METODOLOGÍAS PARA EL DISEÑO DE EQUIPOS MICROCONTROLADOS

1 MÁQUINAS DETERMINÌSTICAS

En una máquina determinística, la operación se realiza en forma **sincrónica** en contraposición al modo **asincrónico** en que trabaja la máquinas por interrupciones.

Las bases fundamentales de la máquina determinística son el barrido, el muestreo, el tiempo real y la estructura por capas.

El Barrido consiste en la ejecución repetitiva de cada una de las partes del programa independientemente del estado del Proceso externo.

El Muestreo consiste en la lectutra y la escritura periódicas desde y hacia el hardware, con retención de la información en la memoria RAM del Microcontrolador

Tiempo Real es la característica que se agrega al barrido para que los programas sean ejecutados periódicamente a intervalos de tiempo regulares y prefijados.

Estructura por Capas es la característica de los programas en los cuáles existe una **BIOS, un Sistema Operativo y una Aplicación.** La BIOS se encarga de realizar los muestreos de la Información; El Sistema Operativo se encarga de controlar el barrido, generar el tiempo real y distribuir las tareas; La Aplicación contiene los bloques de programa que definen la operatividad del microcontrolador.

La principal ventaja consiste en que el programa opera tal como se codifica, lo que optimiza el tiempo de diseño. Adicionalmente no existen Bugs en el programa.

La principal desventaja consiste en que cada bloque de programa tiene que ser desarrollado teniendo en cuenta que se debe entregar el control del flujo del programa a otros bloques.

Programando con Micrograde, este problema se soluciona ya que el control del flujo del programa es administrado previamente por una aplicación de alto nivel. Adicionando los siguientes recursos:

- Multitarea
- Muestreo
- Tiempo Real
- Multifrecuencia
- Programación por capas

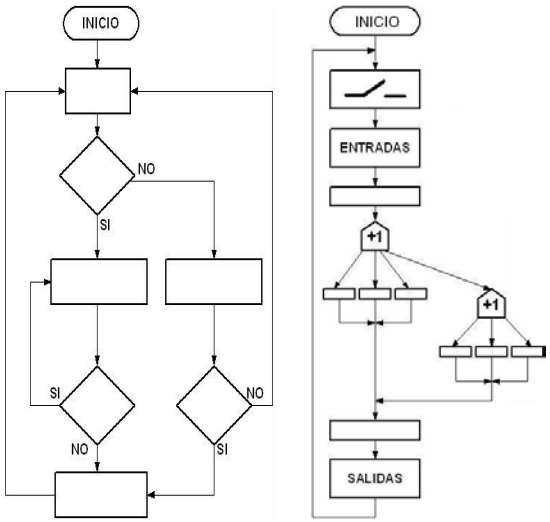
- Máquinas de estado
- Dispositivos Virtuales.

1.1 RECURSOS

En la programación a bajo nivel la productividad es muy baja debido a que se desarrollan N número de funciones por días, semanas e incluso meses.

Al momento de realizar el proceso de programación se puede presentar el "FENOMENO BARRERA", consiste en que el código ofrece una buena simulación, sin embargo, no funciona en el board.

METODOS CONVENCIONALES VS NUEVOS METODOS



TIEMPO **FRECUENCIA**

DOMINIO DE LA FRECUENCIA

Un programa en el dominio del tiempo corre el riesgo de entrar en ciclos infinitos, en el dominio de la frecuencia esto nunca ocurrirá ya que el microcontrolador recorre todas las tareas y condiciones dentro de las tareas que evitan que el programa ejecute las instrucciones en éstas o salte a la siguiente tarea.

El programa que trabaje con el dominio de la frecuencia permite desarrollar el concepto de multitarea ya que esta revisa todas las condiciones de cada una de las tareas, en función del tiempo el programa debe esperar el cumplimiento de una condición para ejecutar una acción de salida.

1.1.1 MULTITAREA

Multitarea es la capacidad de desarrollar varios procesos al mismo tiempo. Es lograr del microcontrolador la vigilancia de todos los procesos sin perder la unidades de tiempo y los parámetros de control, teniendo como principal prioridad la alerta de los sucesos en las entradas y realizar los cambios correspondientes en las salidas.

