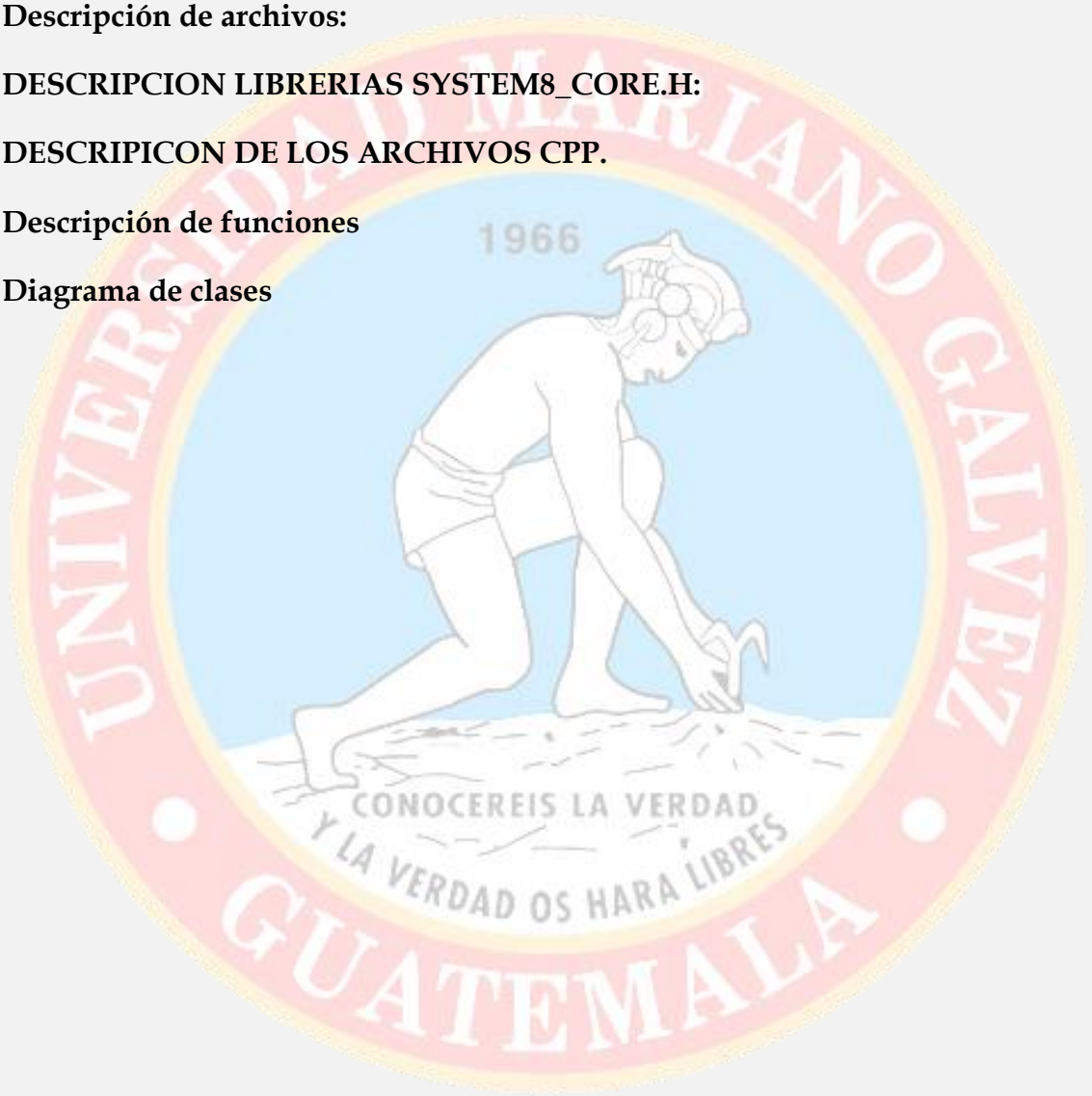
INDICE



DOCUMENTO SYSTEM8 PARA MICROCONTROLADORES ATMEGA328P

Bueno debemos de tener claro que es un sistema operativo y como funciona. Si bien el hardware el de gran importancia el sistema operativo también por ello el curso de sistemas operativos es una mescla de ambos en donde se debe de saber como un sistema multitarea ejecuta varias tareas a la vez.

