PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DEL SOFTWARE DEL SISTEMA EMBEBIDO DE LA BURBUJA ARTIFICIAL NEONATAL (Fase Experimental)

Tesis para optar el Título de:

INGENIERO INFORMÁTICO

Presentado por:

ANDRÉS BARRIOS MONTALVO

LIMA –PERÚ

2006

DESARROLLO DEL SOFTWARE DEL SISTEMA EMBEBIDO DE LA BURBUJA ARTIFICIAL NEONATAL (Fase Experimental)

Andrés Rolando Barrios Montalvo

RESUMEN

En el año 2004 la PUCP patentó un nuevo concepto de atención neonatal mediante un equipo denominado Burbuja Artificial Neonatal (BAN). La BAN es un equipo médico alternativo que cubre las deficiencias de las incubadoras tradicionales y brinda mejores condiciones para el buen desarrollo del recién nacido de alto riesgo. La BAN provee un ambiente estéril, con temperatura uniforme, aire temperado, humedecido y oxigenado.

El presente trabajo plantea el desarrollo del software de tiempo real del sistema embebido de la Burbuja Artificial Neonatal, en su Fase Experimental, capaz de controlar simultáneamente las múltiples variables ambientales como temperatura, porcentaje de humedad relativa, flujo y porcentaje de oxígeno; capaz de permitir la interacción con el usuario; y, capaz de monitorizar los parámetros ambientales.

El análisis de los requerimientos funcionales y no funcionales de la BAN concernientes a sus más importantes involucrados, el neonato de alto riesgo y el personal médico, permitió definir la arquitectura del sistema de software de la BAN, conformada por tres niveles o capaz. El nivel 1 es el núcleo del sistema, denominado *GHOST*. Realiza la planificación y ofrece a los niveles superiores un modelo de procesos secuenciales independientes proveyéndoles de mecanismos de comunicación y sincronización entre procesos. El nivel 2 implementa las tareas que manejan cada uno de los dispositivos de la BAN. Estos son los manejadores de los módulos: Adquisición de Datos de los sensores, Manejo de Dispositivos de Control, Ingreso de Datos, Visualización en LCD y Sonidos. Por último, el nivel 3 contiene los procesos de procesamiento de las señales sensadas, los proceso de control de los parámetros ambientales de la BAN, el proceso de encargado de monitorizar estos

parámetros ambientales, el proceso de supervisión de alarmas y el proceso de comunicación serial con la computadora personal.

Luego de la definición de la arquitectura, se realizó un diseño detallado de cada componente definido, especificando cada uno de sus procedimientos y estructuras de datos. Después de su implementación se procedió a realizar las pruebas del Sistema BAN. Se realizaron dos análisis de planificabilidad: el primero basado en la utilización de CPU y el segundo en el tiempo de respuesta del sistema. Además, se determinaron los valores de las propiedades importantes del núcleo *GHOST*, como lo son los tiempos de latencia, el *jitter* o porcentaje de variación y su porcentaje de utilización de los recursos, los cuales fueron mínimos. Se realizaron también las pruebas de los algoritmos de control, obteniéndose las curvas de la dinámica de la temperatura en la cápsula neonatal de la BAN y del flujo de aire en el circuito neumático.

Estos resultados son información importante y de utilidad para futuros trabajos en el área de desarrollo de software de sistemas embebidos para equipos médicos de soporte de vida.

He combatido el buen combate,

he corrido hasta la meta,

he mantenido la fe.

2 Timoteo 4, 7

I am always doing that which I can not do,

in order that I may learn how to do it.

Pablo Picasso

Al *Padre*, por su Amor, Bendiciones y Dones. A la *Madrecita* también.

A mis padres Rolando y Alicia, madre coraje,
porque el amor incondicional de los dos no conoce el final, infinito e ilimitado.

A mis dos hermanos, Luis Alfredo, el primero, tantas cosas, y
al Rey Leo, tanta vida, que son todo amor.

A toda mi familia, en especial a mi Abe.

A Bruno, amigo y maestro *sui generis.*

Al equipo que desarrolló la BAN Alfa: Bruno, Eduardo, Edwin, Jimmy,
Alejandro, Carlos, Javier, Romy, Raúl y José. Y a todos aquellos que colaboraron,
directa o indirectamente, poniendo su granito de arena.

A mi amigo Miguel, por estar en las buenas y
especialmente en las malas durante el desarrollo de esta tesis.

A mi asesor, Viktor Khlebnikov, por sus conocimientos y paciencia, спасибо.

A todos mis amigos y amigas de Informática, en especial a mis profesores,
que compartieron más que enseñanzas desde las épocas de mis primeros algoritmos.

A todos mis grandes amigos y amigas del GIDEMS, por su amistad,
sus enseñanzas, consejos, apoyo, coraje y entusiasmo. Porque son locos.

A todos aquellos que creyeron y creen en mí.

A todos aquellos que aún creen y creen con fuerza y Fe.

Y a ti, que hoy nos encontramos en el espacio del tiempo
y que haces de esta tesis letra viva.

El autor.

DESARROLLO DEL SOFTWARE DEL SISTEMA EMBEBIDO DE LA BURBUJA ARTIFICIAL NEONATAL (Fase Experimental)

DESARROLLO DEL SOFTWARE DEL SISTEMA EMBEBIDO DE LA BURBUJA ARTIFICIAL NEONATAL (Fase Experimental)

1 INTRODUCCIÓN

En el año 2004 el Grupo de Investigación y Desarrollo de Equipos Médicos y Sistemas (GIDEMS) de la PUCP patentó un nuevo concepto de atención neonatal mediante un equipo denominado Burbuja Artificial Neonatal (BAN); con número de patente EP1380276 en la Oficina de Patentes Europea y número US20040133064 en la Oficina de Patentes y Marcas de los EEUU. La BAN es un equipo médico alternativo a las incubadoras tradicionales capaz de cubrir, debido a su innovación, las deficiencias que presentan estas últimas, tales como: una mala distribución del calor en el habitáculo del bebé, una ventilación inadecuada, una administración inadecuada de oxígeno, la presencia de alto ruido, un ambiente fácilmente contaminable, una deficiente administración de humedad relativa. Así también, la BAN es capaz de brindar mejores condiciones para el buen desarrollo del recién nacido de alto riesgo; proveyendo un ambiente estéril, con temperatura uniforme, aire temperado, humedecido y oxigenado, hasta que el bebé esté listo físicamente para un ambiente normal.

Se quiere desarrollar un equipo basado en la patente con la finalidad de probar y validar el concepto patentado, además de ponerle valor agregado, de forma que se pueda apreciar la verdadera magnitud de las bondades de la patente BAN por los médicos, los empresarios, la comunidad académica y toda la gente involucrada en el desarrollo de la patente.

Por estos motivos se comenzó el Proyecto de Fase Experimental BAN en el que se plantea el desarrollo del primer prototipo de la BAN basado en la patente, el cual es la primera materialización de la idea y es experimental, por lo que está abierto a posibles

perfeccionamientos. Este prototipo requiere un trabajo multidisciplinario que incluye las áreas de diseño industrial, ingeniería mecánica, ingeniería electrónica e ingeniería informática.

Una parte importante de este prototipo es su sistema embebido, compuesto por hardware y software, por lo que es necesario desarrollar el software de este sistema embebido. El software debe ser capaz de controlar simultáneamente las múltiples variables ambientales relevantes como temperatura, porcentaje de humedad relativa, flujo y porcentaje de oxígeno; debe permitir la interacción con el usuario para la configuración del sistema; y, debe monitorizar los parámetros ambientales a fin de mostrarle la información importante al usuario.

El presente trabajo plantea el desarrollo del software de tiempo real de este sistema embebido de forma que provea una solución aceptable al problema propuesto; que provea un núcleo que administre los recursos compartidos y planifique, sincronice y comunique las tareas que han de ejecutarse simultáneamente. El software de este sistema embebido deberá tener la capacidad de poder integrar y coordinar el hardware, que tiene 4 microcontroladores que deben interactuar para realizar las funciones que la BAN le presenta al usuario. Además, deberá tomar en consideración las restricciones puestas por el hardware, como son: el tamaño de la memoria, la capacidad de procesamiento, la velocidad de los microcontroladores.

1.1 Estado del arte

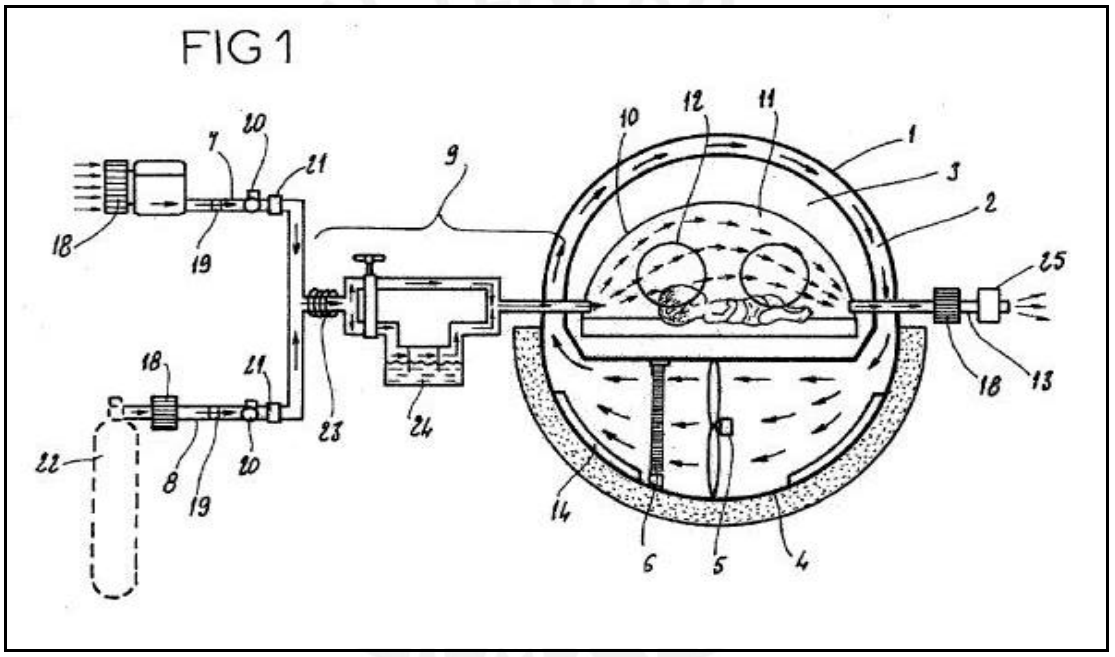
La Burbuja Artificial Neonatal [C&A04] es un equipo médico para mejorar la atención intensiva de neonatos de alto riesgo, que brinda un ambiente estéril con temperatura uniforme y aire mezclado con oxígeno; que previamente fue filtrado, temperado y humedecido. La invención consta de dos circuitos de flujo de gases:

Circuito Cerrado de Aire Temperado: encargado de conservar y mantener uniforme la temperatura en el ambiente artificial intermedio mediante un calefactor y un ventilador que generan un flujo de aire temperado que se usa como medio de propagación de calor. Este circuito no está en contacto con el neonato, atributo que permite instalar filtros acústicos para reducir el ruido.

Circuito Ventilatorio Continuo: conjunto de dispositivos neumáticos conectados consecutivamente para ventilar al neonato con un flujo continuo de aire filtrado, oxigenado, temperado y humedecido. La cantidad de este gas está regulada según los requerimientos de cada neonato. Además, permite utilizar una menor cantidad de oxígeno y mayor tiempo los filtros bacterianos.

El equipo utiliza una cápsula para alojar al neonato, la cual es desechable con la finalidad de evitar la contaminación entre pacientes.

En la Figura 1.1 se presenta el esquema de la patente BAN.



1) Domo 2) Espacio intra-domo 3) Ambiente artificial intermedio 4) Base térmica 5) Ventilador 6) Calefactor 7) Línea de aire 8) Línea de oxígeno 9) Línea colectora de gases 10) Cápsula neonatal 11) Ambiente artificial interno 12) Puertas circulares 13) Línea de salida de la mezcla gaseosa 14) Filtro acústico 18) Filtro bacteriano 19) Válvula Check 20) Válvula proporcional 21) Sensor de flujo 22) Fuente de oxígeno 23) Calefactor 24) Humidificador 25) Sensores de flujo, temperatura, humedad y oxígeno. Fuente: Patente Nº EP1380276 [C&A04].

Figura 1.1. Esquema de la patente BAN.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar el software del sistema embebido de la Burbuja Artificial Neonatal.

1.2.2 Objetivos específicos

- El software debe de poseer la capacidad de integrar y coordinar el hardware de la BAN, el cual tiene 4 microcontroladores que deben interactuar para cumplir con las funcionalidades que se le presentan al usuario.
- El software debe de administrar los recursos de hardware y debe de manejar los dispositivos del sistema embebido.
- El software debe de planificar las tareas que han de ejecutarse simultáneamente.
- El software debe de proveer los mecanismos de comunicación entre los procesos.

1.3 Metodología

Se plantea la siguiente metodología que se desarrollará a lo largo del presente trabajo:

- Análisis y especificación de las funciones del Sistema del Proyecto de Fase Experimental BAN.
- Definición de una arquitectura para el software del Sistema BAN.
- Diseño del software del Sistema BAN.
- Implementación del software del Sistema BAN en el microcontrolador ATmega 128.
- Pruebas del Sistema BAN.

El proceso comienza con la definición de los requerimientos de la BAN concernientes a sus más importantes involucrados, el neonato de alto riesgo y el personal médico. A su vez, se

necesita conocer las especificaciones importantes del hardware y del sistema en general, que llevarán a consideraciones importantes que habrá que tener en cuenta durante todo el ciclo de desarrollo del sistema de tiempo real. Se culmina con la definición de los requerimientos funcionales y no funcionales del sistema.

Luego, se procede a definir la arquitectura del sistema de software de la BAN tomando en cuenta a todo momento las características que debe de tener y la infraestructura donde se va a ejecutar. Este proceso involucra dividir el trabajo que se tiene que realizar en componentes o módulos responsables de una porción del problema.

Una vez que las actividades del diseño arquitectural se hayan completado, se comienza con el diseño detallado de cada componente definido, especificando cada uno de sus procedimientos y estructuras de datos. En este proceso también se definen las características del núcleo sobre el cual se construirá el sistema y los mecanismos o servicios que debe proveer.

Luego, se procede con la implementación del sistema comenzando por los niveles más cercanos al hardware, como lo es el núcleo. El proceso de integración de los módulos se realiza desde los componentes de más bajo nivel hasta los componentes de nivel superior en el sistema.

Durante el proceso, cada componente debe ser analizado usando herramientas que midan sus características, como lo es su tiempo de ejecución en el peor de los casos. Se realizarán dos análisis de planificabilidad: el primero basado en la utilización de CPU y el segundo en el tiempo de respuesta del sistema.

Finalmente, el sistema se somete a un periodo de pruebas funcionales, con la finalidad de validar el sistema respecto a los requerimientos definidos inicialmente.

2 MARCO CONTEXTUAL

2.1 Fase experimental

Es la primera materialización de la idea, donde se prueban y examinan sus virtudes y propiedades. Debido a ser experimental, está abierta a posibles perfeccionamientos [RAE05].

En lo que respecta a este desarrollo, al prototipo se le ha denominado versión Alfa, tomando como referencia lo que se denomina versión Alfa en software, es decir, la primera entrega de un producto que es usualmente probado sólo por los desarrolladores [Wor05] (Ver Anexo F).

2.2 Neonato

Un neonato es un bebé de 4 o menos semanas. El período del neonato está definido y es un tiempo importante porque representa un período corto de la vida cuando los cambios son muy rápidos y cuando se pueden presentar muchas situaciones críticas. Los neonatos se clasifican, según su peso, en las siguientes categorías:

- Peso normal al nacer (más de 2500 g)
- Bajo peso al nacer (BPN, menos de 2500 g)
- Muy bajo peso al nacer (MBPN, menos de 1500 g)
- Extremadamente bajo peso al nacer (EBPN, menos de 1000 g)

El peso y la edad gestacional con que nace el neonato pueden llegar a tener consecuencias mortales. Debemos considerar que la mortalidad del neonato es directamente proporcional al bajo peso y la menor edad gestacional [San05].

2.3 Sistema de tiempo real

Todo sistema en el que es significativo el tiempo en el cual la salida es producida. Esto es usualmente porque la entrada corresponde a algún movimiento en el mundo físico, y la salida está relacionada con ese mismo movimiento. La diferencia entre el tiempo en que ocurre la entrada y el tiempo en que ocurre la salida debe de ser lo suficientemente pequeña para un aceptable cumplimiento de los plazos [Oxf96].

Los sistemas de tiempo real se dividen en dos tipos. Los sistemas de tiempo real duro o estricto (*hard*), aquellos en los que es absolutamente imperativo que las respuestas ocurran dentro de los plazos establecidos. Los sistemas de tiempo real blando o flexible (*soft*), donde los tiempos son importantes pero aún si se perdiera algún plazo el sistema seguiría funcionando correctamente. El uso del término blando no implica un único tipo de requerimiento sino que incorpora diferentes propiedades. Por ejemplo: los plazos pueden ser perdidos ocasionalmente, el servicio puede ocasionalmente ser proporcionado con retraso [B&W96]. El presente trabajo presenta un sistema de tiempo real duro.

2.4 Sistema embebido

Una de las áreas de mayor expansión en el uso de las computadoras, es la que involucra aplicaciones en las cuales su principal función *no* es la de procesar información. Una máquina de lavar controlada por un microcontrolador es un ejemplo de este tipo de sistemas. En este caso, la función principal es lavar la ropa; sin embargo, dependiendo del tipo de ropa a ser lavada, diferentes “programas de lavado” deben ser ejecutados. Este tipo de aplicaciones de computadora son generalmente llamadas embebidas [B&W96].

Un sistema embebido es un sistema de propósito específico, en el cual la computadora está completamente encapsulada por el dispositivo que controla. Un sistema embebido ejecuta tareas predefinidas, usualmente con requerimientos muy específicos, en contraste con un sistema de propósito general [Wik05].

Un sistema embebido es específicamente diseñado para proveer un conjunto dado y restringido de funcionalidades; y, presenta hardware y software como una unidad. Es importante notar que un sistema embebido no está definido por la tecnología usada [Beu03].

2.5 Sistema operativo

Los sistemas operativos pueden verse desde dos perspectivas: administradores de recursos y máquinas extendidas. Desde la perspectiva de administrador de recursos, la tarea del sistema operativo es administrar con eficiencia las diferentes partes del sistema. Desde la perspectiva de máquina extendida, la tarea del sistema operativo es proporcionar a los usuarios una máquina virtual que sea más cómoda de usar que la máquina real [T&W98].

2.6 Núcleo (kernel)

El núcleo (o *kernel*) es el responsable del manejo de los procesos o tareas (p.e. el manejo del tiempo de CPU) y la comunicación entre estas. El servicio fundamental que provee el *kernel* es el cambio de contextos. El uso de un *kernel* de tiempo real generalmente simplifica el diseño de un sistema al permitir que la aplicación sea dividida en múltiples tareas que el *kernel* maneja [Lab02].

2.7 Teoría de control

En ingeniería y matemáticas, la teoría de control se encarga de lidiar con el comportamiento de los sistemas dinámicos. La salida deseada de un sistema es llamada la referencia. Cuando una o más variables de salida de un sistema necesitan seguir una cierta referencia en el tiempo, un controlador manipula las entradas del sistema para obtener el efecto desado en las salidas del sistema [Wik05].

Entre los sistemas de control se encuentran los sistemas de control realimentado, los cuales mantienen una relación determinada entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control [Oga03].

2.8 Interfaz de usuario

Dentro del diseño del sistema los dispositivos de consola forman parte de la interacción básica entre el sistema y el usuario. La consola usualmente consiste en algunos tipos de

visualizadores (*displays*) y teclado. Estos dispositivos pueden variar desde botones hasta terminales gráficos. El diseño de la interfaz de usuario está basado en los mismos principios en que los que están basadas las otras partes de un sistema de tiempo real, el usuario debe de ser capaz de responder a situaciones dentro de restricciones de tiempo establecidas. Adicionalmente, toda la actividad de la interfaz de usuario debe de tomar lugar sin afectar las tareas fundamentales como los son las funciones de control y adquisición de datos [A&T89].

2.9 Monitorizar

Observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías [RAE05].

3 ESPECIFICACIÓN DEL SISTEMA DEL PROYECTO DE FASE EXPERIMENTAL BAN

Introducción

En la presente tesis se utilizó la metodología clásica de desarrollo que comprende las etapas típicas como lo son: la especificación de los requerimientos, el diseño arquitectural, el diseño detallado, la implementación y las pruebas [B&W96]. Donde se siguieron algunas de las recomendaciones de la norma internacional IEC601-1-4 [IEC96], para sistemas médicos eléctricos programables, en lo que se refiere a las etapas que comprende la metodología.

Particularmente en la etapa de especificación de requerimientos se usaron las especificaciones de casos de uso como lo hace la metodología *Mantra* usada por EventHelix Inc. (Diseño de software de tiempo real embebido) [Eve05].

A pesar de que el enfoque es descendente, esta se construye sobre un entendimiento de lo que es viable hacer a bajo nivel [B&W96], por esto se necesitó conocer las especificaciones importantes del hardware y del sistema en general. Esto permitió poder tomar familiaridad con el sistema total e influyó en una especificación más detallada de los requerimientos del sistema.

3.1 Descripción del hardware de la BAN

El sistema electrónico de la BAN está conformado por 6 módulos importantes desde el punto de vista del software del sistema:

- **El Módulo de Procesamiento y Control (Microcontrolador ATmega 128)**, encargado de ejecutar los procesos de lectura de datos, algoritmos de control, movimiento de actuadores, configuración del sistema, etc. Aquí se encuentra el corazón del sistema y se encarga también de la coordinación entre todos los demás módulos. El ATmega 128 es un microcontrolador de 8 bits de arquitectura RISC.

Posee 32 registros de 8 bits, 1 registro de estado, 1 registro de puntero de pila, registros de configuración de los periféricos y registros X(r27:r26), Y, Z de 16 bits de propósito general, especialmente utilizados para acceso indirecto a la memoria (punteros). Cuenta con memorias con espacios de direcciones separados: 128 KB de memoria Flash, 4 KB de memoria EEPROM y 4 KB de memoria SRAM; 4 interrupciones internas, 4 interrupciones externas; 2 temporizadores de 8 bits, 2 temporizadores de 16 bits, 1 contador en tiempo real y 1 temporizador *watchdog* programable. Cuenta también con 6 puertos de 8 bits y 1 de 5 bits configurables como E/S [Atm04].

- **El Módulo de Adquisición de Datos**, encargado de medir todas las variables en la BAN, acondicionar las señales y digitalizarlas. Se conecta con el Módulo de Procesamiento y Control a través de un conversor analógico digital.

- **El Módulo de Manejo de Dispositivos de Control**, encargado de controlar los actuadores de la BAN. Se conecta con el Módulo de Procesamiento y Control a través de un PPI (*Programmable Peripherical Interface*), por el cual pasan las señales de PWM (*Pulse Width Modulation*) y las palabras de control para los estranguladores y para un conversor digital analógico.

- **El Módulo de Ingreso de Datos (Microcontrolador PIC)**, encargado de tomar las señales del teclado tanto de los botones como de la perilla (*encoder*). Las procesa y las envía al Módulo de Procesamiento y Control para que se entere éste último de los nuevos valores de las variables ambientales que el usuario programa, con la finalidad de configurar el sistema. También tiene la función de activar el sonido para las alarmas.

- **El Módulo de Visualización en LCD (Microcontrolador Scenix)**, encargado de mostrar en la pantalla LCD los valores sensados y los valores programados (o seleccionados) por el usuario. Se conecta con el módulo de procesamiento y control a través de un puerto especial al cual este último debe de acceder para poder dibujar en la memoria de video del primero; y por último,

- **El Módulo de Sonidos (Microcontrolador PIC)**, encargado de la reproducción de sonidos intrauterinos y música clásica dentro de la BAN. Se conecta a través de un puerto con el módulo de procesamiento y control.

Estos módulos están esquematizados en el Diagrama de Bloques de la BAN (Figura 3.1).

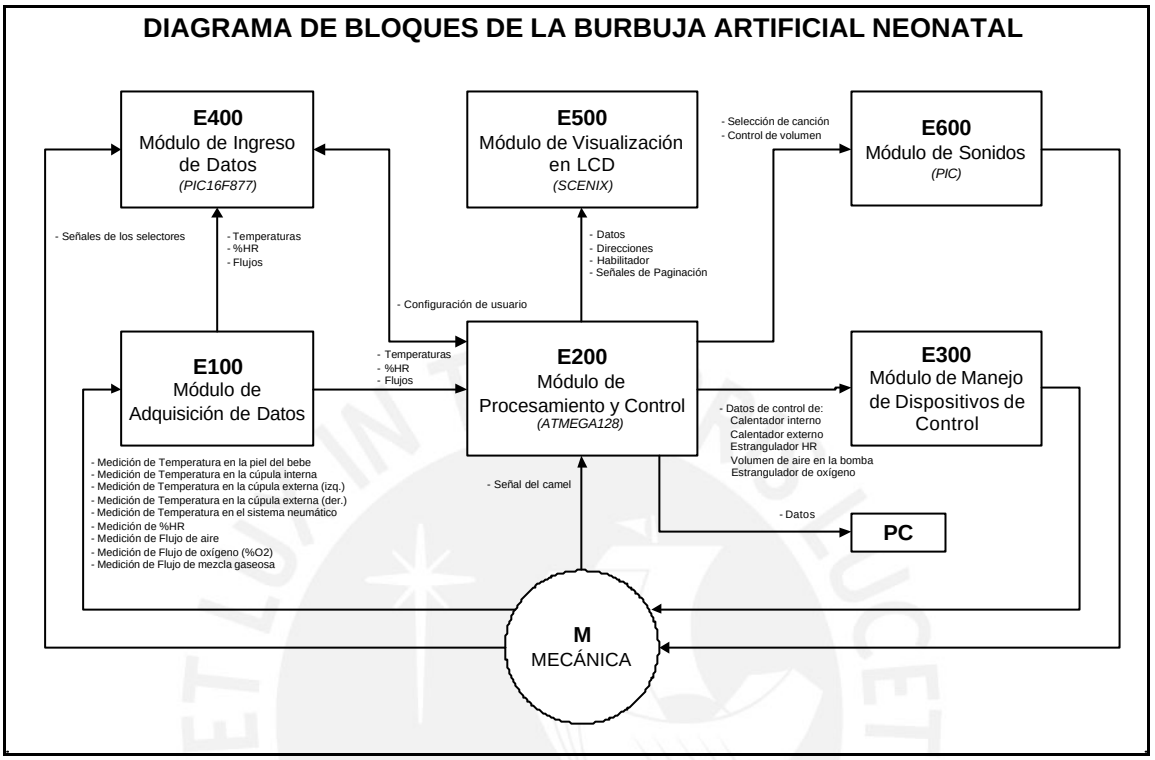


Figura 3.1. Diagrama de bloques de la BAN.

El software de este sistema embebido deberá tener la capacidad de poder integrar y coordinar todo el hardware, que tiene 4 microcontroladores que deben interactuar para realizar las funciones que la BAN le presenta al usuario. El software se encontrará en el Módulo de Procesamiento y Control (ATmega 128).

Además, deberá tomar en consideración las restricciones puestas por el hardware, tales como: el tamaño de la memoria, la capacidad de procesamiento, la velocidad de los microcontroladores.

3.2 Descripción de las funciones de la BAN

En la presente sección se presenta el análisis de todas las funcionalidades que debe de proveer la BAN para cumplir los requerimientos planteados. En las Figuras 3.2, 3.3 y 3.4 se muestran los diagramas de casos de uso. Luego, se muestran unas breves descripciones de los casos de uso. Las especificaciones completas se detallan en el Anexo I.