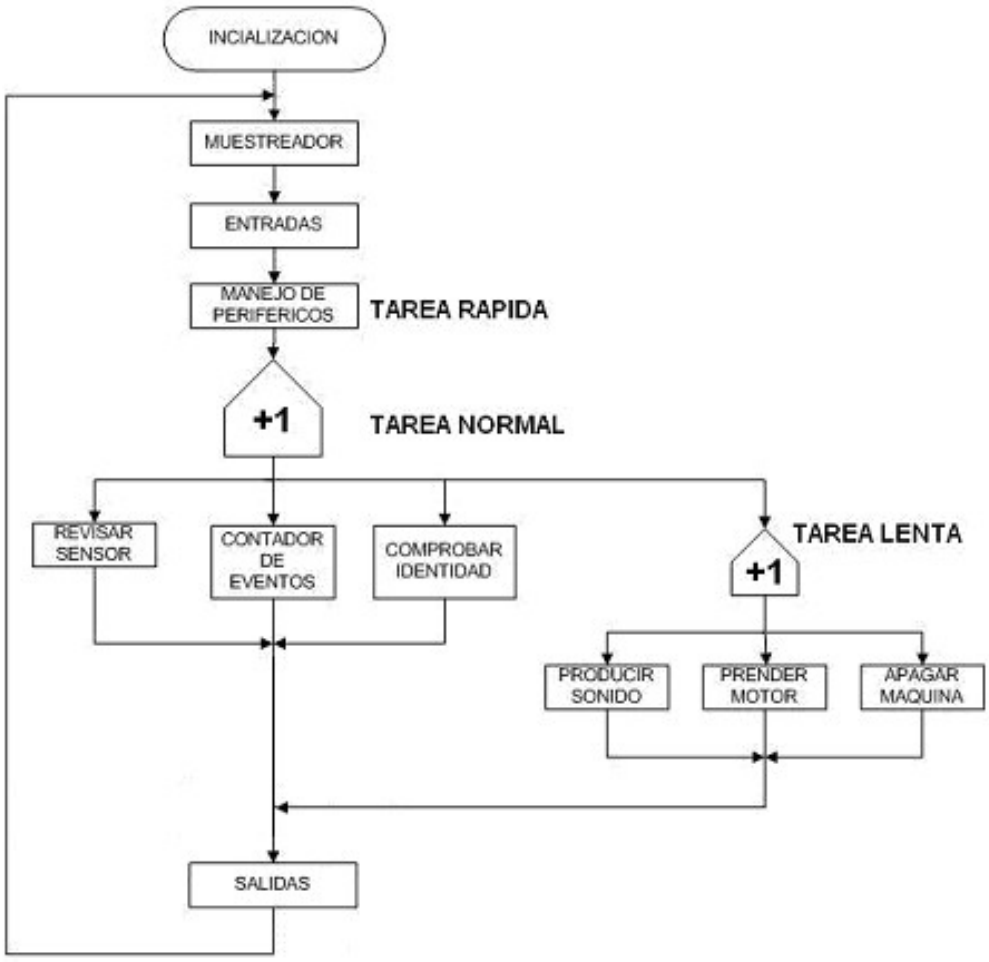
El lograr realizar éstos procesos en forma simultánea sin necesidad de realizar interrupciones, reduce el consumo de RAM y hace que el comportamiento de la aplicación sea más predecible.

"Cada tarea piensa que posee el CPU, pero esta apariencia se controla por el kernel", *Jerry Farmer (Myriad Development Company) Microchip.*

Ejemplo

La multitarea puede ser posible si dividimos una tarea en pequeñas subtareas y se administran a deteminada frecuencia.

veamos un ejemplo:



vemos que existen tres tipos de tareas principales que son Rápida, Normal y Lenta, las cuales pueden desglosarse en subtareas, que serán ejecutadas consecutivamente en cada ciclo de programa.