Debemos de tener la idea que un sistema operativo solo necesita de driver para funcionar con elementos externos. Es por ello que este sistema operativo esta echo para que funcione con cualquier componente electrónico que ud quiera utilizar y programar, ya sean pantallas lcd, led, servo motores, sensores entre otros, lo único que debe de hacer es programar una tarea que funcione mediante el sistema operativo que esta explicado en el documento COMO ENCENDER UN LED.

Antes de empezar con el sistema:

El sistema en si esta codificado para que funcione en un microcontrolador atmega328p, si está utilizando un microcontrolador nuevo vaya a la carpeta ARDU-ATME donde se encuentra la documentación de como cargar el bootloader(secuencia de inicio) al microcontrolador por medio de arduino y del quemador vía serie que hemos diseñado. De lo contrario continúe leyendo este documento

SYSTEM8-328

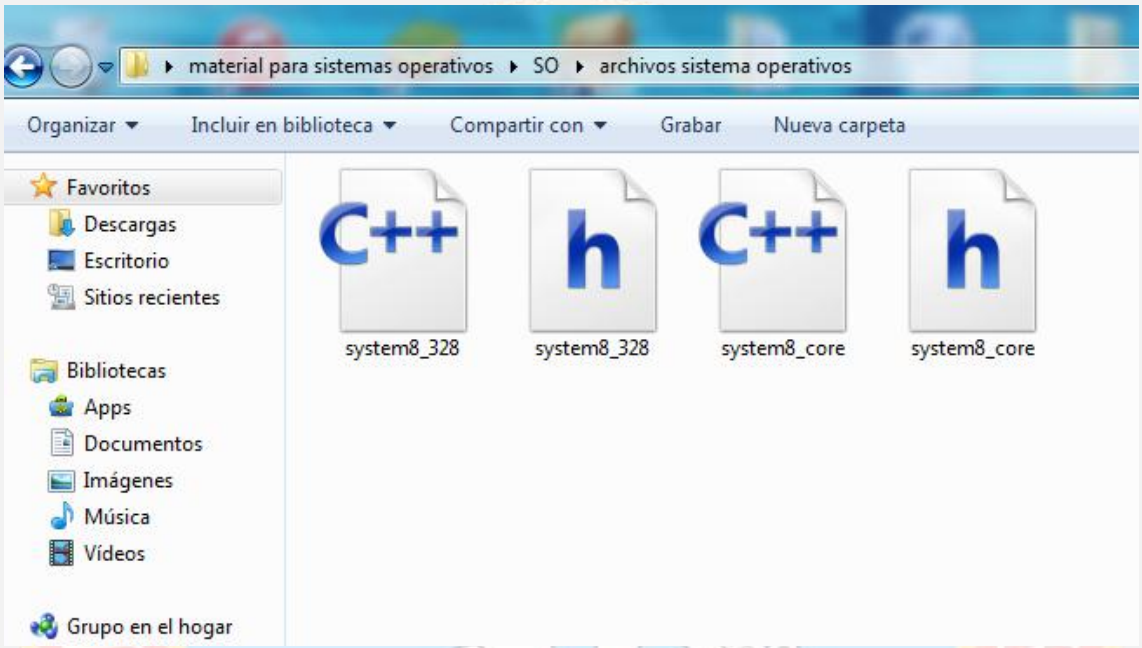
System8-328 es el nombre del sistema operativo para el curso de sistemas operativos abiertos. El nombre system8-328 deriva de que es un sistema operativo de 8 bits y 328 el micro controlador que se utiliza para cargar el sistema.

La idea principal es tener un sistema operativo que sea aplicable a cualquier situación que deseemos aplicar como por ejemplo manejo de cualquier componente, pantalla lcd, sensores, señales analógicas, digitales etc.

Pero además de la utilidad que queramos darle también debemos de saber que system8 es un sistema operativo que funciona mediante multitarea por lo cual ejecuta diversas tareas, haciendo que sea un sistema bien estructurado.

