2.74	5010	2.593
6360	2.612	15776	2.744	5460	2.598
6678	2.618	16591	2.758	5730	2.602
6945	2.62	17011	2.763	5980	2.606
7737	2.635	17351	2.766	6210	2.609
8402	2.647	18360	2.779	6400	2.613
8837	2.649	18968	2.79	6700	2.617
9210	2.659	20310	2.808	6850	2.618
9868	2.67	20910	2.814	7130	2.625
10420	2.677	21990	2.829	7340	2.626
11055	2.691	22910	2.837	8070	2.636
11446	2.691	23710	2.845	8270	2.642
12513	2.709	24621	2.858	8820	2.649
13619	2.724	26483	2.881	8990	2.65
14725	2.74	28664	2.903	9510	2.66
15166	2.746	30476	2.919	10100	2.666
15943	2.758	33483	2.955	10800	2.676
16299	2.759	35885	2.98	11500	2.69
17169	2.768	39736	3.02	12300	2.702
18113	2.78	40836	3.031	13100	2.702
18916	2.791	43422	3.059	14000	2.725
20201	2.809	44873	3.069	14800	2.735
21195	2.82	45206	3.077	15400	2.745
22230	2.83	46333	3.081	16700	2.763
23100	2.845	47127	3.09	17900	2.777

25235	2.869	47753	3.094	18600	2.784
29173	2.91	49093	3.109	19600	2.796
33200	2.956	50775	3.126	20800	2.803
39742	3.029	51547	3.133	21700	2.824
41990	3.05	52703	3.14	23700	2.824
46370	3.085	54545	3.156	23400	2.84
49014	3.115	57223	3.188	24700	2.855
54419	3.169	59069	3.196	27000	2.882
60198	3.216	61036	3.21	37100	2.995
65729	3.263	64166	3.239	46200	3.082
69025	3.286	67761	3.27	49000	3.106
72194	3.311	70456	3.294	52400	3.142
77507	3.355	73314	3.314	56700	3.181
81274	3.399	76572	3.339	60300	3.211
84953	3.41	78391	3.352	64900	3.248
87275	3.424	79916	3.364	68400	3.27
89548	3.443	81220	3.371	73100	3.313
93179	3.469	83304	3.388	78000	3.355
96301	3.489	86721	3.412	80200	3.381
98250	3.503	89871	3.437	86800	3.417
102182	3.535	91934	3.454	91100	3.442
104005	3.545	94034	3.469	93600	3.457
105229	3.556	95097	3.471	97400	3.492
106043	3.561	96646	3.486	99400	3.501
106585	3.563	97606	3.492	101400	3.523
		98560	3.5	104700	3.542
		100165	3.512	106200	3.554
		101424	3.517	107900	3.564
		102304	3.524		
		103.369	3.533		
		104.407	3.539		
		105.602	3.557		
		106.464	3.555		

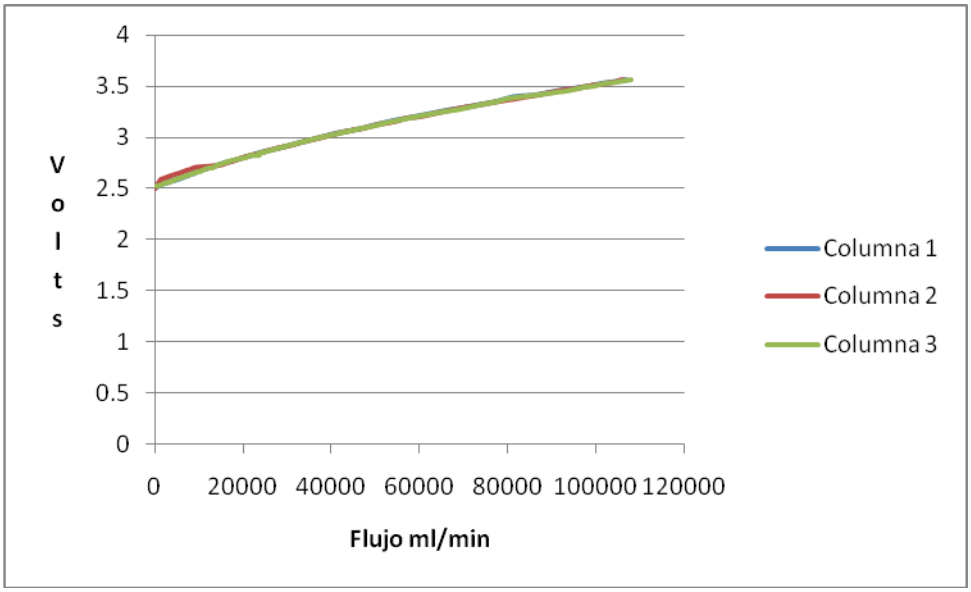


Figura B.1. Gráfica de voltaje vs Flujo.

C Modelado cardiovascular

La simulación del sistema cardiovascular se basa en el flujo de sangre en los vasos sanguíneos provocado por cambios de presión torácica. La simulación se encuentra basada en el trabajo hecho por Babbs[2], en el cual planteó que el sistema está formado por cámaras que se encuentran interconectadas por vasos sanguíneos por medio de una resistencia. Para este proyecto se plantearon siete cámaras, como se muestra en la figura C.1.

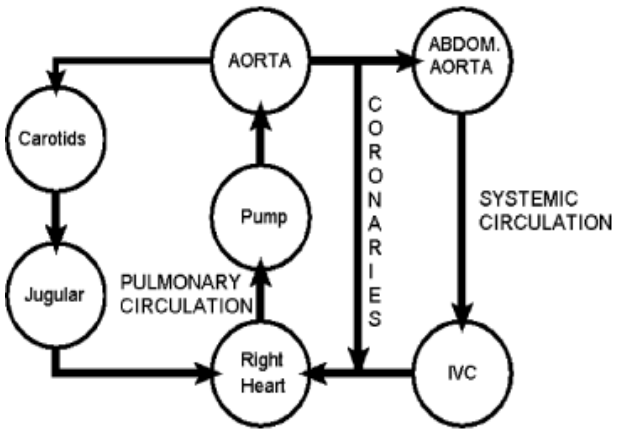


Figura C.1. Modelo del sistema circulatorio humano.

[2] Babbs Charles, "CPR techniques that combine chest and abdominal compression and decompression: hemodynamic insights from a spreandsheet model", Circulation, vol 100, No. 21, nov, 1999, USA, p. 2146-2152.

Se planea que cada cámara se pueda expandir, incrementando su volumen y a su vez la presión interna de la misma, lo que nos lleva a definir la complianza de la cámara:

$$C = \frac{\Delta V_{\text{olumen}}}{\Delta P_{\text{resión}}}$$

La elevación de la presión en las cámaras depende del flujo "i", el cual depende de la diferencia de presión de las cámaras y de la resistencia entre ellas, por lo tanto surgen siete ecuaciones en el sistema que describen el aumento de presión en cada cámara. Estas ecuaciones son las siguientes:

Cámara aorta abdominal (AA)

$$(1)\quad \Delta P_{AA} = \Delta P_{abd} + \frac{1}{C_{AA}}(i_a - i_s)\Delta t = \Delta P_{abd} + \frac{\Delta t}{C_{AA}}\left[\frac{1}{R_a}(P_{Ao} - P_{AA}) - \frac{1}{R_s}(P_{AA} - P_{IVC})\right].$$

Cámara vena cava inferior (IVC)

$$(2)\quad \Delta P_{IVC} = \Delta P_{abd} + \frac{1}{C_{IVC}}(i_s - i_v)\Delta t = \Delta P_{abd} + \frac{\Delta t}{C_{IVC}}\left[\frac{1}{R_s}(P_{AA} - P_{IVC}) - \frac{1}{R_v}(P_{IVC} - P_{RH})\right]$$