1.1.2 MUESTREO

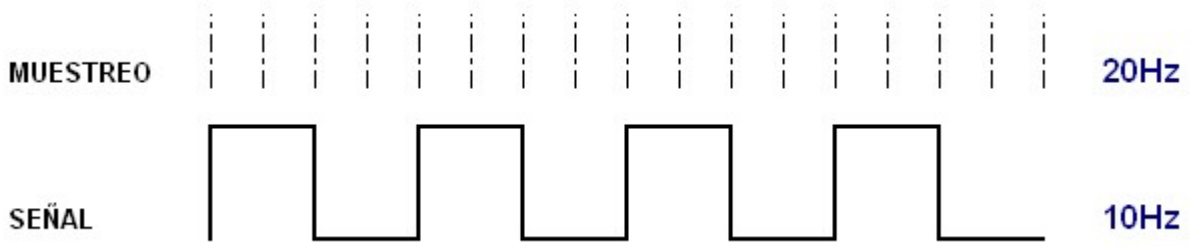
El muestreo consiste en la lectutra y la escritura periódicas desde y hacia el hardware, con retención de la información en la memoria RAM del Microcontrolador

Guardar la información en la memoria RAM se hace necesario para estabilizar la información en el desarrollo de la aplicación.

Además el muestreo se hace bajo la premisa según **Nyquist** de que "la velocidad del muestreo debe ser mínimo el doble de la del proceso".

Ejemplo

Según Nyquist debemos muestrear mínimo al doble de la señal, por lo que si tenemos una señal de 10Hz debemos muestrear como mínimo a 20Hz ó una frecuencia superior.



1.1.3 TIEMPO REAL

Tiempo Real es la característica que se agrega al barrido para que los programas sean ejecutados periódicamente a intervalos de tiempo regulares y prefijados. De ésta forma el microcontrolador puede responder a cualquier tipo de proceso que deba controlar utilizando recursos como:

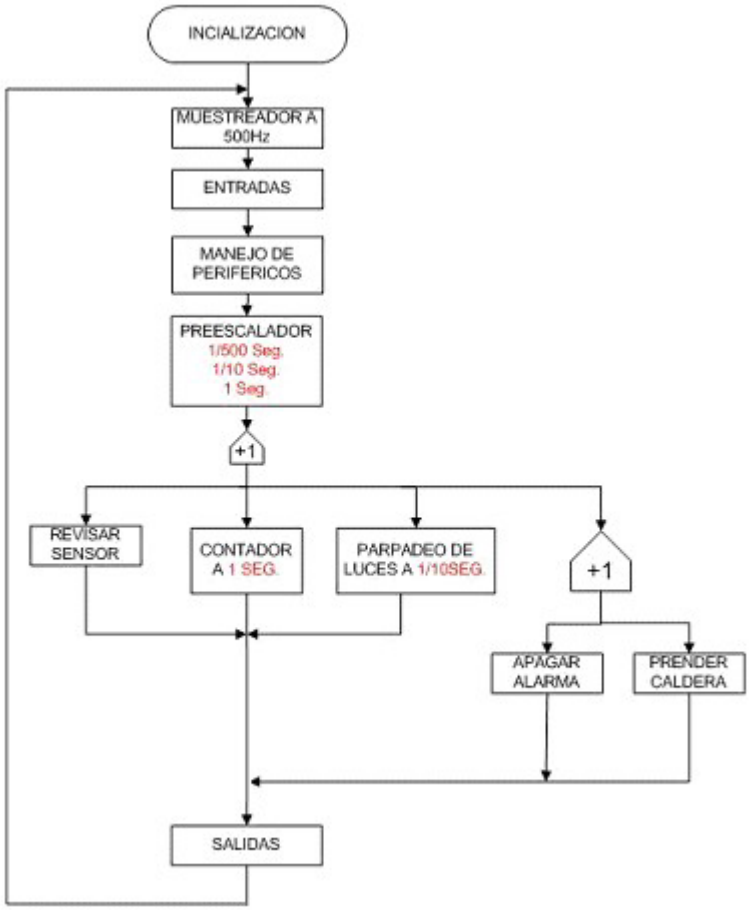
Muestreo el cual es establecido por el desarrollador en diferentes unidades de tiempo.

Cuando el dispositivo de oscilación no es un cristal, sino un resonador o un circuito RC, las bases de tiempo pueden contener errores y el reloj de tiempo real generará pequeños desfaces.

Ejemplo

En una estructura determinística el tiempo real parte de ajustar bases de tiempo a usar según la velocidad de las tareas. Por ejemplo si la tarea rápida es de 500Hz necesitaríamos contar 50 ciclos para una base de 1 /10 de segundo, y a partir de esta base contar 10 ciclos para una nueva base de 1 segundo, y así tenemos 3

bases, una natural de 1 / 500 de segundo (500 Hz), una de 1/10Seg. y la de 1Seg. que podemos utilizar en el programa para alguna labor específica.

