Universidad Nacional de La Matanza

DEPARTAMENTO DE INGENIERÍA E
INVESTIGACIONES TECNOLÓGICAS

ELECTRÓNICA DE POTENCIA

CONTROL DE ENERGÍA POR TIEMPOS DE OPERACIÓN

MANUAL TÉCNICO

Profesores: Ing. Guillermo Luis Miquel

Ing. Oscar Pugliese

Alumno: Nicolás Enrique Agostino
Ignacio Luis Mehle
Santiago Ruiz
Federico Ladislao Sokolic
Pablo Fabián Yujra Ventura

Revisión 002

Índice

Control de Energía por Tiempos de Operación - Manual Técnico

1. Descripción

El dispositivo tiene como principal función la conexión y desconexión de forma automática de un artefacto o carga eléctrica monofásica siguiendo un programa semanal configurable, respetando los feriados según el calendario anual vigente. Este proyecto fue desarrollado a lo largo de la cursada de la asignatura “Electrónica de Potencia”, en el marco de impulsar una mejora en el consumo sustentable de la Universidad.

2. Hardware

El eje central del dispositivo es un microcontrolador ATMEGA328P, encargado de coordinar cada una de las tareas del proyecto. Lo rodean diferentes bloques/módulos que le brindan al equipo funcionalidades para cumplir con su objetivo. Estos bloques se irán definiendo en particular en la sección “Descripción de partes”.

2.1. Diagrama en bloques

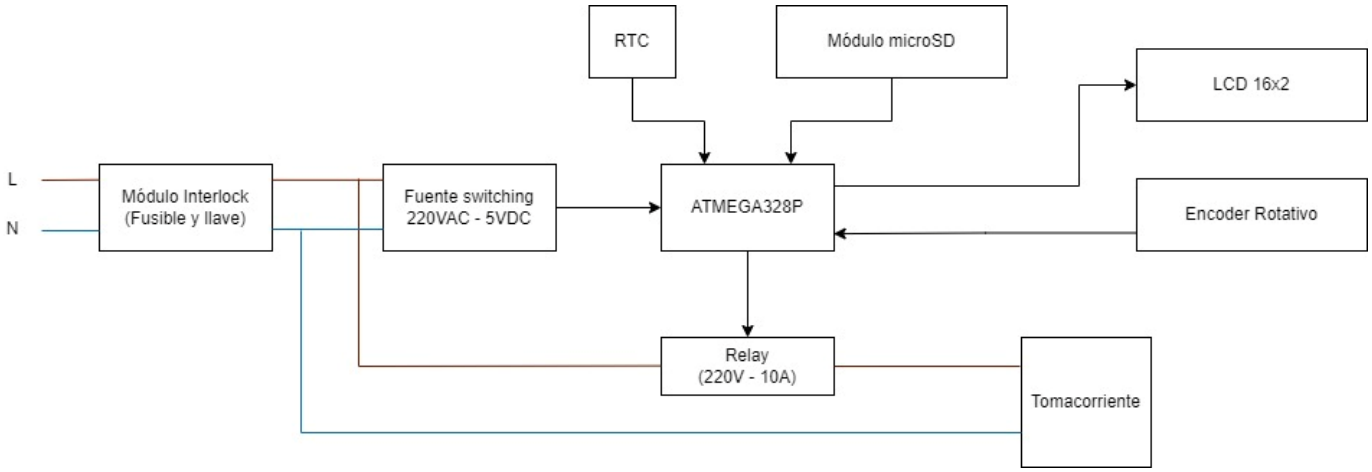

Ilustración 1: Diagrama en bloques

2.2. Descripción de partes

- *ATMEGA328P*

Es un microcontrolador que pertenece a la familia de los microcontroladores AVR de arquitectura RISC-8-bit. En formato DIP, es el que se utiliza en la tarjeta Arduino Uno R3. Posee características como:

- 32 KB de memoria FLASH
- 1 KB de memoria EEPROM
- 2 KB de SRAM
- 23 líneas de I/O de propósito general
- 32 registros de proceso general
- 3 temporizadores flexibles
- Interrupciones internas y externas (timers, adc, ext_int, etc)
- Interfaces UART, I2C e SPI.
- Conversor A/D de 10 bits
- Frecuencia máxima de funcionamiento de 20MHz.
- Voltaje de alimentación de 3,3V a 5V DC.

Ilustración 2: ATMEGA238P

- *Fuente Switching 220VAC – 5VDC*

Para la alimentación del dispositivo, dado que la mayoría de los bloques requerían una tensión constante de 5V, se utilizó una pequeña fuente switching modelo WX-DC12003. Esta permite una sencilla implementación, conectando la alimentación monofásica en su entrada y obteniendo una alimentación de 5V constantes en su salida, con una variación de una centésima de volt. Posee las siguientes características.

- Tensión de entrada: 50-270VAC
- Tensión de salida: 5V DC (±0.3 V)
- Corriente de salida: 700mA
- Potencia de salida: 3,5W
- Ripple: <40mV
- Protección contra cortocircuito/sobretensión/sobrecorriente
- Dimensiones: 23,5x18,15x13,5mm

Ilustración 3: Fuente Switching

- *Pantalla LCD 16x2*

Para la visualización del menú, se colocó una pantalla LCD 16x2, la cual es comúnmente usada en proyectos de electrónica y su manejo es conocido. Junto con el encoder rotativo, integran la interfaz con el usuario del dispositivo. Permite un control del contraste, el cual es ajustado con un preset desde la placa principal. Como el proyecto consiste en un control de la energía, se agregó como funcionalidad del diseño el apagado del backlight de la pantalla cuando se cumple 1 minuto de inactividad. De esta manera el consumo del dispositivo se reduce (ver sección 3 del informe). La pantalla, de la marca DUAITEK, posee las siguientes características:

- Pantalla de 16 caracteres, 2 líneas
- Caracteres de 5x8 puntos, color blanco.
- Permite mostrar letras, números, caracteres especiales y hasta 8 caracteres personalizados por el usuario.
- Backlight color azul
- Interfaz paralela
- Alimentación 5V

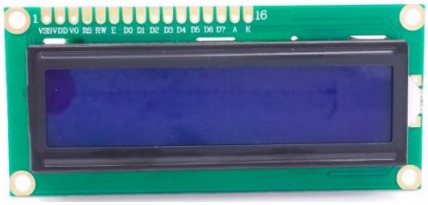

Ilustración 4: Pantalla LCD 16x2

- *Encoder Rotativo*

La segunda parte que integra la interfaz con el usuario, es el encoder rotativo. Este dispositivo permite que usuario pueda ir atravesando las diferentes pantallas que se presentan y de esta manera realizar una correcta configuración del dispositivo. Se utilizó un encoder para simplificar y reducir la cantidad de partes del proyecto, ya que con este se puede generar movimientos hacia la derecha, izquierda y confirmaciones pulsando el switch que incluye. El modelo utilizado es el KY-040 y posee las siguientes características:

- Tipo encoder incremental
- Ciclos por resolución (CPR): 20
- Tensión de trabajo: 0-5V
- Peso: 10g
- Dimensiones: 32x19x30mm

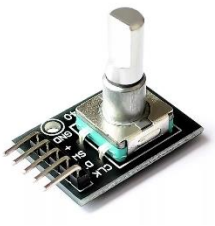

Ilustración 5: Encoder rotativo

- *Módulo RTC (Real Time Clock)*

Para el proyecto era necesario tener como información la fecha y hora actual, así como también la posibilidad de configurarla la primera vez que se utilizaba. Para ello se utilizó un módulo DS3231, el cual es un reloj en tiempo real (RTC) extremadamente preciso y de bajo costo, con un oscilador de cristal con compensación de temperatura (TCXO) y un cristal integrados. Este dispositivo incorpora una entrada de batería, para cuando sea necesario desconectar la fuente de alimentación principal y continúa manteniendo un cronometraje preciso. El oscilador integrado mejora la precisión a largo plazo del dispositivo. RTC mantiene información de segundos, minutos, horas, día, fecha, mes y año. Menos de 31 días del mes, la fecha de finalización se ajustará automáticamente, incluidas las correcciones por año bisiesto. El reloj funciona en la indicación de 24 horas o de banda AM/PM del formato de 12 horas. Proporciona dos despertadores configurables y un calendario que se puede configurar en una salida de onda cuadrada. La dirección y los datos se transfieren en serie a través de un bus bidireccional I2C. A continuación, se enumeran algunas características:

- Tensión de Alimentación: 3,3V – 5V
- RTC de alta precisión DS3231 con oscilador interno
- Exactitud Reloj: 2ppm
- Dirección I2C del DS3132: Read (11010001) Write (11010000)
- Memoria EEPROM AT23C32 (4K*8bit = 32Kbit = 4Kbyte)
- Interfaz I2C
- La batería puede mantener al RTC funcionando por 10 años.