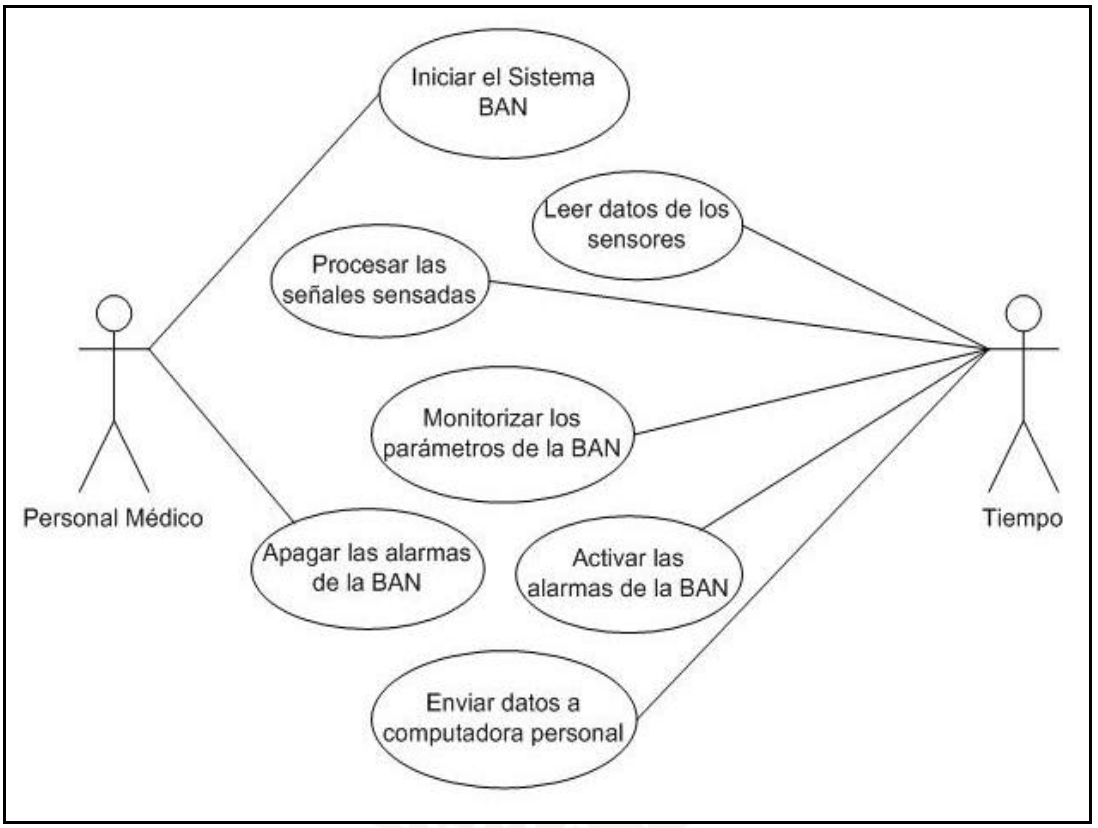


Figura 3.2. Diagrama de Casos de Uso. General.

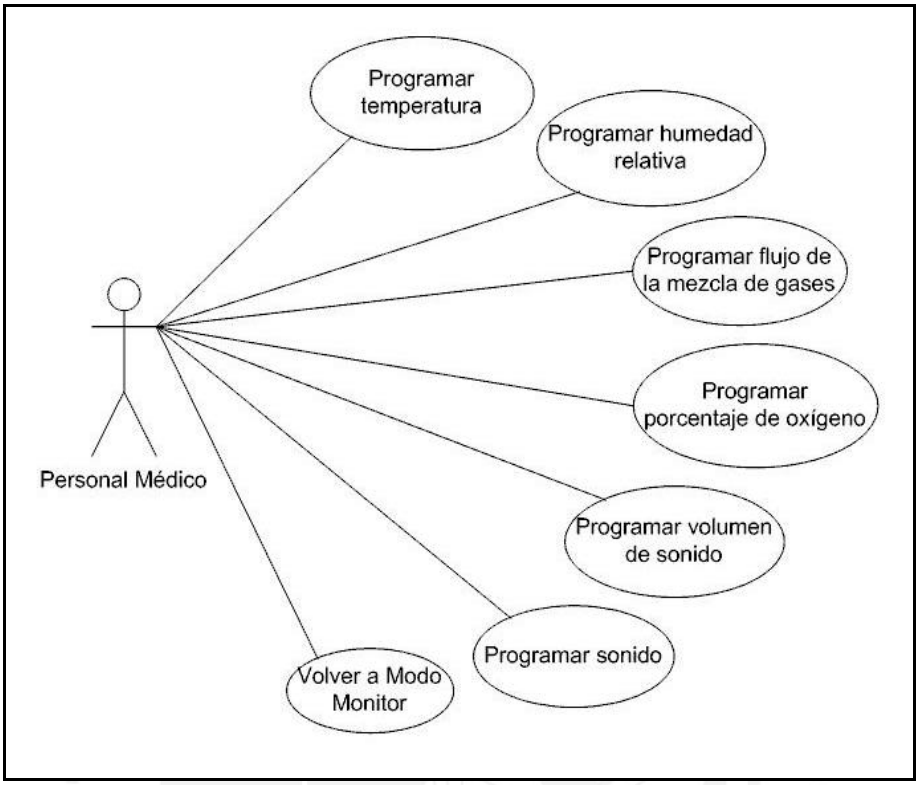


Figura 3.3. Diagrama de Casos de Uso. Programación.

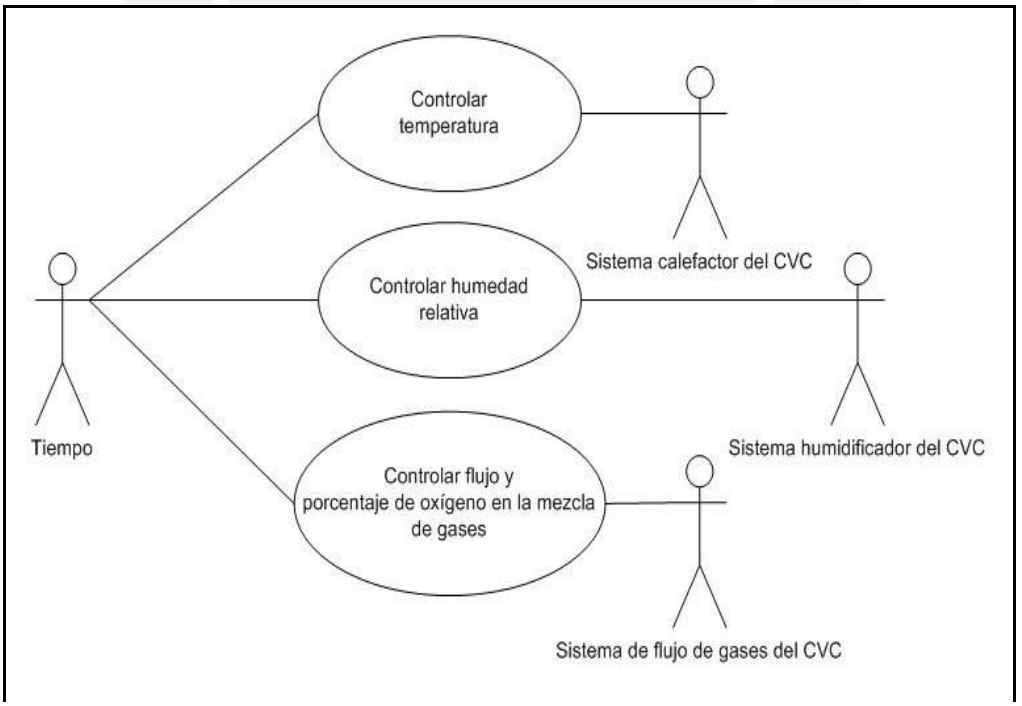


Figura 3.4. Diagrama de Casos de Uso. Control.

3.2.1 Iniciar el Sistema BAN

3.2.1.1 Descripción General

Este caso de uso permite al personal médico inicializar el Sistema BAN para su uso.

3.2.1.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico enciende la BAN.
2. El Sistema BAN deshabilita el Módulo de Ingreso de Datos (teclado).
3. Se inicializa la memoria SRAM.
4. Se configura el ATmega 128.
5. Se inicializa las pantallas de presentación.
6. Se configura todos los dispositivos de HW.
7. Se inicializa el Núcleo y los contextos de los procesos de la BAN.
8. Se inicializan los parámetros de control usando la configuración estándar guardada. Esta consiste en la programación de: temperatura en el ambiente de la cúpula interna a 34 ºC, humedad relativa a 60 %, porcentaje de oxígeno a 21 % y flujo de la mezcla de aire y oxígeno a 2 L/min.
9. Se habilita el Módulo de Ingreso de Datos (teclado).

Flujos Alternativos

No se presentan flujos alternativos.

3.2.2 Leer datos de los sensores

Este caso de uso permite leer los datos de los 9 sensores de la BAN a través de los conversores analógico-digital. Se adquieren los datos de las señales de temperatura: en la cúpula interna, en la cúpula externa (dos sensores: izquierda y derecha), en el sistema neumático CVC (calefactor interno) y en al piel del bebé. Se adquieren los datos de la señal de humedad relativa. Se adquieren los datos de la señal de flujo: en la vía de oxígeno, en la vía de aire y en la vía de la mezcla.

3.2.3 Procesar las señales sensadas

Este caso de uso permite convertir los datos adquiridos a sus unidades físicas respectivas: grados Celcius (ºC), porcentaje de humedad relativa (%), y litros por minuto (Lpm). Además, permite filtrar digitalmente los datos sensados de flujo.

3.2.4 Monitorizar los parámetros de la BAN

Este caso de uso permite mostrar en la pantalla LCD al personal médico, tanto los valores que se programan como los valores medidos de los parámetros de temperatura en la cúpula interna, temperatura en la piel del bebé, porcentaje de humedad relativa, flujo de la mezcla de gases, porcentaje de oxígeno en la mezcla; además, permite mostrar si alguna alarma se ha activado.

3.2.5 Activar las alarmas de la BAN

Este caso de uso se encarga del manejo de las alarmas en la BAN y de mostrarlas en la pantalla LCD. Este sistema de alarmas es desarrollado en conjunto entre el Módulo de Ingreso de Datos (sonidos de alarmas), el Módulo de Procesamiento y Control y el Módulo de Visualización. (La clasificación de las alarmas se muestra en el Anexo C).

3.2.6 Apagar las alarmas de la BAN

Este caso de uso permite al personal médico apagar el sonido de las alarmas de la BAN por un intervalo de 5 minutos. Sin embargo, en todo momento, a pesar de que se desactive el sonido de las alarmas, estas se deberán seguir mostrando en pantalla LCD.

3.2.7 Programar temperatura

Este caso de uso permite al personal médico programar el valor de temperatura en la cúpula interna de la BAN, en un rango de 25°C a 37°C. El valor modificado se visualiza en la región de los parámetros programados de la pantalla LCD. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.8 Programar humedad relativa

Este caso de uso permite al personal médico programar el valor de porcentaje de humedad relativa en el aire de la cúpula interna de la BAN, en un rango de 40% a 90%. El valor modificado se visualiza en la región de los parámetros programados de la pantalla LCD. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.9 Programar flujo de la mezcla de gases

Este caso de uso permite al personal médico programar el valor de flujo de la mezcla de gases dentro de la BAN, en un rango de 3 Lpm a 10 Lpm. El valor modificado se visualiza en la región de los parámetros programados de la pantalla LCD. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.10 Programar porcentaje de oxígeno

Este caso de uso permite al personal médico programar el valor de porcentaje de oxígeno dentro de la BAN, en un rango de 21% a 100%. El valor modificado se visualiza en la región de los parámetros programados de la pantalla LCD. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.11 Programar sonido

Este caso de uso permite al personal médico programar el sonido dentro de la BAN. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.12 Programar volumen de sonido

Este caso de uso permite al personal médico programar el volumen de sonido dentro de la BAN. Si después de un tiempo determinado no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

3.2.13 Volver a Modo Monitor

Este caso de uso permite al personal médico volver a la pantalla de Modo Monitor de la BAN; en este modo el teclado está bloqueado.

3.2.14 Controlar temperatura

Este caso de uso permite el control de la temperatura en la BAN según el valor programado por el personal médico, en un rango de 25 ºC a 37 ºC. Para el control de temperatura se utilizan dos ondas PWM (Modulación por Ancho de Pulso), una por calefactor. El rendimiento de cada onda se calcula sobre la base de la diferencia entre el valor sensado de temperatura y el valor programado.

3.2.15 Controlar humedad relativa

Este caso de uso permite el control del porcentaje de humedad relativa en la BAN según el valor programado por el personal médico, en un rango de 40% a 90% de humedad relativa. Si el valor sensado de humedad relativa difiere del programado, se calcula la distancia que debe de moverse el actuador. Luego se convierte la distancia calculada a un intervalo de tiempo y se mueve el actuador.

3.2.16 Controlar flujo y porcentaje de oxígeno en la mezcla de gases

Este caso de uso permite el control de la mezcla de aire y oxígeno en la BAN según los valores de los parámetros programados por el personal médico, en un rango de 3 Lpm a 10 Lpm para el flujo de la mezcla y en un rango de 21% a 100% para el porcentaje de oxígeno. Para lograr estos objetivos se controlan los flujos en las vías de aire y oxígeno.

En el caso del flujo de aire, si el valor sensado difiere del programado, se calcula la velocidad con la que debe de moverse la bomba de aire para que dé el flujo de aire requerido y se envía la palabra de control.

En el caso del flujo de oxígeno, si el valor sensado difiere del programado, se calcula la distancia que debe de moverse el actuador. Luego se convierte la distancia calculada a un intervalo de tiempo y se mueve el actuador.

3.2.17 Enviar datos a computadora personal

Este caso de uso le brinda al personal médico la posibilidad de poder conectar la BAN a una PC, a fin de poder llevar un registro en el tiempo del estado de los parámetros importantes del Sistema BAN; así como de las programaciones efectuadas de dichos parámetros. Toda la información se envía por una interfaz serial mediante el protocolo RS232.

3.3 Análisis de los requerimientos no funcionales del sistema

En esta sección se muestran a detalle las restricciones no funcionales que tiene la BAN, como son los tiempos asociados a las tareas que se deben de ejecutar o los grupos de prioridades.

3.3.1 Sobre la lectura de datos

El Sistema BAN sensa 4 tipos de señales diferentes mediante el uso de 9 sensores. En la tabla 3.1 se muestran datos relevantes a estos. Las tareas de lectura de datos poseen una prioridad alta en el sistema y su diseño.

Señal	Tiempo de respuesta	Periodo de muestreo
Sensores de temperatura	10 s	4 ms
Sensor de humedad relativa	50 s	4 ms
Sensor de flujo en la vía de aire	60 ms	4 ms
Sensor de flujo en la vía de oxígeno	60 ms	4 ms
Sensor de flujo en la vía de la mezcla	50 ms	4 ms

Tabla 3.1. Tiempos asociados a los sensores de la BAN.

Para la adquisición de la temperatura de la piel del bebé se utiliza el conversor analógico digital ADC Max187, el cual posee un canal y es de tipo serial. Para el resto de variables se utiliza el conversor analógico digital ADC Max186, el cual posee 8 canales y es también del tipo serial.

Ambos dispositivos presentan las siguientes características de interés: 12 bits de resolución, +/- ½ LSB de error, tiempo de conversión máximo 10 µs, tiempo de conversión mínimo 5,5 µs, tiempo de adquisición máximo 1,5 µs.

3.3.2 Sobre el control de los parámetros ambientales

Las tareas de control poseen alta prioridad dentro de todo el sistema, son tareas que se deben de ejecutar en tiempo real debido a que de ellas depende el comportamiento de todo el sistema. En la tabla 3.2 se muestran las frecuencias mínimas a las cuales se deben de ejecutar los controles.

Parámetro	Periodo
Control de temperatura	1 s
Control de humedad relativa	2 s
Control de flujo y porcentaje de oxígeno en la mezcla de gases	5 s

Tabla 3.2. Periodos de los controles de la BAN.

3.3.3 Sobre los dispositivos periféricos (actuadores)

Las tareas encargadas del manejo de los dispositivos periféricos relacionados a las tareas de control (actuadores) son también tareas de alta prioridad ya que regulan las salidas del sistema. En la tabla 3.3 se muestran los tiempos asociados a los actuadores de la BAN.

Actuador	Respuesta
Calefactor Interno	-
Calefactor Externo	-
Estrangulador de humedad relativa	771 ms/mm
Estrangulador de la vía de oxígeno	771 ms/mm
Volumen de aire en la bomba de aire	-

Tabla 3.3. Tiempos asociados a los actuadores de la BAN.

3.3.4 Sobre la tarea de monitorizar los parámetros de la BAN

Esta tarea es la encargada de interactuar con el Módulo de Visualización en LCD. Muestra en la pantalla LCD los valores que se programan como los valores medidos de los parámetros de temperatura en la cúpula interna, temperatura en la piel del bebé, porcentaje de humedad relativa, flujo de la mezcla de gases y porcentaje de oxígeno en la mezcla. Esta tarea tiene una prioridad normal dentro del sistema.

3.3.5 Sobre la supervisión de las alarmas

Al igual que la tarea anterior la revisión de alarmas tiene una prioridad normal dentro de todo el sistema.

3.3.6 Sobre la programación de parámetros de la BAN

Las tareas de programación de los parámetros ambientales (temperatura, humedad relativa, flujo de la mezcla de gases y porcentaje de oxígeno) tienen una prioridad normal dentro del sistema. Para todos los casos se debería volver a Modo Monitor después de transcurridos 10 segundos, en caso de que esto no se haya seleccionado explícitamente por el usuario.

3.3.7 Sobre la programación de sonidos

La tarea de programación de sonidos tiene una prioridad baja debido a que no es una tarea de soporte de vida para el recién nacido.

3.3.8 Sobre el envío de datos a la computadora personal

Esta es una tarea de baja prioridad encargada de enviar información sobre el estado de la BAN a una computadora personal.

A continuación, en la tabla 3.4 se presenta un cuadro resumen de las prioridades dentro del Sistema BAN.

Función	Prioridad
Lectura de datos de los sensores	Alta
Controles de los parámetros ambientales	Alta
Manejadores de los actuadores	Alta
Monitorizar los parámetros ambientales	Normal
Supervisión de las alarmas	Normal
Programación de los parámetros ambientales	Normal
Programación de los sonidos	Baja
Envío de datos a la computadora personal	Baja

Tabla 3.4. Prioridades de las funciones dentro de la BAN.

4 DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE DEL SISTEMA DEL PROYECTO DE FASE EXPERIMENTAL BAN

Introducción

La más importante fase en el desarrollo de cualquier sistema de tiempo real es la generación de un diseño consistente que satisfaga la especificación de los requerimientos [B&W96].

El Sistema BAN es un sistema reactivo que presenta procesos que se ejecutan periódicamente y no terminan; en donde existe una relación continua con el ambiente, donde es este último el que manda.

Durante esta fase se tomó en cuenta a todo momento las características que debía tener; entre otras, la capacidad de poder integrar y coordinar el hardware y sus 4 microcontroladores, los cuales tienen que interactuar para realizar las funciones que la BAN le presenta al usuario. Se buscaba un sistema compacto, que corra sobre un procesador simple (microcontrolador ATmega 128), con restricciones de memoria limitada, respuesta rápida y que tenga un buen rendimiento.

El proceso de diseño de ésta aplicación de tiempo real involucró dividir el trabajo que se tiene que realizar en procesos responsables de una porción del problema. El uso de un núcleo de tiempo real permite esta división del trabajo que se tiene que realizar, lo cual generalmente simplifica el diseño del sistema [Lab02].

Bajo esta concepción se comenzó el proceso de diseño del Sistema BAN definiendo primero su arquitectura en el diseño de alto nivel. Se definieron los componentes o módulos necesarios para cumplir con los objetivos del Sistema BAN.

Luego se prosiguió con el diseño detallado de cada componente definido en la arquitectura, especificando cada uno de sus procedimientos y estructuras de datos. Así, en este proceso también se fueron definiendo los servicios que debía de proveer el núcleo de tiempo real.

Se prosiguió con la implementación del sistema embebido. Se codificó cada uno de sus módulos empezando primero con los del primer nivel hasta terminar con los procesos del nivel 3.

En este capítulo se presentan tablas que contienen las funciones, variables y constantes de cada módulo o paquete de software.

4.1 Arquitectura del Sistema BAN

Bajo la perspectiva planteada se da lugar al modelo que se muestra en la Figura 4.1. Aquí, el nivel 1, el más bajo, es el núcleo del sistema. Este nivel atrapa todas las interrupciones, realiza la planificación y ofrece a los niveles superiores un modelo de procesos secuenciales independientes. Además, provee los mecanismos de comunicación y sincronización entre procesos. También, administra los temporizadores de los procesos.

El nivel 2 implementa los manejadores de cada uno de los dispositivos de la BAN. Entre estos están, los manejadores del Módulo de Adquisición de Datos de los sensores de la BAN, del Módulo de Manejo de Dispositivos de Control, el Módulo de Ingreso de Datos, el Módulo de Visualización en LCD, el Módulo de Sonidos.

El nivel 3 contiene los procesos de procesamiento de las señales sensadas, los proceso de control de los parámetros ambientales de la BAN, el proceso de monitorización, el proceso de supervisión de alarmas y el proceso de comunicación serial con la computadora personal.

Figura 4.1. Estructura del Sistema BAN (Núcleo + Tareas + Procesos).

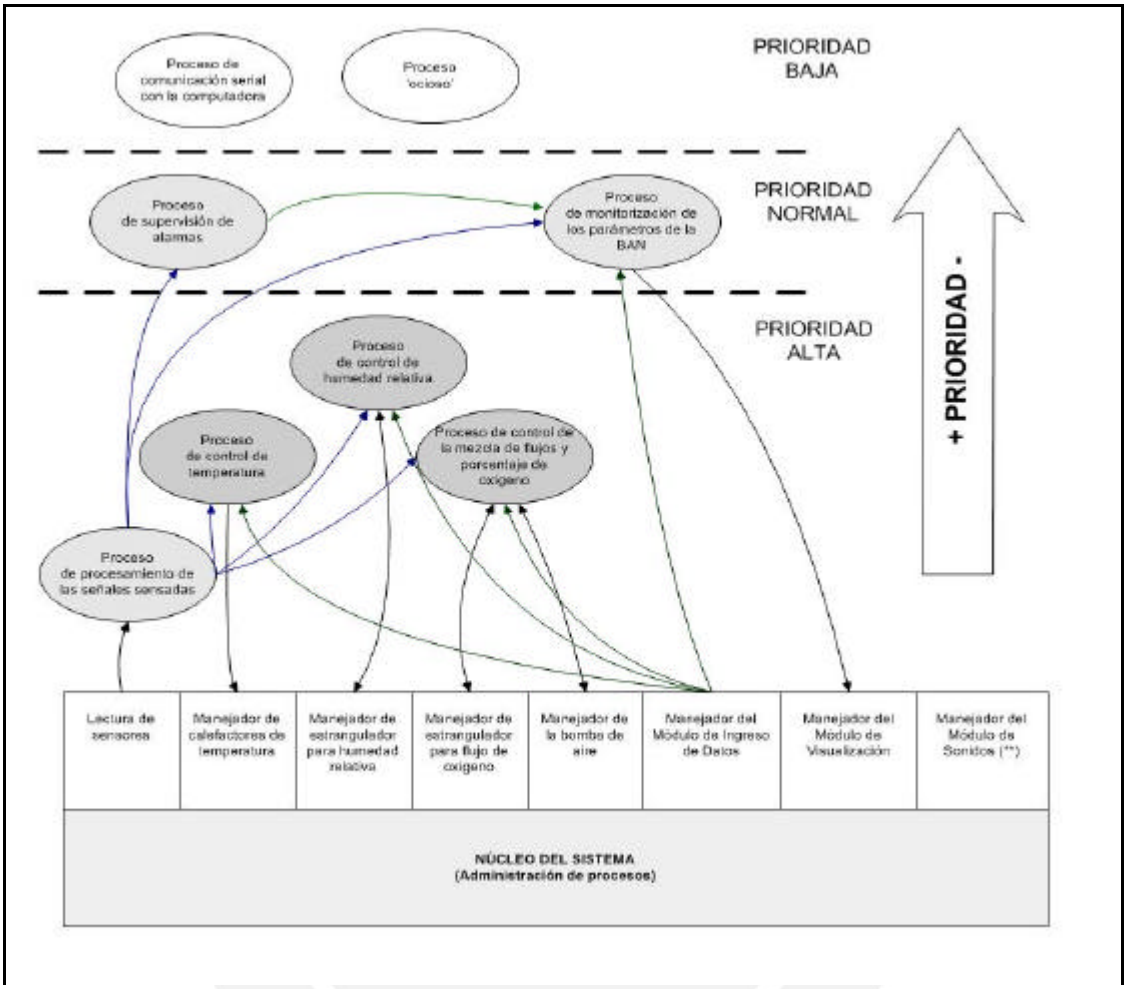


Figura 4.2. Interacción entre los módulos del Sistema BAN.

Tomando en consideración la interacción entre procesos, es útil distinguir entre tres tipos de comportamiento: Independientes, Cooperativos o Competitivos [B&W96]. El Sistema BAN fue diseñado de tal manera que sus procesos sean cooperativos. Los procesos cooperativos son aquellos que regularmente comunican y sincronizan sus actividades con la finalidad de llevar a cabo una tarea común. En el caso de la BAN cooperan principalmente para controlar los parámetros ambientales dentro de límites definidos (Figura 4.2).

La BAN, como todo sistema embebido, tiene un número finito de recursos los cuales son compartidos entre los procesos (p.e. periféricos, memoria, procesador). Para que los procesos obtengan acceso a estos recursos compartidos, deben de competir entre sí. El acto de usar los recursos inevitablemente requiere la comunicación y sincronización entre los procesos del sistema. Sin embargo, aunque los procesos se comuniquen y sincronicen para tener acceso a los recursos, son esencialmente independientes.

4.2 Sistema BAN

Se dividió el Sistema BAN por niveles y estos a su vez se subdividieron en módulos o paquetes de implementación, cada cual con funcionalidad específica (Tabla 4.1). Cabe mencionar que para la implementación de cada una de las funciones se tuvo que desarrollar una convención de llamadas a función, la cual se describe en el Anexo D.

NIVEL 1
BAN_SistemaBAN
LIB_FuncionesGenerales
UC_ATmega128
OS_GHOST
NIVEL 2
TEC_InterfazTeclado
SON_InterfazSonidos
VIS_InterfazVisualizacion
LEC_InterfazLecturaDatos
MG_ManejadoresFuncionesGenerales
MT_ManejadorCalefactoresTemperatura
MH_ManejadorEstranguladorHumedadRelativa
MO_ManejadorEstranguladorFlujoOxigeno
MA_ManejadorBombaAire
NIVEL 3
PCTE_Temperatura
PCHR_HumedadRelativa
PCOF_PorcentajeOxigenoYFlujoMezcla
PSEN_ProcesamientoSenales
PMON_Monitor
PALM_Alarmas
PSRL_TransmisionSerial

Tabla 4.1. Módulos del software del Sistema BAN.

Es conveniente detallar en esta sección los módulos del nivel 1: BAN_SistemaBAN, LIB_FuncionesGenerales y UC_ATmega128. El módulo OS_GHOST se desarrollará en la siguiente sección.

4.2.1 BAN: Módulo principal del Sistema BAN

Este módulo se encarga de llamar al código del resto de módulos del Sistema BAN. Mapea las interrupciones, espera la estabilización con el resto de módulos de hardware, llama a las funciones de inicialización y activa el núcleo.

FUNCIONES
BAN_ESTABILIZAR – Espera la estabilización con el resto de los módulos electrónicos.
BAN_reset – *"Poner en juego la pelota ..."*. Inicializa el puntero de pila y llama a *main()*.
BAN_main – Llama a las funciones de inicialización del sistema, inicializa la primera tarea y empieza la multitarea (el control nunca vuelve a *main()*).
BAN_inicializarCapa2 – Centraliza la inicialización de los módulos del nivel 2.
BAN_inicializarGHOST – Centraliza la inicialización de *GHOST* y le da a conocer los procesos que ejecutará.
VARIABLES
BAN_bPosFinalRAMUtilizada – No es un dato útil. Su valor indica la ultima posición utilizada en la RAM.

Tabla 4.2. Funciones y variables del módulo.

4.2.2 LIB: Librería de funciones generales

Este módulo brinda funciones útiles para el resto de módulos. Retardo de espera activa, rotación y funciones matemáticas.

FUNCIONES
LIB_retardar771ms – Retardo de 771 ms.
LIB_rotarPalabra12Bits – Destructiva. Rota un número natural de 12 bits.
LIB_div8u – Destructiva. 8 / 8 Bits División SIN signo.
LIB_sumar16u – Destructiva. 16 + 16 Bits Suma SIN signo.
LIB_restar16u – Destructiva. 16 - 16 Bits Resta SIN signo.
LIB_mul16u – No Destructiva. 16 * 16 = 16 Bits Multiplicación SIN signo.
LIB_div16u – Destructiva. 16/16 Bits División SIN signo.

Tabla 4.3. Funciones del módulo.

4.2.3 UC: Configuración del microcontrolador ATmega 128

Este módulo contiene todas las funciones relacionadas a la configuración del microcontrolador ATmega 128. Entre las más importantes, la configuración de sus puertos de E/S y el marcapasos del núcleo (*ticks*).