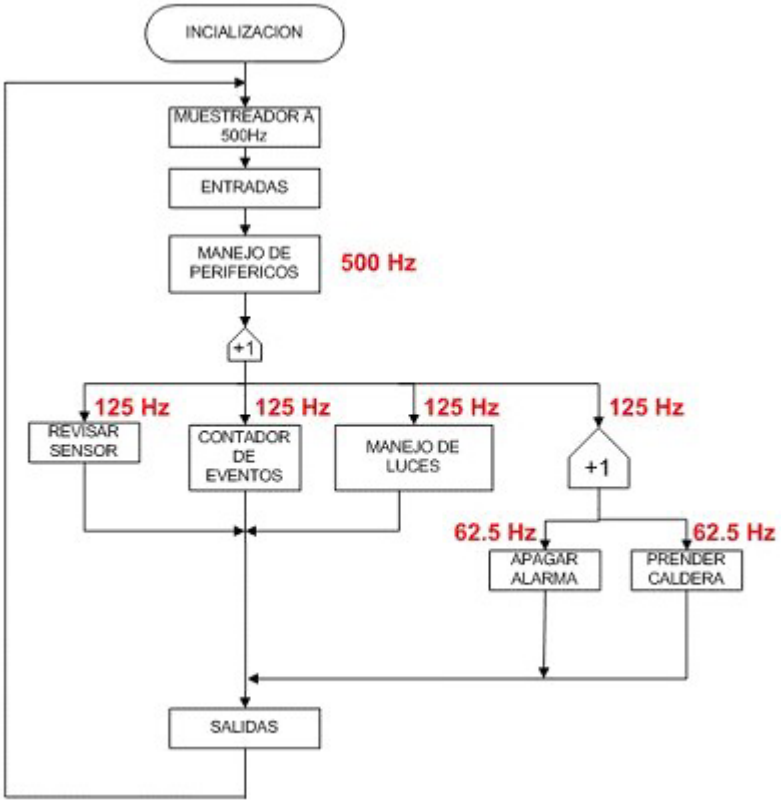


1.1.4 MULTIFRECUENCIA

Es la capacidad de desarrollar tareas a diferentes frecuencias de acuerdo a la evolución de los procesos identificando cuando debo utilizar tareas RAPIDAS , NORMALES O LENTAS.

Ejemplo

La multifrecuencia parte de tener una frecuencia base fundamental, que es la de una tarea rápida, de alli podemos derivar frecuencias en las diferentes tareas.



1.1.5 PROGRAMACIÓN POR CAPAS

La programación estructurada sugiere que los programas deben ser creados mediante una metodología modular, esto es, que cada parte del programa está representado en varios bloques, otra característica de éste tipo de programación es la no existencia de saltos a bloques anteriores, el único salto que existe es una vez se halla recorrido todo el programa, por lo tanto ningún bloque quedará "enganchado" dentro de un ciclo. lo que significa que cada bloque dejará fluir libremente el programa.

Existen tres tipos de estructura:

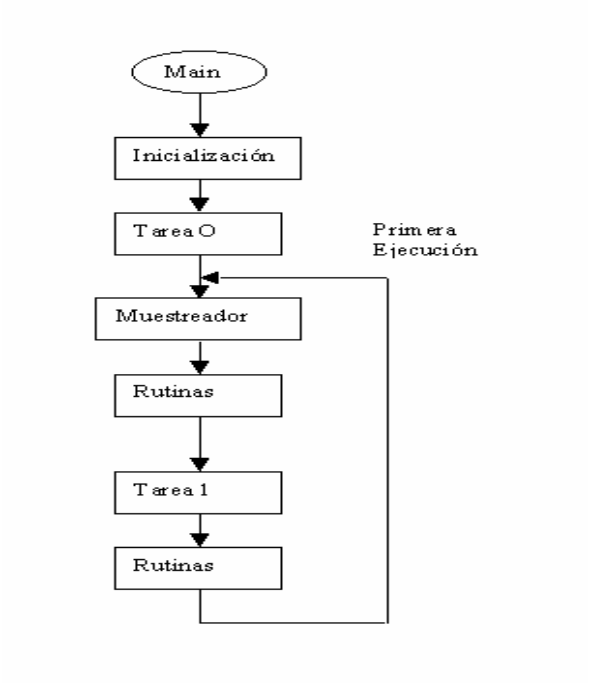
- **Estructura de Secuencia:** Aquellas que cumplen una función específica dentro del programa.
- **Estructura de bifurcación:** Aquellas que seleccionan que tarea se ejecutara al retomar un nuevo ciclo de programa.
- **Estructura Cíclicas:** Representado por el porgrama principal.