Cámara carótida (Car)

$$(3)\quad \Delta P_{car} = \frac{1}{C_{car}}(i_c - i_h)\Delta t = \frac{\Delta t}{C_{car}}\left[\frac{1}{R_c}(P_{Ao} - P_{car}) - \frac{1}{R_h}(P_{car} - P_{jug})\right]$$

Cámara yugular (jug) Siendo N = 1 normalmente e N = 0 cuando se efectúan las compresiones

$$(4)\quad \Delta P_{jug} = \frac{1}{C_{jug}}(i_h - i_j)\Delta t = \frac{\Delta t}{C_{jug}}\left[\frac{1}{R_h}(P_{car} - P_{jug}) - N\frac{1}{R_j}(P_{jug} - P_{RH})\right]$$

Cámara aórtica (Ao) Siendo E = 1 cuando la válvula aórtica se encuentra abierta.

$$(5)\quad \Delta P_{Ao} = \Delta P_{chest} + \frac{1}{C_{Ao}}(i_o - i_c - i_a - i_{ht})\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_{Ao}}\left[E\frac{1}{R_o}(P_p - P_{Ao}) - \frac{1}{R_c}(P_{Ao} - P_{car}) \right.$$
$$\left. - \frac{1}{R_a}(P_{Ao} - P_{AA}) - \frac{1}{R_{ht}}(P_{Ao} - P_{RH}) \right],$$

Cámara corazón derecho (RH) Siendo F = 1 cuando la válvula pulmonar se encuentra abierta.

Cámara de bomba (P)

$$(6)\quad \Delta P_{RH} = \Delta P_{chest} + \frac{1}{C_{RH}}(i_j + i_v + i_{ht} - i_i)\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_{RH}}\left[N\frac{1}{R_j}(P_{jug} - P_{RH}) + \frac{1}{R_v}(P_{IVC} - P_{RH}) \right.$$
$$\left. + \frac{1}{R_{ht}}(P_{Ao} - P_{RH}) - F\frac{1}{R_i}(P_{RH} - P_p) \right],$$

$$(7)\quad \Delta P_p = \Delta P_{chest} + \frac{1}{C_p}(i_i - i_o)\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_p}\left[F\frac{1}{R_i}(P_{RH} - P_p) - E\frac{1}{R_o}(P_p - P_{Ao}) \right].$$

La presión en el pecho es variable y determinada por el sensor y su calibración.

Estas ecuaciones deben ser evaluadas a intervalos de tiempos pequeños para que sean convergentes, y con las mismas se puede obtener la presión en cualquier cámara así como el flujo entre las mismas.

D Modelado de ventilación

El modelado respiratorio se basa en el intercambio de gases como ocurre con los alvéolos de los pulmones. Este modelado depende de la fracción parcial de oxígeno aspirada FiO2, así como de presiones parciales de oxígeno y dióxido de carbono, tanto en el alvéolo como en los vasos sanguíneos que están en contacto.

Se utilizó como ejemplo un metabolismo constante durante la simulación, es decir, el consumo de oxígeno y la producción de dióxido de carbono es constante. La figura D.1 se basa en el trabajo de Tehrani[3].

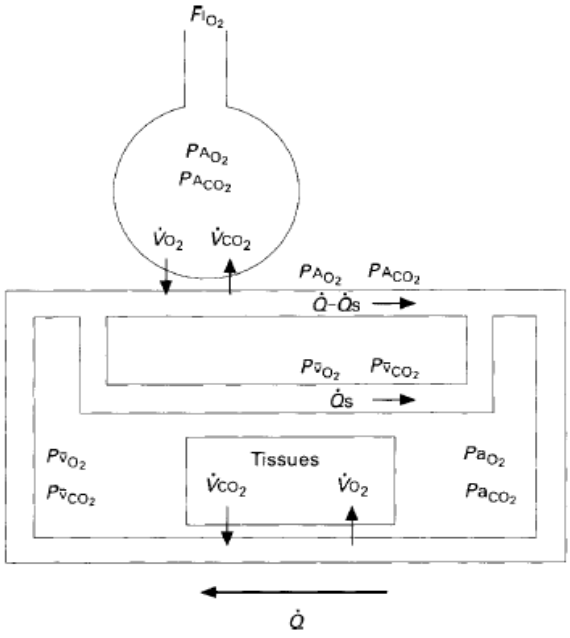


Figura D.1. Modelo del sistema respiratorio humano.

La ecuación del flujo de masa en los alvéolos para el dióxido de carbono:

$$(C_{VTCO_2} - C_{aCO_2})(1-\alpha)Q = \frac{v}{P_b - 47}\frac{dP_{ACO_2}}{dt} + \frac{P_{ACO_2} - P_{ICO_2}}{P_b - 47}\frac{dv}{dt}$$

Y para el oxígeno:

$$(C_{VTO_2} - C_{aO_2})(1-\alpha)Q = \frac{v}{P_b - 47}\frac{dP_{AO_2}}{dt} + \frac{P_{AO_2} - P_{IO_2}}{P_b - 47}\frac{dv}{dt}$$

[3] Tehrani Fleur, "Dinamic modelling of the human respiratory system", PhD Thesis, University of London, 1981.

El primer término es la deferencia de las concentraciones de CO2 y O2, respectivamente, entre las venas y las arterias, multiplicado por un porcentaje *alpha*, que no participa en el intercambio (*shunt fraction*). Y el segundo término es la variación de la concentración parcial en el alvéolo sumado a las presiones parciales del aire inspirado, este valor se vuelve cero durante la expiración. Pb representa la presión barométrica.

Para simplificar el modelo, la relación entre las presiones parciales en el alvéolo y los vasos sanguíneos, es la siguiente:

Al asumir que la mezcla es homogénea entre la sangre venosa y la arterial. Siendo *alpha* la *shunt fraction*.

$$P_{ACO_2} = P_{aCO_2} \qquad P_{AO_2} = P_{aO_2} + K$$

$$C_{amCO_2} = (1-\alpha)C_{aCO_2} + \alpha C_{VTCO_2}$$

$$C_{amO_2} = (1-\alpha)C_{aO_2} + \alpha C_{VTO_2}$$

Y la ecuación para el tejido, la siguiente:

$$C_{VTCO_2}Q = C_{amCO_2}Q + MR_{TCO_2} - S_T\frac{dC_{TCO_2}}{dt}$$

$$C_{VTO_2}Q = C_{amO_2}Q - MR_{TO_2} - S_T\frac{dC_{TO_2}}{dt}$$

Donde MR es la tasa de metabolismo constante, y St una constante de equivalencia de almacenamiento de gas en los tejidos.

Otras ecuaciones que deben ser utilizadas para completar el modelado son las siguientes:

Estas ecuaciones describen las curvas de disociación de los gases.

$$C_{CO_2} = K_3 P_{CO_2} \qquad C_{O_2} = K_4(1 - e^{-K_5 P_{O_2}})^2.$$

E Descripción del protocolo I^2C

Los dos cables, llamados *serial Data* (SDA) y *serial Clock* (SCL), transportan la información entre los dispositivos conectados al *bus*, ambos cables son bidireccionales y se encuentran conectados a una fuente de voltaje positiva. Esta fuente es de corriente o una resistencia *pull up*[4]. Cuando el *bus* está libre las dos líneas se encuentran en su estado alto[5].

La etapa de salida de los dispositivos debe ser de tipo *colector abierto*[6] para que pueda funcionar el arbitraje.

Para que la información sea válida, la línea de datos debe mantenerse estable durante el periodo alto del reloj. El cambio de línea sólo se permite durante el estado bajo[7] del reloj.