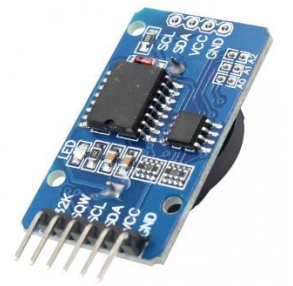

Ilustración 6: Módulo RTC

- *Módulo microSD*

El dispositivo necesita conocer la fecha de los días feriados, así como también la programación semanal con los rangos de trabajo por día. Esto puede configurarse de forma manual, pero teniendo en cuenta una posible reproducción e implementación de este prototipo, aparecía un problema a la hora de la programación de cada uno de los dispositivos, ya que debían ser configurados en forma individual utilizando el encoder rotativo. Esto podría llegar a resultar molesto, pero sobre todo demandaba mucho tiempo. Es por eso que se incluyó un módulo para tarjeta microSD, el cual se cargará con un archivo .txt que contiene los feriados anuales y la programación semanal.

El módulo utilizado cuenta con las siguientes características:

- Soporta tarjetas microSD, microSDHC
- Posee un conversor de nivel para utilizarlo con 3,3V o 5V
- Posee interfaz de comunicación SPI
- Tensión de Alimentación recomendada de 5V
- Dimensiones: 42x23x12mm

Ilustración 7: Módulo microSD

- *Módulo Relay*

La conexión y desconexión del artefacto a controlar estará a cargo del módulo relay, el cual cuenta con transistores para permitir el comando del relay mediante un microcontrolador. Dispone de un contacto simple inversor y la placa presenta un LED indicador de alimentación general. Otras características son:

- Tensión de alimentación: 5V
- Consumo de relay: 50-60mA
- Capaz de manejar una carga de hasta 250VAC-10A/30VDC-10A
- Lógica TTL
- Posee un LED indicador de estado

Ilustración 8: Módulo relay

- *Módulo Interlock*

La alimentación del prototipo será monofásica, para lo cual se empleó un módulo interlock que cuenta con fusible y llave incluidos. De esta manera, utilizando un cable interlock se puede realizar una conexión rápida y segura del equipo.

Ilustración 9: Módulo interlock

- *Tomacorriente*

El lugar para la conexión del artefacto a controlar será en el lateral del dispositivo, a un costado del módulo interlock. Se utilizó un tomacorriente normalizado de 220V-10A de color blanco, embutido en el equipo.

Ilustración 10: Tomacorriente

2.3. Circuito esquemático

El diseño del circuito esquemático fue realizado utilizando el software KiCad versión 7.0.

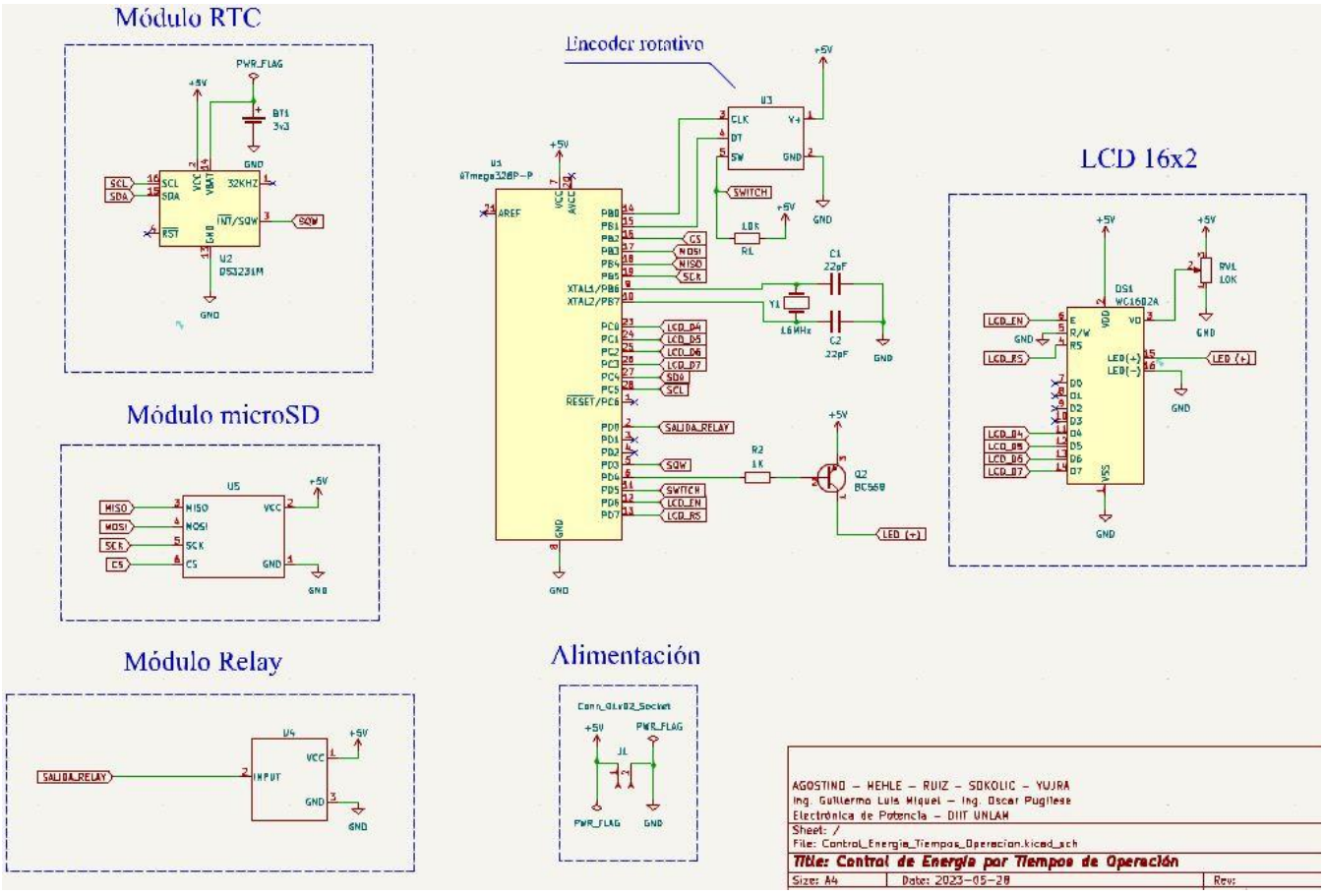

Ilustración 11: Circuito esquemático

2.4. Circuito impreso

El diseño del circuito impreso fue realizado utilizando el software KiCad versión 7.0.