CARACTERISTICAS DE ESTA PROGRAMACION

- Se tiene un programa cíclico, por lo tanto sólo se apaga con la alimentación

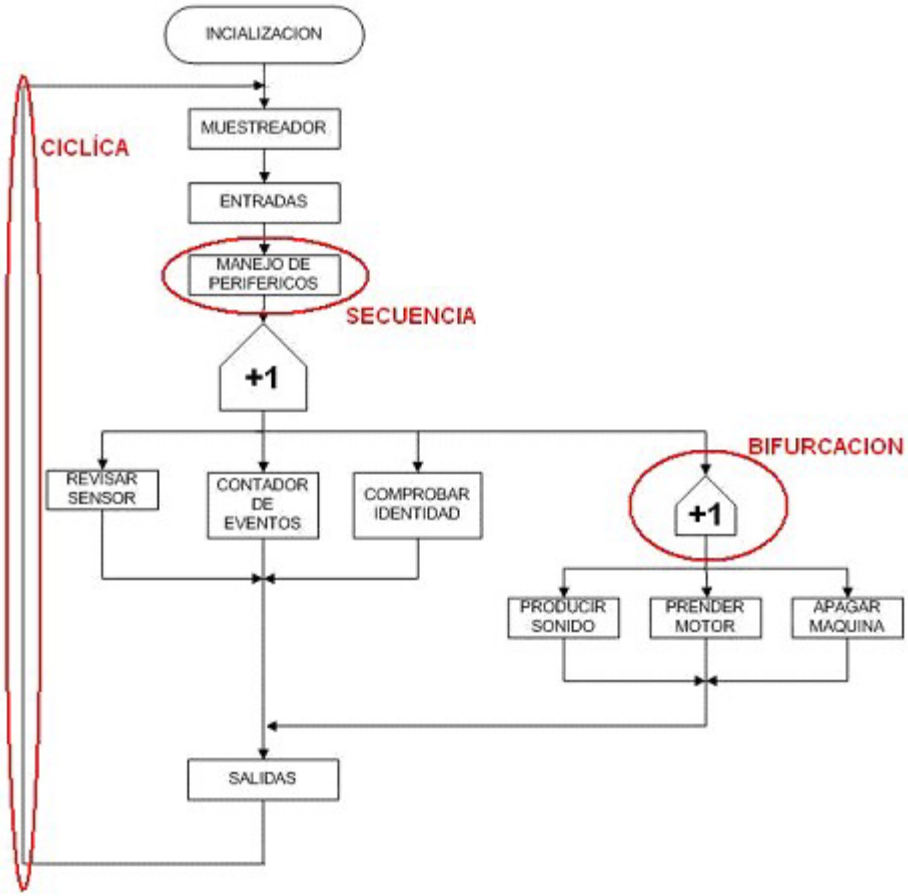
- Es un programa que reparte control
- Opera a una frecuencia Constante
- Realiza muestras periódicas, garantizando filtraje en las señales.
- El control de la aplicación reside en la memoria, se saca el programa a la RAM
- Se trabaja con el dominio de la frecuencia
- Régimen Estacionario
- Permite la programación de dispositivos virtuales
- Ninguna Rutina retiene el control

Ver figura.