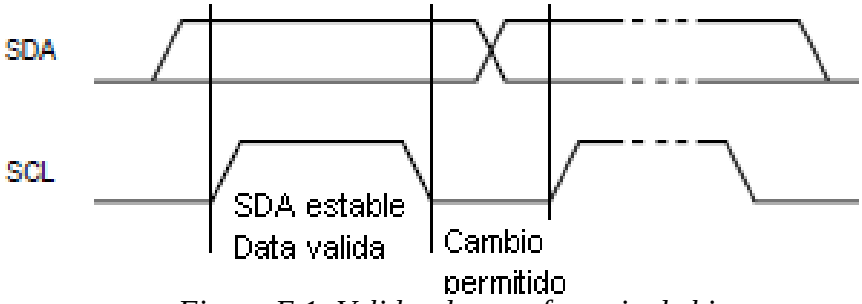


Figura E.1: Validez de transferencia de bit

Toda transferencia debe ser iniciada con una condición de inicio (*Start*), y una condición de fin (*Stop*). Todas estas generadas por el maestro, la condición de inicio se encuentra definida con una transición de alto a bajo de la línea SDA, mientras la línea SCL se encuentra en su estado alto; y la condición de fin estando definida por la transición de bajo a alto durante el estado alto de SCL como se muestra en la figura E.2.

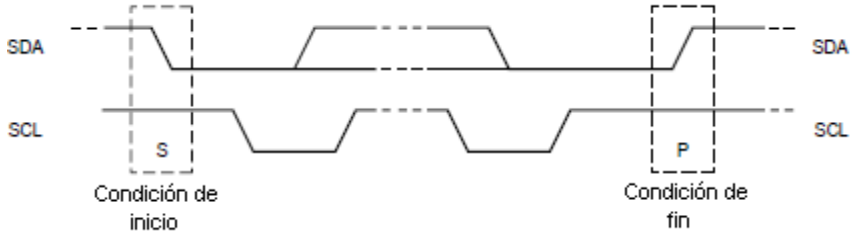


Figura E.2. Condiciones de inicio y de fin en el bus I^2C

---

[4] Se le llama resistencia en *Pull up*, ya que se dice que la fuente 'jala hacia arriba' el voltaje de las terminales de la resistencia al mismo potencial cuando no existe paso de corriente.

[5] Se llama estado alto cuando en los sistemas digitales el valor del voltaje se encuentra arriba del umbral especificado para su valor '1' lógico.

[6] El colector abierto es un tipo de salida de los circuitos integrados, donde en vez de mandar como salida un voltaje o corriente específico, se externaliza directamente la terminal del transistor de salida.

[7] Se llama estado bajo cuando en los sistemas digitales el valor del voltaje se encuentra abajo del umbral especificado para su valor '0' lógico.

Hay que recordar que durante la transmisión de los datos no se debe modificar la línea SDA durante el estado alto de SCL pues se generarían falsos positivos sobre las condiciones de inicio y fin de una transmisión. Es posible repetir una condición de inicio sin tener que dar una condición de fin con la finalidad de no liberar el *bus*.

El formato de los datos debe ser de 8 *bits*[8], enviando primero aquel más significativo (*Most Significant Bit First*), el número de *bytes* (8 *bits*) que puede ser transmitido no es restringido, todos deben ser seguidos por un *bit* de recibido (*ACK*[9]).

El *bit* ACK indica que el *byte* ha sido recibido satisfactoriamente y por lo tanto puede enviarse el siguiente *byte*, es generado por el maestro, pero este libera la línea SDA, para que el esclavo pueda llevarla hacia nivel bajo para generar el *bit* de ACK. Si la línea permanece en alto significa que el *byte* no fue bien recibido (*NACK*[10]) debido a:

- No existe un dispositivo con la dirección generada por el maestro así que ningún dispositivo contestará al *bit* ACK.
- El dispositivo no está listo para recibir porque se encuentra realizando otra tarea.
- Durante la transmisión el dispositivo recibe datos o comandos que no comprende.
- Durante la transferencia, el dispositivo ya no puede recibir más datos.
- El maestro termina la transferencia de datos que el esclavo se encuentra transmitiendo.

La transmisión de datos debe seguir el formato mostrado en la figura E.3.

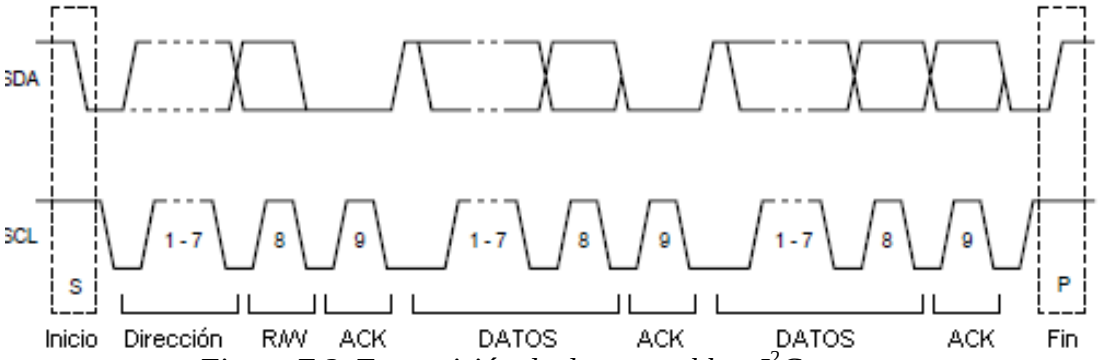


Figura E.3. Transmisión de datos en el bus I^2C

Después de la condición de inicio la dirección del esclavo debe ser enviada, la dirección consta de 7 *bits* de identificación y un *bit* de dirección de la información, el "cero" en este *bit* quiere decir que la información se dirige al esclavo (*write*), el "uno" significa que el esclavo debe transmitir información (*read*).

El esclavo también tiene la capacidad de pausar la transmisión, obligando a la línea SCL a mantenerse en su estado bajo. Esta característica permite que el esclavo tenga tiempo suficiente para procesar la información recibida o preparar la información a transferir.

Cuando se utilizan varios maestros en el mismo *bus* es necesario un método para saber quién va tomar el control del *bus* y realizar su transmisión, esto se logra a través de la

[8] El bit es el acrónimo de **B**inary Dig**it**, que significa Digito binario.
[9] El ACK es el acrónimo de Acknowledge, que significa recibido.
[10] El NACK es el acrónimo de No Acknowledge, que significa no recibido.

sincronización del reloj y el arbitraje. En sistemas de un solo maestro esto no es necesario.

La sincronización del reloj es posible gracias a un AND cableado[11]. Con esto se quiere decir que una transición de alto a bajo en SCL provocará que los demás maestro comiencen a contar su periodo bajo del reloj, este maestro mantendrá SCL en bajo hasta que termine su periodo bajo, pero si algún otro maestro continúa con su periodo bajo la línea mantendrá el mismo estado.

Los maestros que ya hayan acabo de contar su periodo bajo quedarán en estado de espera (*stand by*). Cuando se libera SCL cambia su estado a alto, el contador de estado alto de todos los maestro se inicia y el primero que termine cambia el estado de la línea a bajo. Por lo tanto el periodo del reloj está definido por el contador del maestro con el periodo bajo más largo y el contador del maestro con el periodo alto más corto.

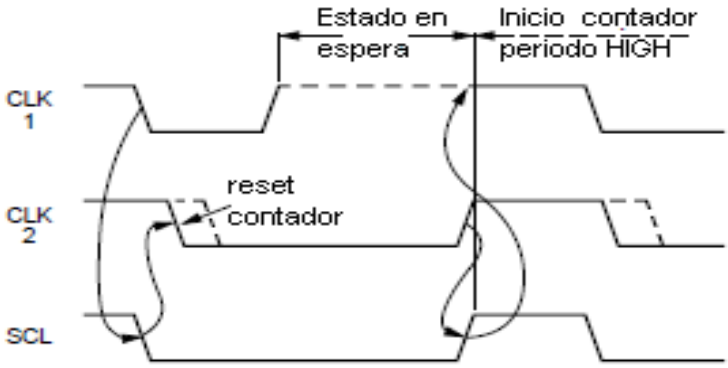


Figura E.4. Sincronización de reloj

El arbitraje entra en operación cuando dos maestros dan una condición de inicio antes del tiempo mínimo[12] de reten de la posición de inicio, por lo tanto el arbitraje debe entrar en operación para determinar qué maestro completará su transmisión.