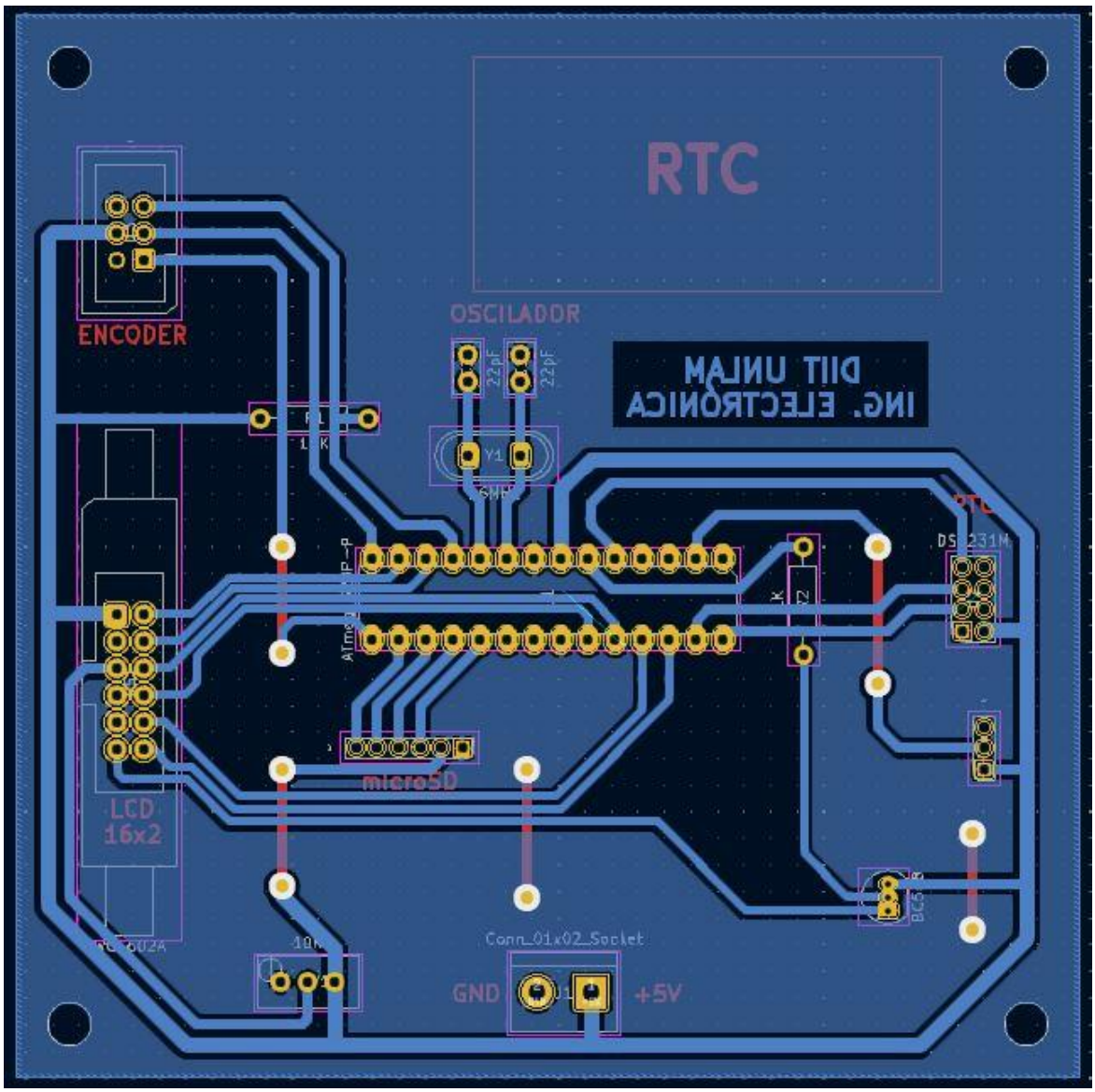

Ilustración 12: Circuito impreso

2.5. Gabinete Plástico

Para el diseño del gabinete plástico, se utilizó el software Fusion 360 de Autodesk. El gabinete consiste en dos partes, la base y una tapa. En la base se encuentra fija la placa principal, la fuente switching, el módulo relay, el tomacorriente y el módulo interlock. Mientras que, en la tapa, se encuentran el LCD, el encoder y el módulo de tarjeta microSD.

El gabinete posee unas guías sobre las cuales la tapa puede deslizarse libremente hasta cubrir la totalidad de la base. Además, se le incluyeron un juego de imanes de neodimio para el cierre de la tapa y evitar el uso de tornillos para el acoplamiento con la base.

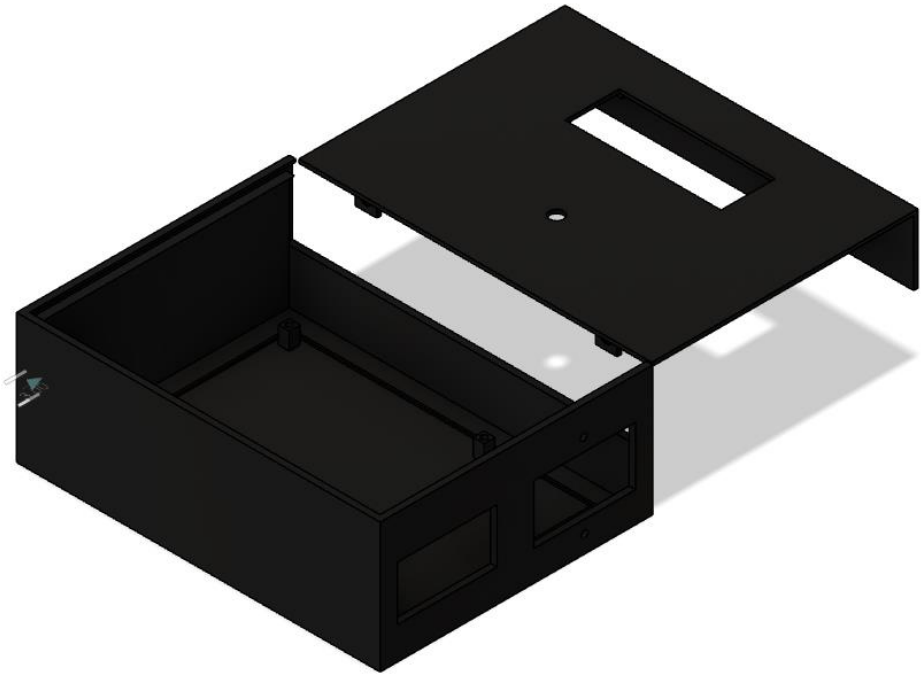

Ilustración 13: Diseño de Gabinete Plástico

2.6. Montaje y ubicación de componentes

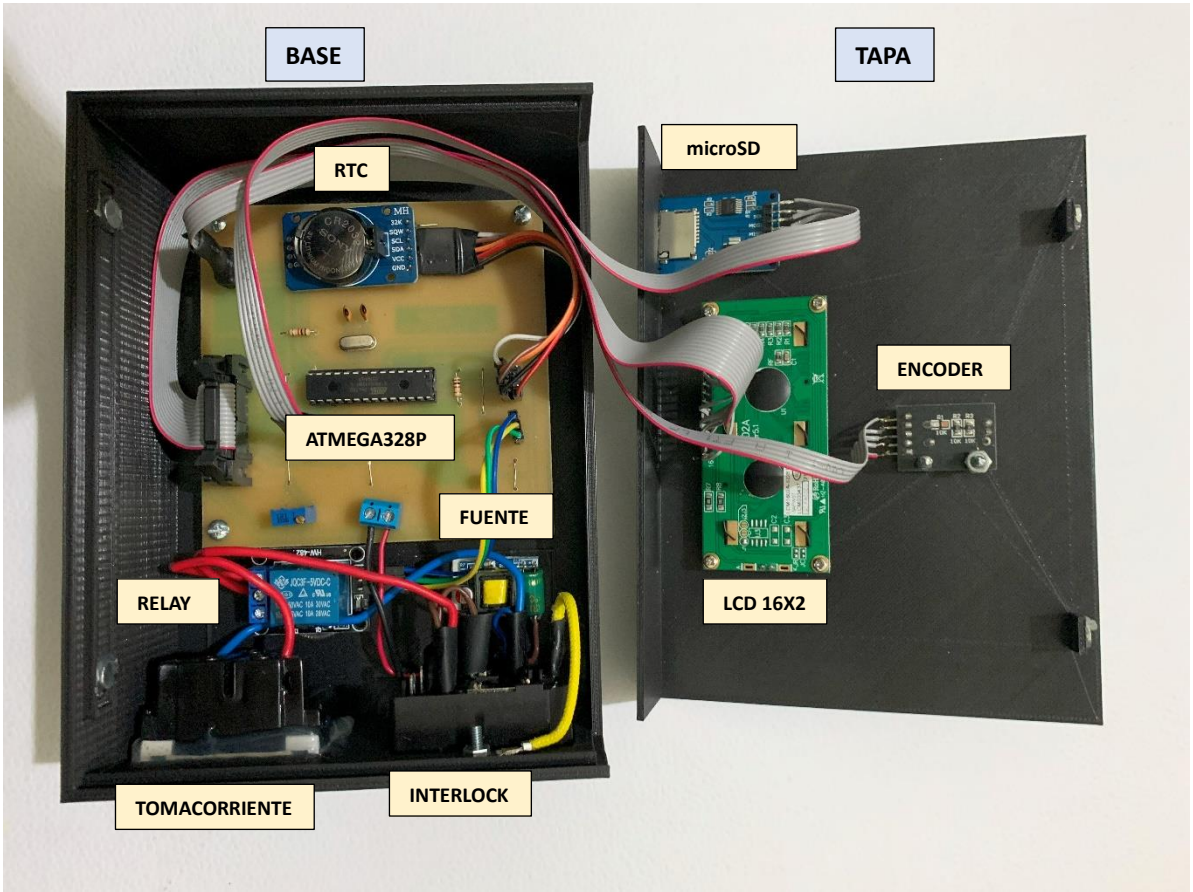

Ilustración 14: Ubicación y montaje

2.7. Costos

A continuación, se muestran los precios de todos los componentes que conforman al producto final, siendo los mismos registrados en junio de 2023.