FUNCIONES
UC_setearRegistros – Asigna a todos los registros el valor de r16.
UC_limpiarRegistros – Destructiva. Se pierde el valor de r16.
UC_limpiarSRAM – Limpia la memoria SRAM. Lleva todos los bytes a cero.
UC_configurarPuertos – Pone los valores en los puertos.
UC_configurarINT6INT7 – Configura las interrupciones INT6 e INT7.
UC_configurarTimer1 – Configura el Timer 1.
UC_resetTimer1 – Lleva a cero el contador del Timer 1.
UC_leerEEPROM – Lee un byte de la EEPROM.
UC_configurarUSART1 – Configura la USART.
UC_enviarByteUSART1 – Envía un byte por la USART.
CONSTANTES
Ticks UC_TIEMPO_NS_INT_TIMER1A = 4000000 (*Tick o marcapasos*) UC_VALOR_OCRA = 6000 (*Prescalador = 1. Puede interpretarse como Cycle Counter*)
USART UC_VALOR_UBRR = 194 (*Baudrate*)

Tabla 4.4. Funciones y constantes del módulo.

4.3 Nivel 1: Núcleo del Sistema BAN - GHOST

Gidems **H**ard(Heart) **O**perating **S**ys**T**em - ***GHOST***, es el corazón del Sistema BAN. Es un núcleo o *kernel* multitarea preemptivo, pequeño, con planificación por prioridades, responsable del manejo de los procesos (p.e. el manejo del tiempo de CPU) y la comunicación entre tareas. El servicio fundamental que provee es el cambio de contextos. Además, cuenta con todos los servicios para la sincronización y comunicación entre los procesos a través de mensajes y memoria compartida; así como para la implementación de retardos (y bloqueo).

GHOST, al ser preemptivo, le da el control del procesador al proceso listo de mayor prioridad, por lo que es determinística la ejecución dicho proceso; es decir, se puede determinar cuando ganará el control del procesador. De esta manera, el tiempo de respuesta a nivel de procesos es minimizado.

Además, *GHOST* es un núcleo capaz de manejar hasta 256 procesos, cada cual con una prioridad distinta. El número exacto de prioridades es determinado por la cantidad de procesos ejecutándose en el sistema.

Implementa también el *Proceso 'Ocioso'* el cual es el de más baja prioridad dentro del sistema.

4.3.1 Procesos

El concepto central de cualquier sistema operativo es el proceso: una abstracción de un programa en ejecución. Cada proceso es una entidad independiente, con su propio contador de programa y estado interno. Los procesos a menudo necesitan interactuar con otros procesos. Un proceso podría generar ciertas salidas que otro podría utilizar como entradas.

A cada proceso se le asigna una prioridad, de acuerdo a su importancia. Estas prioridades son estáticas debido a que no cambian durante la ejecución del sistema. Además, se le asigna su propio conjunto de registros del microcontrolador y su propia área de pila.

A los procesos de la capa inferior se les llamarán tareas. Tanto las tareas como los procesos se crearán en la inicialización del sistema debido a que se conoce el número exacto de los mismos.

4.3.2 Planificador

Un cambio de contexto es el proceso de guardar y restaurar el estado de la CPU y de otras variables, de tal modo que múltiples procesos puedan compartir el CPU y puedan avanzar en sus hilos de ejecución. Este podría pasar cuando un proceso solicita un servicio del sistema (p.e. *OS_esperarMensaje()*); o, cuando ocurre una interrupción del temporizador.

En un cambio de contexto, primero debe salvarse el estado del proceso actualmente corriendo en el área de almacenamiento de su contexto, de modo que pueda ser tomado luego exactamente donde se dejo. Este proceso lo implementan la función *OS_planificar()* y la rutina de servicio a la interrupción *OS_RSI_Tick*. Adicionalmente, la segunda llama a la función *OS_incrementarTiempoTicks()* de manera que lleve la cuenta del tiempo del sistema y de los temporizadores de cada proceso.

El tiempo del *tick* de *GHOST* se determina en tiempo de compilación. Cuando ocurre la interrupción, el hardware salva la dirección de retorno. A continuación, el microcontrolador salta a la dirección especificada en el vector de interrupciones. Esto es todo lo que el hardware hace. De aquí en adelante, el software decide lo que se hace.

Lo primero que hace el procedimiento de servicio de interrupciones es guardar en la pila todos los registros del microcontrolador. El número de proceso actual y un apuntador a su entrada de la tabla de procesos (TCB) se guardan en variables globales a fin de poder encontrarlos rápidamente. Luego, se salva el puntero de pila del proceso que se va a dejar de ejecutar en su entrada de la tabla de procesos. A continuación, se planifica el proceso de mayor prioridad listo para ejecutarse. Se guarda en las variables globales el nuevo número de proceso actual y un apuntador a su entrada de la tabla de procesos. Luego, se hace que el puntero de pila apunte al nuevo contexto. Finalmente, se desapilan los registros del nuevo proceso y se retorna.

Es importante notar que las interrupciones son deshabilitadas, por lo que el núcleo no es interrumpible. Esto pone una restricción en el tiempo de respuesta mínimo de una interrupción (latencia) a eventos externos [A&T89].

4.3.3 Estados de los procesos

Los estados de los procesos de la BAN son los siguientes: Proceso Creado, Proceso Listo, Proceso Esperando, Proceso Corriendo y Rutina de Servicio a la Interrupción Corriendo. En cada momento un proceso puede estar sólo en uno de estos estados (Figura 4.3).

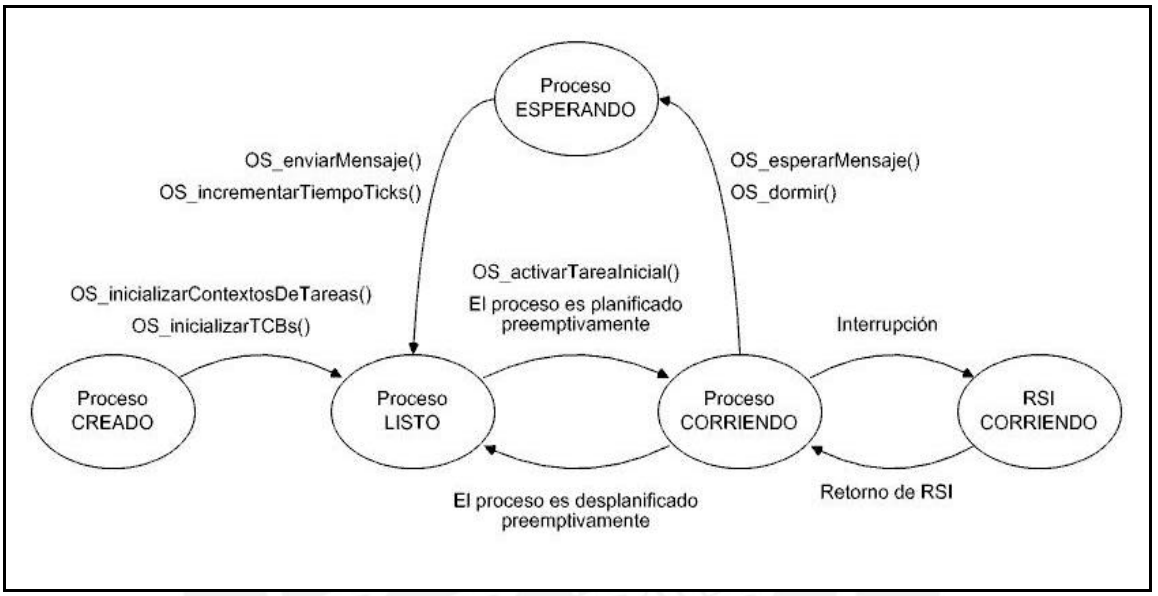


Figura 4.3. Diagrama de Estados de los procesos de la BAN.

Un proceso en estado CREADO es aquel que posee su programa en la EEPROM pero aún su contexto no ha sido creado con *OS_inicializarContextosDeTareas()*, ni sus datos, en su bloque de control, han sido inicializados con *OS_inicializarTCBs()*.

En la inicialización del sistema se obtienen los datos del primer proceso a ejecutarse (el de mayor prioridad) llamando a *OS_obtenerStackPTareaInicial()*, a *OS_obtenerProgCont TareaInicial()* y a *OS_activarTareaInicial()*. Luego se activan las interrupciones y se salta al código del proceso. Es en este momento que comienza la multitarea y el proceso pasa al estado CORRIENDO.

Un proceso puede retardarse una cierta cantidad de tiempo llamando a *OS_dormir()*; así, el proceso se bloquea y pasa al estado ESPERANDO. Cuando se cumple el tiempo que especificó pasa al estado LISTO. Es *OS_incrementarTiempoTicks()* quien lleva la cuenta de los retardos de los procesos.

Si un proceso en ejecución necesita esperar algún evento podría suspenderse también llamando a *OS_esperarMensaje()*, en este caso el siguiente proceso de mayor prioridad es planificado. Tanto otro proceso como una RSI (Rutina de Servicio a la Interrupción) pueden enviar una señal al proceso en espera indicando la ocurrencia del evento con *OS_enviarMensaje()*.

Un proceso en ejecución siempre puede ser interrumpido. En este caso el proceso pasa al estado RSI CORRIENDO. Cuando una interrupción ocurre, la ejecución del proceso se suspende y la RSI toma el control del microcontrolador. La RSI puede llevar al estado LISTO más de un proceso, activándolos mediante el envío de algún mensaje.

Cuando todos los procesos están esperando por algún evento o por el cumplimiento de algún retardo se ejecuta un proceso interno del sistema denominado *Proceso 'ocioso'* (el de más baja prioridad).

4.3.4 Comunicación entre procesos

Si dos procesos desean comunicarse (o un proceso y una RSI), el camino más sencillo es pasar los datos a través de estructuras de datos compartidas previamente especificadas. Sin embargo, se debe acceder a éstas variables dentro de una sección crítica, con la finalidad de prevenir la corrupción de los datos (exclusión mutua). El esquema de mayor simplicidad y rapidez es el de deshabilitar y habilitar las interrupciones; el cual funciona bien cuando el número de variables es pequeño [A&T89]. Especialmente, cuando todos los procesos residen en un espacio de direcciones único y pueden referenciar elementos, tales como variables globales, punteros, *buffers*, listas ligadas y *buffers* circulares [Lab02]. Sin embargo, la desventaja de este método es que las interrupciones son deshabilitadas para todos los procesos inclusive para el temporizador de planificación [A&T89]. *GHOST* provee dos macros que permiten realizar estas operaciones: *OS_ENTRAR_SECCION_CRITICA()*, la cual guarda en la pila el estado del flag de interrupciones deshabilitadas (I) y deshabilita las interrupciones, y *OS_SALIR_SECCION_CRITICA()*, la cual restaura el estado de las interrupciones guardado en la pila. Usando este esquema, si se llama a algún servicio con las interrupciones deshabilitadas o no, el estado es preservado a través de la llamada.

El otro medio de comunicación entre procesos es a través del paso de mensajes. En este caso *GHOST* implementan 3 primitivas. Una llamada a *OS_esperarMensaje()* pone al proceso en estado de espera (o bloqueado). *OS_enviarMensaje()* envía un mensaje a un proceso, devolviéndolo a la cola de proceso listas para ejecutarse. Estas dos primitivas hacen uso de *OS_planificar()* en caso se haya activado con la acción una tarea de mayor prioridad (núcleo preemptivo). *OS_aceptarMensaje()* sólo revisa el campo de mensaje, mas no bloquea al proceso que la llama.

Cabe decir que en este esquema de envío de mensajes el emisor explícitamente nombra al receptor del mensaje, esquema de nombramiento directo y por lo tanto de uno a uno. Además, este esquema es asimétrico debido a que el receptor no especifica la fuente de la cual quiere recibir un mensaje [B&W96].

4.3.5 Sincronización entre procesos

Para lograr la sincronización entre procesos y el manejo de recursos, se utilizan también los mecanismos de *GHOST* discutidos en la sección anterior.

Se puede controlar el acceso a las secciones críticas deshabilitando las interrupciones, lo que permite especificar operaciones atómicas, utilizando las macros *OS_ENTRAR_SECCION_CRITICA()* y *OS_SALIR_SECCION_CRITICA()* [A&T89].

Así también, los mensajes pueden ser utilizados como mecanismos de señalización y sincronización, utilizando las funciones *OS_esperarMensaje()* y *OS_enviarMensaje()*. De ésta manera se pueden eliminar los esquemas de espera ocupada.

4.3.6 Tabla de procesos

GHOST mantiene una tabla de procesos, con una entrada por cada proceso. Cada entrada es una estructura de datos denominada Bloque de Control de Proceso o Tarea (TCB) que posee la información del proceso. Cada bloque posee la siguiente información:

Campo	Tamaño
Puntero de Pila	2 bytes
Flags	1 byte
Retardo	2 bytes
Puntero a mensaje	2 bytes
Total	**7 bytes**

Tabla 4.5. Estructura del Bloque de Control de Tareas.

4.3.7 Organización de la memoria

El sistema utiliza los 3 tipos de memoria que provee el microcontrolador ATmega 128. En su memoria Flash se encuentra tanto el código del núcleo como los códigos de los procesos que se implementan. Adicionalmente, se encuentran las constantes que utilizan (Algunas constantes también se encuentran en la EEPROM).

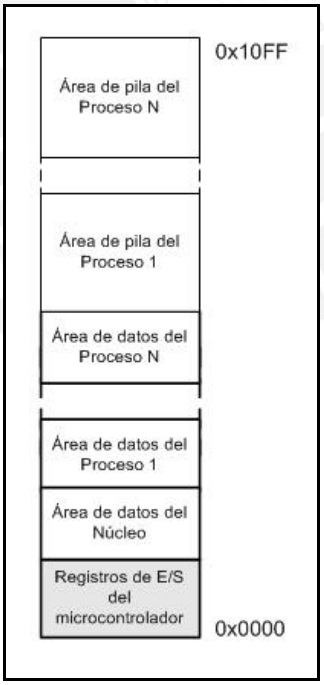


Figura 4.4. Distribución de la memoria SRAM.

En la memoria SRAM se encuentran el espacio de datos requerido por el núcleo (variables y estructuras) y las áreas de datos y pila requeridos por los procesos (Figura 4.4).

4.3.8 Implementación de GHOST

A continación se detallan las funciones implementadas por GHOST:

FUNCIONES (Generales)
OS_PUSH_ALL – Apila todos los registros del microcontrolador.
OS_POP_ALL – Desapila todos los registros del microcontrolador.
OS_PUSH_ALL_R16 – Apila 32 valores con el contenido del registro r16.
FUNCIONES (Sección Crítica)
OS_ENTRAR_SECCION_CRITICA – Marca el inicio de una sección crítica. Protege un área de código, deshabilitando las interrupciones en caso estén activas.
OS_SALIR_SECCION_CRITICA – Sale de la sección crítica marcada. En caso las interrupciones hayan estado activas antes de entrar a la sección critica, las vuelve a activar.
FUNCIONES (RSI, Reloj, Temporizadores)
OS_RSI_Tick – RUTINA DE SERVICIO A LA INTERRUPCIÓN (RSI). Planificador del núcleo a NIVEL de interrupción.
OS_incrementarTiempoTicks – Incrementa el tiempo en *ticks* del sistema; y, decrementa los retardos de las tareas/procesos.
FUNCIONES (Servicios del Núcleo)
OS_planificar – Planificador del núcleo a NIVEL de tareas/procesos.
OS_dormir – Retardo en número de *ticks*. Desplanifica la tarea/proceso actual. Esta pasa a estado de espera.

Tabla 4.6. Funciones generales, de acceso a sección crítica, reloj, temporizadores y planificación del módulo.

OS_planificar(), es el planificador del núcleo a nivel de tareas/procesos; es decir, es un servicio del núcleo que llaman los procesos a través de otros servicios como pedido de retardos o manejo de mensajes.

Lo primero que realiza *OS_planificar()* es desactivar las interrupciones y llamar a la macro *OS_PUSH_ALL()*, que apila los valores de todos los registros de la tarea/proceso que se va a desplanificar. Luego, apila el valor del SREG (*Status Register*) del microcontrolador. Seguidamente, lo que se necesita es salvar el puntero de pila de la tarea/proceso que se va a dejar de ejecutar. A fin de poder encontrar rápidamente el TCB de ésta tarea/proceso que se estaba ejecutando, se cuenta con una variable de 2 bytes llamada *OS_DirecTCBTareaActual*, la cual contiene la dirección del TCB. Al puntero X del

microcontrolador se le asigna ésta dirección y se guarda en el campo respectivo el valor del puntero de pila. Luego, se toma el campo de flags del TCB de la tarea/proceso, se desactiva el OS_BIT_TAREA_ACTIVA y se guarda la modificación.

A continuación, se incrementa el contador de cambios de contextos del sistema, el cual es una variable de 4 bytes llamada *OS_ContCambioContexto*; y,se ejecuta el algoritmo de planificación por prioridades de *GHOST*.

Se utiliza el arreglo de TCBs, de todas las tareas/procesos del Sistema BAN, como cola de los procesos listos. Cabe resaltar que este arreglo contiene ya las tareas/procesos ordenadas de mayor a menor prioridad. Adicionalmente, se tiene una arreglo auxiliar llamado *OS_arrDirecTCB*, que contiene las direcciones de inicio de cada TCB, con la finalidad de hacer la búsqueda más rápida. Al puntero Z se le asigna la dirección de este arreglo. Una a una se revisa cada entrada, asignándole al puntero Y la dirección del TCB de la tarea/proceso que se va a analizar. Esto se realiza accediendo al campo de flags de cada TCB y se verifica si tiene activado el OS_BIT_TAREA_LISTA. De ser así, se ha encontrado en el sistema la tarea/proceso de mayor prioridad lista para ejecutarse (su identificador se guarda en la variable *OS_bTareaActual*). En caso contrario, se continúa con la búsqueda.

Una vez que se ha determinado la tarea/proceso de mayor prioridad que se va a planificar, se guarda en la variable *OS_DirecTCBTareaActual* la dirección su TCB (valor que se encuentra en el puntero Y). Seguidamente, se accede al campo de puntero de pila del TCB, se toma su valor y se le asigna al SP (*Stack Pointer*) del microcontrolador. También, se accede al campo de flags del TCB y se activa el OS_BIT_TAREA_ACTIVA.

Finalmente, se desapila el valor del SREG del nuevo proceso que se va a planificar y se desapilan los valores de todos los demás registros llamando a la macro *OS_POP_ALL()*. Se ha concluido la planificación y el cambio de contexto, lo último que se hace es retornar al código donde se dejó al nuevo proceso que gozará del procesador, activando las interrupciones.

OS_RSI_Tick(), es el planificador del núcleo a nivel de interrupción; es decir, es llamado cada vez que se cumple el tiempo de un tick o marcapasos del núcleo. En general, realiza las mismas funciones que *OS_planificar()*; sin embargo, se diferencia de ésta en dos aspectos fundamentales. El primero es llevar la cuenta del tiempo del sistema, incrementando el valor de la variable *OS_Ticks*; y, el segundo y más importante aspecto es llevar la cuenta del tiempo de los temporizadores de cada proceso, decrementando los retardos de los procesos que se bloquearon, por una cantidad de ticks determinados, haciendo uso del servicio que brinda el núcleo para esta finalidad, *OS_dormir()*. Estos dos procedimientos los implementa a través de la función *OS_incrementarTiempoTicks()*.

Al inicio de *OS_incrementarTiempoTicks()*, se le asigna al puntero Y la dirección del arreglo de TCBs llamado *OS_arrTCB*. Para cada tarea/proceso, se accede al campo de retardo de su TCB y se analiza si su valor es mayor que cero. De ser así, se decrementa en uno su valor y se guarda. Luego, se verifica si el retardo es igual a cero. En caso de que esto sea cierto, significa que se ha cumplido el tiempo que la tarea/proceso pidió bloquearse y dormir; por lo que se accede al campo de flags de su TCB, se desactiva el OS_BIT_TAREA_ESPERANDO_RETARDO y se activa el OS_BIT_TAREA_LISTA, lo que la pone otra vez en la cola de procesos listos.

OS_dormir(), es el servicio que el núcleo provee a los procesos para dormir una cantidad determinada de ticks que se le pasan como parámetro. Lo primero que se hace es llamar a la macro *OS_ENTRAR_SECCION_CRITICA()*. Luego, se hace que el puntero Y apunte al TCB de la tarea/proceso que se está ejecutando. Se accede al campo de flags del TCB, se desactiva el OS_BIT_TAREA_LISTA y se activa el OS_BIT_TAREA_ESPERANDO_RETARDO, pasando así a bloquearse en espera del evento del temporizador. También, se le asigna al campo de retardo del TCB la cantidad de tiempo en ticks que se pasó como parámetro. Finalmente, se llama a *OS_SALIR_SECCION_CRITICA()* y a *OS_planificar()*, para que se planifique a la siguiente tarea de mayor prioridad lista para ejecutarse.

FUNCIONES (Manejo de Mensajes)
OS_enviarMensaje – Envía un mensaje a una tarea/proceso y la pone en la cola de procesos como lista.
OS_esperarMensaje – Espera por un mensaje. La tarea/proceso se bloquea.
OS_aceptarMensaje – Revisa el campo de mensaje. Es no bloqueante.

Tabla 4.7. Funciones de manejo de mensajes del módulo.

OS_enviarMensaje(), es el servicio del núcleo que permite a los procesos enviar un mensaje a otro proceso. Se pasan como parámetro, el identificador de la tarea/proceso y la dirección del mensaje. Lo primero que se hace es llamar a la macro *OS_ENTRAR_SECCION_CRITICA()*. Luego, se hace que el puntero Y apunte al TCB de la tarea/proceso que se está ejecutando. A continuación, se accede al campo de puntero al mensaje del TCB y se guarda la dirección del mensaje. Luego, se accede al campo de flags del TCB y se analiza si el OS_BIT_TAREA_ESPERANDO_MENSAJE está activado. De ser así, la tarea/proceso se encontraba ya esperando un mensaje. Se desactiva el OS_BIT_TAREA_ESPERANDO_MENSAJE debido a que la tarea/proceso ya recibió un mensaje y se activa el OS_BIT_TAREA_LISTA devolviéndola a la cola de procesos listos. Luego se llama a *OS_SALIR_SECCION_CRITICA()* y a *OS_planificar()*, para que se planifique a la siguiente tarea de mayor prioridad lista para ejecutarse. En caso contrario, se accede al campo de flags del TCB, se activa el OS_BIT_TAREA_MENSAJE_RECIBIDO. Finalmente, se llama a OS_SALIR_SECCION_CRITICA() y se retorna.

OS_esperarMensaje(), es el servicio del núcleo que permite a los procesos esperar por un mensaje; y, tiene como parámetro de salida la dirección del mensaje recibido. Lo primero que se hace es llamar a la macro *OS_ENTRAR_SECCION_CRITICA()*. Luego, se hace que el puntero Y apunte al TCB de la tarea/proceso que se está ejecutando. Se accede al campo de flags del TCB y se analiza si el OS_BIT_TAREA_MENSAJE_RECIBIDO está activado. De ser así, la tarea/proceso ya cuenta con un mensaje recibido. Se desactiva el OS_BIT_TAREA_MENSAJE_RECIBIDO, para futuros envíos de mensajes, se llama a *OS_SALIR_SECCION_CRITICA()* y se retorna. En caso contrario, se accede al campo de flags del TCB, se activa el OS_BIT_TAREA_ESPERANDO_MENSAJE y se desactiva el OS_BIT_TAREA_LISTA, lo que provoca que la tarea/proceso se bloquee hasta que reciba un mensaje. Luego, se llama a *OS_SALIR_SECCION_CRITICA()* y a *OS_planificar()*, para que se planifique a la siguiente tarea de mayor prioridad lista para ejecutarse. Al final se le asigna al parámetro de salida la dirección del mensaje recibido y se retorna.

OS_aceptarMensaje(), por su parte sólo revisa el campo de mensaje y retorna su valor. Es una función no bloqueante debido a que regresa inmediatamente después de revisar el campo del mensaje. Lo primero que se hace es llamar a la macro *OS_ENTRAR_SECCION_CRITICA()*. Luego, se hace que el puntero Y apunte al TCB de la tarea/proceso que se está ejecutando. Se asigna al parámetro de salida el valor del campo de puntero al mensaje del TCB, se llama a *OS_SALIR_SECCION_CRITICA()* y se retorna.

FUNCIONES (Activación de la Tarea Inicial)
OS_activarTareaInicial – Marca como activa la tarea/proceso inicial.
OS_obtenerStackPTareaInicial – Obtiene el puntero de pila de la tarea/proceso inicial.
OS_obtenerProgContTareaInicial – Obtiene el contador de programa de la tarea/proceso inicial.
FUNCIONES (Inicialización)
OS_insertarEnArrPCIniciales – Inserta en el arreglo de contadores de programa la dirección de inicio de una tarea/proceso.
OS_inicializarTCBs – Inicializa los bloques de control de cada tarea/proceso (TCB).
OS_inicializarContextosDeTareas – Inicializa los contextos de las tareas/procesos.
OS_inicializarVariables – Inicializa las variables de módulo.
FUNCIONES (Procesos del Sistema)
OS_ProcesoOcioso – Proceso ocioso del sistema. Se planifica cuando es el único proceso en la cola de listos.

Tabla 4.8. Funciones de inicialización del módulo y proceso ocioso.

VARIABLES
OS_bTareaActual – Tarea actual. *OS_arrPCIniciales* – Arreglo de PC's de las tareas/procesos. *OS_arrTCB* – Arreglo de TCB's. *OS_arrDirecTCB* – Arreglo de las direcciones de los TCB's de las tareas./procesos. *OS_bDirecTCBTareaActualL* – LSB del puntero al TCB de la tarea/proceso actualmente ejecutándose. *OS_bDirecTCBTareaActualH* – MSB. *OS_bTicksLL* – LSB del valor actual del tiempo del sistema. *OS_bTicksLH* – 2do. Byte. *OS_bTicksHL* – 3er. byte. *OS_bTicksHH* – MSB. *OS_bContCambioContextoLL* – LSB del contador de cambios de contexto. *OS_bContCambioContextoLH* – 2do. Byte. *OS_bContCambioContextoHL* – 3er. Byte. *OS_bContCambioContextoHH* – MSB.
CONSTANTES
Ticks por segundo OS_TICKS_POR_SEG = 250
Tareas OS_NUMERO_TAREAS_APLICACION = 10 OS_NUMERO_TAREAS_SISTEMA = 1 OS_NUMERO_TAREAS = 11
Tamaño de segmentos OS_TAMANNO_SEGMENTO_DATOS_SO = 32 OS_TAMANNO_SEGMENTO_DATOS_TAREA = 250
TCB OS_TAMANNO_TCB = 8 (*2(Ptr. Pila) + 1(Flags) + 2(Retardo) + 2(Direc. Mensaje) + 1(Padding)*) OS_TAMANNO_ARREGLO_TCB = 88 OS_TAMANNO_ARREGLO_DIREC_TCB = 22
Arreglo de PC iniciales OS_TAMANNO_ARREGLO_PC_INICIALES = 22
Configuración de los bits en los bytes OS_BIT_TAREA_ACTIVA = 0 (*Tarea en ejecución*) OS_BIT_TAREA_LISTA = 1 (*Tarea en la cola de tareas listas*) OS_BIT_TAREA_ESPERANDO_RETARDO = 4 (*Tarea bloqueada por retardo*) OS_BIT_TAREA_ESPERANDO_MENSAJE = 5 (*Tarea bloqueada por espera de mensaje*) OS_BIT_TAREA_MENSAJE_RECIBIDO = 7 (*Tarea tiene un mensaje recibido*)
Misceláneos FALSO = 0 VERDADERO = 1

Tabla 4.9. Variables y constantes del módulo.

4.4 Nivel 2: Manejo de los dispositivos de entrada / salida

4.4.1 TEC: Programación de los parámetros de la BAN

Este módulo implementa el manejador que se encarga de procesar los comandos que le llegan del Módulo de Ingreso de Datos. Debido a que no hay manera de predecir cuando los comandos llegarán, porque estos son generados cuando el usuario usa la interfaz, para evitar hacer un uso ineficiente del microcontrolador al hacer que este revisando constantemente por nuevos comandos, se utilizó un esquema de procesamiento asíncrono (interrupción) [A&T89] que recoge el comando que llega, modifica el valor del parámetro y lo guarda dejándolo listo para el uso de las otras tareas que lo necesiten. Además, en el caso de que llegue un comando relacionado a sonidos, envía los comandos respectivos al Módulo de Sonidos.

FUNCIONES
TEC_RSI_Teclado – RSI de manejo de los comandos provenientes del Módulo de Ingreso de Datos.
TEC_RSI_Sonidos – RSI que inicializa el puerto conectado al Módulo de Sonidos.
TEC_TareaTeclado – Ejecuta los cambios en la programación de los parámetros de la BAN. *TEC_ejecutarAccionDelCodigoTeclado* – Ejecuta la acción respectiva según el comando proveniente del Módulo de Ingreso de Datos.
TEC_calcularDatoProgFlujoOxiYFlujoAire – Calcula el valor programado de Flujo de oxígeno y Flujo de aire.
TEC_convertirUnidadesTempeA12Bits – Convierte el dato de grados Celsius a un número natural de 12 bits.
TEC_convertirUnidadesFlujoA12Bits – Convierte el dato de litros por minutos (Lpm) a un número natural de 12 bits.
TEC_inicializarVariables – Inicializa las variables del módulo.
TEC_inicializarBytesTeclado – Inicializa los bytes de estado del teclado.
TEC_inicializarDatosProgramados – Inicializa los valores programados.