Se procede *bit* por *bit*, cada maestro revisa la línea SDA para ver si ésta coincide con lo que él está mandando. Si en algún momento la línea no coincide, es porque el maestro ha perdido el arbitraje, entonces deberá cesar la transmisión de información y pasar de inmediato al modo esclavo, por lo tanto no hay pérdida de información.

Es importante reconocer que la prioridad en el arbitraje la tienen los datos de menor valor, sea en la dirección del esclavo o en la información a transmitir.

[11] El AND-cableado es una configuración donde la funcional lógica del AND se encuentra implícita en el cableado.

[12] Para mayor información sobre el tiempo mínimo consultar The I2C Bus especification, versión 2.1, 2000

F Descripción del protocolo USB

El cable está formado por terminales; alimentación (Vcc), tierra (Gnd), D+ y D-. Estos dos último son utilizados para transferir la información.

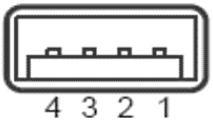


Figura F.1. Numeración de pines, USB specification, rev 2.0, 2000

Al igual que el *bus* I^2C, la comunicación para el *bus* universal serial es iniciada únicamente por el *host*[13] No se pueden conectar dos *host* entre ellos, por lo tanto se han diseñado dos tipos de conectores una para el *host* tipo A y otro para el dispositivo tipo B. La forma de los conectores se muestra en la figura F.2.

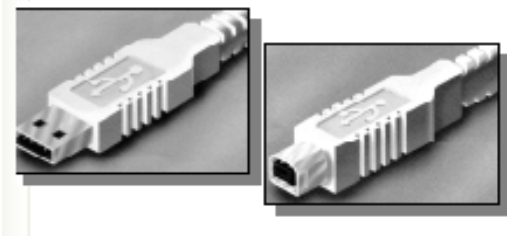


Figura F.2. Tipos de conectores

La transmisión de los datos se hace vía diferencial, es decir, se mide la diferencia entre los dos cables y este dato proporciona el valor de la señal. Este modo de transferencia permite mayores velocidades de transmisión, con cables relativamente largos.

En este proyecto sólo se utilizó el modo *Full Speed*, entonces será el descrito aquí.

La señal diferencial debe contener las siguientes características para que pueda ser detectado como un “uno” diferencial o un “cero” diferencial, respectivamente.

Estado del Bus	Niveles de las señales		
	En la salida del Host	En la entrada del dispositivo	
		Requerida	Aceptable
Diferencial "1"	D+ > 2.8 V y D- < 0.3 V	(D+) - (D-)> 200mV y D+ > 2.0V	(D+) - (D-)> 200mV
Diferencial "0"	D- > 2.8 V y D+ < 0.3 V	(D-) - (D+)> 200mV y D- > 2.0V	(D-) - (D+)> 200mV

Tabla F.1 Niveles de las señales en el bus USB

[13] Se le llama Host al computador que funciona como punto de inicio y final de las transferencias de datos.

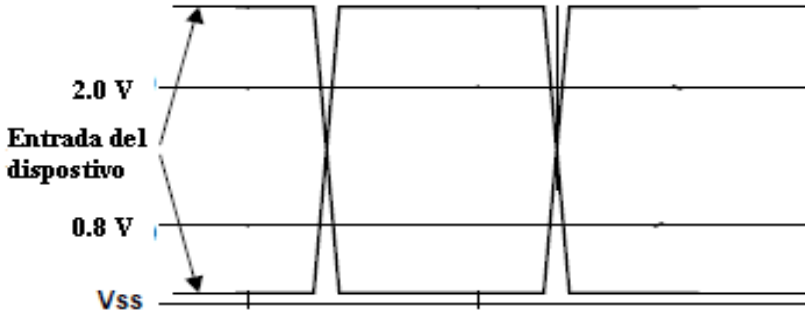
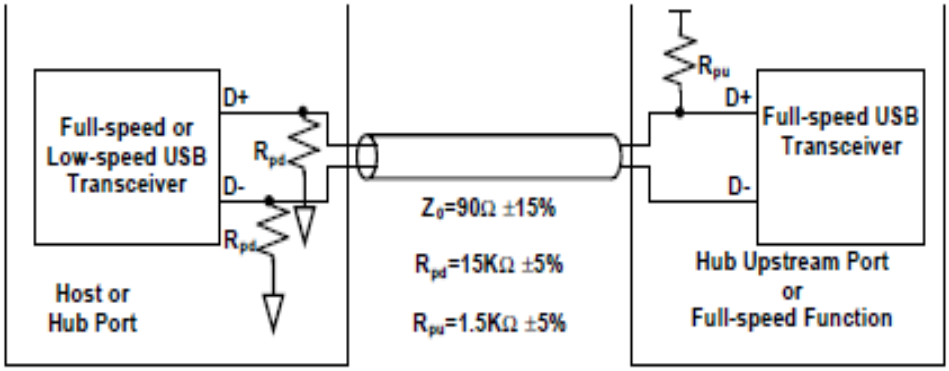
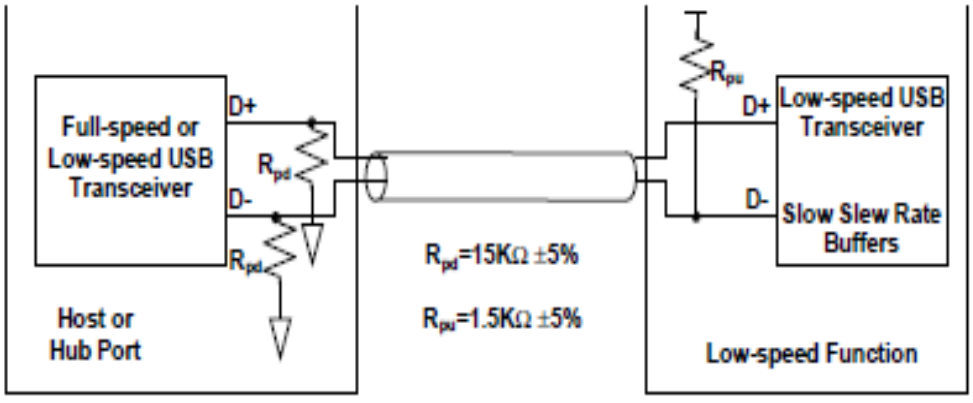


Figura F.3 Forma de onda de las señales

Para distinguir entre las diferentes velocidades de los dispositivos, el *host* revisa cuál es el estado del *bus* cada que es conectado un dispositivo. Si el *bus* se encuentra detecta un "uno" diferencial, se trata de un dispositivo *Full Speed*; si el estado es "cero" diferencial entonces el dispositivo es *Low Speed*. Para un dispositivo *High Speed* primero se debe identificar como un *Full Speed* y después cambiar al modo a *High Speed*. Este comportamiento puede comprenderse al ver el arreglo de resistencias para el bus mostrado en la figura F.4.



a)



b)

Figura F.4. Configuración del cable y resistencias, a) Full speed, b) Low speed

Cuando no hay algún dispositivos conectado las dos líneas diferenciales se encontrarán en un nivel bajo, de esta manera el *host* puede detectar la conexión de un dispositivo cuando alguna de ellas cambia de estado.

Para hablar de los estados lógicos de manera general, deben definirse dos nuevos, así no se tendrá que invertir los estados dependiendo de la tasa de transferencia.