Componente	Precio	
Fuente Switching 220 VAC 5 VDC 700 mA	AR$	1850,00
ATMEGA328 con zócalo, cristal y capacitores	AR$	2829,00
Display LCD 16x2	AR$	1623,60
Módulo Encoder Rotativo Ky-040	AR$	565,50
Módulo RTC DS3231	AR$	1789,04
Módulo Relay 5v 10A	AR$	798,94
Módulo lector de memorias microSD	AR$	379,00
Módulo Interlock con llave y fusible	AR$	2225,90
Placa Pertinax 10cm x 10cm	AR$	408,27
Transistor NPN BC548	AR$	34,03
Bornera azul de 2 contactos	AR$	126,42
Conector hembra 1x40	AR$	563,93
Conector hembra 2x40	AR$	781,96
Conector hembra 2x7	AR$	253,60
Header macho 2x7	AR$	157,08
Resistencia 10K ohm 1/4 Watt	AR$	17,69
Resistencia 1K ohm 1/4 Watt	AR$	17,69
Preset Potenciómetro Multivuelta 10K ohm	AR$	419,00
Cable plano de 14 conductores	AR$	607,93
Plástico de base del gabinete	AR$	1900,00
Plástico de tapa del gabinete	AR$	760,00
Total	**AR$**	**18108,58**

3. Consumos

A la hora de evaluar el consumo del dispositivo, era importante considerar que, si la carga a controlar era de hasta 10 A, limitada por el relay, el consumo del dispositivo debía ser mucho menor que este valor. Para este análisis, se plantearon 4 posibles condiciones de trabajo a las cuales puede estar sometido el equipo. Las mismas son:

- Que el backlight del LCD y el relay estén activados (**PEOR CONDICIÓN**)
- Que el backlight del LCD esté activado y el relay desactivado
- Que el backlight del LCD esté desactivado y el relay activado
- Que el backlight del LCD y el relay estén desactivados (**MEJOR CONDICIÓN**)

En función de lo planteado, se realizaron las mediciones correspondientes obteniendo los valores que se observan en la Ilustración 15.

		RELAY	
		ON	OFF
BACKLIGHT	ON	110mA	40mA
	OFF	97mA	25mA

Ilustración 15: Tabla de consumos

De esta manera, se puede establecer que el equipo tiene un consumo que varía desde los 25mA hasta los 110mA aproximadamente, lo cual permite afirmar que el consumo del equipo no es representativo en comparación con el consumo del artefacto al cual busca controlar.