Tabla 4.10. Funciones del módulo.

VARIABLES
BNibbleLSB – Primer nibble enviado por el Módulo de Ingreso de Datos. *BCodigoEnviado* – Comando enviado por el Módulo de Ingreso de Datos. *BFlagsTeclado* – Banderas asociadas al módulo. *BDatoProgTempeInter* – Valor programado en °C. *BDatoProgHumRelat* – Valor programado en °C. *BDatoProgFlujoMezcla* – Valor programado en Lpm. *BDatoProgPorcOxi* – Valor programado en unidades porcentuales. *BDatoProgTempeCVCCalculado* – Valor programado en °C. *BDatoProgFlujoOxiCalculado* – Valor programado en Lpm. *BDatoProgFlujoAireCalculado* – Valor programado en Lpm. *BdatoProgTempeInter12BitsL* – LSB del valor programado en 12 bits. *BdatoProgTempeInter12BitsH* – MSB del valor programado en 12 bits. *BdatoProgHumRelat12BitsL* – LSB del valor programado en 12 bits. *BdatoProgHumRelat12BitsH* – MSB del valor programado en 12 bits. *BDatoProgTempeCVCCalculado12BitsL* – LSB del valor programado en 12 bits. *BDatoProgTempeCVCCalculado12BitsH* – MSB del valor programado en 12 bits. *BdatoProgFlujoOxiCalculado12BitsL* – LSB del valor programado en 12 bits. *BdatoProgFlujoOxiCalculado12BitsH* – MSB del valor programado en 12 bits. *BdatoProgFlujoAireCalculado12BitsL* – LSB del valor programado en 12 bits. *BdatoProgFlujoAireCalculado12BitsH* – MSB del valor programado en 12 bits.
CONSTANTES
Valores iniciales de los Datos Programados VAL_INI_DATOPROG_n *(para cada parámetro n, el valor depende del parámetro)* *Rangos de los Datos Programados* VAL_MINIMO_DATOPROG_n *(para cada parámetro n)* VAL_MAXIMO_DATOPROG_n
Códigos del teclado del Módulo de Ingreso de Datos SEL_n *(para cada parámetro n)* INC_n DEC_n

Tabla 4.11. Variables y constantes del módulo.

4.4.1.1 Protocolo de comunicación

A continuación se presenta el algoritmo del Módulo de Ingreso de Datos (teclado) y se especifica el protocolo de comunicación desarrollado por el GIDEMS. Primero se sensan constantemente los botones del teclado: monitor, temperatura, humedad relativa, oxígeno, flujo de la mezcla, tipo de sonido, volumen de sonido. Se espera a que se suelte el botón seleccionado. Luego, cuando se mueve el *encoder* se envía el código correspondiente, según el parámetro seleccionado. Este código se envía en 2 nibbles, donde cada uno genera una interrupción en el ATmega 128. El ATmega 128 por su parte, almacena el primer nibble y cuando llega el segundo nibble los ordena de manera de poder interpretar el código.

En el caso especial de que se quiera modificar un valor correspondiente al Módulo de Sonidos se genera una interrupción adicional antes de enviar el código, de manera que el ATmega 128 pueda limpiar lo que anteriormente haya colocado en el puerto de comunicación al teclado.

4.4.2 SON: Interfaz con el Módulo de Sonidos

Este módulo implementa el manejador que se encarga de enviar los comandos respectivos al Módulo de Sonidos, según la acción que haya realizado en usuario desde la interfaz del sistema. Implementa todas las funciones para manejar dicho módulo.

FUNCIONES
SON_seleccionarSonido – Envía el comando que selecciona la melodía.
SON_incrementarVolumenSonido – Envía el comando que incrementa el volumen.
SON_decrementarVolumenSonido – Envía el comando que decrementa el volumen.
SON_apagarSonido – Envía el comando que apaga la melodía.
SON_ejecutarSonido0 – Envía el comando que toca la melodía SONIDO0_PRUEBA.
SON_ejecutarSonido1 – Envía el comando que toca la melodía SONIDO1_VOZ.
SON_ejecutarSonido2 – Envía el comando que toca la melodía SONIDO2_CLASICA1.
SON_ejecutarSonido3 – Envía el comando que toca la melodía SONIDO3_CLASICA2.
SON_resetearPuertoASonidos – Envía el comando que resetea las líneas conectadas al Módulo de Sonidos.
CONSTANTES
Configuración de los bits en los bytes BIT_ME0 = 0 BIT_ME1 = 1 BIT_ME2 = 6 BIT_COK = 7

Tabla 4.12. Funciones y constantes del módulo.

4.4.2.1 Protocolo de comunicación

El protocolo de comunicación desarrollado por el GIDEMS especifica 4 líneas para la comunicación entre el Módulo de Sonidos y el Módulo de Procesamiento y Control. El ATmega 128 codifica todas las acciones que permite el Módulo de Sonidos en 3 bits de datos que escribe en el puerto, según el comando que le llegue del Módulo de Ingreso de

Datos. Finalmente, activa el bit de confirmación de código. Cuando el Módulo de Sonidos recibe esta señal comienza a procesar el código enviado como dato.

4.4.3 VIS: Interfaz con el Módulo de Visualización

Este módulo implementa el manejador que se encarga del Módulo de Visualización. Cubre la funcionalidad de dibujar un píxel, especificando su color, en alguna página de la memoria de video de dicho módulo; así como, la de manejo de la paginación de la memoria de video al poder indicarle que página mostrar en la pantalla LCD. Implementa la primitiva de dibujo de rectángulos en la pantalla LCD, la cual es la base de los dibujos, y la función de dibujo de una imagen en la pantalla LCD, tomando como parámetro la dirección de alguna imagen que haya sido guardada, en el formato adecuado, en la memoria flash del microcontrolador ATmega 128.

FUNCIONES
VIS_dibujarRect – Dibuja una rectángulo en la página activa del Módulo de Visualización.
VIS_cambiarPaginaMostrar – Cambia a la página que se va a ver en el Módulo de Visualización.
VIS_dibujarImagen – Dibuja una imagen en la página activa del Módulo de Visualización.
VIS_intercambiarPaginaEscribirEnPuerto – Intercambia el valor de la página activa actual y lo pone en el puerto.
VIS_inicializarVariables – Inicializa las variables del módulo.
VARIABLES
BPaginaEscribir – Página en la cual se está dibujando.
CONSTANTES
Dirección de cambio de página FIL_DIRECCAMBIOPAG = 0xFF COL_DIRECCAMBIOPAG = 0xFF

Tabla 4.13. Funciones, variables y constantes del módulo.

4.4.3.1 Protocolo de comunicación

El Módulo de Procesamiento y Control se conecta de forma paralela con el Módulo de Visualización a través de un bus de datos, un bus de direcciones y un bus de control. El protocolo de comunicación fue desarrollado por el GIDEMS. La funcionalidad básica que

provee el protocolo es la de dibujo de un solo píxel en cualquiera de las páginas de la memoria de video. Se especifica la dirección, compuesta por la página a la cual se va a direccionar (bits más significativos) y el desplazamiento u *offset* dentro de ésta página (bits menos significativos). Luego, se coloca el byte de datos en el puerto, que especifica los píxeles que se van a dibujar (2 píxeles por byte). Finalmente, se generan las señales de control en el bus de control.

4.4.4 LEC: Lectura de datos de los sensores

Este módulo implementa los manejadores de los conversores analógico a digital (ADC's) del Módulo de Adquisición de Datos. La tarea de lectura de datos de los sensores se encarga de leer periódicamente todas las variables ambientales que se sensan en un formato de números naturales de 12 bits. Luego se encarga de filtrarlas.

Se diseño un filtro FIR (Finite Impulse Response) de Medias Móviles para filtrar las señales sensadas de flujo de aire y oxígeno. Un filtro FIR es la suma de los productos de los datos muestreados con los coeficientes del filtro; en el caso de Medias Móviles todos valores de los coeficientes son iguales. El número de coeficientes del filtro o *taps* determina la "suavidad" del filtro, así como también el retardo de cada dato filtrado (la longitud de tiempo que le toma a un dato recién ingresado salir del bloque del filtro). Entre sus principales características está que son siempre estables y son capaces de tener una respuesta de fase que es lineal, lo que equivale a decir que su respuesta tiene un retraso constante [Low05][Iri05] [ASL05][Bar05].

Se especificaron 256 coeficientes para el filtro, teniendo como base el criterio de Nyquist debido a la frecuencia de 100 Hz de la señal de flujo de aire.

Adicionalmente, se implementó un *buffer* circular para el filtro y se optimizó el código guardando en variables los cálculos ya realizados en la iteración anterior de tal forma que sirvan en la actual. De esta manera en vez de realizar 256 operaciones sólo se realizan dos.

FUNCIONES
LEC_TareaLecturaDatos12Bits – Tarea encargada de la lectura de las variables ambientales en un formato de números naturales de 12 bits.
LEC_leerCanales12BitsParametrosAmbientales – Lectura de todas las variables ambientales sensadas.
LEC_filtrarDatos – Implementa un Filtro FIR de Medias Móviles.
LEC_leerCanalMAX186 – Manejador para leer un canal del MAX186.
LEC_leerCanalMAX187 – Manejador para leer un canal del MAX187.
LEC_inicializarVariables – Inicializa las variables del módulo.
VARIABLES
ArrDatoSen12BitsTempePiel – Arreglo de datos sensados en 12 bits. *ArrDatoSen12BitsTempeInter* – Para CONTROL. Arreglo de datos en 12 bits. *ArrDatoSen12BitsTempeCCATIzq* – Arreglo de datos en 12 bits. *ArrDatoSen12BitsTempeCCATDer* – Arreglo de datos en 12 bits. *ArrDatoSen12BitsTempeCVC* – Para CONTROL. Arreglo de datos en 12 bits. *ArrDatoSen12BitsHumRelat* – Para CONTROL. Arreglo de datos en 12 bits. *ArrDatoSen12BitsFlujoOxi* – Para CONTROL. Arreglo de datos en 12 bits. *ArrDatoSen12BitsFlujoAire* – Para CONTROL. Arreglo de datos en 12 bits. *ArrDatoSen12BitsFlujoMezcla* – Arreglo de datos en 12 bits.
CONSTANTES
Periodo de la Tarea LEC_PERIODO_LEC_MILISEG = 4 LEC_PERIODO_LEC_TICKS = 1
Número de muestras a tomar por parámetro ambiental LEC_NUMERO_DATOSEN12BITS_n (*para cada parámetro n*) LEC_NUMERO_DATOSEN12BITS_FLUJOn = 256 (*para cada flujo n*) (*256 por Nyquist. La frecuencia mayor es 100 Hz*)
Tamaño de los buffer de datos LEC_NUMBYTES_CABECERA_ARREGLO = 10 (*2(dato filtrado) + 4(suma) + 2(índice) + 2(tamaño del buffer). Luego vienen 2*Nmuestras bytes*) LEC_NUMBYTES_DATOSEN12BITS_n (*para cada parámetro n*)
Palabras de control del ADC MAX186 CANAL_DATO_n (para cada parámetro n)
Palabras de control del ADC MAX187 CANAL_DATO_TEMPEPIEL = 0x00

Tabla 4.14. Funciones, variables y constantes del módulo.

4.4.5 MG: Funciones compartidas por los manejadores

Este módulo implementa las funciones generales para los manejadores de los actuadores del Módulo de Manejo de Dispositivos de Control. Debido a que se comparten puertos y pines para los diferentes dispositivos de control, implementa un acceso síncrono para la escritura en dichos puertos.

FUNCIONES
MG_escribirPPIPuertoA – Escribe un byte en el puerto A del Módulo de Manejo de Dispositivos de Control.
MG_escribirPPIPuertoB – Escribe un byte en el puerto B del Módulo de Manejo de Dispositivos de Control.
MG_inicializarActuadoresGeneral – Inicializa las líneas conectadas al Módulo de Manejo de Dispositivos de Control.
VARIABLES
BbufferActuadoresPPIPuertoA – Mapea las líneas de los Calefactores y Estranguladores de las vías de flujo de oxígeno y flujo humedecido. *BbufferActuadoresPPIPuertoB* – Mapea las líneas del Bus de Control del DAC TLC1257 (Bomba de Aire).
CONSTANTES
Configuración de los bits en los bytes BIT_PULSOHABIL_LATCH_A = 0 BIT_PULSOHABIL_LATCH_B = 1

Tabla 4.15. Funciones, variables y constantes del módulo.

4.4.6 MT: Calefactores de temperatura

Este módulo implementa el manejador de los calefactores del Módulo de Manejo de Dispositivos de Control, y sus primitivas. La tarea de manejo de los calefactores se encarga de generar las ondas PWM de los calefactores de temperatura.

El método de actuación al ser implementado por software requiere una atención constante por parte del microcontrolador. Esta generación de la onda se realiza normalmente con mucho más frecuencia que el intervalo de control y no es síncrona con el lazo de control.

Cabe resaltar que es útil controlar los calefactores a través del uso de ondas PWM debido a que es la manera más eficiente de modular la potencia de los calefactores mediante el uso

de líneas que sólo proveen estados de prendido y apagado [A&T89]. La potencia promedio de cada calefactor se controla por el *duty cycle* de cada onda PWM.

FUNCIONES
MT_TareaManejoCalefactores – Tarea que maneja los rendimientos las ondas PWM de los calefactores de temperatura.
MT_controlarOndasPWMCalefactores – Controla que se cumplan los rendimientos las ondas PWM de los calefactores de temperatura.
MT_manejarRendimientoOndasPWMCalefactores – Función Manejador de las ondas PWM de los calefactores. Configura los nuevos rendimientos de las ondas PWM de los calefactores.
MT_encenderCalefactorCupulaCCAT – Primitiva que enciende el calefactor de la cúpula CCAT.
MT_apagarCalefactorCupulaCCAT – Primitiva que apaga el calefactor de la cúpula CCAT.
MT_encenderCalefactorNeumaticoCVC – Primitiva que enciende el calefactor del circuito neumático CVC.
MT_apagarCalefactorNeumaticoCVC – Primitiva que apaga el calefactor del circuito neumático CVC.
MT_inicializarActuador – Inicializa los calefactores (apagados).
MT_inicializarVariables – Inicializa las variables del módulo.
VARIABLES
BcontPasosEncendidoCalefCCAT – Número de Pasos que lleva encendido el calefactor de la cúpula CCAT. *BcontPasosEncendidoCalefCVC* – Número de Pasos que lleva encendido el calefactor del circuito neumático CVC. *BnumPasosRendimientoCalefCCAT* – Número de Pasos que debe de estar encendido el calefactor de la cúpula CCAT. *BnumPasosRendimientoCalefCVC* – Número de Pasos que debe de estar encendido el calefactor circuito neumático CVC.
CONSTANTES
Periodo de la Tarea MT_PERIODO_MT_TICKS = 2 MT_PERIODO_MT_SEG = 0,008
Pasos de las ondas PWM NUM_PASOS_PERIODO_ONDAPWM_CCAT = 30 NUM_PASOS_PERIODO_ONDAPWM_CVC = 100
Identificadores de los calefactores ID_CALEFACTOR_CUPULA_CCAT = 0 ID_CALEFACTOR_NEUMAT_CVC = 1

Tabla 4.16. Funciones, variables y constantes del módulo.

4.4.7 MH: Estrangulador de humedad relativa

Este módulo implementa el manejador del estrangulador de la vía humedecida del Módulo de Manejo de Dispositivos de Control, y sus primitivas. Mueve el estrangulador, avanzándolo o retrocediéndolo, una cantidad de milímetros especificada. Tal cantidad de milímetros se traduce a un intervalo de tiempo y este a *ticks* y se bloquea por la cantidad de *ticks* calculados haciendo uso del servicio que brinda el núcleo para este caso.

FUNCIONES
MH_manejarEstranguladorHumRelatCVC – Función Manejador del estrangulador de la vía humedecida del Módulo de Manejo de Dispositivos de Control. Mueve el estrangulador la cantidad de milímetros pasada como parámetro.
MH_detenerEstranguladorHumRelatCVC – Primitiva que detiene el movimiento del estrangulador.
MH_avanzarEstranguladorHumRelatCVC – Primitiva que avanza el estrangulador. Menos flujo humedecido.
MH_retrocederEstranguladorHumRelatCVC – Primitiva que retrocede el estrangulador. Más flujo humedecido.
MH_inicializarActuador – Inicializa el estrangulador a su posición inicial.
CONSTANTES
Retardo para un milímetro MH_RETARDO_MH_MILISEG_PARA_1MM = 771 MH_RETARDO_MH_TICKS_PARA_1MM = 192
Configuración de los bits en los bytes MH_BIT1_AVANZAR = 3 *(Cuando esta en 1 avanza para cerrar la vía de flujo humedecido)* MH_BIT0_RETROCEDER = 2 *(Cuando esta en 1 retrocede para abrir la vía de flujo humedecido)*
Palabras de control del manejador MH_AVANZAR_ESTRANGULADOR = 0 MH_RETROCEDER_ESTRANGULADOR = 1

Tabla 4.17. Funciones y constantes del módulo.

4.4.8 MO: Estrangulador de flujo de oxígeno

Este módulo implementa el manejador del estrangulador de la vía oxigenada del Módulo de Manejo de Dispositivos de Control, y sus primitivas. Su funcionamiento es similar al funcionamiento del manejador del estrangulador de humedad relativa.

FUNCIONES
MO_manejarEstranguladorFlujoOxiCVC – Función del Manejador del estrangulador de la vía oxigenada del Módulo de Manejo de Dispositivos de Control. Mueve el estrangulador la cantidad de milímetros pasada como parámetro.
MO_detenerEstranguladorFlujoOxiCVC – Primitiva que detiene el movimiento del estrangulador.
MO_avanzarEstranguladorFlujoOxiCVC – Primitiva que avanza el estrangulador. Menos flujo oxigenado.
MO_retrocederEstranguladorFlujoOxiCVC – Primitiva que retrocede el estrangulador. Más flujo oxigenado.
MO_inicializarActuador – Inicializa el estrangulador a su posición inicial.
CONSTANTES
Retardo para un milímetro MO_RETARDO_MO_MILISEG_PARA_1MM = 771 MO_RETARDO_MO_TICKS_PARA_1MM = 192
Configuración de los bits en los bytes MO_BIT1_AVANZAR = 1 *(Cuando esta en 1 avanza para cerrar la vía de oxígeno)* MO_BIT0_RETROCEDER = 0 *(Cuando esta en 1 retrocede para abrir la vía de oxígeno)*
Palabras de control del manejador MO_AVANZAR_ESTRANGULADOR = 0 MO_RETROCEDER_ESTRANGULADOR = 1

Tabla 4.18. Funciones y constantes del módulo.

4.4.9 MA: Manejador de la bomba de aire

Este módulo implementa el manejador de la bomba de aire del Módulo de Manejo de Dispositivos de Control, y sus primitivas. Controla el conversor digital a analógico (DAC) conectado a la bomba de aire, donde la palabra de control es proporcional a la cantidad de flujo requerida.

FUNCIONES
MA_incrementarFlujoAire – Incrementa el flujo de aire, según el cambio en el valor almacenado en la palabra de control.
MA_decrementarFlujoAire – Decrementa el flujo de aire, según el cambio en el valor almacenado en la palabra de control.
MA_enviarDatoDACTLC1257 – Manejador del conversor digital analógico. Envía la palabra de control.
MA_inicializarActuador – Inicializa la bomba de aire con el flujo mínimo.
MA_inicializarVariables – Inicializa las variables del módulo.
VARIABLES
BpalabraControlBombaAireL – LSB de la Palabra de control de DAC LTC1257. *BpalabraControlBombaAireH* – MSB de la Palabra de control de DAC LTC1257.
CONSTANTES
Cantidad de pasos a incrementar/decrementar a la Palabra de Control MA_CANTIDADPASOS_INC = 50 MA_CANTIDADPASOS_DEC = 50
Configuración de los bits en los bytes MA_DIN = 0 MA_SCLK = 1 MA__LOAD = 2
Rango de la palabra de control MA_PALABRACONTROL_MAXIMA = 4050 *(Menos flujo de aire)* MA_PALABRACONTROL_MINIMA = 2050 *(Más flujo de aire)*

Tabla 4.19. Funciones, variables y constantes del módulo.

4.5 Nivel 3: Procesos

4.5.1 PSEN: Procesamiento de los datos sensados

Este proceso se encarga de recolectar todos los valores sensados de las variables ambientales en números naturales de 12 bits, los procesa y convierte los valores a sus unidades físicas respectivas para el monitoreo y el control.

FUNCIONES
PSEN_ProcesoProcesamientoSenales – Proceso encargado del procesamiento de las señales sensadas. Convierte los datos sensados de números naturales de 12 bits a sus unidades físicas respectivas.
PSEN_convertir12BitsAUnidadesParametrosAmbientales – Convierte los datos sensados de números naturales de 12 bits a las unidades físicas respectivas.
PSEN_convertir12BitsAUnidadesTempePiel – Convierte el dato sensado de temperatura en la piel del bebé de número natural de 12 bits a grados Celsius.
PSEN_convertir12BitsAUnidadesTempe – Convierte el dato sensado de temperaturas en la BAN de un número natural de 12 bits a grados Celsius.
PSEN_convertir12BitsAUnidadesHumRelat – Convierte el dato sensado de humedad relativa en la BAN de un número natural de 12 bits a unidades porcentuales.
PSEN_convertir12BitsAUnidadesFlujo – Convierte el dato sensado de flujo de oxígeno o flujo de aire de un número natural de 12 bits a litros por minuto.
PSEN_convertir12BitsAUnidadesFlujoMezcla – Convierte el dato sensado por el GISS de flujo de la mezcla de un número natural de 12 bits a litros por minuto.
PSEN_calcularDatoSenFlujoMezclaYPorcentajeOxigeno – Calcula los datos de flujo total de la mezcla y porcentaje de oxígeno en la mezcla a partir de los datos tomando como entradas el flujo de oxígeno y el flujo de aire.
PSEN_inicializarVariables – Inicializa las variables del módulo.
PSEN_inicializarDatosSensados – Inicializa los datos sensados.

Tabla 4.20. Funciones del módulo.

VARIABLES
BdatoSenTempePiel – Valor sensado de la temperatura en la piel del bebé en grados Celsius. *BdatoSenTempeInter* – Valor sensado de la temperatura dentro de la cúpula de la BAN en grados Celsius. *BdatoSenTempeCCATIzq* – Valor sensado de la temperatura en el sector izquierdo de la cúpula de la BAN en grados Celsius. *BdatoSenTempeCCATDer* – Valor sensado de la temperatura en el sector derecho de la cúpula de la BAN en grados Celsius. *BdatoSenTempeCVC* – Valor sensado de la temperatura en el circuito CVC de la BAN en grados Celsius. *BdatoSenHumRelat* – Valor sensado de la humedad relativa en la cúpula de la BAN en unidades porcentuales. *BdatoSenFlujoOxi* – Valor sensado del flujo de oxígeno en el circuito CVC de la BAN en litros por minuto. *BdatoSenFlujoAire* – Valor sensado del flujo de aire en el circuito CVC de la BAN en litros por minuto. *BdatoSenFlujoMezcla* – (Se mide). Valor sensado del flujo total de la mezcla en el circuito CVC de la BAN en litros por minuto. *BdatoSenFlujoMezclaCalculado* – (Se calcula). Valor calculado del flujo total de la mezcla en el circuito CVC de la BAN en litros por minuto. *BdatoSenPorcOxiCalculado* – (Se calcula). Valor calculado del porcentaje de oxígeno en la mezcla en el circuito CVC de la BAN en unidades porcentuales.
CONSTANTES
Periodo del Proceso PSEN_PERIODO_PSEN_MILISEG = 4 PSEN_PERIODO_PSEN_TICKS = 1

Tabla 4.21. Variables y constantes del módulo.

4.5.2 PCTE: Control de temperatura

En esta Fase Experimental de la BAN los procesos de control de parámetros se implementan mediante algoritmos basados en el error entre el valor sensado y el valor programado.

El proceso de Control de Temperatura es un proceso periódico que se encarga del control de temperatura en la BAN, tanto en la cápsula neonatal como en el circuito neumático CVC. Para ambos casos la estrategia de control que se implementa se basa en el error entre el valor sensado de temperatura y el valor programado (meta de control), según el cual se modifica el valor medio de la onda PWM respectiva.

FUNCIONES
PCTE_ProcesoControlTemperatura – Proceso encargado del control de la temperatura tanto en la Cápsula Neonatal como en el Circuito Neumático CVC.
PCTE_controlarTemperaturaCCATNormal – Control de la temperatura en la Cápsula Neonatal.
PCTE_controlarTemperaturaCVCNormal – Control de la temperatura en el Circuito Neumático CVC.
CONSTANTES
Periodo del Proceso PCTE_PERIODO_PCTE_MILISEG = 8 PCTE_PERIODO_PCTE_TICKS = 2

Tabla 4.22. Funciones y constantes del módulo.

4.5.3 PCHR: Control de humedad relativa

Este es un proceso periódico que se encarga del control de la humedad relativa en la BAN. La estrategia de control que se implementa se basa en el error entre el valor sensado de humedad relativa y el valor programado (meta de control), si el valor sensado es mayor al valor programado se invoca al manejador para que desplace el estrangulador con tal de que se cierre un poco más (una cantidad definida de milímetros, en este caso 1 mm) la vía de flujo humedecido, en caso contrario se desplaza el actuador con tal de que se abra un poco más.

FUNCIONES
PCHR_ProcesoControlHumedadRelativa – Proceso encargado del control de la humedad relativa en la Cápsula Neonatal.
PCHR_controlarHumedadRelativa – Control de la humedad relativa en la BAN.
CONSTANTES
Periodo del Proceso PCHR_PERIODO_PCHR_SEG = 2 PCHR_PERIODO_PCHR_TICKS = 500

Tabla 4.23. Funciones y constantes del módulo.

4.5.4 PCOF: Control de mezcla de flujos y porcentaje de oxígeno

Este es un proceso periódico que se encarga del control del flujo total de la mezcla y del control del porcentaje de oxígeno en la mezcla, a través del control del flujo de oxígeno y del control del flujo de aire.

En el caso del flujo de aire, la estrategia de control que se implementa se basa en el error entre el valor sensado de flujo de aire y el valor programado (meta de control), si el valor sensado es mayor al valor programado se invoca al manejador de la bomba de aire para que decremente el flujo de aire, en caso contrario se le pide que incremente el flujo de aire.

En el caso del flujo de oxígeno, la estrategia de control que se implementa también se basa en el error entre el valor sensado de flujo de oxígeno y el valor programado (meta de control), si el valor sensado es mayor al valor programado se invoca al manejador para que desplace el estrangulador con tal de que cierre un poco más (una cantidad definida de milímetros, en este caso 1 mm) la vía de flujo de oxígeno, en caso contrario se desplaza el estrangulador con tal de que se abra un poco más.

FUNCIONES
PCOF_ProcesoControlPorcentajeOxigenoYFlujoMezcla – Proceso encargado del control del porcentaje de oxígeno y del control del flujo de la mezcla en el Circuito Neumático CVC.
PCOF_controlarFlujoOxi – Control del flujo de oxígeno en el circuito neumático CVC.
PCOF_controlarFlujoAire – Control del flujo de aire en el circuito neumático CVC.
CONSTANTES
Periodo del Proceso PCOF_PERIODO_PCOF_MILISEG = 5000 PCOF_PERIODO_PCOF_TICKS = 1250

Tabla 4.24. Funciones y constantes del módulo.

4.5.5 PMON: Acción de monitorizar los parámetros de la BAN

Este es un proceso periódico que se encarga de mostrar los datos sensados y programados de las variables ambientales. También, muestra en pantalla si se ha activado alguna alarma. Cabe decir, que en cada refresco de pantalla sólo se redibujan las áreas de datos para una mayor eficiencia.

Los números fueron guardados en formato vectorial en la memoria EEPROM del microcontrolador ATmega 128, con la finalidad de optimizar el espacio ocupado. En lo que se refiere a la pantalla de la BAN, se desarrolló un programa especial llamado ***Bitmap2ScenixFrame*** capaz de pasar la información de la imagen en formato GIF al formato que entiende el microcontrolador Scenix del Módulo de Visualización, con la finalidad de facilitar su diseño en computador. La imagen se guarda en la memoria Flash del microcontrolador ATmega 128, la cual trabaja como un *buffer* de video.