Ejemplo

Veamos la correspodencia a cada estructura sobre un programa:



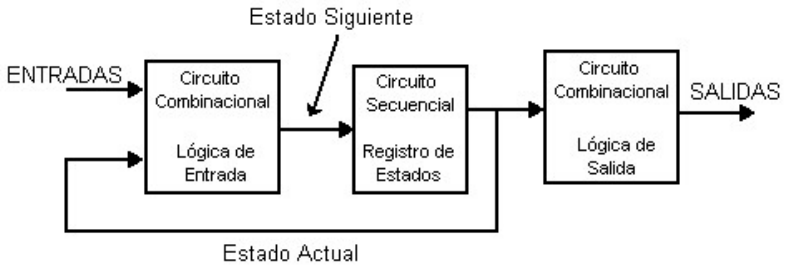
1.1.6 MÁQUINAS DE ESTADO

Las máquina de estadoson circuitos secuenciales que tienen un número determinado de estados (2n) siendo n la cantidad de salidas de un sistema, pueden ser retroalimentados (flip flops, biestables) o máquinas sincrónicas temporizadas. En cualquier caso se tienen unas salidas unas entradas y unos estados.

Existen dos tipos de máquinas de estado:

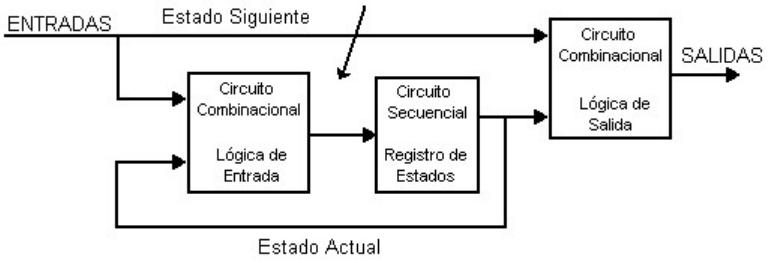
MOORE.

La salida únicamente depende del estado actual.



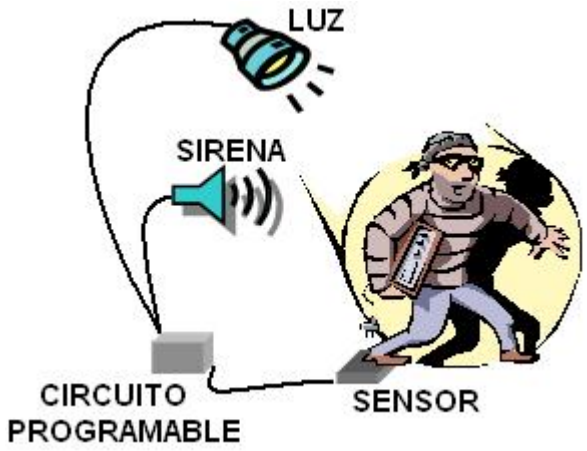
MEALY.

La salida depende del estado actual y de las entradas

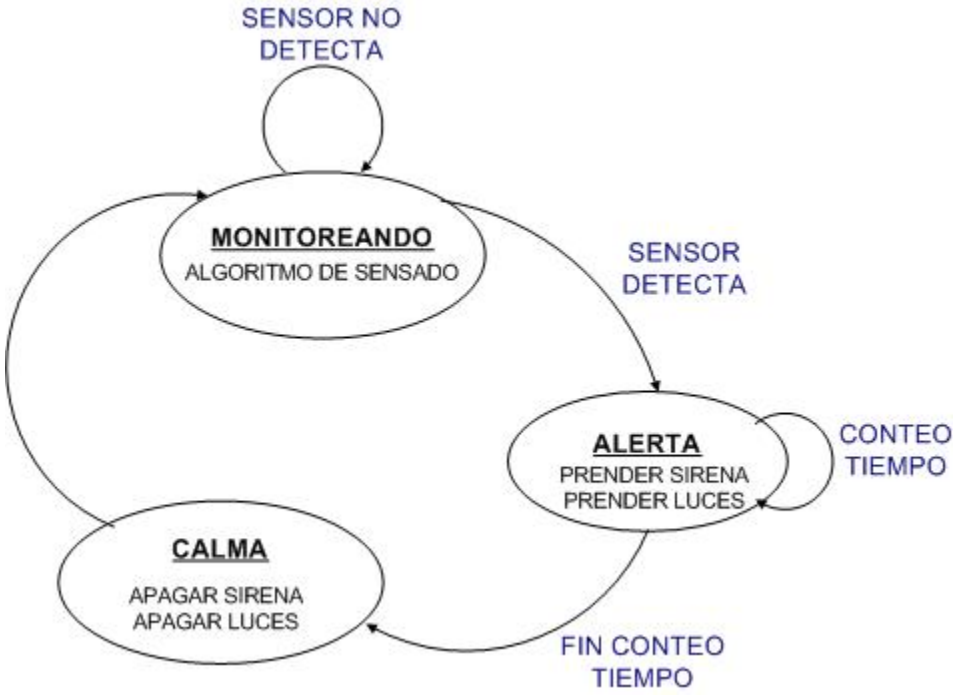


Ejemplo

Si tuviesemos un proceso sencillo como una alarma, que cuando se active un sensor se prenda una sirena y una luz durante 30 segundos.