Se definen de la siguiente manera:

	Full Speed	*Low Speed*
Estado **J**	Diferencial "1"	Diferencial "0"
Estado **K**	Diferencial "0"	Diferencial "1'

Tabla F.2 Definición de los estados lógicos en el bus USB
(Elaboración propia a partir de USB Specification, Rev. 2.0, 2000)

Debido a que el *bus* no cuenta con señal de reloj, la transmisión debe ser iniciada con *bit* de sincronización, esto sucede en el inicio de paquete (SOP[14]) y es señalado con el cambio de estado **J**(estado *idle*[15]) a **K**, este representa al primer *bit* que sincroniza. La secuencia llamada SYNC[16] requiere que sean enviados 3**KJ** seguidos por 2**K** para un total de 8 símbolos, como se muestra en la figura 2.13, al terminar la transmisión el paquete es finalizado (EOP[17]) mandando las dos líneas diferenciales a *gnd*, definiendo el estado SE0[18].

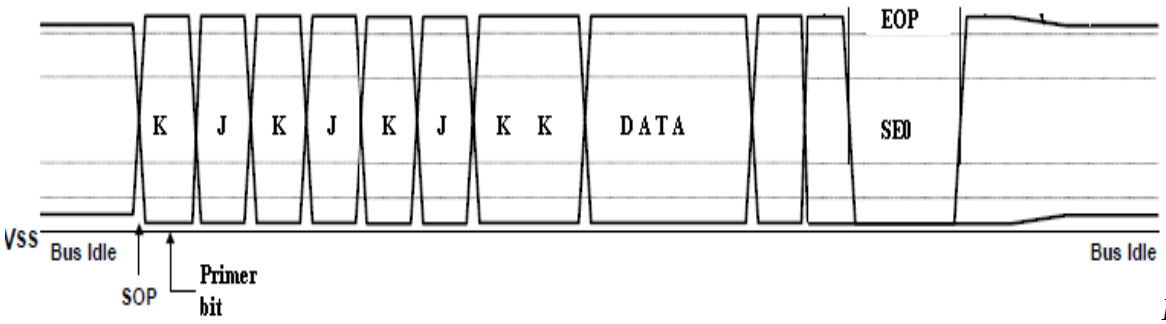


Figura F.5. Forma de onda de paquete en el bus USB

El estado SE0 también es utilizado para mandar señales de *reset* al dispositivo que esté conectado en el puerto.

Los datos son transmitidos mediante la codificación NRZI[19], en esta codificación la falta de cambio de nivel lógico es representada por los "uno" y los "cero" designan un cambio en el nivel. La figura F.6 muestra un ejemplo de la codificación.

[14] SOP es el acrónimo Start of Packet, significa inicio de paquete.
[15] El estado Idle es un estado de espera, es decir se esta a la escucha de un nuevo evento.
[16] SYNC es el acrónimo de Synchronization, que significa sincronización.
[17] EOP es el acrónimo de End of Packet, que significa fin de paquete.
[18] SE0 es el acrónimo de Single Ended Zero, que significa que las dos líneas se encuentran en estado bajo.
[19] NZRI por sus siglas en inglés, Non Return To Zero Inverted

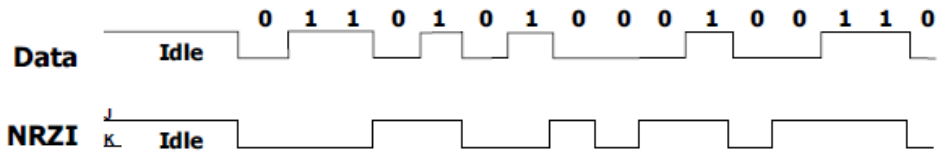


Figura F.6. Ejemplo de codificación NZRI

Para evitar la desincronización de los relojes, un "cero" es agregado después de seis "uno" consecutivos en la información a transmitir, antes de ser codificada en NRZI. El receptor debe reconocer estos *bits* y descartarlos.

Distribución de energía

Los dispositivos conectados al *bus* pueden ser alimentados por tener propia alimentación. Si el dispositivo es alimentado por el *bus* puede obtener hasta 100 mA[20] de éste.

La configuración se encuentra en baja energía pues todo dispositivo recién conectado está configurado como de baja energía. Se debe configurar por medio del *software* si se requiere mayor corriente, con la posibilidad de sustraer hasta 500 mA del *bus*. Ningún dispositivo puede entregar energía al bus, solo le está permitido sustraerla.

El dispositivo tampoco deberá proporcionar energía a las líneas diferenciales. Al no estar presente el *Vbus*[21] se removerá la energía en menos de 10 segundos.

Protocolo

El elemento principal del protocolo es el paquete y debe iniciarse con las secuencia de sincronización SYNC seguido por su identificador de paquete PID[22]. Existen tres tipos de paquetes; *token*, *data* y *handshake*.

Es importante comentar que los *bits* son enviados por el *bus* siendo el menos significativo el primero en salir.

Los paquetes son divididos en estructuras llamadas sub campos, los cuales pueden conformar los distintos paquetes. Un sub campo que debe estar incluido en todos los paquetes es el identificador de éstos.

El identificador de paquete está formado por 4 *bits* seguido por otro grupo de cuatro *bits* como se muestra en la figura F.7.

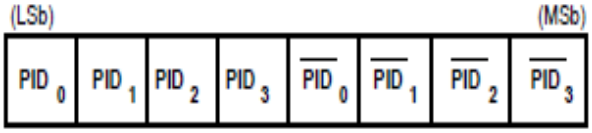


Figura F.7.Formato del identificador del paquete

Los bits de confirmación aseguran la correcta identificación del paquete formándolos con el *complemento uno*[23] del PID original.

Si el PID es inválido se considera que está incorrecto y todo el paquete es descartado.

---

[20] mA es la abreviación de miliAmperes, unidad de medida de corriente eléctrica.
[21] El Vbus es el voltaje de alimentación que otorga el host a los dispositivos.
[22] PID es el acrónimo de Packet Identifier, significa el identificador de paquete.
[23] El Complemento uno es una operación binaria, la cual consta de invertir todos los valores del número binario.

Si un PID válido llega a un dispositivo que no cuenta con la función solicitada debe ignorar al paquete recibido, por ejemplo, si a un dispositivo le llega una petición de información y éste sólo está programado para recibirla. Los identificadores de paquete son descritos en la tabla F.3.

Tipo PID	Nombre PID	PID <3:0>	Descripción
Token	OUT	0001B	Información de Host → Dispositivo
	IN	1001B	Información de Dispositivo -> Host
	SOF	0101B	Inicio de Frame
	SETUP	1101B	Información de Setup de Host → Dispositivo
Data	DATA0	0011B	Información par
	DATA1	1011B	Información non
Handshake	ACK	0010B	Información recibida sin errores
	NACK	1010B	No se puede recibir ni transmitir información
	STALL	1110B	El endpoint se encuentra desactivado
	NYET	0110B	Sin respuesta

Tabla F.3 Tabla de los valores del identificador de paquete

Los dispositivos se identifican utilizando dos sub campos; la dirección y el *endpoint*[24]. El dispositivo debe decodificar por completo los dos sub campos.

La dirección ADDR[25] es la fuente o el destino del paquete dependiendo del PID del paquete, está formada por 7 *bits* ADDR<6:0> que dan un total de 128 direcciones. La dirección "cero" está reservada ya que es la dirección predeterminada de todo dispositivo recientemente conectado al *bus*, éste después deberá ser enumerado.

El *endpoint* son 4 bits adicionales, para la identificación de la función dentro del dispositivo, por ejemplo una *webcam* puede tener un micrófono, para lo cual podría contener un *endpoint* para el video y otro para el audio. El *endpoint* "cero" se encuentra reservado, como el de control por defecto. Un dispositivo puede soportar hasta 16 *endpoints*.