Los archivos del sistema operativo son los siguientes que se muestran en la imagen explicaremos cada uno de ellos detalladamente para una mejor compresión.

Paquete de archivos que contiene el sistema operativo:



Descripción de archivos:

El sistema se divide en cuatro archivos escritos en el lenguaje de programación c++ para editar cada uno de los archivos y ver el código fuente necesitan de un compilador como por ejemplo visual studio o cualquier otro compilador para archivos c++. Como podemos observar en la imagen aparecen dos archivos de los cuales dos son archivos.h y los otros dos son archivos.cpp explicaremos cada uno mas detalladamente.

DESCRIPCION LIBRERIAS SYSTEM8_CORE.H:

Estas son dos librerías del sistema operativo de las cuales **system8_core.h** es la librería que contiene el sistema operativo, y la **system8_328.h** contiene las características del microcontrolador atmega328p para system8.

DESCRIPICON DE LOS ARCHIVOS CPP.

Estos archivos contienen los archivos de como se crean las tareas en el sistema, además cuenta con los planificadores de las tareas como por ejemplo el planificador básico, planficador cíclico, planificador por prioridad etc. En estos archivos también definidos los procesos por los cuales pasa una tarea y como es ejecutada.

Descripción de funciones en librerías system8_328.h y system8_core.h:

Si abrimos archivo con el nombre system8_core.h encontraremos varias funciones de las cuales vamos a describir algunas a efecto de que comprenda como funciona todo. Esta librería contiene funciones escritas en ensamblador porque mediante esta forma se prepara en el microcontrolador los espacios de memoria que serán utilizados por las funciones, tareas, variables etc

```
#define SYSTEM8_getStack() \
  asm volatile\
  (\
    "LDS R26, stack_addr     \n\t"\
    "LDS R27, stack_addr + 1 \n\t"\
    "IN  R0, __SP_L__        \n\t"\
    "ST  X+, R0              \n\t"\
    "IN  R0, __SP_H__        \n\t"\
    "ST  X+, R0              \n\t"\
  );
```

Esta es una función llamada system8_getstack la cual esta escrita en lenguaje ensamblador por la razón que esta es la librería que contiene las características del microcontrolador. Esta librería se encarga de tomar las tareas que están almacenadas en la pila para luego ser ejecutadas por el planificador de tareas.

```
#define SYSTEM8_setStack() \
  asm volatile\
  (\
    "LDS R26, stack_addr     \n\t"\
    "LDS R27, stack_addr + 1 \n\t"\
    "LD  R0, X+              \n\t"\
```

```
    "OUT __SP_L__, R0        \n\t"\
    "LD  R0, X+              \n\t"\
    "OUT __SP_H__, R0        \n\t"\
  );
```

Esta es set_stack() manda a las tareas a la pila para luego ser ejecutadas de acuerdo al planificador.

```
class PlanificadorCiclico : public PlanificadorBasico
  {
  protected:
    byte pos;
    section sect;
    virtual Tarea *getPlanificador();
    virtual Tarea *ejecutarTarea();
    virtual Tarea *get();

  public:
    PlanificadorCiclico();
  };
```

El planificador cíclico es uno de los planificadores con que cuenta system8 para nuestro proyecto hemos utilizado el planificador cíclico que se muestra en el código anterior. El planificador cíclico se encarga de tomar las tareas que se encuentran en la pila y ejecutarlas en el orden asignado para luego repetirlas de forma cíclica. Primeramente hay una clase llamada **planificadorCiclico**: que a su vez extiende de planificadorBasico que está definida en la clase **system8_core.h** luego se crea una variable **protected** (protegida) de tipo byte con el nombre pos, también se crean tres variables **virtuales** que son utilizadas para llamar a funciones externas en otras clases, para finalizar hacer un llamado al método planificadorCiclico() y se ejecuta de forma recursiva dentro de la clase.