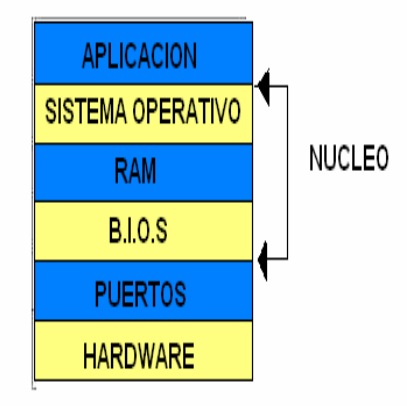


La máquina de estados sería de la siguiente forma:



1.2 EL NÚCLEO DE UNA APLICACIÓN

Cuando se cuenta con un sistema operativo sabemos que se tiene varios componentes que lo conforman para que este pueda operar, la siguiente figura muestra la estructura del sistema operativo de MICROGRADES.



Como podemos ver es el Sistema operativo es el que administra los recursos del Chip y el encargado de mantener la información incorruptible, por lo que cubre a la aplicación del usuario para que esta funcione deacuerdo a los programado.

El Kernel lo conforma el S.O sobre la aplicación, la RAM y la BIOS, estos tres son la base para que la aplicación cuente con información estable y local.

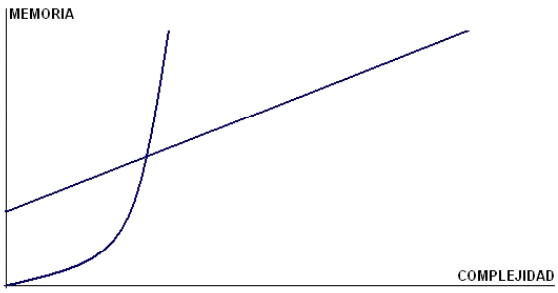
La parte mas cercana a la aplicación se llama BIOS (BASIC INPUT OUTPUT SYSTEM) que se encarga de leer o escribir los puertos a través de la información que se encuentra en la RAM, es decir, un conjunto de subrutinas que se encargan de llevar la información de la RAM a los puertos o viceversa, como por ejemplo, subrutinas de entradas digitales, análogas, manejos de display, manejo de teclados, manejo de PWM y otros.

La programación orientada a objetos hace más estructurado el manejo del tiempo en una aplicación.



Cuando se cuenta con un sistema operativo se empieza a utilizar recursos del chip inicialmente para administrarlos esto hace que una aplicación muy sencilla ocupe más de lo normal, pero que un aplicación más robusta ocupe menos de lo que esperamos.

Algunas de las funciones del Sistema operativo son:



- Detectar caídas de voltaje para salvar información
- Conservar el estado de la información en RAM
- Conservar el muestreador estable.

Existen dos formas de constituir un núcleo (kernel)

El Nucleo No-preventivo (**Non-Preemptive Kernel)**

El Nucleo No-preventivo también se llama un "El Nucleo cooperativo" porque las tareas sólo entregan el mando cuando ellas necesitan estar en coordinación con otras tareas, y eventos.

El Nucleo preventivo (**Preemptive Kernel)**

En un Nucleo Preventivo, una tarea corriente puede hacer intercambio con una tarea de prioridad más alta cuando esta está lista.

1.2.1 HARDWARE

La parte física de la aplicación es el mundo exterior es decir, los circuitos y componentes físicos que hagan parte de un aplicativo. Entre dichos componentes se encuentra la configuración reprogramable, aquellos circuitos lógicos que pueden variar según la necesidad, es decir un microcontrolador.

El microcontrolador cumple la vital función de disponer de sus propios recursos como de aquellos que en el circuito se conecten a el.

Esto crea un circuito inteligente, pues podemos agregar un programa al microcontrolador para que realice procesos con su perifericos.