La información se encuentra contenida en el sub campo *data* y su tamaño puede variar desde 0 *bytes* hasta 1024, y debe ser un número entero de *bytes*.

La verificación de la información se hace mediante el sub campo *Cyclic Redundancy Check*[26] (CRC) teniendo un tamaño de 5 *bits* para el paquete tipo *token* y 16 *bits* para el *data*. Todo paquete que falle esta verificación será descartado.

El paquete tipo *token* consiste de un PID, puede ser IN OUT o SET UP, de una dirección y del *enpoint*. Una transacción OUT o SET UP significa que el subsecuente paquete *data* contendrá información para el dispositivo especificado en este *token*.

Una Transacción tipo IN, significa que el dispositivo especificado en el *token* deberá emitir un paquete tipo DATA.

[24] El endpoint es el nombre que se la da a la entidad final de un canal de información.
[25] ADDR es el acrónimo de Address, que significa dirección.
[26] La Cyclic Redundancy Check es un tipo de verificación de errores en el mensaje, se detalla en el apéndice A.

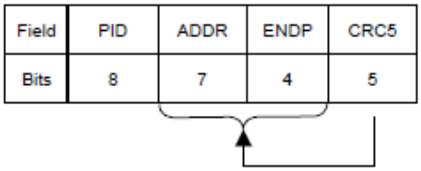


Figura F.8. Formato del paquete Token.

Estando el paquete verificado mediante un CRC, como se muestra en la figura número F.8, solo el *host* podrá emitir los paquetes tipo *token*, es decir que ningún dispositivo podrá iniciar una comunicación.

Los paquetes tipo DATA están formados por un PID, la DATA, que puede contener de 0-1024 *bytes*, y su CRC, como se muestra en la figura F.9. El PID puede variar entre DATA0 y DATA1, esto permite realizar una sincronización por alternado de la información.

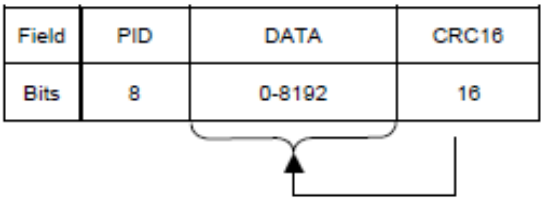


Figura F.9. Formato del paquete Data

Los paquetes tipo *Hand Shake* son utilizados en la última etapa de transmisión para informar el estado de la transacción, están formados únicamente por su PID y sólo están permitidos en algunos tipos de transacciones, además pueden sustituir la etapa de datos. Si después de emitir un paquete tipo *handshake* no se recibe un EOP, entonces después de un tiempo de *bit*, ya terminado el envío, el paquete *handshake* será descartado.

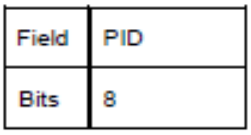


Figura F.10. Formato paquete Handshake

Si el *handshake* es del tipo ACK indica cuál fue el paquete que se transmitió de manera correcta.

Si el *handshake* es del tipo NACK quiere decir que el dispositivo no está listo para recibir el otro paquete o que no hay información para transferir.

Si el *handshake* es del tipo STALL significa que el dispositivo no puede soportar la función solicitada.

Los paquetes que conforman una transacción varía dependiendo del tipo de la misma, existen cuatro tipos de transacciones; *bulk*, *control*, i*nterrupt* e *isochronous*.

Flujo de información

El USB está diseñado para que al usuario le sea simple conectar un dispositivo, como en la figura F.11 inciso a. En realidad es más complejo, como en la figura F.11 inciso b. Su diseño es en forma de capas, por lo tanto cada capa presenta cierta independencia de la otra para que el desarrollador se concentre únicamente en su aplicación.

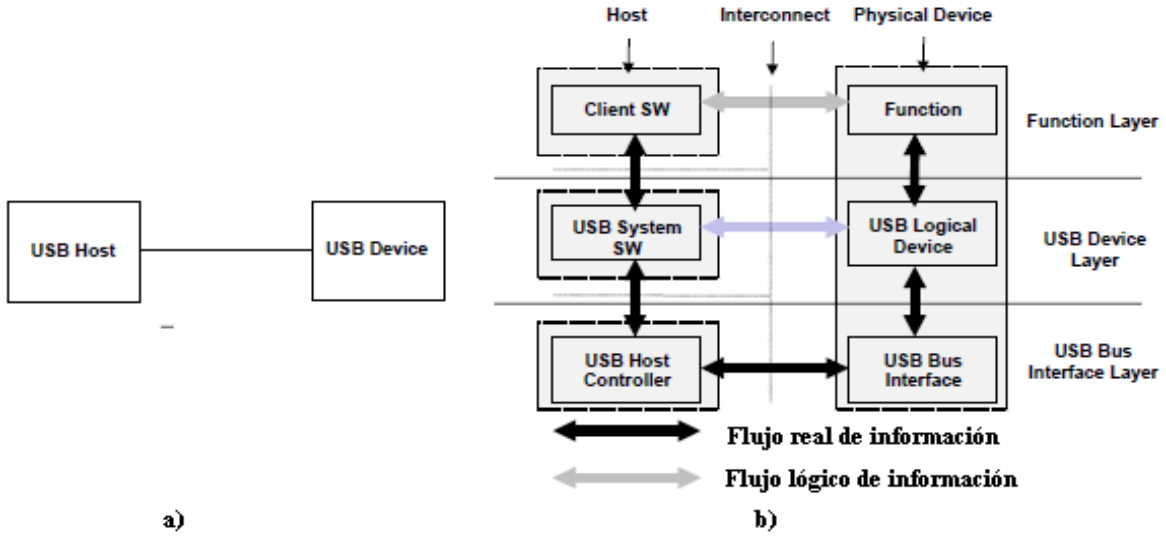


Figura F.11. Flujo de información.
a) Aparente, b) Real

Los dispositivos se encuentran conectados físicamente al *bus* mediante una topología tipo estrella, con puntos de conexión en dispositivos especiales llamados *hubs*. Esta conexión permite hasta siete niveles de interconexión, incluyendo al *root hub*. Por ejemplo, en la figura F.12, el *root hub* pertenece al primer nivel, los dispositivos y *hubs* conectados a él pertenecen a un segundo nivel y los dispositivos conectados a los *hubs* pertenecen a un tercer nivel.

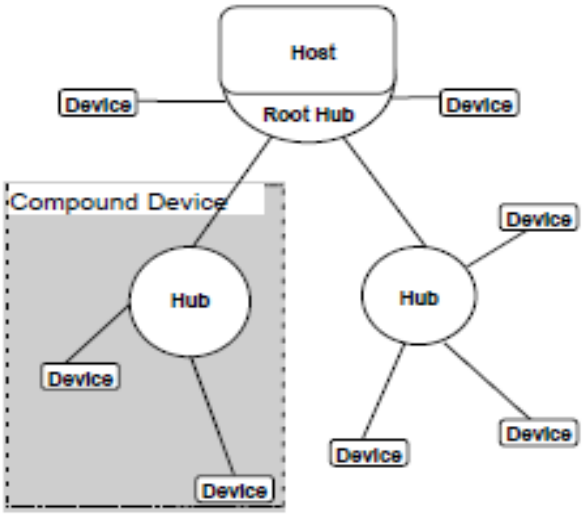


Figura F.12. Topología del bus USB

Estos niveles no son visibles, parecería que todos los dispositivos están conectados directamente al *host*, por lo mismo, es conveniente definir el concepto de tubería o PIPE, estos son puntos de conexión lógicos entre el cliente (*host*) y el dispositivo (*endpoint*). Así, se define una comunicación más simple como la presentada en la figura 2.19, inciso a.