FUNCIONES
PMON_ProcesoMonitor – Proceso encargado del dibujo de todas las variables ambientales en la pantalla LCD del Módulo de Visualización.
PMON_limpiarPantalla – Limpia las zonas que se invalidan en la pantalla al hacer un cambio.
PMON_dibujarAlarmasActivas – Dibuja en el Módulo de Visualización las alarmas activas.
PMON_dibujarParametrosAmbientales – Dibuja los valores de todas las variables ambientales en la pantalla LCD del Módulo de Visualización.
PMON_dibujarParametro – Dibuja el valor de una variable ambiental en la pantalla LCD del Módulo de Visualización.
PMON_dibujarNumero – Dibuja un dígito en la pantalla LCD del Módulo de Visualización.
PMON_inicializarPantallaEnPaginas – Inicializa el módulo. Dibuja la pantalla de la BAN en ambas páginas del Módulo de Visualización.

Tabla 4.25. Funciones del módulo.

CONSTANTES EN MEMORIA
ArrDireccNumeros – Arreglo con las direcciones donde se encuentran almacenadas la información en formato vectorial de cada número en base 10. *DireccPantallaBAN* – Dirección donde se encuentra el mapa de píxeles de la pantalla de la BAN. Se encuentra en el archivo "PMON_PantallaScenix.inc".
CONSTANTES
Periodo del Proceso PMON_PERIODO_PMON_SEG = 1 PMON_PERIODO_PMON_TICKS = 250
Datos Programados XINI_DATOPROG_n *(para cada parámetro n)* YINI_DATOPROG_n
Datos Sensados XINI_DATOSEN_n *(para cada parámetro n)* YINI_DATOSEN_n
Alarmas XINI_ALARMA_n *(para cada alarma n)* YINI_ALARMA_n

Tabla 4.26. Constantes del módulo.

4.5.6 PALM: Supervisión de alarmas

Este es un proceso periódico que supervisa las alarmas definidas en el Sistema BAN. (Ver Anexo C). Recoge los valores sensados de todos los parámetros ambientales y supervisa que se encuentren dentro de los rangos definidos, de no ser así activa la alarma respectiva.

FUNCIONES
PALM_ProcesoMonitorAlarmas – Monitorea los parámetros ambientales. Activa las alarmas.
PALM_evaluarAlarmas – Activa o desactiva las alarmas evaluando los parámetros ambientales.
PALM_inicializarVariables – Inicializa las variables de módulo.

Tabla 4.27. Funciones del módulo.

VARIABLES
BFlagsTemperatura – Flags de las seis alarmas relacionadas con la Temperatura. *BFlagsHumRelatYGases* – Flags de las dos alarmas relacionadas con la Humedad Relativa y de las cinco alarmas relacionadas con las vías de Oxígeno, Aire y la Mezcla.
CONSTANTES
Periodo del Proceso PALM_PERIODO_PALM_SEG = 1 PALM_PERIODO_PALM_TICKS = 250
Rangos de las alarmas VAL_MINIMO_ALARMA_n *(para cada alarma n)* VAL_MAXIMO_ALARMA_n

Tabla 4.28. Variables y constantes del módulo.

4.5.7 PSRL: Comunicación serial con la computadora

Este es un proceso periódico que se encarga de la transmisión serial de la información de la BAN a una computadora personal (PC). Transmite los valores sensados y programados de los parámetros de la BAN y los estados de las alarmas.

FUNCIONES
PSRL_ProcesoTransmisionSerial – Transmite la información de la BAN a una PC.
PSRL_transmitirDatos – Transmite la información de la BAN a una PC.
CONSTANTES
Periodo del Proceso PSRL_PERIODO_PSRL_SEG = 1 PSRL_PERIODO_PSRL_TICKS = 250

Tabla 4.29. Funciones y constantes del módulo.

5 PRUEBAS Y RESULTADOS

Introducción

Se ha logrado desarrollar el software del sistema embebido de la Burbuja Artificial Neonatal (Fase Experimental) de acuerdo a los objetivos planteados. El software desarrollado es capaz de controlar simultáneamente las múltiples variables ambientales relevantes como temperatura, porcentaje de humedad relativa, flujo y porcentaje de oxígeno; permite la interacción con el usuario para la configuración del sistema a través de sus interfaces; y, monitoriza los parámetros ambientales y las alarmas respectivas a fin de mostrarle la información importante.

Además, el software desarrollado logró la integración y coordinación del hardware de la BAN y de los 4 microcontroladores con los que cuenta, implementando así las funcionalidades de la BAN.

Así también, se desarrolló un núcleo denominado *GHOST* que maneja los recursos de hardware y dispositivos del sistema embebido, planifica los procesos del sistema y provee los mecanismos de comunicación y sincronización entre los procesos de la BAN.

A continuación se detallan las pruebas realizadas, así como, los resultados obtenidos. Adicionalmente, se muestran las curvas de la dinámica de la temperatura en la cápsula neonatal de la BAN y del flujo de aire en el circuito neumático.

5.1 Sistema BAN

El Sistema BAN se desarrolló bajo la estructura de 20 módulos divididos en 3 niveles. Cuatro módulos el Nivel 1: BAN, LIB, UC y OS. Nueve en el Nivel 2: TEC, SON, VIS, LEC, MG, MT, MH, MO y MA. Siete en el Nivel 3: PCTE, PCHR, PCOF, PSEN, PMON, PALM y PSRL.

Estos módulos implementan los 11 procesos de la BAN: las tareas MT, LEC y TEC; los procesos PSEN, PCOF, PCTE, PCHR, PALM, PSRL y PMON; y, el proceso de sistema denominado OCIOSO.

El uso de las memorias internas del ATmega 128, por el Sistema BAN, se muestra en la siguiente tabla:

Segmento	Inicio	Fin	Código (bytes)	Datos (bytes)	Usada (bytes)	Tamaño (bytes)	% de uso
Flash	0x000000	0x00C31E	11 514	38 400	49 914	131 072	38,1%
SRAM	0x000100	0x000620	0	*1 312	*1 312	4 096	*32,0%
EEPROM	0x000000	0x0000CE	0	206	206	4 096	5,0%

* Utilización por parte de variables estáticas. Nótese que el separador decimal es la coma.

Tabla 5.1. Uso de las memorias del ATmega 128 por el Sistema BAN.

Nótese que en el caso de la memoria Flash de los 48,74 KB (38,1%) que se utilizaron, 11,24 KB (8,8%) pertenecen a todos los códigos del Sistema BAN, mientras que los restantes 37,5 KB (29,3%) son de la pantalla de la BAN.

En el caso de la memoria SRAM se utilizaron 1,28 KB de los 4 KB (32%) en variables estáticas. Lo que dejó 250 bytes para la pila de cada uno de los 11 procesos de la BAN; utilizándose finalmente el 100% de la memoria SRAM.

También, se obtuvo para cada proceso el tiempo de ejecución del peor de los casos mediante el análisis del código fuente. El modelo del microcontrolador que se utilizó fue el que brinda el Entorno de Desarrollo Integrado (IDE) AVRStudio 4.10.356 del fabricante Atmel [Atm05].

Nº	Proceso	Número de ciclos	Tiempo de cómputo, C	Periodo, T	Utilización, U
1	MT	260	17,333 µs	8 ms	0,217%
2	LEC	8 230	548,667 µs	4 ms	13,717%
3	TEC	593	39,533 µs	1 s	0,004%
4	PSEN	7 546	503,067 µs	4 ms	12,577%
5	PCOF	2 025	135,000 µs	5 s	0,003%
6	PCTE	1 089	72,600 µs	8 ms	0,908%
7	PCHR	335	22,333 µs	2 s	0,001%
8	PALM	602	40,133 µs	1 s	0,004%
9	PSRL	561 864	37,458 ms	1 s	3,746%
10	PMON	425 600	28,373 ms	1 s	2,837%

Tabla 5.2. Tiempos de ejecución en el peor de los casos de los procesos BAN.

Con estos resultados medimos el porcentaje de utilización del procesador utilizando la siguiente fórmula:

$$Utilización = \sum_{i=1}^{N} \left(\frac{Ci}{Ti} \right)$$ Ecuación 5.1

Donde, para el Sistema BAN la utilización resulta de un ***34,014%***.

5.1.1 Asignación de prioridades a los procesos de la BAN

En el Sistema BAN se utilizó el esquema de asignación de prioridades basado en la tasa monótona de plazos de ejecución (*DMPO, Deadline monotonic priority ordering*) de Leung y Whitehead (1982) [L&W82], donde a cada proceso se le asigna una única prioridad basada en su plazo de ejecución (a diferencia del algoritmo de tasa monótona, el cual se basa en el periodo del proceso); así, para plazos más bajos, prioridades más altas, $D_i < D_j \Rightarrow P_i > P_j$).

En la siguiente tabla se muestran todos los atributos relevantes de los procesos de la BAN. En lo respecto a prioridades, el valor cero (0) corresponde a la máxima prioridad.

Nº	Proceso	Tipo	Tiempo de cómputo, C	Periodo, T	Plazo, D	Prioridad, P
1	MT	Periódico	17,333 µs	8 ms	1 ms	0
2	LEC	Periódico	48,667 µs	4 ms	2 ms	1
3	TEC	Esporádico	39,533 µs	1 s	4 ms	2
4	PSEN	Periódico	503,067 µs	4 ms	4 ms	3
5	PCOF	Periódico	135,000 µs	5 s	5 ms	4
6	PCTE	Periódico	72,600 µs	8 ms	8 ms	5
7	PCHR	Periódico	22,333 µs	2 s	8 ms	6
8	PALM	Periódico	40,133 µs	1 s	1 s	7
9	PSRL	Periódico	37,458 ms	1 s	1 s	8
10	PMON	Periódico	28,373 ms	1 s	1 s	9
11	OCIOSO	-	0,200 µs	-	-	10

Tabla 5.3. Atributos de los procesos BAN.

5.1.2 Análisis de planificabilidad basado en la utilización

Se aplicó un test de planificabilidad sencillo, el cual a pesar de no ser exacto es atractivo debido a su simplicidad. Liu y Layland (1973) [L&L73], demostraron que se puede obtener un test de planificabildad considerando sólo la utilización del conjunto de procesos del sistema. Si la siguiente condición es verdadera entonces todos los N procesos cumplirán sus plazos (*deadlines*):

$$\sum_{i=1}^{N}\left(\frac{Ci}{Ti}\right) < N\,(2^{1/N} - 1) \qquad \text{Ecuación 5.2}$$

Nótese que la segunda parte de la ecuación tiende asintóticamente a 0,693.

La utilización combinada de todos los procesos es 34,014%, la cual está debajo del límite calculado en 71,545%; por lo tanto se garantiza que el conjunto de procesos BAN alcanzarán todos sus plazos (*deadlines*).

En la Figura 5.1 se muestra la línea de tiempo de la ejecución de los procesos BAN (La gráfica fue realizada mediante la utilización del software Visual Micro Lab 3.11). Según Liu y

Layland (1973) [L&L73], para un conjunto de procesos que comparten un tiempo común de lanzamiento o instante crítico, puede ser demostrado que una línea de tiempo igual al tamaño del periodo más largo es suficiente. Por lo tanto, si todos los procesos cumplen sus primeros plazos (*deadlines*) entonces también alcanzaran los siguientes.

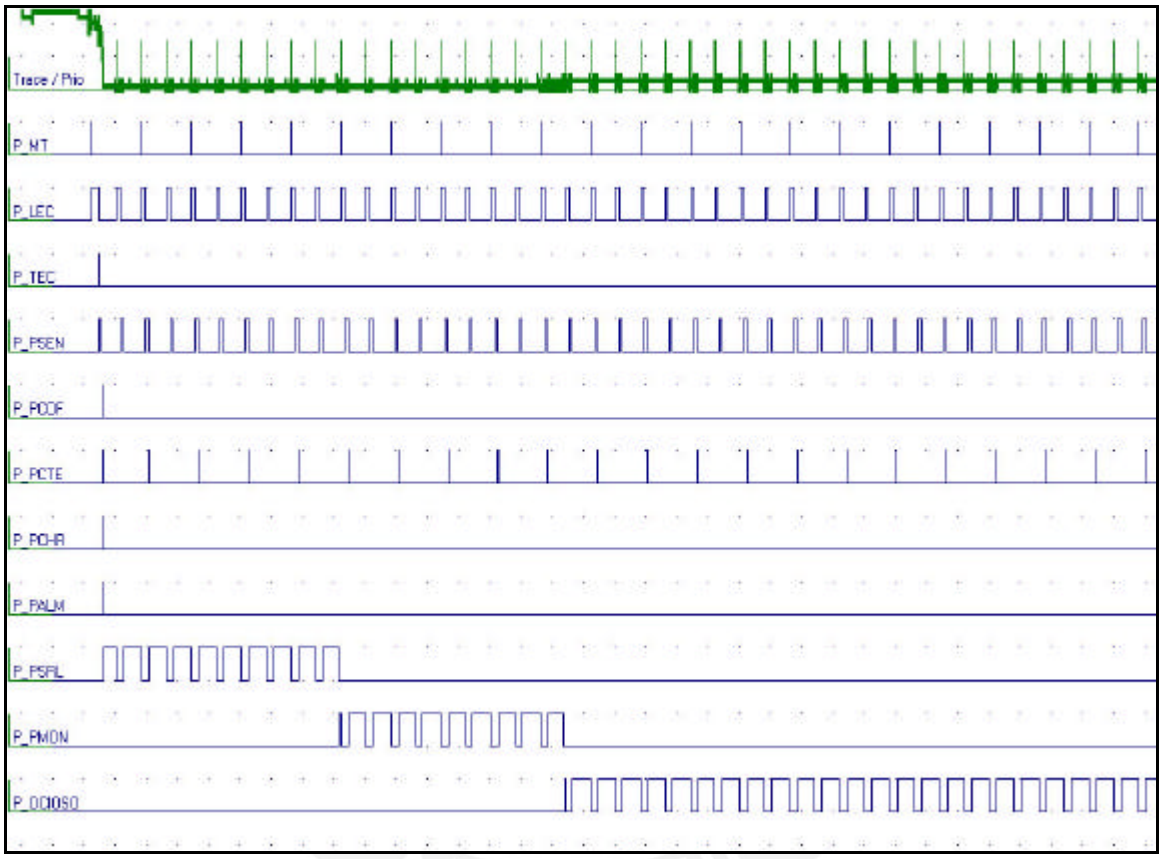


Figura 5.1. Ejecución de los procesos BAN en el tiempo.

5.1.3 Análisis de planificabilidad basado en el tiempo de respuesta

A continuación se aplicó un test diferente, denominado Test de Análisis del Tiempo de Respuesta. Primero, se utiliza un método analítico para predecir el tiempo de respuesta en el peor de los casos (R) de cada proceso del Sistema BAN. Luego los valores son comparados con los plazos (*deadlines*) de los procesos.

Para el proceso de mayor prioridad, su tiempo de respuesta en el peor de los casos es igual a su tiempo de cómputo (R = C). Los otros procesos sufren la interferencia de los procesos de mayor prioridad. Así:

$$Ri = Ci + Ii \qquad \text{Ecuación 5.3}$$

Donde, Ri es el tiempo de respuesta en el peor de los casos del proceso i; Ci es su tiempo de ejecución en el peor de los casos; e, Ii es la máxima interferencia que el proceso i puede experimentar en cualquier intervalo de tiempo [t, t+Ri[. Luego, se puede obtener la interferencia máxima (Ii) de la siguiente ecuación (Joseph y Pandya, 1986) [J&P86]:

$$Ii = \left\lceil \frac{Ri}{Tj} \right\rceil Cj \qquad \text{Ecuación 5.4}$$

Donde, $\lceil \ \rceil$ es la función mayor entero. Así, la ecuación 5.3 queda:

$$Ri = Ci + \sum_{j \in hp(i)} \left\lceil \frac{Ri}{Tj} \right\rceil Cj \qquad \text{Ecuación 5.5}$$

Donde, hp(i) es el conjunto de procesos con prioridad mayor a la prioridad del proceso i.

La ecuación es difícil de resolver debido a la presencia del mayor entero. Por este motivo se aplica el cálculo numérico. El método más simple es formar una relación de recurrencia (Audsley et al.) [Aud93]:

$$Wi^{n+1} = Ci + \sum_{j \in hp(i)} \left\lceil \frac{Wi^{n}}{Tj} \right\rceil Cj \qquad \text{Ecuación 5.6}$$

Los valores de w_i^n son monótonamente no decrecientes y la condición de parada se da cuando $w_i^n = w_i^{n+1}$; es decir, la ecuación tiene solución. Si w_i^n llega a ser mayor que su plazo (*deadline*), la ecuación no tiene solución y por lo tanto el sistema no es planificable.

Para el Sistema BAN se desarrolló el programa ***RTA_DMPO***, el cual utiliza la ecuación 5.6 para determinar el tiempo de respuesta en el peor de los casos para los procesos BAN; el algoritmo se detalla en el Anexo E. Después de correr el programa se obtuvieron los siguientes resultados:

Proceso	Tiempo de respuesta, R	Plazo, D
MT	17,333 µs	1 ms
LEC	566,000 µs	2 ms
TEC	605,533 µs	4 ms
PSEN	1 108,600 µs	4 ms
PCOF	1 243,600 µs	5 ms
PCTE	1 316,200 µs	8 ms
PCHR	1 338,533 µs	8 ms
PALM	1 378,666 µs	1 s
PSRL	51 996,672 µs	1 s
PMON	91 337,010 µs	1 s

Tabla 5.4. Tiempos de respuesta en el peor de los casos de los procesos BAN.

Todos los tiempos de respuesta fueron menores que sus respectivos plazos, por lo que se concluye que el Sistema BAN ***es planificable***.

Cabe mencionar que éste análisis de los tiempos de respuesta en el peor de los casos tiene la ventaja de que es suficiente y necesario.

5.2 GHOST

Gidems Hard (Heart) Operating SysTem - GHOST, es el corazón del Sistema BAN. Es un núcleo o *kernel* multitarea preemptivo con planificación por prioridades, responsable del manejo de los procesos (p.e. el manejo del tiempo de CPU) y la comunicación entre tareas. El servicio fundamental que provee es el cambio de contextos. Además, cuenta con todos los servicios para la sincronización y comunicación entre los procesos a través de mensajes y memoria compartida; así como para la implementación de retardos (y bloqueo).

En la siguiente tabla se muestra el uso de las memorias del ATmega 128 por *GHOST*:

Segmento	Código (bytes)	Datos (bytes)	Usada (bytes)	Tamaño (bytes)	% de uso
Flash	1 436	0	*1 436*	131 072	1,1%
SRAM	0	131	*131*	4 096	3,2%
EEPROM	0	0	*0*	4 096	0,0%

Tabla 5.5. Uso de las memorias del ATmega 128 por *GHOST*.

En términos de tiempo de procesamiento, el tiempo de planificación de la próxima tarea (función del núcleo *OS_RSI_Tick*) es de 51,733 µs (776 ciclos); lo que representa para el periodo del reloj de *GHOST*, de 4 ms, un porcentaje de **1,293%**.

Cuando se programa un valor en el teclado, se envía el código correspondiente a través de 2 interrupciones del teclado, como se explicó anteriormente. En la Figura 5.2 se muestra la llegada de las dos interrupciones. La rutina de servicio de la interrupción manda un mensaje a la tarea TEC, la cual se encuentra siempre en estado de espera del evento del teclado, y por lo tanto se pone a la tarea en el estado de lista para ejecutarse. Cuando esto ocurre durante la ejecución de una tarea de mayor prioridad, ésta última sigue su ejecución después del retorno de la interrupción, debido a que la tarea del teclado recién activada tiene menor prioridad.

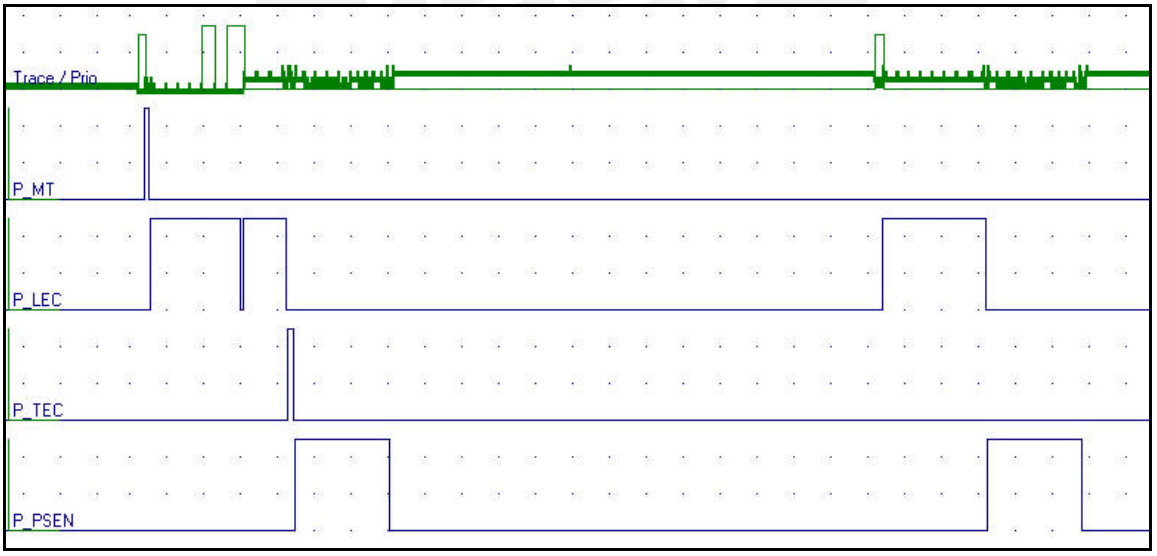


Figura 5.2. Activación de un proceso durante la ejecución de uno de mayor prioridad.

En la Figura 5.3 se muestra también la llegada de las dos interrupciones. La rutina de servicio de la interrupción manda un mensaje a la tarea TEC, la cual se encuentra siempre en estado de espera del evento del teclado, y por lo tanto se pone a la tarea en el estado de lista para ejecutarse. Sin embargo, cuando esto ocurre durante la ejecución de una tarea de menor prioridad, a ésta última se le desplanifica y se pasa a ejecutar la tarea de mayor prioridad, en este caso la tarea del teclado recién activada.

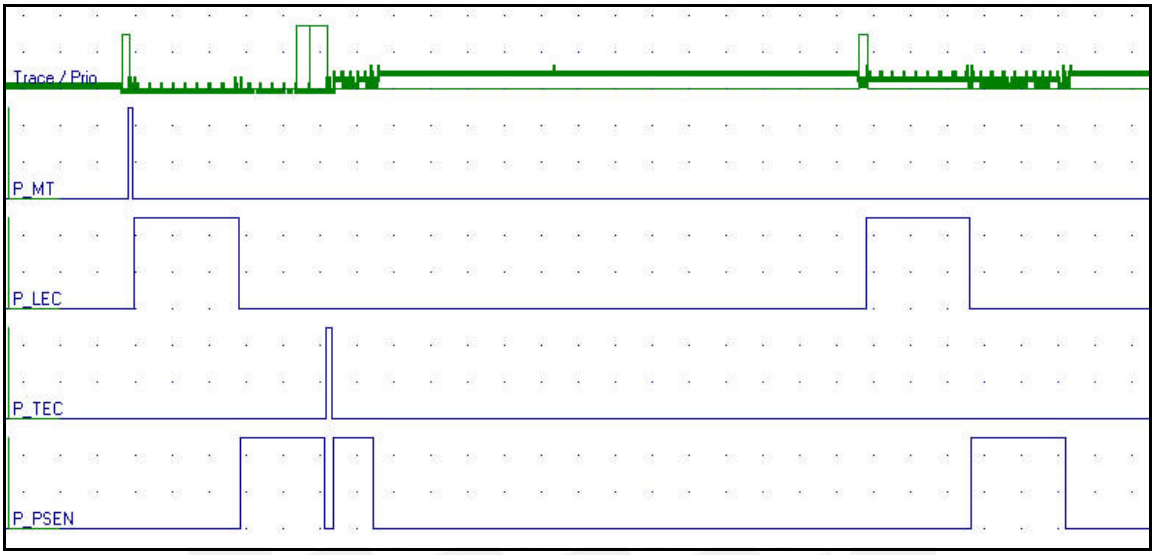


Figura 5.3. Activación de un proceso durante la ejecución de uno de menor prioridad.

Las gráficas anteriores muestran la naturaleza preemptiva de *GHOST*.

5.2.1 Tiempo de latencia de la interrupción

Probablemente la especificación más importante de un núcleo de tiempo real es la cantidad de tiempo en que las interrupciones están deshabilitadas. Mientras más tiempo estén deshabilitadas más alto será el tiempo de latencia de la interrupción [Lab02].

Tiempo de latencia de la interrupción = *MAX(Instrucción más larga, deshabilitación de las interrupciones por el programador, deshabilitación de las interrupciones por el núcleo)* + *Salto a la RSI* — Ecuación 5.7

La siguiente tabla muestra los valores para *GHOST*:

Característica	Número de ciclos	Tiempo
Instrucción más larga	4	0,267 µs
Deshabilitación de las interrupciones por el programador	1 520	101,333 µs
Deshabilitación de las interrupciones por el núcleo	776	51,733 µs
Salto a la RSI	11	0,733 µs

Tabla 5.6. Datos para el cálculo del tiempo de latencia de la interrupción.

Por lo tanto, para *GHOST* el tiempo de latencia de la interrupción es de **102,066 µs**.

5.2.2 Tiempo de respuesta de la interrupción

Está definido como el tiempo comprendido entre la recepción de la interrupción y el comienzo del código de la aplicación que maneja la interrupción [Lab02].

$$\begin{aligned} \textit{Tiempo de respuesta de la interrupción} &= \textit{Tiempo de latencia de la interrupción} \\ &+ \textit{Guardar el contexto de la CPU} \\ &+ \textit{Función del núcleo de entrada a la RSI} \end{aligned}$$

Ecuación 5.8

La siguiente tabla muestra los valores para *GHOST*:

Característica	Número de ciclos	Tiempo
Tiempo de latencia de la interrupción	1 531	102,066 µs
Guardar el contexto de la CPU	67	4,467 µs
Función del núcleo de entrada a la RSI	0	0,000 µs

Tabla 5.7. Datos para el cálculo del tiempo de respuesta de la interrupción.

Así, para *GHOST* el tiempo de respuesta de la interrupción es de **106,533 µs**.

Por lo tanto, el *jitter* o porcentaje de variación sobre el periodo del reloj de *GHOST*, el cual se ejecuta cada 4 ms, es de **2,663%**.

5.2.3 Tiempo de recuperación de la interrupción

Está definido como el tiempo requerido por el procesador para retornar al código donde se produjo la interrupción ó al código de un proceso de mayor prioridad, en caso se hubiera activado [Lab02].

$$\begin{aligned} \textit{Tiempo de recuperación de la interrupción} = {} & \textit{Tiempo de búsqueda del proceso de mayor prioridad} \\ & + \textit{Restauración del contexto del proceso de mayor prioridad} \\ & + \textit{Retorno de la interrupción} \end{aligned}$$

Ecuación 5.9

La siguiente tabla muestra los valores para *GHOST*:

Característica	Número de ciclos	Tiempo
Tiempo de búsqueda del proceso de mayor prioridad	211	14,067 µs
Restauración del contexto del proceso de mayor prioridad	83	5,533 µs
Retorno de la interrupción	4	0,267 µs

Tabla 5.8. Datos para el cálculo del tiempo de recuperación de la interrupción.

Así, para *GHOST* el tiempo de recuperación de la interrupción es de **19,867 µs**.

5.3 Calibración de la temperatura

Durante las pruebas se calibraron los sensores de temperatura. Primero se procedió a medir los voltajes que arrojaban los sensores a las distintas temperaturas a las que fue llevada la BAN. Para la medición de la temperatura se utilizó un termómetro digital, de 2 dígitos decimales de resolución, de la marca *Ertco-eutechnics* modelo *4400*. Luego se procedió a realizar una regresión lineal en el rango de trabajo.

En la Figura 5.4 se muestran los resultados de la calibración.