4. Firmware

4.1. Código

```
//*********************************************************************
// Archivo:              control_de_energia.ino
//
// Proyecto:             CONTROL DE ENERGÍA POR TIEMPOS DE OPERACIÓN
//
// Materia:              ELECTRÓNICA DE POTENCIA
//
// Fecha de creación:    1er CUATRIMESTRE 2023
//
// Profesores:           Ing. Guillermo Luis Miquel
//                       Ing. Oscar Pugliese
//
//  Autores:             Nicolás Enrique Agostino
//                       Ignacio Luis Mehle
//                       Santiago Ruiz
//                       Federico Ladislao Sokolic
//                       Pablo Fabián Yujra Ventura
//  Descripción:
//    El dispositivo tiene como principal función la conexión y desconexión de
//    forma automática de un artefacto o carga eléctrica monofásica siguiendo
//    un programa semanal configurable, respetando los feriados según el
//    calendario anual vigente. Este proyecto fue desarrollado a lo largo de
//    la cursada de la asignatura "Electrónica de Potencia", en el marco de
//    impulsar una mejora en el consumo sustentable de la Universidad.
//
//*********************************************************************************

#include <LiquidCrystal.h>
#include <EEPROM.h>
#include <DS3231.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#ifdef TIMER_INTERNO
#include <TimerOne.h> //Para simular el reloj del RTC
#endif
```

```
//*****************************************************************
//        ETIQUETAS
//*****************************************************************
//Descomentar para habilitar el debug por puerto serie
//#define DEBUG_SERIE

//Menu
#define MENU_PRINCIPAL          0
#define MENU_AJUSTE_FECHA_HORA  1
#define MENU_CONFIGURACION      2
#define MENU_PROGRAMA_SEMANAL   3
#define MENU_FERIADOS           4
#define MENU_CAMBIOS_SD         5

//Dias de la semana
#define LUNES     1
#define MARTES    2
#define MIERCOLES 3
#define JUEVES    4
#define VIERNES   5
#define SABADO    6
#define DOMINGO   7

//Rangos
#define INICIO_1   0
#define FIN_1      1
#define INICIO_2   2
#define FIN_2      3
#define INICIO_3   4
#define FIN_3      5
#define INICIO_4   6
#define FIN_4      7

#define PRIMERA_POSICION_EVENTOS  0
#define ULTIMA_POSICION_EVENTOS   PRIMERA_POSICION_EVENTOS+55
#define PRIMERA_POSICION_FERIADOS ULTIMA_POSICION_EVENTOS+1
#define ULTIMA_POSICION_FERIADOS  PRIMERA_POSICION_FERIADOS+365

#define MAX_VALOR_HORARIOS  95 //correspondiente a las 23:45

#define CANT_RANGOS 4

//Encoder
#define DERECHA   1
#define IZQUIERDA 2
#define ENTER     3

//Para lectura de EEPROM
#define HORA    0
#define MINUTOS 1

//Para seteo de feriados
#define ES_FERIADO    1
#define NO_ES_FERIADO 0

#define TIEMPO_PARA_APAGAR_LCD  60 //en segundos
```

```
//Pin usado para disparar la interrupción, la salida SQW del RTC debe conectarse a el
pin a usar
#define CLINT 3

#define SSPin 10 //CS de la uSD

//*********************************************************************
//        VARIABLES
//*********************************************************************
const int Encoder_OuputA PROGMEM = 9;
const int Encoder_OuputB PROGMEM = 8;
const int Encoder_Switch PROGMEM = 5;
const int Salida_Rele PROGMEM = 0;
const int Salida_Backlight PROGMEM = 4;

int Previous_Output;
int diadelasemana=0;
int estado_menu=0;
int estado_menu_anterior=0;
int estado_ajuste=0;
int estado_prog_sem=0;
int estado_prog_rango=0;
int estado_menu_feriados=0;
int proximo_menu=0;
int seg_anterior=0;
int rango_a_programar=0;
char Rango_num[CANT_RANGOS*2];
char Rangos_hoy[CANT_RANGOS*2];
char codigo_hora_actual=0;
char codigo_hora_actual_anterior=0;
char ultimo_dia=0;
bool modificacion_realizada=0;
bool cursor_feriado=0;
bool feriado=0;
int tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;

//ESTRUCTURA DE FECHA Y HORA
struct RTC_Time
{
  //Dia de la semana
  char DoW;
  //FECHA
  char dia; //[1,31]
  char mes; //[1,12]
  char anio; //[0,255]
  //HORA
  char seg; //[0,59]
  char min; //[0,59]
  char hora; //[0,23]
};

RTC_Time ActualTime;
RTC_Time AuxTime;
```

```
// Setup clock
DS3231 RTC;

//Estructura para el archivo de la uSD
//File myFile;


volatile byte tick = 1;

byte alarmBits = 0b00001110; // Cada un min

bool Century  = false;
bool h12=0;//Modo 24HS
bool PM ;

//Cantidad de dias por mes, usado para calcular la posicion en memoria
const int cant_dias_mes[] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

//Mention the pin number for LCD connection
const int rs PROGMEM = 7;
const int en PROGMEM = 6;
const int d4 PROGMEM = 14;
const int d5 PROGMEM = 15;
const int d6 PROGMEM = 16;
const int d7 PROGMEM = 17;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int min_ant,seg_ant;

bool estado_uSD=0;
//*********************************************************************
// Función:                                    setup
//
// Descripción:              Configuración inicial del programa
//
//*********************************************************************
void setup()
{
  /***Letras seleccionadas***/
  byte L[8] = {
  0b01111,
  0b01111,
  0b01111,
  0b01111,
  0b01111,
  0b01111,
  0b00000,
  0b11111
  };

  byte M[8] = {
  0b01110,
  0b00100,
  0b01010,
  0b01010,
```

```
0b01110,
0b01110,
0b01110,
0b11111
};

byte a[8] = {
0b11111,
0b11111,
0b10001,
0b11110,
0b10000,
0b01110,
0b10000,
0b11111
};

byte i[8] = {
0b11011,
0b11111,
0b10011,
0b11011,
0b11011,
0b11011,
0b10001,
0b11111
};

byte J[8] = {
0b10000,
0b11101,
0b11101,
0b11101,
0b11101,
0b01101,
0b10011,
0b11111
};

byte V[8] = {
0b01110,
0b01110,
0b01110,
0b01110,
0b01110,
0b10101,
0b11011,
0b11111
};

byte S[8] = {
0b10001,
0b01110,
0b01111,
0b10001,
0b11110,
```

```
  0b01110,
  0b10001,
  0b11111
  };

  byte D[8] = {
  0b00011,
  0b01101,
  0b01110,
  0b01110,
  0b01110,
  0b01101,
  0b00011,
  0b11111
  };
  // Begin I2C communication
  Wire.begin();

  //Interrupción de RTC
  configuraInterrupcionRTC();

  // put your setup code here, to run once:
  lcd.begin(16, 2); //Initialise 16*2 LCD

  lcd.createChar(0,D);
  lcd.createChar(1,L);
  lcd.createChar(2,M);
  lcd.createChar(3,a);
  lcd.createChar(4,i);
  lcd.createChar(5,J);
  lcd.createChar(6,V);
  lcd.createChar(7,S);

#ifdef DEBUG_SERIE
  Serial.begin(9600);
#endif

  pinMode (Encoder_OuputA, INPUT);
  pinMode (Encoder_OuputB, INPUT);
  pinMode (Encoder_Switch, INPUT);
  Previous_Output = digitalRead(Encoder_OuputA); //Read the inital value of Output A
  pinMode (Salida_Rele, OUTPUT);
  pinMode (Salida_Backlight, OUTPUT);

  Leer_Fecha_Hora_RTC(ActualTime);

#ifdef TIMER_INTERNO
  //Para tener un timer de 1seg
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(ISR_Segundo);
#endif

  Rango_num[INICIO_1]=Rango_num[FIN_1]='1';
  Rango_num[INICIO_2]=Rango_num[FIN_2]='2';
  Rango_num[INICIO_3]=Rango_num[FIN_3]='3';
  Rango_num[INICIO_4]=Rango_num[FIN_4]='4';
```

```
  //Cargo los rangos del día de hoy
  Cargar_Rangos();
  ultimo_dia=ActualTime.DoW;

  tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;

  //Lee si hay feriados en la uSD para agregar
  leerFeriadosDeSd();

}

//**********************************************************************
// Función:                  leerFeriadosDeSd
//
// Descripción: Lee los feriados y eventos de la uSD y los agrega en la EEPROM
//
// Parámetros:               void
// Valor devuelto: void
//
//**********************************************************************
void leerFeriadosDeSd(void)
{
#ifdef DEBUG_SERIE
  char BufferAux[25];
#endif
  //Estructura para el archivo de la uSD
  File myFile;
  int feriado_anterior=PRIMERA_POSICION_FERIADOS;
  bool hubo_cambio=0;

#ifdef DEBUG_SERIE
  Serial.print(F("Inicializando tarjeta...\n"));
  delay(50);
#endif

  if (!SD.begin(SSPin))
  {
    estado_uSD=0;
#ifdef DEBUG_SERIE
    Serial.print(F("Falló la inicialización de la uSD\n"));
    delay(50);
#endif
  }
  else
  {
    //Si entra acá es porque está la uSD presente
    //La leo solo si no la leí previamente desde que está colocada

    if(!estado_uSD)
    {
      //Abre el archivo de feriados
      myFile = SD.open("feriados.txt");

      if(myFile)
      {
```

```
        String feriados;
        String eventos;

        estado_uSD = 1;

#ifdef DEBUG_SERIE
        Serial.print(F("if TRUE\n"));
        delay(50);
#endif

//      while (myFile.available())
//      {
          //Feriados
          feriados = myFile.readStringUntil('|');

          int feriados_len = feriados.length() + 1;
          String evento = "";
          int eventos_len;

          char char_feriados[feriados_len];
          feriados.toCharArray(char_feriados, feriados_len);

          String feriado = "";

          for (int i = 0; i < feriados.length(); i++)
          {
            if (feriados[i] == ',')
            {
              if((feriado.toInt()>=PRIMERA_POSICION_FERIADOS)&&
              (feriado.toInt()<=ULTIMA_POSICION_FERIADOS))
              {
                if(EEPROM.read(feriado.toInt())!=1)
                {
                  EEPROM.write(feriado.toInt(), 1);
                  hubo_cambio=1;
#ifdef DEBUG_SERIE
                  Serial.print(F("Lo guardo!"));
#endif
                }
#ifdef DEBUG_SERIE
                sprintf(BufferAux,"feriado: %d \n",feriado.toInt());
                Serial.print(BufferAux);
#endif
                //Borro todas las posiciones que no estén en la microSD
                for(int j=feriado_anterior;j<feriado.toInt();j++)