Hay dos tipos de PIPE, los de mensaje y los de tipo *stream*. El tipo *stream* no contienen ninguna estructura, los datos que entran a la tubería son igualmente entregados al dispositivo y son unidireccionales. Los PIPE tipo de mensaje contienen una estructura definida, y aunque los datos no son procesados por el *bus* se necesita que la transmisión inicie con un *request* por parte del *host*, que luego siga una etapa de datos y se termine con el estado de la transmisión. Se puede enviar un *request* a la vez, esto hasta que ya haya sido procesado, el siguiente se podrá enviar en el orden de llegada (*First In First Out*). Este tipo de tubería es bidireccional.

Para dar más flexibilidad a los dispositivos que utiliza el *bus*, hay 4 tipos de transacciones, cada una con sus características y prioridad dentro del *bus*. La primera es la transferencia de *control*. Esta permite el acceso a distintas partes del dispositivo, su intención es soportar comunicaciones del tipo *configuración/comando/estado*, es una tubería de mensaje, por lo tanto, tiene una estructura definida, como se puede observar en la figura F.13. Todos los datos enviados en la misma transferencia deben estar en la misma dirección, la cual, se encuentra indicada en el PID. El tamaño de la etapa de datos debe ser de 8, 16, 32, y 64 *bytes*. Si ocurre un error durante alguna etapa, la transferencia será retransmitida.

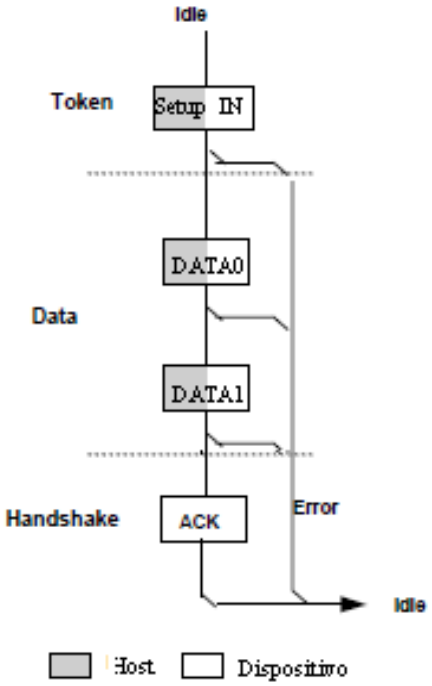


Figura F.13. Ejemplo de transferencia tipo Control

La segunda transferencia es de tipo *Bulk*. Está se encuentra diseñada para transmitir información relativamente grande en tiempos variables cuando el *bus* esté disponible. Es unidireccional, para poder transmitir en dos direcciones deberá utilizar dos tuberías. El tamaño de su etapa de datos debe ser 8, 16, 32 y 64 *bytes*. Al presentarse un error durante alguna etapa la transferencia será retransmitida.

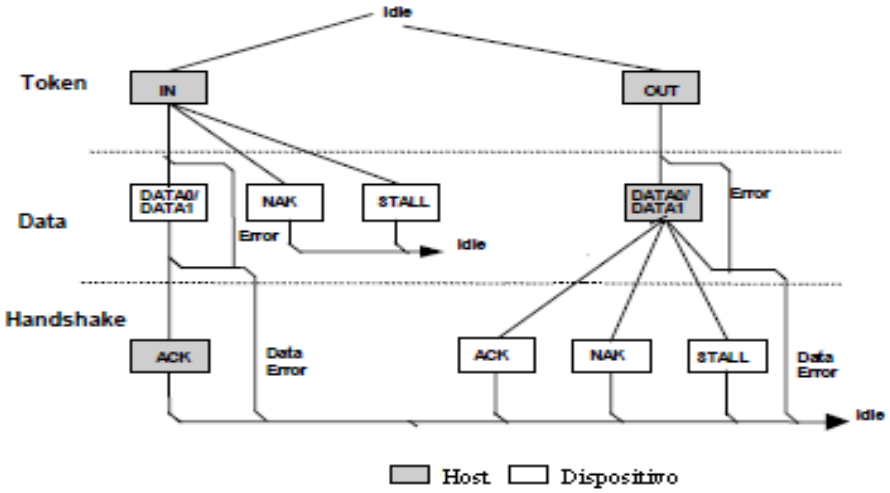


Figura F.14. Ejemplo de transferencia tipo Bulk

Transferencia tipo *Interrupt*. Este tipo de transacción es utilizada en dispositivos que necesitan transmitir información de manera no frecuente que debe ser verificada en periodos específicos. La transferencia tipo *interrupt* es unidireccional, cuenta con un máximo de 64 *bytes* para la etapa de datos. La información se retransmite hasta que se reciba un ACK como respuesta de aceptación.

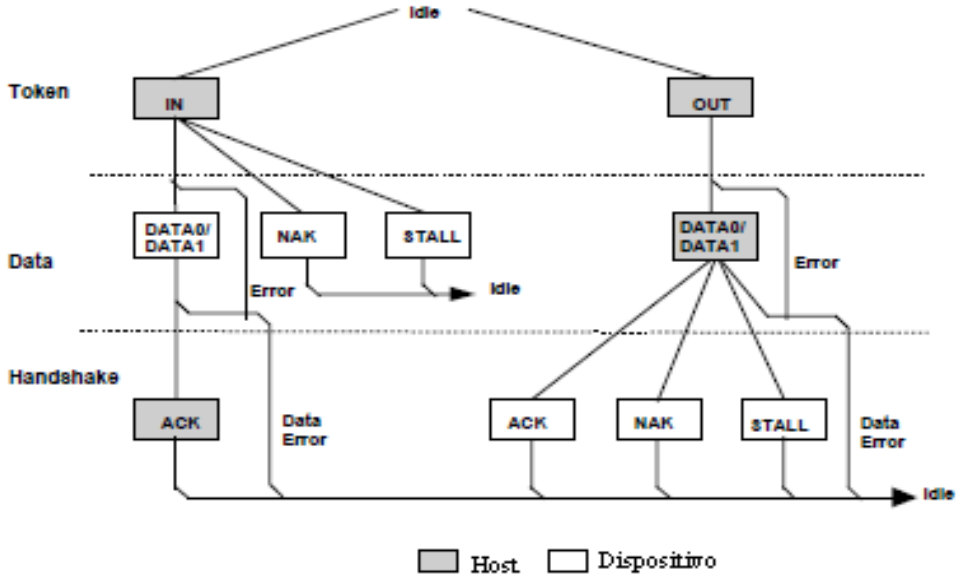
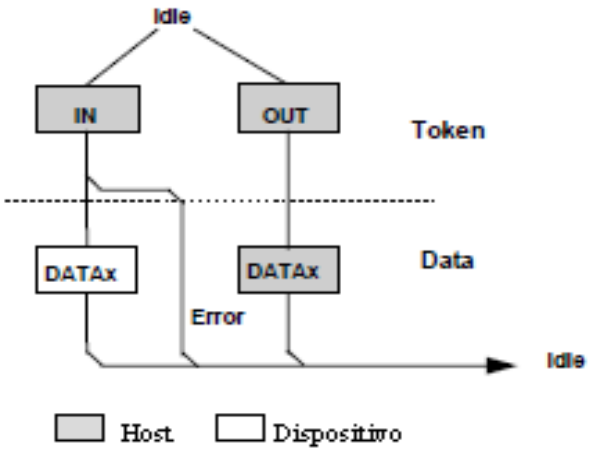


Figura F.15. Ejemplo de transferencia tipo Interrupt.

Por último, tipo de transferencia i*sochronous*. Este indica una tasa de transferencia constante con tolerancia de errores. Necesita un acceso al *bus* con una latencia constante. Tiene un máximo de 1023 *bytes* para la etapa de datos. La información no es retransmitida en caso de error.