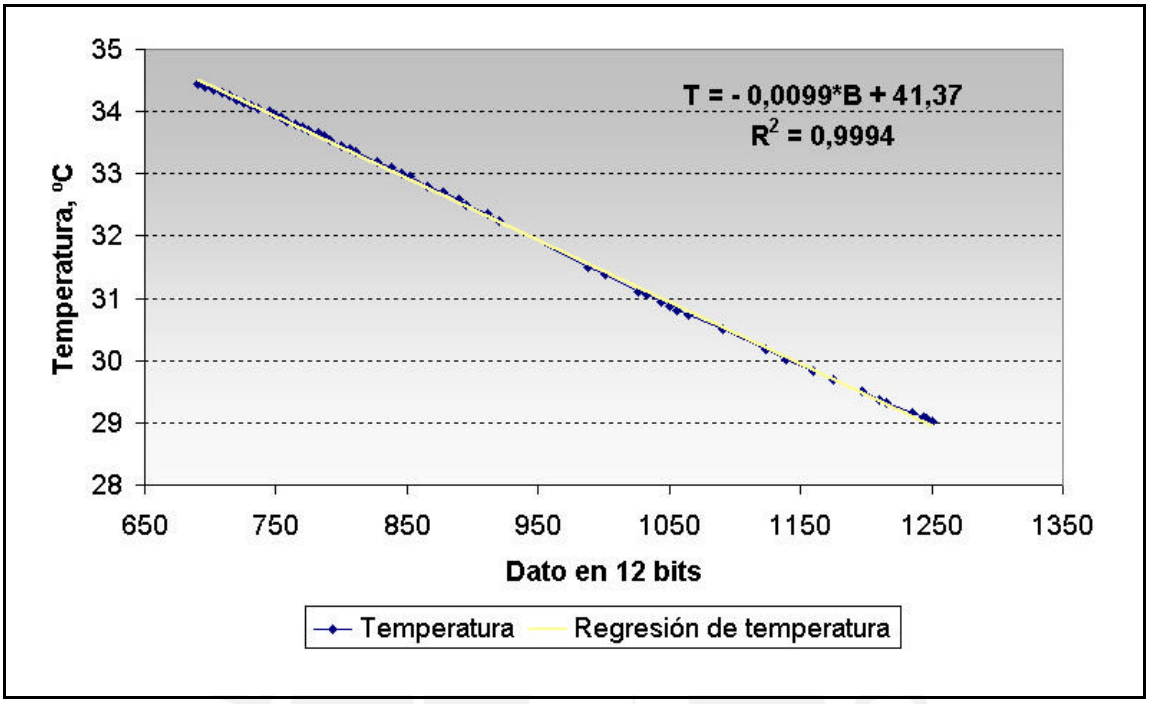


Figura 5.4. Recta de regresión de temperatura vs. dato en número natural de 12 bits.

Se obtuvo la recta de calibración de temperatura con coeficiente $R^2 = 0,9994$.

5.4 Filtro para las señales de flujo

Otro resultado fue la implementación de un filtro FIR (*Finite Impulse Response*) de Medias Móviles para filtrar las señales sensadas de flujo de oxígeno y aire. Se especificaron 256 coeficientes para el filtro; y entre sus principales características están su estabilidad y que tiene una respuesta de fase lineal, es decir, su respuesta tiene un retraso constante.

De las dos señales, la señal de flujo de aire es la que más necesita del filtro implementado debido a la característica de su actuador, que genera el flujo bombeando aire.

En la Figura 5.4 se muestra un cuadro comparativo entre la señal de flujo de aire antes del filtro y la señal filtrada. Se logró reducir el error de +/- 0,49 Lpm (litros por minuto) en el estado transitorio y el error de +/- 0,14 Lpm en el estado estable a un error de +/- 0,01 Lpm.

5.5 Resultados del control

Se realizaron pruebas relacionadas con el control de temperatura y el control de flujo de aire dentro de la BAN. No se realizaron las pruebas de los algoritmos de control de oxígeno ni de control de humedad relativa debido a que el hardware asociado a estos está todavía en proceso de implementación.

5.5.1 Control de flujo de aire

Las incubadoras tradicionales son incapaces de proporcionar un flujo de aire enriquecido con oxígeno al recién nacido, por lo que en los hospitales y clínicas se utiliza un equipo adicional denominado *blender* para realizar esta tarea.

Por este motivo, el control de flujo de aire es una característica importante dentro de BAN porque logra que se incorpore la funcionalidad del *blender* al equipo logrando una diferencia importante con las incubadoras tradicionales.

Para el flujo de aire, la estrategia de control que se implementó se basa en el error entre el valor sensado de flujo de aire y el valor programado (meta de control), si el valor sensado es mayor al valor programado se invoca al manejador de la bomba de aire para que decremente el flujo de aire, en caso contrario se le pide que lo incremente.

Se realizaron pruebas con diferentes valores de incremento/decremento de la palabra de control de la bomba de aire en cada ejecución del control, con la finalidad de determinar valores adecuados. Así también, durante las pruebas se determinó un valor adecuado para el periodo de ejecución del proceso de Control de Flujo de Aire.

Finalmente, el periodo del control se programó en 5 segundos y la palabra de control se definió en el rango de 1550 a 4050 con un incremento/decremento de 10.

Característica	Valor
Sobreimpulso	0 Lpm
Tiempo de establecimiento	6,5 minutos
Error en el estado estable	+/- 0,01 Lpm

Tabla 5.9. Características del control de flujo de aire.

En la Figura 5.5 se observa el control del flujo de aire para un valor programado de 2 Lpm.

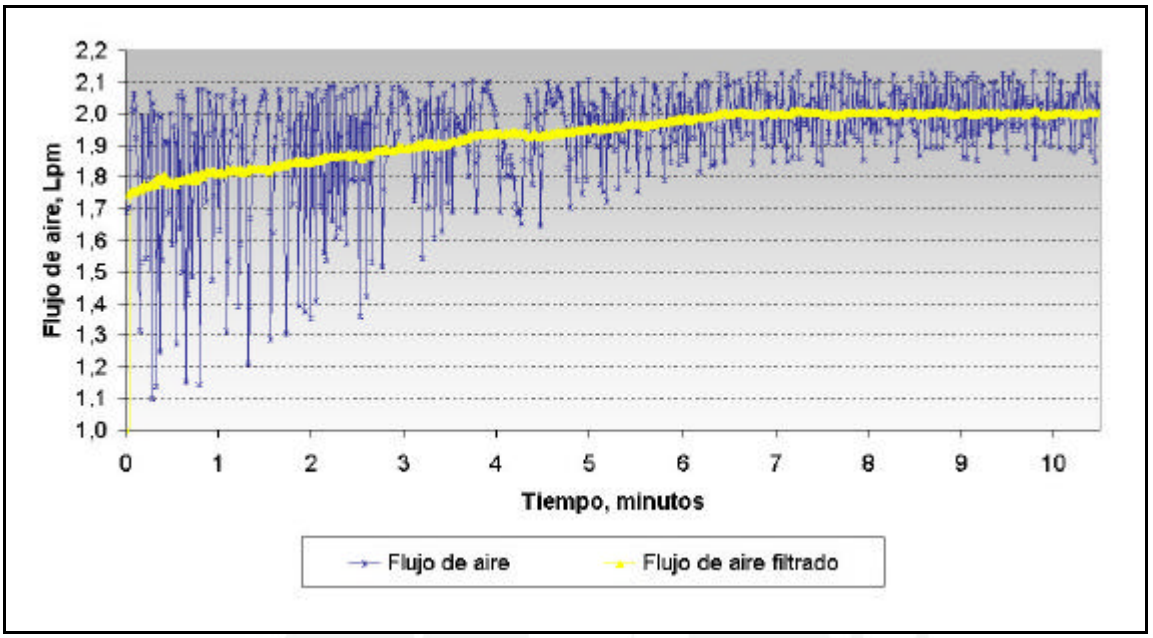


Figura 5.5. Flujo de aire: Señal sensada, filtrada y controlada.

5.5.2 Control de temperatura

Durante esta etapa se probaron diferentes estrategias de control con la finalidad de disminuir el tiempo de establecimiento de la temperatura y el error presentado respecto al valor programado. Durante las pruebas sólo se programó el control de temperatura en el circuito CCAT dejando el rendimiento de la onda PWM del calefactor del circuito CVC en 1%.

5.5.2.1 Control ON/OFF

La primera estrategia de control que se implementó se basa en el error entre el valor sensado de temperatura y el valor programado (meta de control), si el valor sensado es mayor al valor programado se disminuye el rendimiento de la onda PWM respectiva, en caso contrario se aumenta dicho rendimiento. Debido a que se programó un periodo de ejecución del control cada 8 ms, en 1 segundo ya se habían alcanzado los valores de 0% y 100% de la onda PWM según el caso, dejándonos con un control denominado ON/OFF.

En la Figura 5.6 se observan los resultados de este primer control que dio como resultados un sobreimpulso de 3 ºC, un tiempo de establecimiento de 2,3 horas y un error en el estado estable de +/- 0,68 ºC.

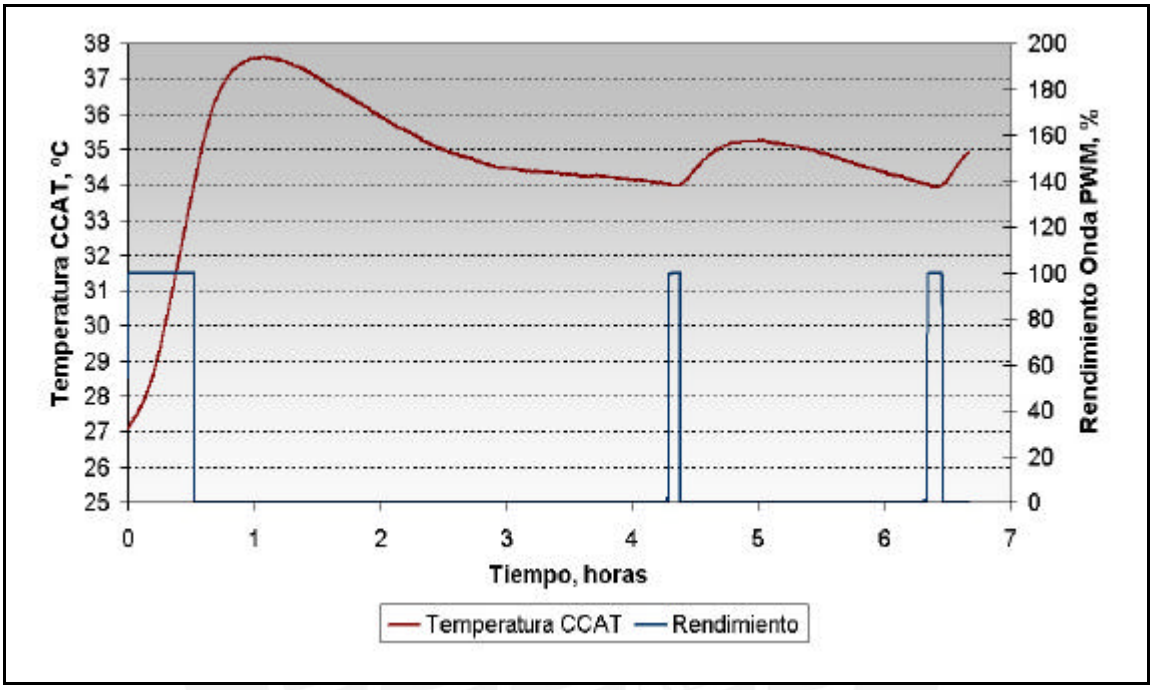


Figura 5.6. Primera prueba del control ON/OFF de temperatura.

Luego se modificó esta estrategia de control adicionando un tiempo inicial en el cual el rendimiento se mantenía en un cierto porcentaje. Se probaron valores de 25%, 50% y 100%. Con los valores de 25% y 50% si bien el sobreimpulso era menor, el tiempo de establecimiento aumentaba; por lo que se optó en dejar el rendimiento inicial en un 100% en los primeros 40 minutos.

En la Figura 5.7 se observan los resultados de esta primera modificación que dio como resultados un sobreimpulso de 3,47 ºC, un tiempo de establecimiento de 2,5 horas y un error en el estado estable de +/- 0,7 ºC. Sin embargo, los resultados no mejoraron debido a que en los primeros 40 minutos iniciales ya se había alcanzado la meta de control.

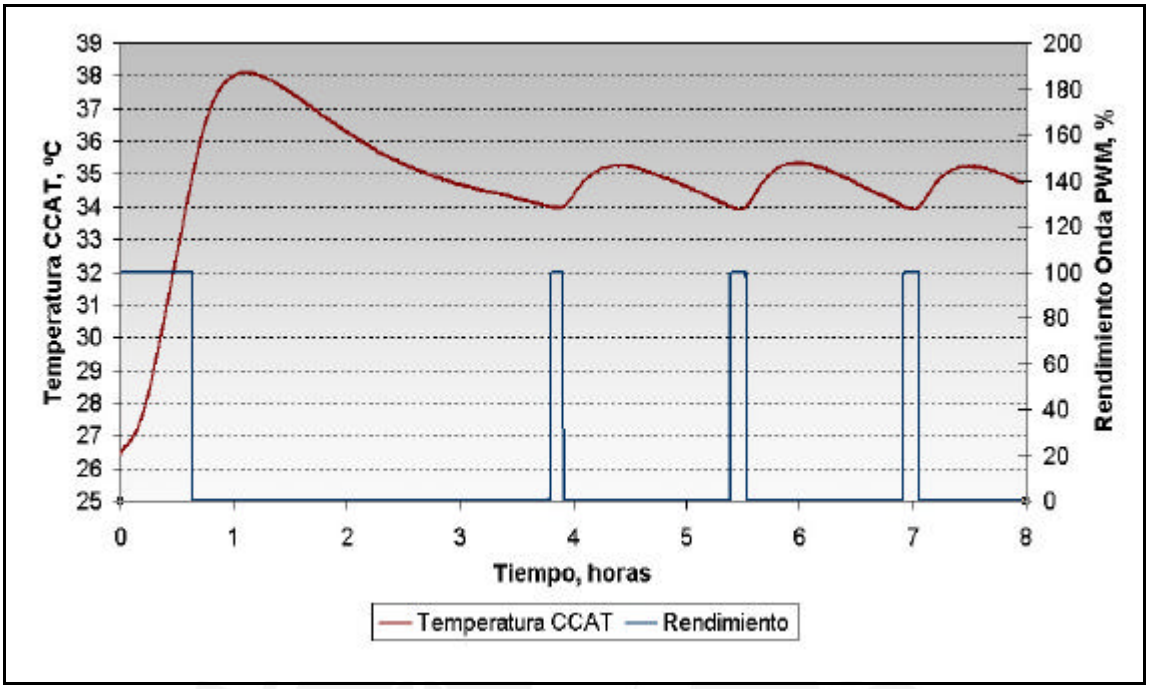


Figura 5.7. Segunda prueba del control ON/OFF de temperatura.

Teniendo como base esta experiencia se decidió tratar el tiempo inicial como una cantidad variable. En este tiempo inicial el rendimiento se mantenía en 100% hasta llegar a tener una diferencia de 2,5 ºC con la meta de control, cantidad basada en los valores de los sobreimpulsos observados durante las pruebas. Luego se mantenían los calefactores apagados durante 15 minutos, con la finalidad de aprovechar la inercia térmica y lograr un menor sobreimpulso.

En la Figura 5.8 se observan los resultados de esta segunda modificación que dio como resultados un sobreimpulso de 0,96 ºC, un tiempo de establecimiento de 1,17 horas y un error en el estado estable de +/- 0,75 ºC. Notablemente superiores a los primeros obtenidos.

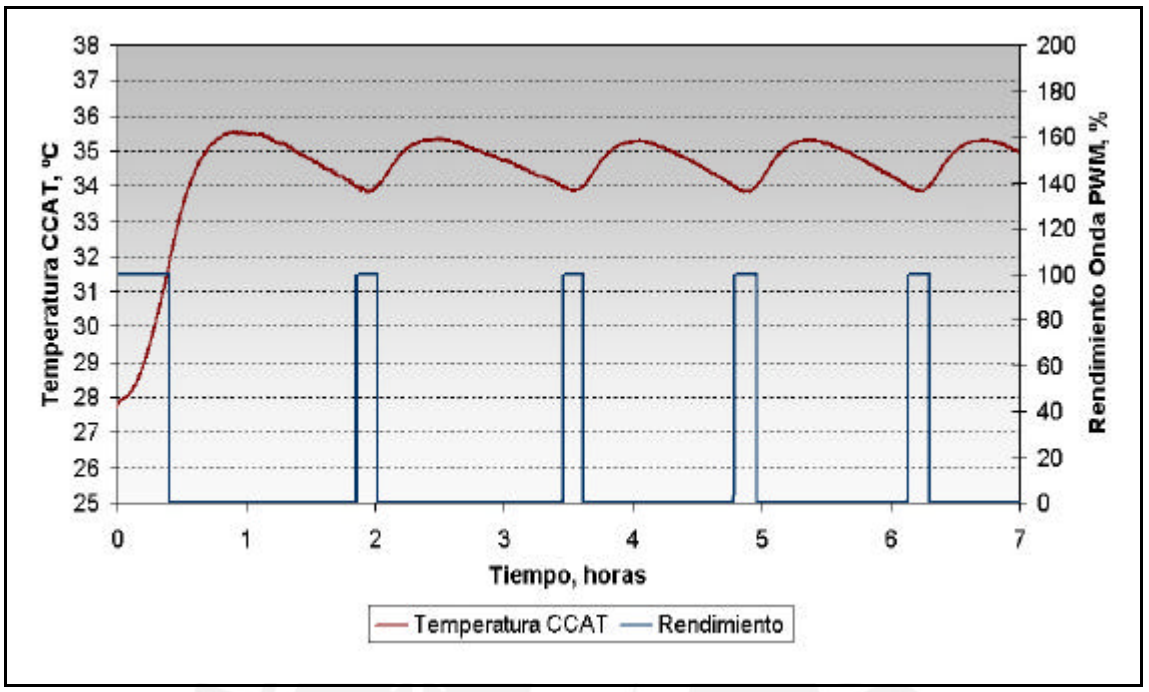


Figura 5.8. Tercera prueba del control ON/OFF de temperatura.

5.5.2.2 Control proporcional

La segunda estrategia que se implementó fue la de un control proporcional, la cual se basa también en el error entre el valor sensado de temperatura y el valor programado (meta de control). En el caso de que se esté por debajo de la meta de control, el rendimiento de la onda PWM se obtiene al dividir el error entre el rango de temperatura, en el caso contrario simplemente se le asigna cero. Además, se definió el incremento del rendimiento de la onda en pasos de 3,3% debido a la limitación de las operaciones con números de 16 bits (el porcentaje es equivalente a 8 ms según la configuración del sistema, como se muestra en la Figura 5.9). En la siguiente figura se muestra la onda del calefactor en el CCAT.

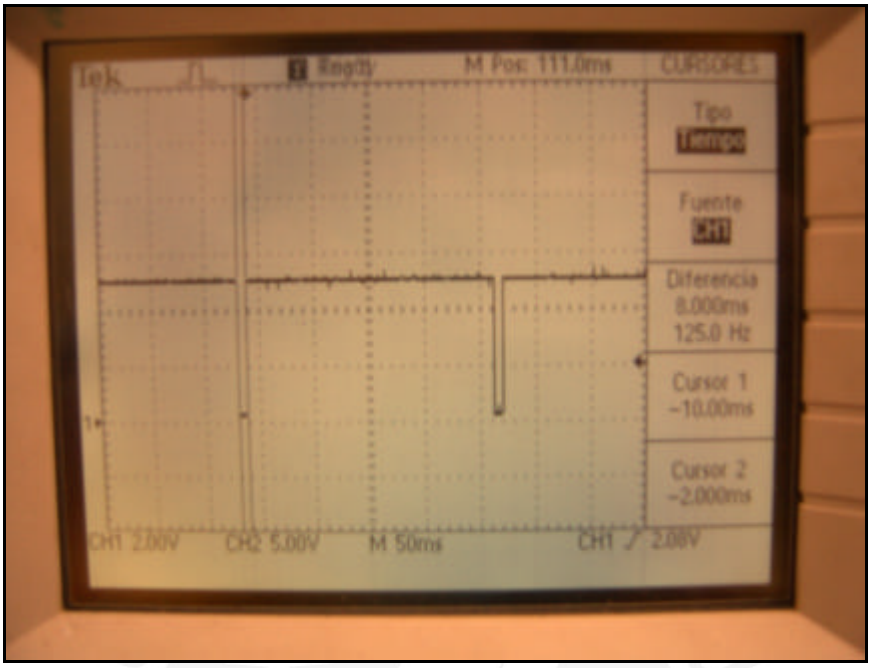


Figura 5.9. Onda PWM del calefactor CCAT (Rendimiento en lógica negativa).

En la Figura 5.10 se observan los resultados de este primer control proporcional, que dio como resultados un tiempo de establecimiento de 3 horas y un error en el estado estable de 0,14 ºC, mucho menor a los errores anteriores. Sin embargo, trajo consigo un efecto de desplazamiento (*offset*) de 1,6 ºC por debajo de la meta de control [Oga03].

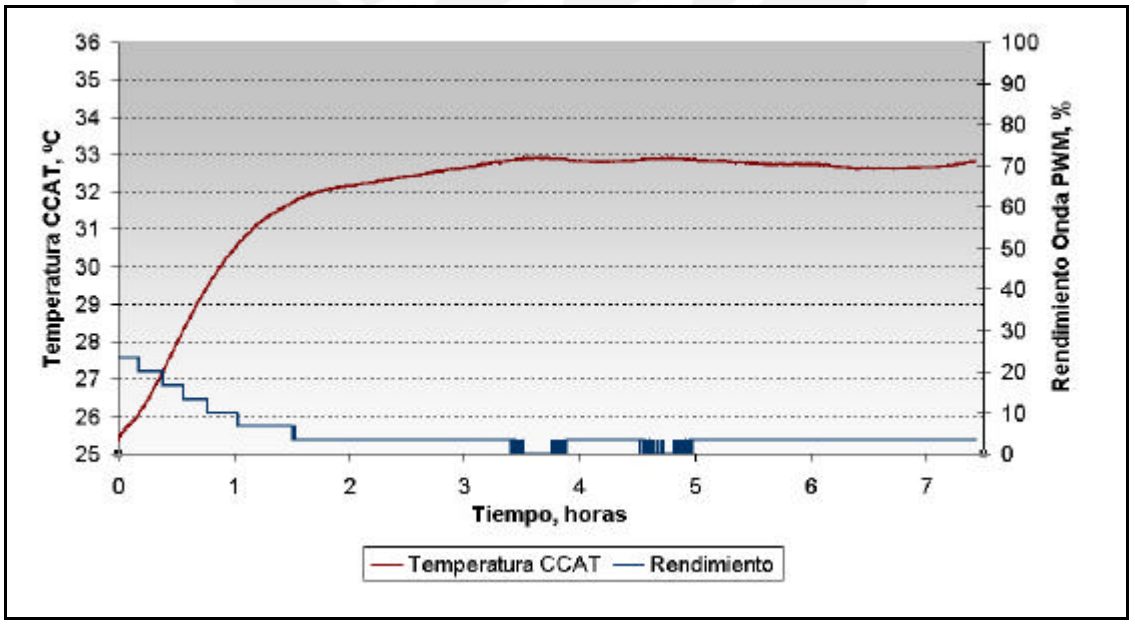


Figura 5.10. Primera prueba del control proporcional de temperatura.

Por este motivo y teniendo en consideración las experiencias anteriores también se decidió modificar el algoritmo de tal manera que se tenga un tiempo inicial de calentamiento al 100% del rendimiento de la onda PWM hasta que se alcancen la temperatura 2,5 ºC por debajo de la meta de control.

En la Figura 5.11 se observan los resultados de esta primera modificación, que dio como resultados un sobreimpulso de 1,12 ºC, un tiempo de establecimiento de 3 horas y un error en el estado estable de +/- 0,16 ºC. Sin embargo, aún se tenía un *offset* de 0,58 ºC por debajo de la meta de control.

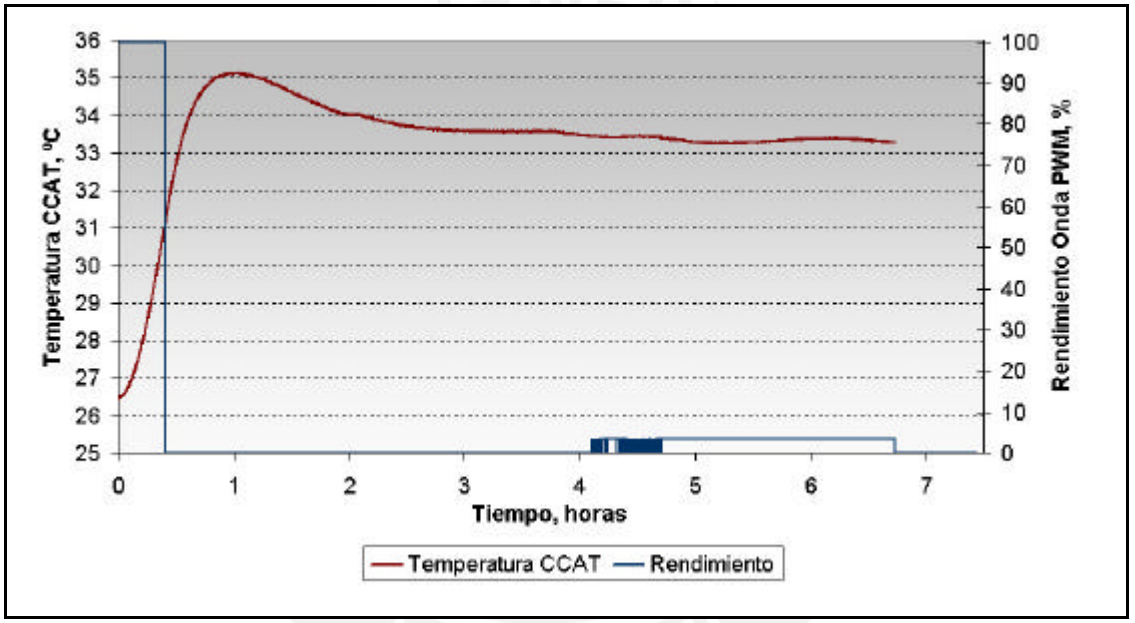


Figura 5.11. Segunda prueba del control proporcional de temperatura.

Finalmente, se llegó a la conclusión de que el *offset* anterior era debido a un error de cuantización al momento de hallar el rendimiento de la onda PWM debido principalmente a se está trabajando con aritmética entera. La solución que se le dio fue que cuando el cálculo de rendimiento de cómo resultado 0 (cero), se incremente en un paso el rendimiento de la onda, al estar el valor sensado por debajo del valor programado.

En la Figura 5.12 se observan los resultados de esta última modificación, que dio como resultados un sobreimpulso de 1,8 ºC, un tiempo de establecimiento de 2,4 horas, un error en el estado estable de +/- 0,13 ºC; y, la eliminación del *offset*.

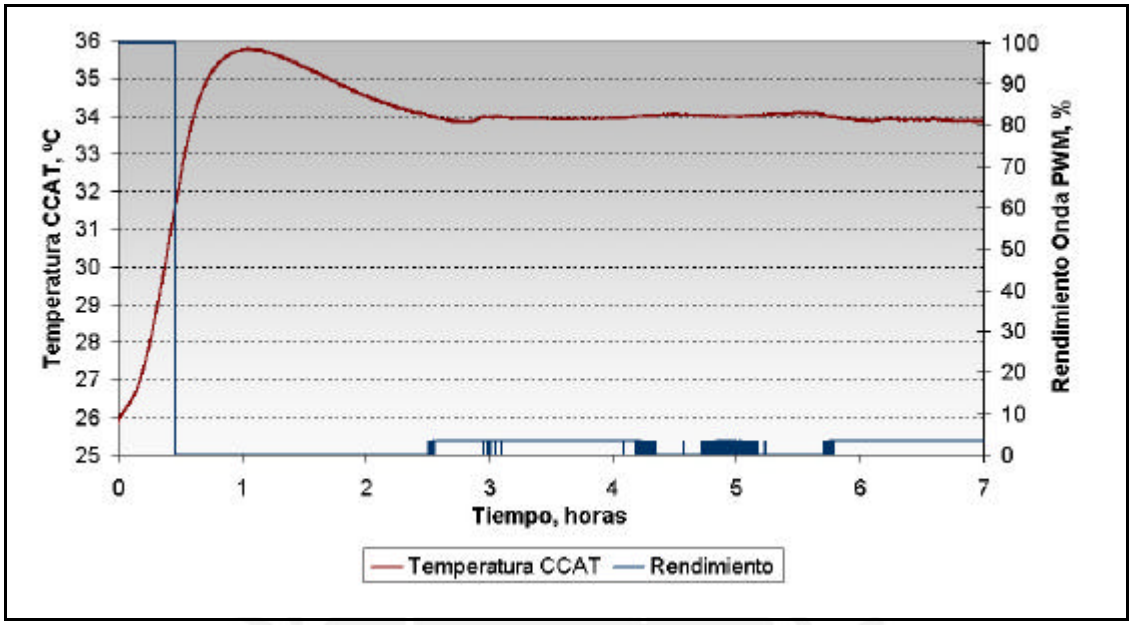


Figura 5.12. Tercera prueba del control proporcional de temperatura.

En la siguiente tabla se muestran las características finales del control de temperatura:

Característica	Valor
Sobreimpulso	1,8 ºC
Tiempo de establecimiento	2,4 horas
Error en el estado estable	+/- 0,13 ºC

Tabla 5.10. Características del control de temperatura.