                {
                  if(EEPROM.read(j)!=0)
                  {
                    EEPROM.write(j, 0);
                    hubo_cambio=1;
#ifdef DEBUG_SERIE
                    Serial.print(F("Lo borro!"));
#endif
                  }
                }
```

```
                feriado_anterior = feriado.toInt()+1;
              }
              feriado = "";
            }
            else
            {
              feriado += feriados[i];

#ifdef DEBUG_SERIE
              Serial.print(feriado);
              Serial.print(F("\n"));
#endif
            }
          }


if((feriado_anterior>PRIMERA_POSICION_FERIADOS)&&(feriado_anterior<=ULTIMA_POSICION_FERI
ADOS))
          {
            //Borro todas las posiciones que faltan de la uSD
            for(int i=feriado_anterior;i<=ULTIMA_POSICION_FERIADOS;i++)
            {
              if(EEPROM.read(i)!=0)
                EEPROM.write(i, 0);
            }
          }

        while (myFile.available())
        {
          eventos = "";
          evento = "";
          //Eventos
          eventos = myFile.readStringUntil(',');

          //if (eventos[i] != '.')
          if (eventos[0] != '.')
          {
            eventos += ',';
            eventos_len = eventos.length() + 1;

#ifdef DEBUG_SERIE
            Serial.print(eventos);

            delay(100);

            sprintf(BufferAux,"\n eventos_len: %d\n",eventos_len);
            Serial.print(BufferAux);
#endif


            //char char_eventos[eventos_len];
            //eventos.toCharArray(char_eventos, eventos_len);

            for (int i = 0; i < eventos.length(); i++)
            {
```

```
#ifdef DEBUG_SERIE
              sprintf(BufferAux,"eventos[%d]: %c\n",i,eventos[i]);
              Serial.print(BufferAux);
#endif
              if (eventos[i] != ',')
              {
                //Mientras no llegue la coma, armo el string
                evento += eventos[i];

#ifdef DEBUG_SERIE
                Serial.print(evento);
                Serial.print(F("\n"));
#endif
              }
              else
              {
                String posicionEnMemoriaEvento = evento.substring(0,2); //Agarro la
posicion (antes del ':')
                String valorAGuardarEnPosicion = evento.substring(3,5);   //Agarro el
codigo a guardar (despues del ':')

#ifdef DEBUG_SERIE
                sprintf(BufferAux,"posicionEnMemoriaEvento:
%d\n",posicionEnMemoriaEvento.toInt());
                Serial.print(BufferAux);
                sprintf(BufferAux,"valorAGuardarEnPosicion:
%d\n",valorAGuardarEnPosicion.toInt());
                Serial.print(BufferAux);
#endif


if((posicionEnMemoriaEvento.toInt()>=PRIMERA_POSICION_EVENTOS)&&(posicionEnMemoriaEvento
.toInt()<=ULTIMA_POSICION_EVENTOS)&&(valorAGuardarEnPosicion.toInt()<=
MAX_VALOR_HORARIOS))
                {
                  //Si es valido, me fijo si es diferente a lo que ya está guardado

if(EEPROM.read(posicionEnMemoriaEvento.toInt())!=valorAGuardarEnPosicion.toInt())
                  {
                    //Si es diferente, lo guardo
                    EEPROM.write(posicionEnMemoriaEvento.toInt(),
valorAGuardarEnPosicion.toInt());
                    hubo_cambio = 1;

#ifdef DEBUG_SERIE
                    Serial.print(F("Lo guardo!"));
#endif
                  }
                }

                evento = "";
              }
            }
          }
        }
```

```
        if(hubo_cambio)
          estado_menu = MENU_CAMBIOS_SD;

        // Cerrar el archivo
        myFile.close();

      }
      else
      {
        estado_uSD=0;
#ifdef DEBUG_SERIE
        Serial.print(F("Error leyendo feriados.txt"));
#endif
      }
    }
  }
}

//*****************************************************************
// Función:                 Leer_Fecha_Hora_RTC
//
// Descripción:    Carga en la estructura la fecha y hora guardada en el RTC
//
// Parámetros:     RTC_Time Time (estructura a cargar)
// Valor devuelto: void
//
//*****************************************************************
void Leer_Fecha_Hora_RTC(RTC_Time& Time)
{
  //Leo los datos del RTC
  Time.DoW=RTC.getDoW();
  //FECHA
  Time.dia=RTC.getDate();
  Time.mes=RTC.getMonth(Century);
  Time.anio=RTC.getYear();
  //HORA
  Time.hora=RTC.getHour(h12, PM);
  Time.min=RTC.getMinute();
  Time.seg=RTC.getSecond();
}

//*****************************************************************
// Función:                 Cargar_Rangos
//
// Descripción:             Carga los rangos de encendido del día de hoy
//
// Parámetros:              void
// Valor devuelto: void
//
//*****************************************************************
void Cargar_Rangos(void)
{
#ifdef DEBUG_SERIE
  char BufferAux[50];
#endif
```

```
  for(int i=0; i<CANT_RANGOS*2; i++)
  {
    Rangos_hoy[i]=EEPROM.read((ActualTime.DoW-1)*8+i);

#ifdef DEBUG_SERIE
      if(i%2==0)
        Serial.print(F("Inicio "));
      else
        Serial.print(F("Fin    "));

      sprintf(BufferAux,"%c: %d\n",Rango_num[i],Rangos_hoy[i]);
      Serial.print(BufferAux);
#endif
  }
}

//*****************************************************************
// Función:            Leer_Encoder
//
// Descripción:       Devuelve la dirección del enconder o si se apretó el
//                   pulsador, teniendo prioridad el pulsador
//
// Parámetros:               void
// Valor devuelto: int  IZQUIERDA(antihorario), DERECHA(horario) o ENTER
//
//*****************************************************************
int Leer_Encoder(void)
{
  int giro_encoder=0;

  if (digitalRead(Encoder_OuputB) != Previous_Output)
  {
    if (digitalRead(Encoder_OuputA) != Previous_Output)
    {
      giro_encoder = DERECHA;
      tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;
    }
    else
    {
      giro_encoder = IZQUIERDA;
      tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;
    }
  }

  Previous_Output = digitalRead(Encoder_OuputB);

  //El ENTER tiene prioridad ante los giros
  if(digitalRead(Encoder_Switch)==0)
  {
    while(digitalRead(Encoder_Switch)==0)
    {
      //Espero a que suelte el pulsador
    };

    if(tiempo_sin_pulsar>0)  //Si está el backlight apagado, no realiza ninguna acción
el enter, solo prende el backlight
```

```
      return(ENTER);

    tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;
  }

  return(giro_encoder);
}

//*****************************************************************
// Función:                  Programacion_Semanal
//
// Descripción:    Se programan los rangos de encendido de cada día de la semana
//
// Parámetros:               void
// Valor devuelto: void
//
//*****************************************************************
void Programacion_Semanal(void)
{

  char BufferAux[5];

  switch(estado_prog_sem)
  {
    default:
      estado_prog_sem=0;
    break;

    case 0:
      lcd.setCursor(0, 0);
      lcd.print(F("Seleccionar dia"));
      lcd.setCursor(0, 1);
      lcd.print(F("D L Ma Mi J V S"));

      //Selecciono DOMINGO
      //lcd.setCursor(0, 1);
      //lcd.write(byte(0));//D seleccionada
      //diadelasemana=DOMINGO;

      //delay(1000);

      diadelasemana=ActualTime.DoW;

      estado_prog_sem++;
    break;

    case 1:
      switch(Leer_Encoder())
      {
        default:
          break;

        case DERECHA:
          diadelasemana ++;
          if(diadelasemana > DOMINGO)
            diadelasemana = LUNES;
```

```
#ifdef DEBUG_SERIE
              Serial.print(F("Variable: "));
              Serial.print(diadelasemana);
              Serial.print(F("\n"));
#endif

          delay(50);
        break;

        case IZQUIERDA:
          diadelasemana--;
          if(diadelasemana < LUNES)
            diadelasemana = DOMINGO;

#ifdef DEBUG_SERIE
              Serial.print(F("Variable: "));
              Serial.print(diadelasemana);
              Serial.print(F("\n"));
#endif

          delay(50);
        break;

        case ENTER:
          estado_prog_sem++;
        break;
      }

      Seleccion_dias();

      delay(10);
    break;

    case 2:
    //Presento el día en pantalla
    lcd.clear();
    lcd.setCursor(0, 0);
    switch(diadelasemana)
    {
      default:
        diadelasemana=DOMINGO;
        lcd.print(F("DOMINGO"));
      break;

      case DOMINGO:
        lcd.print(F("DOMINGO"));
      break;

      case LUNES:
        lcd.print(F("LUNES"));
      break;

      case MARTES:
        lcd.print(F("MARTES"));
      break;
```

```
      case MIERCOLES:
        lcd.print(F("MIERCOLES"));
      break;

      case JUEVES:
        lcd.print(F("JUEVES"));
      break;

      case VIERNES:
        lcd.print(F("VIERNES"));
      break;

      case SABADO:
        lcd.print(F("SABADO"));
      break;
    }


    estado_prog_sem++;
    rango_a_programar=0;
    estado_prog_rango=0;
    break;

    case 3:
      if(Programar_Rango(rango_a_programar)==1)
        rango_a_programar++;

      if(rango_a_programar >= (CANT_RANGOS*2))
        estado_prog_sem++;
    break;

    case 4:
      //Trato de guardar los cambios si hay uSD
      Guardar_Cambios_uSD();

      lcd.setCursor(0, 0);
      lcd.print(F("Config guardada "));
      lcd.setCursor(0, 1);
      lcd.print(F("   con exito    "));
      delay(3000);

      estado_menu=MENU_PRINCIPAL;
      estado_prog_sem=0;
      diadelasemana=0;
      Cargar_Rangos();
      modificacion_realizada=1;
    break;
  }
}