F*igura F.16. Ejemplo de transferencia tipo Isochronous*

Para poder manejar los distintos tipos de transferencia, y hacerlo en el momento indicado, el *bus* encapsula las transacciones en *frames*, estos duran un milisegundo, cada transacción tiene un ancho de banda máximo reservado en cada *frame*. Las siguientes tablas muestran algunos ejemplos dependiendo de la carga de datos y tipo de transferencia.

Protocol Overhead (45 bytes)			9 SYNC *bytes*, 9 PID *bytes*, 6 *endpoints* +CRC *bytes*, 6 CRC *bytes*, 8 *setup data bytes* y 7 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	28	224 / 1260	4 / 14.9 / 84 %	16	1792 kbps
16	24	384 / 1080	4 / 25.6 / 72 %	36	3072 kbps
32	19	608 / 855	5 / 40.5 / 57 %	37	4864 kbps
64	13	832 / 585	7 / 55.5 / 39 %	83	6656 kbps

Tabla F.4 Limites de transferencia de control

** Ancho de banda porfFrame por transferencia/bytes útiles/bytes no útiles*

Protocol Overhead (13 *bytes*)			3 SYNC *bytes*, 3 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 3 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	71	568 / 923	1 / 37.8 / 61.5 %	9	4544 kbps
16	51	816 / 663	2 / 54.4 / 44.2 %	21	6528 kbps
32	33	1056 / 429	3 / 70.4 / 28.6 %	15	8448 kbps
64	19	1216 / 247	5 / 81 / 16.4 %	37	9728 kbps

Tabla F.5 Limites de transferencia Bulk
** Ancho de banda por frame por transferencia/Bytes útiles/Bytes no útiles*

Protocol Overhead (9 *bytes*)			2 SYNC *bytes*, 2 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 1 de retraso entre los paquetes(EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	Bytes restantes	Tasa de transferencia
128	10	1280 / 90	9 / 85.3 / 6 %	130	10240 kbps
256	5	1280 / 45	18 / 85.3 / 3 %	175	10240 kbps
512	2	1024 / 18	35 / 68.2 / 1.2 %	458	8192 kbps
1023	1	1023 / 9	69 / 68.2 / 16.4 %	468	8184 kbps

Tabla F.6 Limites de transferencia Isochronous
** Ancho de banda por frame por transferencia/Bytes útiles/Bytes no útiles*

Protocol Overhead (13 *bytes*)			3 SYNC *bytes*, 3 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 3 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por frame	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	71	568 / 923	1 / 37.8 / 61.5 %	9	4544 kbps
16	51	816 / 663	2 / 54.4 / 44.2 %	21	6528 kbps
32	33	1056 / 429	3 / 70.4 / 28.6 %	15	8448 kbps
64	19	1216 / 247	5 / 81 / 16.4 %	37	9728 kbps

Tabla F.7 Límites de transferencia Interrupción
** Ancho de banda por Frame por Transferencia / Bytes útiles / Bytes no útiles*

Configuración del dispositivo

Debido a que los dispositivos pueden ser conectados y desconectados dinámicamente del *bus*, este debe pasar por una serie de estados antes de poder ser utilizados:

- *Attached*. Este estado es justo cuando el dispositivo es conectado al *bus.*
- Energizado. Cuando el dispositivo es alimentado y sus líneas D+ y D- se encuentran en el estado J (estado *idle*). El *host* es capaz de detectar el dispositivo en este momento y puede reiniciar el ordenador (*reset*) para poner al dispositivo en el siguiente estado.
- *Default*. El dispositivo puede ser alcanzado mediante la dirección "cero", es capaz de responder a los *request* de su descriptor y configuración.
- Enumerado. El *host* le ha asignado una dirección única, por lo tanto, el dispositivo ya no responderá a la dirección por defecto.
- Configurado. El *host* ha seleccionado la configuración que utilizará del dispositivo y se lo ha indicado.

El dispositivo ahora se encuentra listo para ser utilizado por el *software.* Las configuraciones que tiene el dispositivo deben estar especificadas en el *device descriptor*.

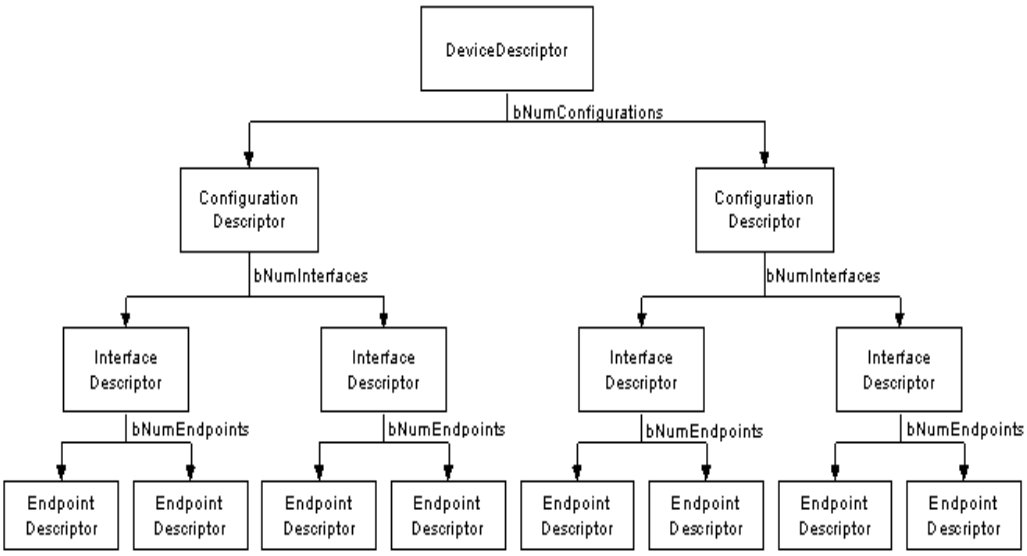


Figura F.17. Arreglo de descriptores en un dispositivo USB

Un dispositivo puede tener una o varias interfaces, estas interfaces con más de una configuración, cada configuración con dos o más *endpoints*, hasta 16. Estos archivos de configuración son abordados en la sección del microcontrolador maestro.

Bibliografía

1 Colectivo de autores, Tendencias pedagógicas contemporáneas, Universidad de La Habana. Centro de Estudios para el Perfeccionamiento de la Educación Superior (CEPES), 1996, La habana, Cuba, p. 131, Paira. Editora e Impresora Siglo 17.

2 Dolmans, Diana et all., Problem-based learning: future challenges for educational practice and research, Medical Education, Vol 39 No 7, 2005,USA, pp. 732-741, Blackwell publishing.

3 María Nolla Domenjó, El proceso cognitivo y el aprendizaje profesional, Educación Médica, Vol 9, No 1, 2006, España, pp. 11-16, Sociedad Española De Educación Médica.

4 American Heart Association, American Heart Association Guidelines for Cardiopulmonary Resuscitation and Emergency Cardiovascular Care, Circulation,Vol 112 No 24, 2005, USA, pp. 211, American Heart Association.

5 European Resuscitation Council, Resuscitation Council (ERC) Guidelines for Resuscitation, Resuscitation Vol 67 Suplemento 1, 2005,Unión europea, pp.189, Elsevier.

6 Pérez, Juan Carlos; Sergio Sáez, Estudio de un sistema operativo, futura.disca.upv.es, 2010, España, Universidad Politécnica de Valencia

7 Gerard H. Holzmann, Design and Validation of Computer Protocols, 1990,USA, p. 12, Prentice Hall

8 Heath, Steve , Embedded systems design 2da edición , 2003, UK, p.2, Newnes.

9 Philips Semiconductor, The I2C bus specification, version 2.1, 2000, USA, pp.46

10 USB Implementers Forum, Inc, USB Specification, Rev. 2.0, 2000, USA, pp.650