Si abrimos el archivo **system_328.h** nos desplegara el código que contiene las características del atmega328p para system8, en ella encontraremos una serie de variables externas que nos servirán para llamar a funciones externas de otras clases.

```
#ifndef SYSTEM8TEM8_328_H   // cabeceras donde llamamos a las clases las cuales
#define SYSTEM8TEM8_328_H   // usaremos en este archivo
#include <arduino.h>
#include <arv.h>

#include "system8_core.h"

extern void** stack_addr;

namespace SYSTEM8
{
  extern Tarea* tareaActual;
```

```
  extern PlanificadorBasico* master;
  extern void* emptyStack;
  extern int (*funcionActual)();
  extern bool primero;
  extern bool estado;
  extern byte cnt;
}
```

Aquí es la parte en donde se inicia el sistema luego de que a sido cargado al microcontrolador como prueba de su funcionamiento correcto se habilita uno de los pines en este caso el numero 13 como una señal que puede ser interpretada por un led como muestra de la misma.

```
#define SYSTEM8_iniciarSistema() \
{\
  pinMode(13, OUTPUT);\
  stack_addr = &SYSTEM8::emptyStack;\
  TCCR0A |= (1 << WGM01);\
  OCR0A = 0xF9;\
  TIMSK0 |= (1 << OCIE0A);\
  E\
  TCCR0B |= (1 << CS01) | (1 << CS00);\
}\
```

Se define una instrucción habilitar esta clase de instrucciones son interpretadas como tokens por el compilador como una serie es por eso que el uso de las diagonales inversas sirven para decirle al compilador que el trozo de código en realidad es una sola línea. Todas las funciones que se muestran a continuación están detalladas en el diagrama de clases que esta adjunto en este documento como material de apoyo para la comprensión de la jerarquía utilizada. también estaremos adjuntando por aparte el código comentariado.

```
#define SYSTEM8_habilitar \
ISR(TIMER0_COMPA_vect) __attribute__ ((signal, naked));\
\
ISR(TIMER0_COMPA_vect)  \
{\
  SYSTEM8_guardarContexto();\
  SYSTEM8_getStack();\
  {\
    {\
      SYSTEM8::master->milli();\
      SYSTEM8::primero = false;\
      SYSTEM8::cnt++;\
      if(SYSTEM8::cnt >= 100)\
      {\
        SYSTEM8::estado = !SYSTEM8::estado;\
        digitalWrite(13, SYSTEM8::estado);\
        SYSTEM8::cnt = 0;\
      }\
    }\
    {\
      SYSTEM8::tareaActual = SYSTEM8::master->get();\
      if(SYSTEM8::tareaActual == (SYSTEM8::Tarea*)0x0000)\
      {\
        stack_addr = &SYSTEM8::emptyStack;\
```

```
      }\
      else\
      {\
        if(SYSTEM8::tareaActual->llamarTarea == false)\
        {\
          SYSTEM8::primero = true;\
          SYSTEM8::funcionActual = SYSTEM8::tareaActual->main;\
          SYSTEM8::tareaActual->llamarTarea = true;\
        }\
        stack_addr = &(SYSTEM8::tareaActual->stack);\
      }\
    }\
  }\
  if(SYSTEM8::primero)\
  {\
    SYSTEM8_setStack();\
    SYSTEM8_invocarEjecucion(SYSTEM8::funcionActual)\
  }\
  else\
  {\
    SYSTEM8_setStack();\
    SYSTEM8_cargarContexto();\
    asm volatile("RETI");\
  }\
}

#endif // SYSTEM8TEM8_328_H
```

DESCRIPCION DE LOS ARCHIVOS SYSTEM8_328.CPP Y SYSTEM_CORE.CPP

Si nos hemos dado cuenta cada archivo de los 4 que componen el sistema está relacionado con el nombre de una librería por ejemplo system_328.h , system_328.cpp. eso es porque cada archivo creado hace referencia a su propia librería. Los archivos. Para una mejor compresión ahora es necesario que habran los archivos adjuntos del sistema operativo en esta documentación, ya que estan comentariados y es mas sencillo entenderlo de esa forma.

DIAGRAMA JERARQUICO DE CLASES

Hemos realizado este diagrama para que vean como una clase depende de la otra y de los componentes que cada una contiene