//*****************************************************************
// Función:                  Programar_Rango
//
// Descripción:    Se programan las horas y minutos de cada rango
//
```

```
// Parámetros:       int rango: INICIO1,FIN1,INICIO2,FIN2...etc
// Valor devuelto: bool: devuelve un 1 cuando ya se programó ese rango, sino 0
//
//*********************************************************************
bool Programar_Rango(int rango)
{
  //Por cada rango a programar va a entrar acá
  char BufferAux[5];

  switch(estado_prog_rango)
  {
    default:
      estado_prog_rango=0;
    break;

    case 0:
      lcd.setCursor(10, 0);
      lcd.print(F("Rango"));
      lcd.setCursor(15, 0);
      lcd.print(Rango_num[rango]);

      if(rango%2==0)
      {
        lcd.setCursor(0, 1);
        lcd.print(F("Encendido:"));//Rangos pares son encedido
      }
      else
      {
        lcd.setCursor(0, 1);
        lcd.print(F("Apagado:  "));//Rangos impares son apagado
      }

      AuxTime.hora = Leer_Rangos_EEPROM(diadelasemana, rango, HORA);
      AuxTime.min  = Leer_Rangos_EEPROM(diadelasemana, rango, MINUTOS);

      lcd.setCursor(11, 1);
      if((AuxTime.hora<0 || AuxTime.hora>23)||(AuxTime.min<0 || AuxTime.min>45))
      {
        AuxTime.hora = 0;
        AuxTime.min  = 0;
      }
      else
      {
        sprintf(BufferAux,"%02d:%02d",AuxTime.hora,AuxTime.min);
        lcd.print(BufferAux);
      }
      estado_prog_rango++;
    break;

    case 1:
      //Programo la hora
      lcd.setCursor(11, 1);
      sprintf(BufferAux,"%02d:%02d",AuxTime.hora,AuxTime.min);
      lcd.print(BufferAux);
      Efecto_Titilar(11,1,2,100);
```

```
    switch(Leer_Encoder())
    {
      default:
      break;

      case IZQUIERDA:
        AuxTime.hora--;
        if(AuxTime.hora<0)
          AuxTime.hora=23;
      break;

      case DERECHA:
        AuxTime.hora++;
        if(AuxTime.hora>23)
          AuxTime.hora=0;
      break;

      case ENTER:
        estado_prog_rango++;
      break;
    }
  break;

  case 2:
    //Programo los minutos
    lcd.setCursor(11, 1);
    sprintf(BufferAux,"%02d:%02d",AuxTime.hora,AuxTime.min);
    lcd.print(BufferAux);
    Efecto_Titilar(14,1,2,100);

    switch(Leer_Encoder())
    {
      default:
      break;

      case IZQUIERDA:
        AuxTime.min=AuxTime.min-15;
        if(AuxTime.min<0)
          AuxTime.min=45;
      break;

      case DERECHA:
        AuxTime.min=AuxTime.min+15;
        if(AuxTime.min>45)
          AuxTime.min=0;
      break;

      case ENTER:
        estado_prog_rango++;
      break;
    }
  break;

  case 3:
    //Guardo el rango en EEPROM
```

```
      #warning "Habria que hacer una logica que se fije que el inicio sea antes del fin
en cada rango, y que no se superpongan"

      Escribir_Rangos_EEPROM(diadelasemana,rango,AuxTime.hora,AuxTime.min);


      estado_prog_rango=0;
      return(1);
    break;

  }
  return(0);
}

//*********************************************************************
// Función:                  Leer_Rangos_EEPROM
//
// Descripción:   Se leen las horas o minutos (según se solicite) guardado
//                  en la EEPROM del día solicitado en el rango solicitado.
//
// Parámetros:        int dia (LUNES, MARTES... etc)
//                  int rango (INICIO1, FIN1, INICIO2... etc)
// Valor devuelto: bool hora_min: aclaras si queres recibir la hora o los mins
//                  (HORA o MINUTOS)
//
//*********************************************************************
char Leer_Rangos_EEPROM(int dia, int rango, bool hora_min)
{
  char codigo_hora=0;
  int direccion_eeprom=0;

  //Leo la hora y fecha programada
  direccion_eeprom= (dia-1)*8+rango;
  codigo_hora = EEPROM.read(direccion_eeprom);

  if(hora_min == HORA)
    return(codigo_hora/4);
  else //if(hora_min == MINUTOS)
  {
    codigo_hora=codigo_hora%4; //Agarro el resto (de 0 a 3)
    codigo_hora=codigo_hora*15;//Lo multiplico por 15
    return(codigo_hora);
  }

}

//*********************************************************************
// Función:                  Escribir_Rangos_EEPROM
//
// Descripción:    Se escribe en la EEPROM el código generado por las horas y
//               minutos en la dirección generada por el día y rango indicados
//
// Parámetros:        int dia (LUNES, MARTES... etc)
//                  int rango (INICIO1, FIN1, INICIO2... etc)
//                  int hora
//                  int min (múltiplo de 15 preferentemente)
```

```
//
// Valor devuelto: void
//
//*********************************************************************
void Escribir_Rangos_EEPROM(int dia, int rango, int hora, int min)
{
  char codigo_hora=0;
  int direccion_eeprom=0;

  direccion_eeprom= (dia-1)*8+rango;

  codigo_hora=((hora*4)+(min/15));

  EEPROM.write(direccion_eeprom,codigo_hora);

}

//*********************************************************************
// Función:                  Seleccion_dias
//
// Descripción:  Se utiliza el encoder para seleccionar el día de la semana
//
// Parámetros:               void
//
// Valor devuelto: void
//
//*********************************************************************
void Seleccion_dias(void)
{
  switch(diadelasemana)
  {
    default:
      diadelasemana=DOMINGO;
    break;

    case DOMINGO:
      //Selecciono DOMINGO
      lcd.setCursor(0, 1);
      lcd.write(byte(0));//D seleccionada
      //Deselecciono resto de dias

      //Deselecciono SABADO
      lcd.setCursor(14, 1);
      lcd.print(F("S"));
      //Deseleccionar_dias(diadelasemana);

      //Deselecciono LUNES
      lcd.setCursor(2, 1);
      lcd.print(F("L"));
    break;

    case LUNES:
      //Selecciono LUNES
      lcd.setCursor(2, 1);
      lcd.write(byte(1));//L seleccionada
```

```
    //Deselecciono DOMINGO
    lcd.setCursor(0, 1);
    lcd.print(F("D"));

    //Deselecciono MARTES
    lcd.setCursor(4, 1);
    lcd.print(F("Ma"));
  break;

  case MARTES:
    //Selecciono MARTES
    lcd.setCursor(4, 1);
    lcd.write(byte(2));//M seleccionada
    lcd.write(byte(3));//a seleccionada

    //Deselecciono LUNES
    lcd.setCursor(2, 1);
    lcd.print(F("L"));

    //Deselecciono MIERCOLES
    lcd.setCursor(7, 1);
    lcd.print(F("Mi"));
  break;

  case MIERCOLES:
    //Selecciono MIERCOLES
    lcd.setCursor(7, 1);
    lcd.write(byte(2));//M seleccionada
    lcd.write(byte(4));//i seleccionada

    //Deselecciono MARTES
    lcd.setCursor(4, 1);
    lcd.print(F("Ma"));

    //Deselecciono JUEVES
    lcd.setCursor(10, 1);
    lcd.print(F("J"));
  break;

  case JUEVES:
    //Selecciono JUEVES
    lcd.setCursor(10, 1);
    lcd.write(byte(5));//J seleccionada

    //Deselecciono MIERCOLES
    lcd.setCursor(7, 1);
    lcd.print(F("Mi"));

    //Deselecciono VIERNES
    lcd.setCursor(12, 1);
    lcd.print(F("V"));
  break;

  case VIERNES:
    //Selecciono VIERNES
    lcd.setCursor(12, 1);
```

```
      lcd.write(byte(6));//V seleccionada

      //Deselecciono JUEVES
      lcd.setCursor(10, 1);
      lcd.print(F("J"));

      //Deselecciono SABADO
      lcd.setCursor(14, 1);
      lcd.print(F("S"));
    break;

    case SABADO:
      //Selecciono SABADO
      lcd.setCursor(14, 1);
      lcd.write(byte(7));//S seleccionada

      //Deselecciono VIERNES
      lcd.setCursor(12, 1);
      lcd.print(F("V"));

      //Deselecciono DOMINGO
      lcd.setCursor(0, 1);
      lcd.print(F("D"));
    break;
  }
}

//*****************************************************************
// Función:         Menu_Principal
//
// Descripción:    Muestra la fecha y la hora y utiliza el encoder para elegir
//                  ajustar la hora/fecha o configurar la programación
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*****************************************************************
void Menu_Principal(void)
{

  char BufferAux[16];

  //Muestro la fecha y hora actual

  lcd.setCursor(0, 0);
  switch(ActualTime.DoW)
  {
    default:
      //Distingo si me trae basura
      lcd.print(F("?"));
    break;

    case DOMINGO:
      lcd.print(F("D"));
    break;
```

```
    case LUNES:
      lcd.print(F("L"));
    break;

    case MARTES:
      lcd.print(F("M"));
    break;

    case MIERCOLES:
      lcd.print(F("X"));
    break;

    case JUEVES:
      lcd.print(F("J"));
    break;

    case VIERNES:
      lcd.print(F("V"));
    break;

    case SABADO:
      lcd.print(F("S"));
    break;
  }

  lcd.setCursor(1, 0);
  sprintf(BufferAux," %02d/%02d/%02d
%02d:%02d",ActualTime.dia,ActualTime.mes,ActualTime.anio,ActualTime.hora,ActualTime.min)
;
  lcd.print(BufferAux);

  lcd.setCursor(1, 1);
  lcd.print(F("Reloj"));
  lcd.setCursor(10, 1);
  lcd.print(F("Config"));


  //Para la primera vez que entra aca
  //if((proximo_menu != MENU_CONFIGURACION)&&(proximo_menu != MENU_AJUSTE_FECHA_HORA))
  if(proximo_menu != MENU_CONFIGURACION)
  {
    proximo_menu = MENU_AJUSTE_FECHA_HORA;
    lcd.setCursor(0, 1);
    //lcd.write(byte(8));//Flecha señalando Reloj
    lcd.print(F(">"));
  }

/*
#ifdef DEBUG_SERIE
    Serial.print(F("proximo_menu:"));
    Serial.println(proximo_menu);
#endif
*/

  Seleccion_Principal();
```

```
  //Hago titilar el ':' para que se vea que está vivo
  Efecto_Titilar(13,0,1,50);
}

//*****************************************************************
// Función:        Seleccion_Principal
//
// Descripción:    Dibuja una flecha en el menú seleccionado dentro del
//                  principal. Si se presiona ENTER, entra a ese menú.