Cabe resaltar que la temperatura de mando fue de 34 ºC, siempre superior en 3 ºC a la temperatura ambiente. Además, se debe resaltar que el valor del error es menor a +/- 0,5 ºC y que el valor del sobreimpulso es menor a 2 ºC; valores especificados por la norma internacional *IEC 60601-2-19: Medical electrical equipment. Part 2: Particular requirements for the safety of baby incubators*, de la *International Electrothecnical Commission* [IEC90].

5.6 Resultados adicionales

A continuación se presentan los resultados adicionales alcanzados.

- El módulo TEC implementa el protocolo de comunicación entre el ATmega 128 y el Módulo de Ingreso de Datos, procesando los comandos que le llegan.
- El módulo SON implementa el protocolo de comunicación entre el ATmega 128 y el Módulo de Sonidos. Envía los comandos de que sonido se desea escuchar, así como el volumen.
- El módulo VIS implementa el protocolo de comunicación entre el ATmega 128 y el Módulo de Visualización. Envía los comandos acerca del píxel que se desea dibujar, especificando su color, en alguna página de la memoria de video de dicho módulo. Maneja también la paginación de la memoria de video de la pantalla LCD.
- Desarrollo, en lenguaje de programación java, del programa ***Bitmap2ScenixFrame*** (Figura 5.13) capaz de pasar la información de la imagen en formato GIF al formato que entiende el microcontrolador Scenix del Módulo de Visualización.

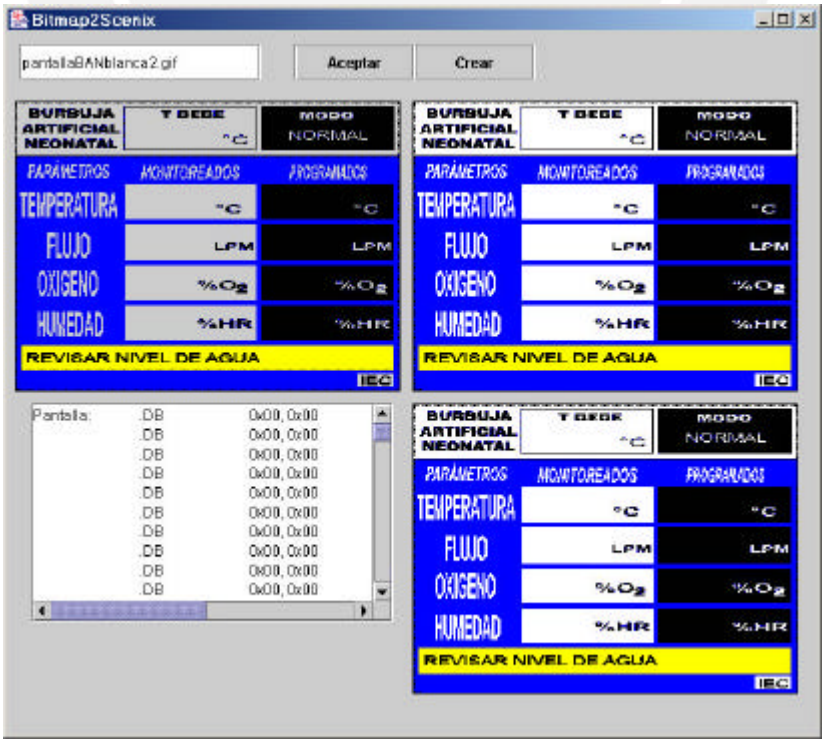


Figura 5.13. Programa Bitmap2ScenixFrame.

- Desarrollo, en lenguaje de programación Visual Basic, del programa ***CommSistemaBAN*** (Figura 5.14) que se comunica con el Sistema BAN a través del puerto serial de la computadora personal, recibe los datos que le envía, los visualiza y los guarda en un archivo con formato CSV.

Figura 5.14. Programa CommSistemaBAN.

- Desarrollo, en lenguaje de programación C, del programa ***RTA_DMPO*** (Figura 5.15) que realiza el análisis de tiempo de respuesta para un conjunto de procesos en un sistema multitarea.

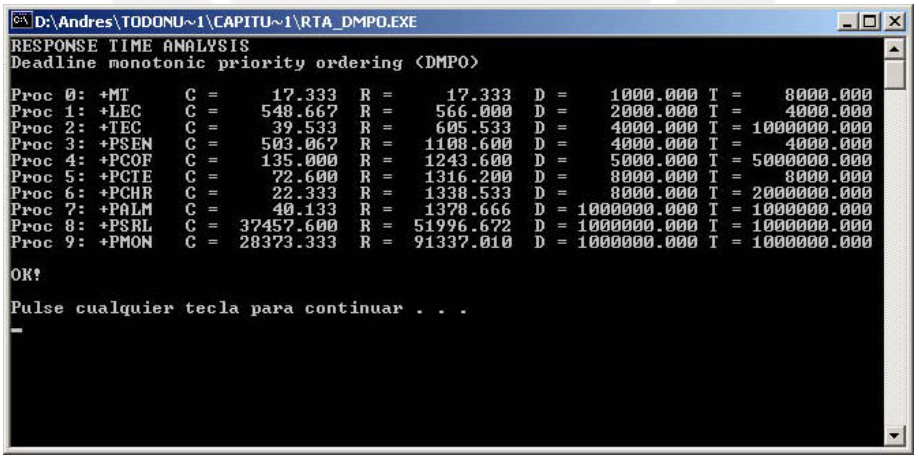


Figura 5.15. Programa RTA_DMPO.

- Se estructuró una **metodología** preliminar para el desarrollo de software de sistemas embebidos de tiempo real. La que se implementó a lo largo del desarrollo de esta tesis.

Figura 5.16. Burbuja Artificial Neonatal.

6 CONCLUSIONES Y RECOMENDACIONES

Introducción

Todos los objetivos propuestos en este trabajo fueron cumplidos plenamente. El desarrollo del software del sistema embebido de la Burbuja Artificial Neonatal (Fase Experimental), fue realizado de forma satisfactoria y exitosa. Se ha demostrado que el sistema BAN es un sistema de tiempo real planificable que cumple tanto con su especificación funcional como con su especificación no funcional.

6.1 Conclusiones

6.1.1 Respecto al Sistema BAN

Al iniciarse el desarrollo no se tenían experiencias anteriores en el desarrollo de sistemas similares, por lo que no se tenía un dato aproximado de cuánta memoria se iba a necesitar, ni se sabía cuánta capacidad de procesamiento se iba a requerir para cumplir con los requerimientos temporales. Se terminó usando el 8,8% de la memoria Flash, del microcontrolador ATmega 128, para los códigos del Sistema BAN y el 29,3% de la misma para datos. El caso de la memoria SRAM sí fue crítico, se terminó usando el 100% de esta memoria.

En lo que respecta a la capacidad del procesador utilizada, ésta fue del 34,014% del total. Lo que nos permite pensar, sin poner en riesgo la planificabilidad del sistema, en la implementación de estrategias de control más sofisticadas mediante el uso de números en punto flotante o enteros escalados en las siguientes etapas del Proyecto Burbuja Artificial Neonatal.

Por otro lado, lo que demandó más tiempo fue la implementación en lenguaje ensamblador, el cual se eligió por los motivos expuestos. Sin embargo, haber desarrollado el Sistema BAN en este lenguaje permitió la codificación de rutinas eficientes tanto en espacio como en tiempo. Especialmente, se tuvo control de este último parámetro en todo momento. Cabe resaltar también, que es posible la reutilización del núcleo *GHOST*, así como las rutinas del Nivel 2 del Sistema BAN que se consideren necesarias en futuros desarrollos.

Así mismo, se postula que sistemas de esta naturaleza, con la preponderancia de procesos periódicos, pueden ser resueltos estrictamente mediante el uso de *timers* (*OS_dormir()*) y el uso de memoria compartida con acceso protegido (*OS_ENTRAR_SECCION_CRITCA()* y *OS_SALIR_SECCION_CRITICA()*). Sólo se hace necesario el uso de mensajes en el caso de presencia de procesos esporádicos; o, en el caso de que un proceso necesite activar a dos o más procesos que necesiten llevar la cuenta del tiempo que transcurre simultáneamente.

6.1.2 Respecto a *GHOST*

El uso del núcleo *GHOST* permitió el uso eficiente de un único procesador. La duración de la rutina del planificador es muy pequeña, 51,733 µs, lo que representa un porcentaje de utilización del procesador de 1,293% para un *tick* de 4 ms. Además, cuenta con un valor bajo de *jitter*, 2,663%, lo que mejora la precisión del sistema.

Asimismo, se considera de mucha importancia tener un núcleo como *GHOST*, que ayudará a reducir el tiempo de desarrollo de futuros equipos que se proyectan implementar en el GIDEMS, como es el caso del equipo CPAP (*Continuous Positive Airway Pressure*) para neonatos, o de desarrollos de otros grupos de investigación.

6.1.3 Respecto al control

En lo que respecta al control de flujo de aire, el algoritmo de control implementado no presenta sobreimpulso respecto a la meta de control. El tiempo de establecimiento logrado fue de 6,5 minutos, con un error mínimo en el estado estable, el cual fue de +/- 0,01 Lpm.

Cabe resaltar que debido a la variabilidad de la señal de flujo de aire, la utilización del filtro digital fue fundamental en el proceso de control.

Para el caso del control de temperatura, se probaron la estrategia de control ON/OFF y la de control proporcional. Además, se realizaron una serie de modificaciones con la finalidad de mejorar sus características.

Bajo los resultados arrojados por las pruebas que se realizaron, el control ON/OFF proporcionaba magnitudes menores para el sobreimpulso, 0,96 ºC, y para el tiempo de establecimiento, 1,17 horas. Esto se debe al hecho de que en el inicio se le proporcionaba más energía al sistema, con lo que se llegaba más rápido a la meta de control y, por ende, la temperatura se establecía más rápido. Sin embargo, esta estrategia de control presentaba un error en el estado estable de +/- 0,75 ºC, lo que es crítico debido a que las variaciones térmicas tienen implicancias en la salud del neonato de alto riesgo.

El control proporcional arrojó un error mucho menor en el estado estable con un valor de +/- 0,13 ºC; el cual es un valor menor al +/- 0,5 ºC especificado por la norma internacional IEC 60601-2-19. El sobreimpulso que arrojó, de 1,8 ºC, también es menor a los 2 ºC especificados en dicha norma. Aunque el tiempo de establecimiento, de 2,4 horas, es mayor al del primer control implementado, no se especifica ningún valor en dicha norma.

6.1.4 Respecto al proceso de desarrollo

Haber usado un simulador de software y hardware, como el VMLab, permitió reducir el tiempo de desarrollo del sistema de la BAN. Especialmente en la etapa de pruebas, donde se probó las funciones del Sistema BAN antes de bajarlos a las memorias del microcontrolador ATmega 128 para su final ejecución.

El desarrollo de la herramienta *Bitmap2ScenixFrame* facilitó enormemente el trabajo con imágenes para la pantalla LCD, al permitir la edición de éstas en un formato conocido (GIF) por cualquier editor de imágenes de computadora, debido a que puede transformar este formato al formato entendido por el Módulo de Visualización.

La implementación de una interfaz compatible con el puerto serial de una computadora personal ha aportado beneficios en cuanto al desarrollo, al servir de medio de transferencia de información entre el equipo y la PC. La herramienta *CommSistemaBAN* permitió también la automatización de la toma de datos durante la fase de pruebas, guardando la información que le enviaba la BAN en un archivo del tipo CSV (Comma Separated Value). Los datos recopilados eran luego analizados en su conjunto.

Además, la metodología de desarrollo de software de sistemas embebidos de tiempo real que se siguió puede ser tomada como base para la consolidación de una mucho más madura.

Finalmente, el presente trabajo sienta las bases para futuras investigaciones en el área de desarrollo de software de sistemas embebidos para equipos médicos de soporte de vida.

6.2 Recomendaciones

Habiéndose desarrollado el prototipo experimental de la BAN se recomienda continuar con el trabajo de investigación hasta lograr un equipo que pueda ser utilizado en las Unidades de Neonatología de los hospitales y/o clínicas. A continuación, se presentan las recomendaciones:

- En los esquemas como los de este equipo, en los que se usan más de un microcontrolador se recomienda siempre utilizar protocolos de comunicación bidireccionales de tal manera que se pueda enviar una confirmación de comando recibido, con lo que se podría implementar esquemas de seguridad que actúen según el estado en el que se encuentran los microcontroladores, dispositivos y demás componentes; con la finalidad de hacer más robusto el equipo.

- Se recomienda también, la implementación de una librería que permita trabajar con números reales, como una librería de punto flotante o una librería que trabaje con enteros escalados, con la finalidad de poder implementar estrategias de control más sofisticadas.

- Por otro lado, se recomienda implementar un esquema de auditoria capaz de guardar en la memoria no volátil la información asociada a todos los eventos relevantes en el equipo como lo son: cambios en la programación del equipo, activación de alarmas, etc. (p.e. fecha y hora de la ocurrencia del evento). Además, estos datos podrían ser transmitidos a una computadora y almacenados en una base de datos, con el fin de facilitar un programa de tecnovigilancia.

- Finalmente, al ser la confiabilidad un tema clave en un equipo médico, sobretodo si se trata de uno de soporte de vida, se recomienda **fuertemente** iniciar cada nuevo desarrollo con un análisis de riesgos, como lo recomienda la norma internacional *IEC601-1-4: Medical Electrical Equipment. Part 1: General requirements for safety. 4. Collateral Standard: Programmable electrical medical systems*, de la *International Electrothecnical Commission*. Con ésta finalidad, se podrían usar métodos como el *Fault Tree Analysis (FTA)* o el *Failure Mode and Effect Analysis (FMEA)*.

7 BIBLIOGRAFÍA

[A&T89] David Auslander y Cheng Tham, *Real-time software for control: program examples in C*. 1989.

[ASL05] Aerospace Software Ltd., *The Very Basics on FIR Filters*. 2005. URL disponible en: http://www.aerospacesoftware.com/fir.htm

[Atm04] Atmel Corporation, *Hoja de datos del microcontrolador ATmega 128*, Rev. 2467M-AVR-11/04. Noviembre 2004.

[Atm05] Atmel Corporation. 2005. URL disponible en: http://www.atmel.com/

[Aud93] N. Audsley, A. Burns, M. Richardson, K. Tindell y A.J. Wellings, *Applying new scheduling theory to static priority pre-emptive scheduling*. Software Engineering Journal, 8(5), 284-92. 1993.

[B&W96] Alan Burns y Andy Wellings, *Real-Time Systems and Programming Languages*. Segunda Edición 1996.

[Bar05] Richard Baraniuk, *Practical Filters*. 2005. URL disponible en: http://cnx.rice.edu/content/m10126/latest/

[Beu03] Danilo Beuche, *Composition and Construction or Embedded Software Families, Dissertation thesis*. Diciembre 2003. URL disponible en: http://wwwivs.cs.uni-magdeburg.de/~danilo/diss-lyx-online.pdf

[C&A04] Bruno Castillón y Eduardo Ajito, *Patent Number EP1380276. Neonatal Bubble*. Enero 2004.

[ENP04] Europe's Network of patent databases. 2004. URL disponible en: http://es.espacenet.com

[Eve05] EventHelix.com Inc., *Real-time and embedded software design*. 2005. URL disponible en: http://www.eventhelix.com/RealtimeMantra/Basics

[IEC90] International Electrothecnical Commission, *International Standard IEC601-2-19: Medical electrical equipment. Part 2: Particular requirements for safety of baby incubators*. Primera Edición 1990.

[IEC96] International Electrothecnical Commission, *International Standard IEC601-1-4: Medical Electrical Equipment. Part 1: General requirements for safety. 4. Collateral Standard: Programmable electrical medical systems*. Primera Edición 1996.

[Iri05] Andoni Irizar, *Tratamiento Digital de Señales. Diseño de Filtros Digitales*. Escuela Superior de Ingenieros Tecnum Universidad de Navarra. 2005. URL disponible en: http://www.tecnun.com/asignaturas/tratamiento%20digital/TEMA9/sld001.htm

[J&P86] M. Joseph y P. Pandya, *Finding response times in a real-time system*. BCS Computer Journal, 29(5), 390-5. 1986.

[L&L73] C. Liu y J. Layland, *Scheduling algorithms for multiprogramming in a hard real-time environment*. JACM, 20(1), 46-61. 1973.

[L&W82] J. Leung y J. Whitehead, *On the complexity of fixed-priority scheduling of periodic, real-time tasks*. Performance Evaluation (Netherlands), 2(4),237-50. 1982.

[Lab02] Jean Labrosse, *MicroC/OS-II The Real-Time Kernel*. Segunda Edición 2002.

[Low05] Lowegian's dspGuru, *Digital Signal Processing Information*. 2005. URL disponible en: http://dspguru.com/info/faqs/

[Oga03] Katsuhiko Ogata, *Ingeniería de Control Moderna*. Cuarta Edición 2003.

[OMS98] Organización Mundial de la Salud, *El Desarrollo de la Evaluación de las Tecnologías en Salud en América Latina y el Caribe*. 1998.

[Oxf96] Oxford, *Dictionary of Computing*. 1996.

[RAE05] Real Academia Española, *Diccionario de la lengua española*. Vigésima segunda Edición. 2005. URL disponible en: http://www.rae.es

[San05] Norberto Santos et al., Actualización de temas neonatales Tomo I. Asociación Argentina de Perinatología. 2005.

[T&W98] Andrew Tanenbaum y Albert Woodhull, *Sistemas Operativos: Diseño e implementación*. Segunda Edición 1998.

[Wik05] WIKIPEDIA The Free Encyclopedia. 2005. URL disponible en: http://en.wikipedia.org

[Wor05] WorNet (r) 1.7, *Diccionario en línea*. 2005. URL disponible en: http://dict.die.net/alpha%20software

8 ANEXOS

Anexo A. Glosario de Términos

Patente BAN: Patente Burbuja Artificial Neonatal.

BAN: Así se le denomina al aparato construido sobre la base de la patente BAN.

Sistema BAN: Software del sistema embebido de la BAN.

Circuito Cerrado de Aire Temperado (CCAT): encargado de conservar y mantener uniforme la temperatura en el ambiente artificial intermedio mediante un calefactor y un ventilador que generan un flujo de aire temperado que se usa como medio de propagación de calor. Este circuito no está en contacto con el neonato, atributo que permite instalar filtros acústicos para reducir el ruido.

Circuito Ventilatorio Continuo (CVC): Conjunto de dispositivos neumáticos conectados consecutivamente para ventilar al neonato con un flujo continuo de aire filtrado, oxigenado, temperado y humedecido La cantidad de este gas es regulada según los requerimientos de cada neonato lo cual permite utilizar menor cantidad de oxígeno y mayor tiempo los filtros bacterianos.

Modo Monitor: Modo en el que se encuentra normalmente la BAN, monitorizando el sistema. No permite la modificación de ningún valor programado debido a que el teclado está bloqueado.

Modo de Programación: Similar al Modo Monitor con la diferencia que se puede cambiar el valor programado de alguna variable.

Duty Cycle (ciclo de trabajo): La razón entre la duración de un pulso y el periodo del pulso.

Habitáculo del bebé: La parte del equipo destinada a alojar al bebé.

Anexo B. Patente BAN

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 380 276 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.01.2004 Bulletin 2004/03

(51) Int Cl.7: **A61G 11/00**

(21) Application number: **03356111.9**

(22) Date of filing: **11.07.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR
Designated Extension States:
AL LT LV MK

(30) Priority: **12.07.2002 PE 62202**

(71) Applicant: **Pontificia Universidad Catolica del Peru**
San Miguel (PE)

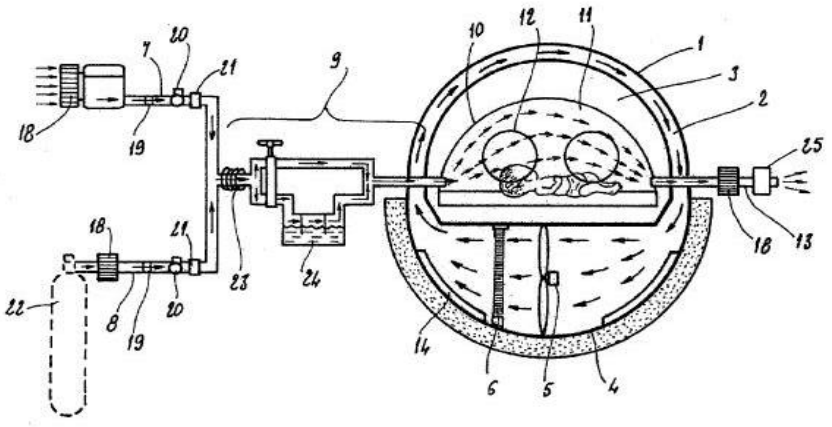
(72) Inventors:
- **Castillon Levano, Claudio Bruno**
  San Juan de Miraflores, Lima (PE)
- **Ajito Lam, Eduardo**
  Bellavista, Callao (PE)

(74) Representative: **Maureau, Philippe et al**
Cabinet GERMAIN & MAUREAU,
BP 6153
69466 Lyon Cedex 06 (FR)

(54) Neonatal bubble

(57) The present invention relates to a medical appliance for improving the intensive care of high-risk newborns, said appliance being characterized in that it comprises: firstly, a tempered air closed circuit (1, 4) enclosing a neonatal capsule (10) and comprising a dome (1) composed of two concentric layers defining an intradome space (2) therebetween, through which tempered air can circulate, to maintain the temperature in the intermediate artificial environment (3) created between the neonatal capsule (10) and the tempered air closed circuit (1, 4), and secondly, a continuous ventilation circuit (4) comprising air and oxygen inlet lines (7, 8) and a mixture outlet line ,for administrating a continuous and regulated air flow of filtered, oxygenated, tempered and humidified air to the newborn child inside the neonatal capsule (10), and thirdly, a set of doors (12, 15, 16) through said neonatal capsule and dome, providing access inside said neonatal capsule.

FIG 1



EP 1 380 276 A1

Printed by Jouve, 75001 PARIS (FR)

Anexo C. Clasificación de Alarmas

Las alarmas previstas para el Sistema BAN son:

Temperatura

T0 - Exceso de temperatura en la cúpula interna (Rango de trabajo: de 25 a 37 ºC).

T1 - Defecto de temperatura en la cúpula interna (Rango de trabajo: de 25 a 37 ºC).

T2 - Exceso de temperatura en la cúpula externa (Rango de trabajo: de 25 a 37 ºC).

T3 - Defecto de temperatura en la cúpula externa (Rango de trabajo: de 25 a 37 ºC).

T4 - Exceso de temperatura en la piel del bebé (Rango de trabajo: de 34 a 37 ºC).

T5 - Defecto de temperatura en la piel del bebé (Rango de trabajo: de 34 a 37 ºC).

Humedad Relativa

H0 - Exceso de humedad relativa en el CVC (Rango de trabajo: de 40 a 90%).

H1 - Defecto de humedad relativa en el CVC (Rango de trabajo: de 40 a 90%).

Mezcla de Gases

G0 - Exceso del flujo de oxígeno (Rango de trabajo: de 0 a 5 Lpm).

G1 - Exceso del flujo de aire (Rango de trabajo: de 1 a 5 Lpm).

G2 - Defecto del flujo de aire (Rango de trabajo: de 1 a 5 Lpm).

G3 - Exceso de flujo en la vía de la mezcla (Rango de trabajo: de 1 a 5 Lpm).

G4 - Defecto de flujo en la vía de la mezcla (Rango de trabajo: de 1 a 5 Lpm).

Anexo D. Convención de llamada a función

A continuación, se presenta la convención de la llamada a una función en el Sistema BAN.

Dentro de la función se apilan los valores de los registros que se utilizarán como variables locales y antes del retorno de función se desapilan nuevamente. El cuerpo de una función puede lucir como el siguiente ejemplo:

```
;-------------------------------------------------------------------
; PSEN_convertir12BitsAUnidadesTempe

PSEN_convertir12BitsAUnidadesTempe:
    ; PARAMETROS DE ENTRADA
    ;   r16: LSB Dato sensado de temperatura en 12 bits.
    ;   r17: MSB Dato sensado de temperatura en 12 bits.

    ; PARAMETROS DE SALIDA
    ;   r18: Dato sensado de temperatura en grados celsius.

    ; VARIABLES LOCALES
    ;   r24: Auxiliar. LSB Dato.
    ;   r25: Auxiliar. MSB Dato.
    push    r24
    push    r25
;*
;* Inicio
;*


; Código ...


; mover el resultado al parámetro de salida r18
    mov     r18,r24

;*
;* Fin
;*
    pop     r25
    pop     r24

    ret
```

Es en el cuerpo de la función que la va a llamar donde se apilan los parámetros de entrada y salida, como se muestra en el siguiente ejemplo.

```
; 3 * Tbits
    push    r16
    push    r17
    push    r20
    push    r21
    push    r22
    push    r23

    movw    r21:r20 ,r17:r16
    ldi     r22     ,low (3)
    ldi     r23     ,high(3)
    ; Param. entrada: r23:r22 * r21:r20 | Param. salida: r17:r16(resultado)
    call    LIB_mul16u
    movw    r25:r24 ,r17:r16

    pop     r23
    pop     r22
    pop     r21
    pop     r20
    pop     r17
    pop     r16
```

Anexo E. Algoritmo para el Análisis de Tiempo de Respuesta

A continuación, se presenta el algoritmo para el análisis de tiempo de respuesta.

```
void main(){
    int i = 0, n = 0;
    long double Wn = 0, Wn1 = 0;

    leerEntrada();

    for (i = 0; i < iNumProc; i++) {
        n = 0;
        Wn = C[i];

        while (1) {

            Wn1 = calcularWn1(i, Wn);

            if ( abs(Wn1-Wn) <= PRECISION ) {
                R[i] = Wn;
                break;
            }

            if ( Wn1 > D[i] ) {
                R[i] = -1;
                break;
            }

            Wn = Wn1;
            n++;
        } // fin while
    } // fin for

    imprimir();
    system("PAUSE");

} // fin main


long double calcularWn1( int p_i, long double p_Wn) {
    int j = 0;
    long double Wn1 = 0;

    for (j = 0; j < p_i; j++) {
        Wn1  = Wn1 + ceill(p_Wn/T[j])*C[j];
    }

    Wn1 = C[p_i] + Wn1;

    return Wn1;
}
```

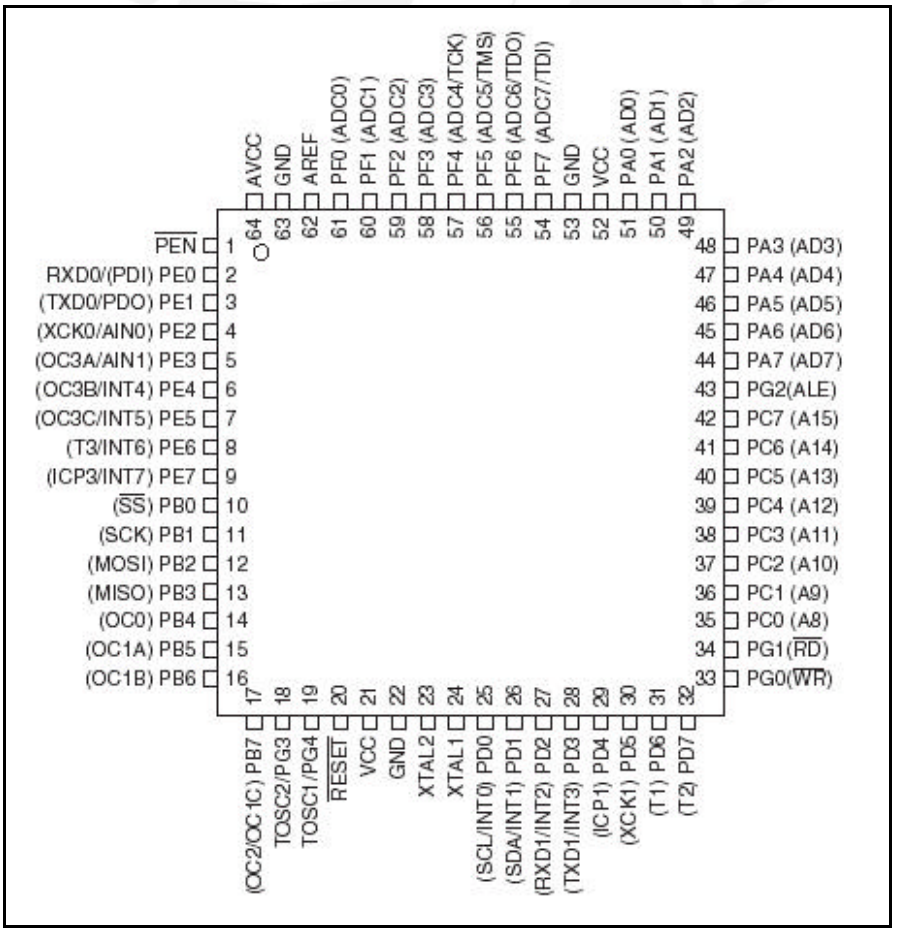
Anexo F. Fases del Proyecto BAN

1era Fase | Prototipo Experimental (BAN Alfa)
Objetivo: Desarrollar un prototipo experimental del aparato BAN basado en la patente.
Usuarios Finales: Personal Médico
Norma: IEC 601-2-19
Etapas: Captura de requerimientos Diseño del sistema Construcción del sistema Prueba del desarrollo
2da Fase | Prototipo Funcional (BAN Beta)
Objetivo: Desarrollar el prototipo funcional del aparato BAN basado en el prototipo experimental y su rediseño.
Usuarios Finales: Neonato de Alto Riesgo y Personal Médico
Norma: IEC 601-2-19
Etapas: Captura de requerimientos Diseño del sistema (regido por estándares internacionales) Construcción del sistema Prueba del desarrollo Prueba por parte del usuario Presentación del sistema
3era Fase | Producto (BAN Versión 1)
Objetivo: Desarrollar un prototipo de producto del Aparato BAN.
Usuarios Finales: Neonato de Alto Riesgo y Personal Médico.
Norma: IEC 601-2-19.
Etapas: Captura de requerimientos Diseño del sistema (regido por estándares internacionales) Construcción del sistema Prueba del desarrollo Prueba por parte del usuario Presentación del sistema Certificación

Anexo G. Microcontrolador ATmega 128

El ATmega 128 es un microcontrolador de 8 bits de arquitectura RISC. Posee 32 registros de 8 bits, 1 registro de estado, 1 registro de puntero de pila, registros de configuración de los periféricos y registros X(r27:r26), Y, Z de 16 bits de propósito general, especialmente utilizados para acceso indirecto a la memoria (punteros).