Pero este hardware por su naturaleza real, està sometido a inclemencias climaticas, o golpes, pero ante todo, ruido electromagnético, el cual puede hacer que los procesos que el microcontrolador esté ejecutando no se cumplan correctamente y el dispositivo empiece a fallar.

Los niveles de ruido dependen del ambiente en donde trabaje la aplicación, puede ser un hogar, un restaurante, una fabrica,etc.

Lo importante es como discriminar este fenómeno para que el microcontrolador opere correctamente.

1.2.2 BIOS

Basic Input Ouput System

Este sistema tiene como labor principal encargarse del estado en los pines del microcontrolador, para que la aplicaciòn pueda toma decisiones.

Un pin puede ser configurado como una entrada o como una salida, esto determina que las condiciones de entrada cambien las de salida.

La BIOS toma el estado de las entradas y las pone en RAM, a disposición de la aplicación, posteriomente es la aplicación quien analiza estos estados y determinauna accion con las salidas, la cual es puesta de nuevo en RAM, de nuevo la BIOS toma estainformación y la coloca en los pines de salida, para luego comenzar el ciclo otravez.

Cuando la BIOS es la barrera que separa los pines de programa, logramos un primer nivel de estabilidad en la información que se verá reflejado en todo el sistema.

1.2.3 SISTEMA OPERATIVO

La forma de operar los recursos de un chip, determinan la eficiencia y calidad de un producto electrónico, por eso el sistema operativo debe garantizar que los recursosse estan utilizando correctamente y que la información no se está perdiendo ni deteriorando.

Hablar de sistemas operativos en microcontroladores es un paradigma roto desde hace muchos años, pues aunque no se tiene la misma complejidad que un sistema operativo para un computador personal si existen estructuras identicas para administrar los recursos de un chip, que tiene conectado a él diversos perifericos.

1.2.4 APLICACIÓN

Es el pedazo de la arquitectura del kernel que contempla los requerimientos del usuario o desarrollador, para que maneje los recursos a voluntad.

En este nivel existe entonces un entorno de programación, el cual se convierte en la herramienta para construir ideas, de ella depende que el programador tome o no bastante tiempo en construit una idea y ponerla a funcionar.

La programación por medio de o texto, hace que este proceso de desarrollo tome bastante tiempo, que se olvide rápidamente el código y que los prototipos sean de difícil mantenimiento.

La programación gráfica permite recordar con gran facilidad la estructura de una idea, hace rápido el desarrollo de un proceso y sin qu el programador lo conciba, orienta el prototipo al mantenimiento.

2 MÁQUINAS POR INTERRUPCIONES

Esta máquinas operan bajo el principio de las **Interrupciones.**

Una **Interrupción** es el corte súbito en la ejecución de un programa suscitado por un evento externo o interno del microcontrolador, a lo cual se ejecuta una rutina de servicio. Este método es Universalmente utilizado en los programas para microcontroladores y microprocesadores.

Su principal ventaja está en la posibilidad de correr procesos que requieren la máquina exclusiva para ellos.

Ejemplo. Cada bloque de programa es desarrollado para que cumpla una función específica sin importar si en el mismo programa, hay otros bloques que requieren de tiempo de máquina. En éste caso se asume que cada dispositivo de hardware que quiera interactuar con el programa solicitará tiempo de máquina a través de una **Interrupción.** Si se trata de un bloque de procesamiento que no corresponde a Hardware la petición de recurso de máquina será hecha por un **disparador de tiempos** acompañado de un distribuidor de tareas.

La principal desventaja consiste en la limitación que se genera en los chips más pequeños (de menor capacidad) debido a la escacez de recursos que los caracteriza. Obligando a utilizar chips de mayor capacidad y más costosos.

Cuando se desarrolla bajo esta arquitectura no existen estructuras predefinidas pues es totalmente libre al usuario y a la forma de programar en estas, esto abre el paso también al error que puede contener la métodología de programación del usuario.

3 WEBGRAFÍA

[1] A Real-Time Operating System for PICmicro' Microcontrollers

www.microchip.com/download/appnote/pic16/00585b.pdf

[2] Software Interrupt Techniques

www.microchip.com/download/appnote/devspec/16c5x/00514c.pdf

[3] Understanding Small Microcontrollers "The Paced Loop"

e-www.motorola.com/files/microcontrollers/doc/ref_manual/M68HC05TB.pdf