El microcontrolador cuenta con memorias internas con espacios de direcciones separados: 128 KB de memoria Flash, 4 KB de memoria EEPROM y 4 KB de memoria SRAM. Además, cuenta con 4 interrupciones internas, 4 interrupciones externas; 2 temporizadores de 8 bits, 2 temporizadores de 16 bits, 1 contador en tiempo real y 1 temporizador *watchdog* programable. Cuenta también con 6 puertos de 8 bits y 1 de 5 bits configurables como E/S [Atm04].



Microcontrolador ATmega 128.

En la siguiente figura se muestra el diagrama de bloques de la arquitectura del microcontrolador ATmega 128.

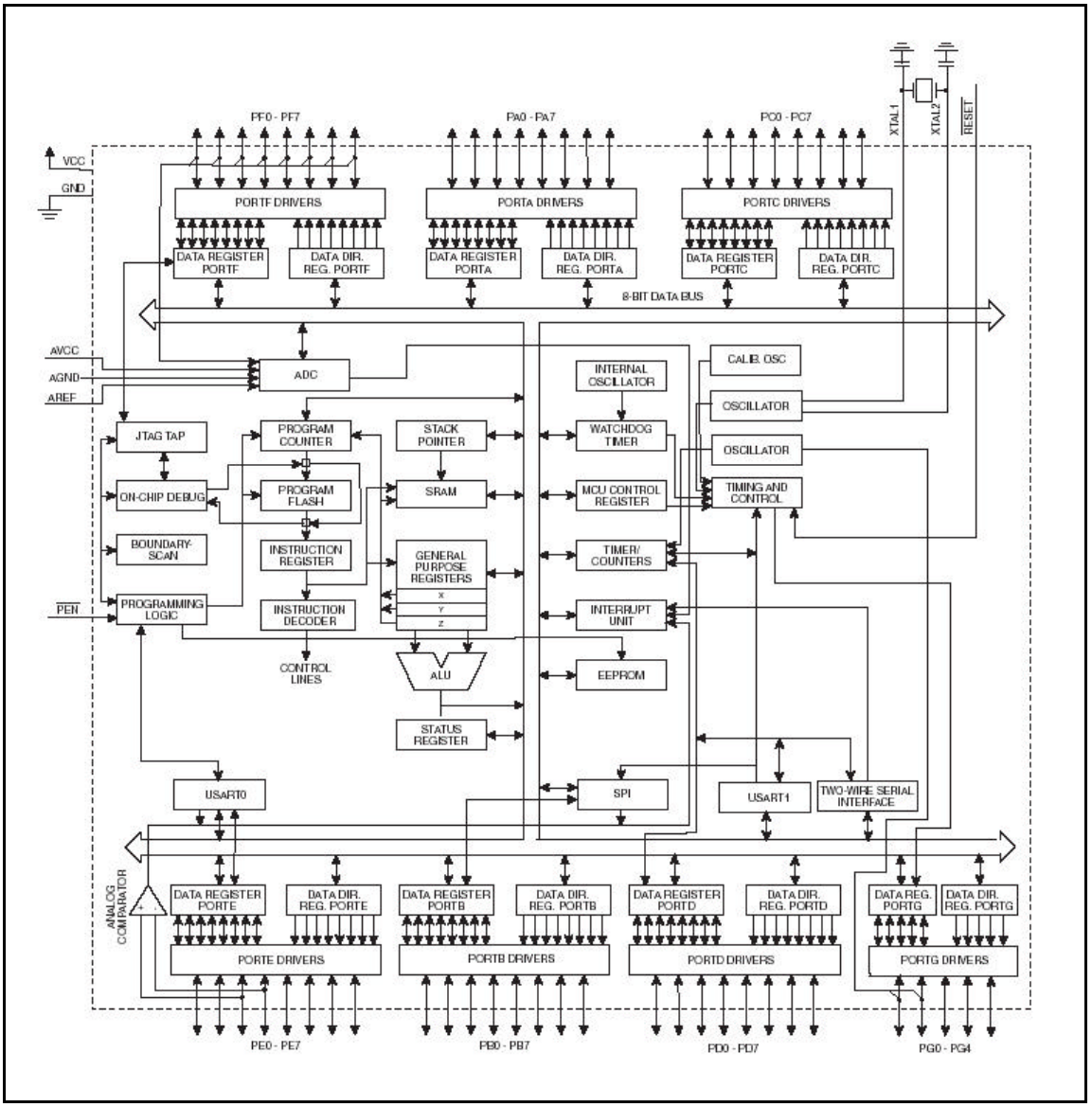


Diagrama de bloques del microcontrolador ATmega 128.

Anexo H. Descripción de las interfaces y pines del ATmega 128

En el archivo “E200 Interfaces - Descripción.pdf”, desarrollado por el GIDEMS, se muestra la descripción de las interfaces entre el Módulo de Procesamiento y Control y los demás módulos de la BAN.

Para cada una de las interfaces con los otros módulos, se definen los pines utilizados en cada puerto del ATmega 128, se define si es de entrada y/o salida, y se da una descripción de la función del pin.

Anexo I. Especificación de Casos de Uso del Sistema BAN

En el presente anexo se muestran a detalle las especificaciones de casos de uso. En las siguientes figuras se tienen los diagramas de casos de uso:

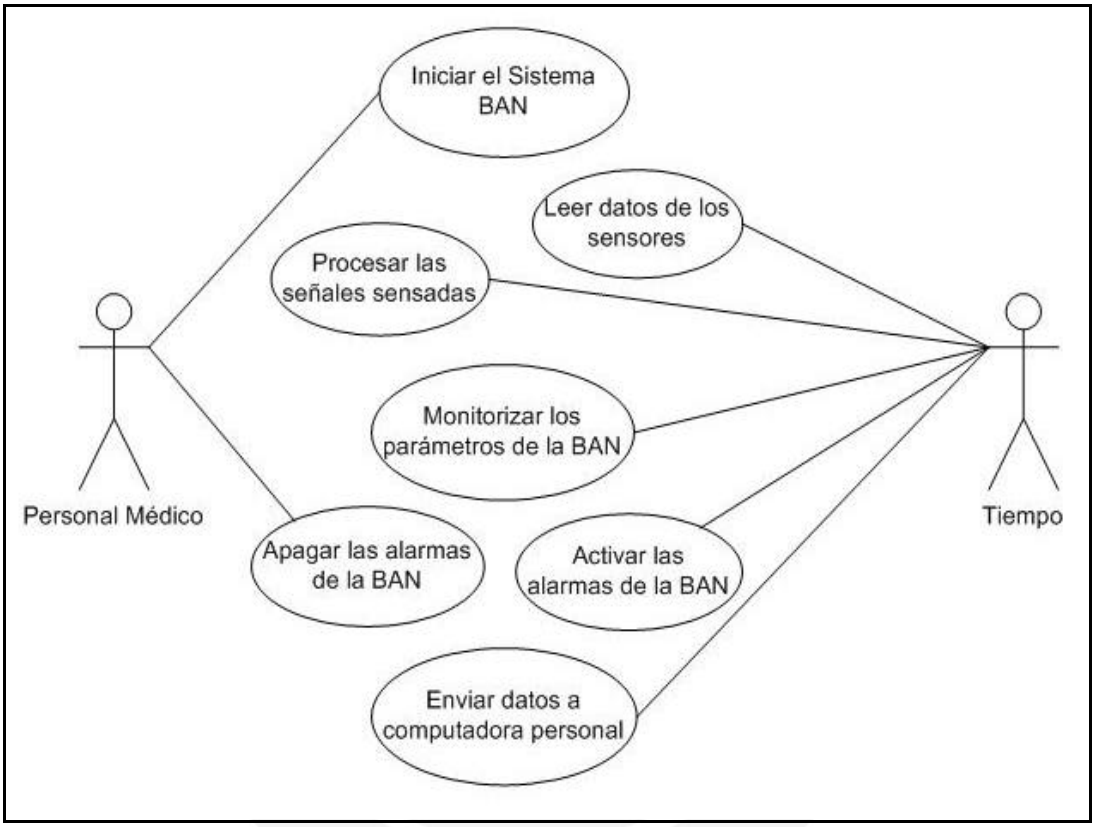


Diagrama de Casos de Uso. General.

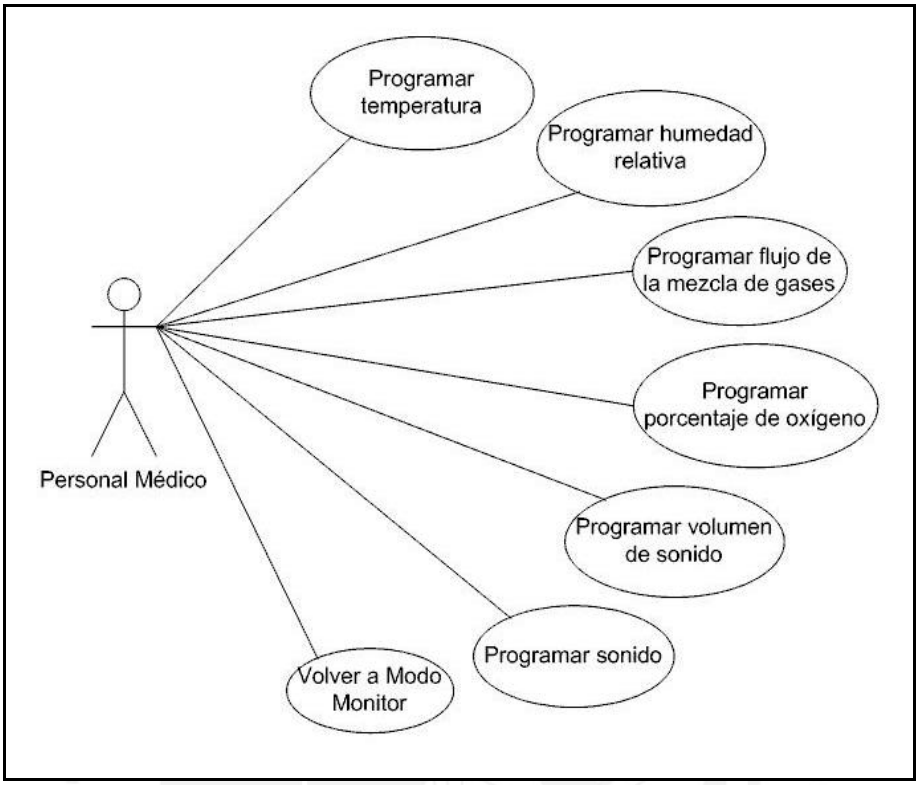


Diagrama de Casos de Uso. Programación.

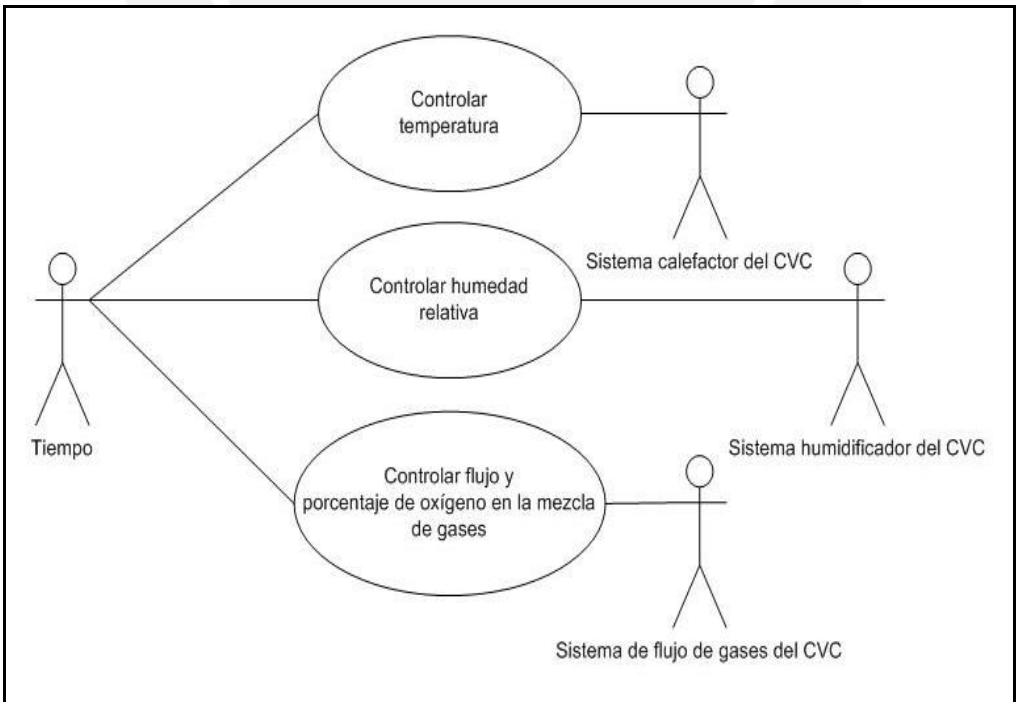


Diagrama de Casos de Uso. Control.

1 Iniciar el Sistema BAN

1.1 Descripción General

Este caso de uso permite al personal médico inicializar el Sistema BAN para su uso.

1.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico enciende la BAN.
2. El Sistema BAN deshabilita el Módulo de Ingreso de Datos (teclado).
3. Se inicializa la memoria SRAM.
4. Se configura el ATmega 128.
5. Se inicializa las pantallas de presentación.
6. Se configura todos los dispositivos de HW.
7. Se inicializa el Núcleo y los contextos de los procesos de la BAN.
8. Se inicializan los parámetros de control usando la configuración estándar guardada. Esta consiste en la programación de: temperatura en el ambiente de la cúpula interna a 34 ºC, humedad relativa a 60 %, porcentaje de oxígeno a 21 % y flujo de la mezcla de aire y oxígeno a 2 L/min.
9. Se habilita el Módulo de Ingreso de Datos (teclado).

Flujos Alternativos

No se presentan flujos alternativos.

2 Leer datos de los sensores

2.1 Descripción General

Este caso de uso permite leer los datos de los 9 sensores de la BAN.

2.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se adquieren los datos de las señales de temperatura: en la cúpula interna, en la cúpula externa (dos sensores: izquierda y derecha), en el sistema neumático CVC (calefactor interno) y en al piel del bebé; a través del conversor analógico-digital.
2. Se adquieren los datos de la señal de humedad relativa.
3. Se adquieren los datos de la señal de flujo: en la vía de oxígeno, en la vía de aire y en la vía de la mezcla.
4. Se almacenan los valores adquiridos.

Flujos Alternativos

No se presentan flujos alternativos.

3 Procesar las señales sensadas

3.1 Descripción General

Este caso de uso permite convertir los datos adquiridos a sus unidades respectivas; y aplicar filtros a los datos sensados de flujo.

3.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se convierten los datos de temperatura adquiridos a grados Celsius (ºC): de la cúpula interna, de la cúpula externa (dos sensores: izquierda y derecha), del sistema neumático CVC (calefactor interno) y de la piel del bebé.
2. Se convierten los datos de humedad relativa adquiridos a unidades porcentuales (%).
3. Se aplican algoritmos de filtrado a los datos de la señal de flujo: en la vía de oxígeno, en la vía de aire.
4. Se convierten los datos de los flujos adquiridos a litros por minuto (Lpm).
5. Se almacenan los valores.

Flujos Alternativos

No se presentan flujos alternativos.

4 Monitorizar los parámetros de la BAN

4.1 Descripción General

Este caso de uso permite mostrar al personal médico, tanto los valores que se programan como los valores medidos de los parámetros de temperatura en la cúpula interna, temperatura en la piel del bebé, porcentaje de humedad relativa, flujo de la mezcla de gases, porcentaje de oxígeno en la mezcla; además, permite mostrar si alguna alarma se ha activado.

4.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se toman los valores de los parámetros sensados, los valores programados y los estados de las alarmas. Estos valores pueden ser modificados por la adquisición de datos, el Módulo de Ingreso de Datos (programación de parámetros) o por el chequeo de alarmas.
2. Se actualiza la región de la memoria de video del Scenix (Módulo de Visualización), con los nuevos parámetros y los nuevos estados de las alarmas.
3. Se vuelve al paso 1.

Flujos Alternativos

No se presentan flujos alternativos.

5 Activar las alarmas de la BAN

5.1 Descripción General

Este caso de uso se encarga del manejo de las alarmas en la BAN. Este sistema de alarma es desarrollado en conjunto entre el Módulo de Ingreso de Datos (sonidos de alarmas), el Módulo de Procesamiento y Control y el Módulo de Visualización. (La clasificación de las alarmas se muestra en el Anexo C).

5.2 Escenario o Flujo de Eventos

Flujo Básico

1. El Módulo de Ingreso de Datos detecta un estado de alarma en la BAN, al también sensar las señales de la BAN.
2. El Módulo de Ingreso de Datos manda a sonar el sonido correspondiente a la alarma que se ha activado.
3. El procesador ATmega 128 por su parte también evalúa si un parámetro está fuera del rango establecido, y de darse el caso activa el estado de alarma.
4. El Módulo de Procesamiento y Control se encarga de reflejar el estado de la alarma en la pantalla LCD, comunicándose con el Módulo de Visualización (Scenix).

Flujos Alternativos

No se presentan flujos alternativos.

6 Apagar las alarmas de la BAN

6.1 Descripción General

Este caso de uso permite al personal médico apagar el sonido de las alarmas de la BAN.

6.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico presiona un botón de desactivación de las alarmas.
2. El PIC 16F877 del Módulo de Ingreso de Datos desactiva el sonido de las alarmas.
3. El PIC 16F877 del Módulo de Ingreso de Datos inhabilita, por un intervalo de tiempo de 5 minutos, la activación de las alarmas a fin de que no se vuelva a activar inmediatamente el sonido de la alarma.
4. En todo momento, a pesar de que se desactive el sonido de las alarmas, estas se seguirán mostrando en pantalla.

Flujos Alternativos

No se presentan flujos alternativos.

7 Programar temperatura

7.1 Descripción General

Este caso de uso permite al personal médico programar el valor de temperatura en la cúpula interna de la BAN, en un rango de 25°C a 37°C. El valor modificado se visualiza en la región de los parámetros programados de la pantalla.

7.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico presiona un botón de programación de temperatura.
2. El PIC 16F877 del Módulo de Ingreso de Datos (teclado) detecta que se ha activado la programación de temperatura.
3. El personal médico puede modificar el valor del parámetro moviendo el *encoder* del panel de control. Los valores permitidos están en un rango de 25°C a 37°C.
4. Cada vez que se mueve el *encoder*, el PIC16F877 interrumpe al Módulo de Procesamiento y Control (ATmega 128), señalándole que se está aumentando o disminuyendo el valor de temperatura.
5. El Sistema BAN modifica el valor programado de temperatura.
6. El cambio se debe de ver reflejado en la pantalla LCD del Módulo de Visualización (Scenix).
7. Luego de un tiempo determinado, si no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

Flujos Alternativos

No se presentan flujos alternativos.

8 Programar humedad relativa

8.1 Descripción General

Este caso de uso permite al personal médico programar el valor de porcentaje de humedad relativa en el aire de la cúpula interna de la BAN, en un rango de 40% a 90%. El valor modificado se visualiza en la región de los parámetros programados de la pantalla.

8.2 Escenario o Flujo de Eventos

Flujo Básico

1. El flujo es similar al de la función “Programar temperatura”. Se presiona un botón de programación de la humedad relativa; y, los valores permitidos están en un rango de 40% a 90%.

Flujos Alternativos

No se presentan flujos alternativos.

9 Programar flujo de la mezcla de gases

9.1 Descripción General

Este caso de uso permite al personal médico programar el valor de flujo de la mezcla de gases dentro de la BAN, en un rango de 3 Lpm a 10 Lpm. El valor modificado se visualiza en la región de los parámetros programados de la pantalla.

9.2 Escenario o Flujo de Eventos

Flujo Básico

1. El flujo es similar al de la función “Programar temperatura”. Se presiona un botón de programación de flujo; y, los valores permitidos están en un rango de 3 Lpm a 10 Lpm.
2. Adicionalmente, se leen los valores programados de porcentaje de oxígeno y de flujo total de la mezcla, y empleando las fórmulas que relacionan estas dos variables se hallan los valores de flujo de aire y de flujo de oxígeno requeridos para cumplir con los valores programados.

Flujos Alternativos

No se presentan flujos alternativos.

10 Programar porcentaje de oxígeno

10.1 Descripción General

Este caso de uso permite al personal médico programar el valor de porcentaje de oxígeno dentro de la BAN, en un rango de 21% a 100%. El valor modificado se visualiza en la región de los parámetros programados de la pantalla.

10.2 Escenario o Flujo de Eventos

Flujo Básico

1. El flujo es similar al de la función “Programar temperatura”. Se presiona un botón de programación de porcentaje de oxígeno; y, los valores permitidos están en un rango de 21% a 100%.
2. Adicionalmente, se leen los valores programados de porcentaje de oxígeno y de flujo total de la mezcla, y empleando las fórmulas que relacionan estas dos variables se hallan los valores de flujo de aire y de flujo de oxígeno requeridos para cumplir con los valores programados.

Flujos Alternativos

No se presentan flujos alternativos.

11 Programar sonido

11.1 Descripción General

Este caso de uso permite al personal médico programar el sonido dentro de la BAN.

11.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico presiona un botón de programación de sonidos.
2. El PIC16F877 del Módulo de Ingreso de Datos (teclado) detecta que se ha activado la programación de sonidos.
3. El personal médico puede programar la melodía moviendo el *encoder* del panel de control.
4. Cada vez que se mueve el *encoder*, el PIC16F877 interrumpe al Módulo de Procesamiento y Control (ATmega 128), enviándole la nueva melodía programada.
5. El Módulo de Procesamiento y Control envía un mensaje al Módulo de Sonidos para que reproduzca la melodía programada.
6. Se escucha el sonido respectivo.
7. Luego de un tiempo determinado, si no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

Flujos Alternativos

No se presentan flujos alternativos.

12 Programar volumen de sonido

12.1 Descripción General

Este caso de uso permite al personal médico programar el volumen de sonido dentro de la BAN.

12.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico presiona un botón de programación de volumen.
2. El PIC16F877 del Módulo de Ingreso de Datos (teclado) detecta que se ha activado la programación de volumen.
3. El personal médico puede programar el volumen moviendo el *encoder* del panel de control.
4. Cada vez que se mueve el *encoder*, el PIC16F877 interrumpe al Módulo de Procesamiento y Control (ATmega 128), enviándole el nuevo volumen programado.
5. El Módulo de Procesamiento y Control envía un mensaje al Módulo de Sonidos para que cambie el volumen.
6. Luego de un tiempo determinado, si no se ha vuelto a Modo Monitor de manera manual, el Sistema BAN vuelve a Modo Monitor.

Flujos Alternativos

No se presentan flujos alternativos.

13 Volver a Modo Monitor

13.1 Descripción General

Este caso de uso permite al personal médico volver a la pantalla de Modo Monitor de la BAN; en este modo el teclado está bloqueado.

13.2 Escenario o Flujo de Eventos

Flujo Básico

1. El personal médico presiona un botón de vuelta a Modo Monitor.
2. El PIC 16F877 del Módulo de Ingreso de Datos (teclado) interrumpe al Módulo de Procesamiento y Control (ATmega 128).
3. El Sistema BAN detecta que se ha vuelto al Modo Monitor.

Flujos Alternativos

No se presentan flujos alternativos.

14 Controlar temperatura

14.1 Descripción General

Este caso de uso permite el control de la temperatura en la BAN según el valor programado por el personal médico, en un rango de 25 ºC a 37 ºC.

14.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se lee el valor programado de temperatura.
2. Se leen los valores sensados de temperatura en la cúpula interna y en el sistema neumático CVC (calefactor interno).
3. Para el control de temperatura se utilizan dos ondas PWM (Modulación por Ancho de Pulso), una por calefactor. El rendimiento de cada onda se calcula sobre la base de la diferencia entre el valor sensado de temperatura y el valor programado.
4. Se mandan las palabras de control al Módulo de Manejo de Dispositivos de Control de modo que los calefactores se prendan y apaguen en proporción al rendimiento calculado para cada onda.
5. Se vuelve al paso 1.

Flujos Alternativos

No se presentan flujos alternativos.

15 Controlar humedad relativa

15.1 Descripción General

Este caso de uso permite el control del porcentaje de humedad relativa en la BAN según el valor programado por el personal médico, en un rango de 40% a 90% de humedad relativa.

15.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se lee el valor programado de humedad relativa.
2. Se lee el valor sensado de humedad relativa.
3. Si el valor de humedad relativa difiere, se calcula la distancia que debe de moverse el actuador. Luego se convierte la distancia calculada a un intervalo de tiempo.
4. Se manda la palabra de control para que el actuador que regula la humedad relativa se mueva según lo especificado.
5. Cuando se cumple el tiempo calculado, se manda la palabra de control para que el actuador se detenga.
6. Se vuelve al paso 1.

Flujos Alternativos

No se presentan flujos alternativos.

16 Controlar flujo y porcentaje de oxígeno en la mezcla de gases

16.1 Descripción General

Este caso de uso permite el control de la mezcla de aire y oxígeno en la BAN según los valores de los parámetros programados por el personal médico, en un rango de 3 Lpm a 10 Lpm para el flujo de la mezcla y en un rango de 21% a 100% para el porcentaje de oxígeno.

16.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se leen los valores sensados de flujo de aire, flujo de oxígeno y flujo de la mezcla actuales.
2. Si los valores difieren, se controlan los actuadores de aire y oxígeno para que lleguen al valor calculado.
3. Si el valor de aire difiere, se calcula la velocidad con la que debe de moverse la bomba de aire para que dé el flujo de aire requerido.
4. Se manda la palabra de control para la bomba.
5. Si el valor de oxígeno difiere, se calcula la distancia que debe de moverse el regulador de oxígeno. Luego, se convierte la distancia calculada a un intervalo de tiempo.
6. Se mandan la palabra de control para que el actuador que regula el oxígeno se mueva según lo especificado.
7. Cuando se cumple el tiempo calculado, se manda la palabra de control para que el actuador se detenga.
8. Se vuelve al paso 1.

Flujos Alternativos

No se presentan flujos alternativos.

17 Enviar datos a computadora personal

17.1 Descripción General

Este caso de uso le brinda al personal médico la posibilidad de poder conectar la BAN a una PC, a fin de poder llevar un registro en el tiempo del estado de los parámetros importantes del Sistema BAN; así como de las programaciones efectuadas de dichos parámetros.

17.2 Escenario o Flujo de Eventos

Flujo Básico

1. Se leen los datos sensados de las variables de: temperatura en la piel del bebé, temperatura en la cúpula externa (izquierda), temperatura en la cúpula externa (derecha), temperatura en la cúpula interna, temperatura en el CVC, humedad relativa, flujo de oxígeno, flujo de aire, flujo de la mezcla.
2. Se leen los datos programados de las variables de: temperatura en la cúpula interna, humedad relativa, porcentaje de oxígeno y flujo de la mezcla.
3. Se lee los estados de las alarmas del Sistema BAN.
4. Se organiza toda la información en una trama de datos.
5. Se envía toda la información, por una interfaz serial mediante el protocolo RS232.

Flujos Alternativos

No se presentan flujos alternativos.