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*****************************************************************
void Seleccion_Principal(void)
{
  switch(Leer_Encoder())
  {
    default:
      break;

    case DERECHA:
      lcd.setCursor(0, 1);
      lcd.print(F(" "));
      lcd.setCursor(9, 1);
      //lcd.write(byte(8));//Flecha señalando Config
      lcd.print(F(">"));
      proximo_menu = MENU_CONFIGURACION;
      delay(250);
    break;

    case IZQUIERDA:
      lcd.setCursor(0, 1);
      //lcd.write(byte(8));//Flecha señalando Reloj
      lcd.print(F(">"));
      lcd.setCursor(9, 1);
      lcd.print(F(" "));
      proximo_menu = MENU_AJUSTE_FECHA_HORA;
      delay(250);
    break;

    case ENTER:
      estado_menu = proximo_menu;
    break;
  }
}

//*****************************************************************
// Función:        Seleccion_Configuracion
//
// Descripción:    Muestra el menú Configuración y dibuja una flecha en el menu
//                  seleccionado. Si se presiona ENTER, entra a ese menu.
//
// Parámetros:                      void
```

```
//
// Valor devuelto: void
//
//*********************************************************************
void Seleccion_Configuracion(void)
{
  lcd.setCursor(1, 0);
  lcd.print(F("Prog. semanal"));

  lcd.setCursor(1, 1);
  lcd.print(F("Feriados"));

  //Para la primera vez que entra aca
  if((proximo_menu != MENU_PROGRAMA_SEMANAL)&&(proximo_menu != MENU_FERIADOS))
  {
    lcd.setCursor(0, 0);
    //lcd.write(byte(8));//Flecha señalando Prog Semanal
    lcd.print(F(">"));
    lcd.setCursor(0, 1);
    lcd.print(F(" "));
    proximo_menu = MENU_PROGRAMA_SEMANAL;

    delay(1000);
  }

  switch(Leer_Encoder())
  {
    default:
      break;

    case IZQUIERDA:
      lcd.setCursor(0, 0);
      //lcd.write(byte(8));//Flecha señalando Prog Semanal
      lcd.print(F(">"));
      lcd.setCursor(0, 1);
      lcd.print(F(" "));
      proximo_menu = MENU_PROGRAMA_SEMANAL;
      delay(250);
    break;

    case DERECHA:
      lcd.setCursor(0, 0);
      lcd.print(F(" "));
      lcd.setCursor(0, 1);
      //lcd.write(byte(8));//Flecha señalando Feriados
      lcd.print(F(">"));
      proximo_menu = MENU_FERIADOS;
      delay(250);
    break;

    case ENTER:
      estado_menu = proximo_menu;
    break;
  }

}
```

```
//*********************************************************************
// Función:        Reloj
//
// Descripción:    Reloj en base al timer de 1seg para simular el RTC
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*********************************************************************
  /*
void Reloj(void){
  if(ActualTime.seg >59)
  {
    ActualTime.seg=0;
    ActualTime.min++;
    if(ActualTime.min >59)
    {
      ActualTime.min=0;
      ActualTime.hora++;
      if(ActualTime.hora>23)
        ActualTime.hora=0;  //Faltaría cambiar de día
    }
  }
}
*/
#ifdef TIMER_INTERNO
//*********************************************************************
// Función:        ISR_Segundo
//
// Descripción:    Timer de 1seg
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*********************************************************************
void ISR_Segundo(void){

  ActualTime.seg++;

  if(tiempo_sin_pulsar>0)
    tiempo_sin_pulsar--;

}
#endif
//*********************************************************************
// Función:        Ajuste_Reloj
//
// Descripción:    Menú para modificar la hora y fecha actual usando el encoder
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*********************************************************************
```

```
void Ajuste_Reloj(void)
{
  char BufferAux[16];

  switch(estado_ajuste)
  {
    default:
      estado_ajuste=0;
    break;

    case 0:
      lcd.setCursor(0, 0);
      lcd.print(F("Seleccione hora "));
      lcd.setCursor(0, 1);
      lcd.print(F("y fecha actual  "));
      delay(3000);
      estado_ajuste++;
    break;

    case 1:
      //Muestro la fecha y hora actual

      lcd.setCursor(0, 0);
      lcd.print(F("Dia: "));

      switch(ActualTime.DoW)
      {
        default:
          //Distingo si me trae basura
          lcd.print(F("?? "));
        break;

        case DOMINGO:
          lcd.print(F("Do "));
        break;

        case LUNES:
          lcd.print(F("Lu "));
        break;

        case MARTES:
          lcd.print(F("Ma "));
        break;

        case MIERCOLES:
          lcd.print(F("Mi "));
        break;

        case JUEVES:
          lcd.print(F("Ju "));
        break;

        case VIERNES:
          lcd.print(F("Vi "));
        break;
```

```
      case SABADO:
        lcd.print(F("Sa "));
      break;
    }

    sprintf(BufferAux,"%02d/%02d/%02d",ActualTime.dia,ActualTime.mes,ActualTime.anio);
    lcd.print(BufferAux);

    lcd.setCursor(0, 1);
    sprintf(BufferAux,"Hora:   %02d:%02d     ",ActualTime.hora,ActualTime.min);
    lcd.print(BufferAux);

    AuxTime = ActualTime;//Copio la estructura para modificarla

    estado_ajuste++;
  break;

  case 2:
    //Modifico el día de la semana
    lcd.setCursor(5, 0);

    switch(AuxTime.DoW)
    {
      default:
        //Distingo si me trae basura
        lcd.print(F("?? "));
      break;

      case DOMINGO:
        lcd.print(F("Do "));
      break;

      case LUNES:
        lcd.print(F("Lu "));
      break;

      case MARTES:
        lcd.print(F("Ma "));
      break;

      case MIERCOLES:
        lcd.print(F("Mi "));
      break;

      case JUEVES:
        lcd.print(F("Ju "));
      break;

      case VIERNES:
        lcd.print(F("Vi "));
      break;

      case SABADO:
        lcd.print(F("Sa "));
      break;
    }
```

```
    if(digitalRead(Encoder_Switch)==0)//ENTER
    {
      estado_ajuste++;
      while(digitalRead(Encoder_Switch)==0)
      {
        //Espero a que suelte el pulsador
      };
    }
    else
    {
      AuxTime.DoW=Modificar_Variable(AuxTime.DoW,1,7);

      Efecto_Titilar(5,0,2,100);
    }
  break;

  case 3:
    //Modifico el día
    lcd.setCursor(8, 0);
    sprintf(BufferAux,"%02d",AuxTime.dia);
    lcd.print(BufferAux);

    if(digitalRead(Encoder_Switch)==0)//ENTER
    {
      estado_ajuste++;
      while(digitalRead(Encoder_Switch)==0)
      {
        //Espero a que suelte el pulsador
      };
    }
    else
    {
      AuxTime.dia=Modificar_Variable(AuxTime.dia,1,31);

      Efecto_Titilar(8,0,2,100);
    }
  break;

  case 4:
    //Modifico el mes
    lcd.setCursor(11, 0);
    sprintf(BufferAux,"%02d",AuxTime.mes);
    lcd.print(BufferAux);

    if(digitalRead(Encoder_Switch)==0)//ENTER
    {

      if(Validar_Fecha(AuxTime.dia,AuxTime.mes)==false)
      {
        lcd.setCursor(0, 0);
        lcd.print(F("La fecha elegida"));
        lcd.setCursor(0, 1);
        lcd.print(F("   no existe    "));
        delay(3000);
        estado_ajuste=0;
```

```
        estado_menu=MENU_PRINCIPAL;
      }
      else
        estado_ajuste++;

      while(digitalRead(Encoder_Switch)==0)
      {
        //Espero a que suelte el pulsador
      };
    }
    else
    {
      AuxTime.mes=Modificar_Variable(AuxTime.mes,1,12);

      Efecto_Titilar(11,0,2,100);
    }


  break;

  case 5:
    //Modifico el año
    lcd.setCursor(14, 0);
    sprintf(BufferAux,"%02d",AuxTime.anio);
    lcd.print(BufferAux);

    if(digitalRead(Encoder_Switch)==0)//ENTER
    {

      if((AuxTime.dia==29)&&(AuxTime.mes==2)&&(AuxTime.anio%4!=0))
      {
        lcd.setCursor(0, 0);
        lcd.print(F("La fecha elegida"));
        lcd.setCursor(0, 1);
        lcd.print(F("   no existe    "));
        delay(3000);
        estado_ajuste=0;
        estado_menu=MENU_PRINCIPAL;
      }
      else
        estado_ajuste++;

      while(digitalRead(Encoder_Switch)==0)
      {
        //Espero a que suelte el pulsador
      };
    }
    else
    {
      AuxTime.anio=Modificar_Variable(AuxTime.anio,0,99);//De 2023 a 2099

      Efecto_Titilar(14,0,2,100);
    }
  break;
```

```
case 6:
  //Modifico la hora
  lcd.setCursor(8, 1);
  sprintf(BufferAux,"%02d",AuxTime.hora);
  lcd.print(BufferAux);

  if(digitalRead(Encoder_Switch)==0)//ENTER
  {
    estado_ajuste++;
    while(digitalRead(Encoder_Switch)==0)
    {
      //Espero a que suelte el pulsador
    };
  }
  else
  {
    AuxTime.hora=Modificar_Variable(AuxTime.hora,0,23);

    Efecto_Titilar(8,1,2,100);
  }
break;


case 7:
  //Modifico los minutos
  lcd.setCursor(11, 1);
  sprintf(BufferAux,"%02d",AuxTime.min);
  lcd.print(BufferAux);

  if(digitalRead(Encoder_Switch)==0)//ENTER
  {
    estado_ajuste++;
    while(digitalRead(Encoder_Switch)==0)
    {
      //Espero a que suelte el pulsador
    };
  }
  else
  {
    AuxTime.min=Modificar_Variable(AuxTime.min,0,59);

    Efecto_Titilar(11,1,2,100);
  }
break;

case 8:

  lcd.setCursor(0, 0);
  lcd.print(F("  Modificacion  "));
  lcd.setCursor(0, 1);
  lcd.print(F("    exitosa     "));

  //Copio la hora seleccionada en la actual
  AuxTime.seg = 0;
  ActualTime = AuxTime;
```

```
      //Como la fecha es válida guardo fecha y hora en el RTC
      RTC.setSecond(0);
      RTC.setMinute(ActualTime.min);
      RTC.setHour(ActualTime.hora);
      RTC.setDoW(ActualTime.DoW);
      RTC.setDate(ActualTime.dia);
      RTC.setMonth(ActualTime.mes);
      RTC.setYear(ActualTime.anio);
      RTC.setClockMode(h12);

      delay(3000);
      estado_ajuste=0;
      estado_menu=MENU_PRINCIPAL;
      modificacion_realizada=1;
    break;
  }
}


//*********************************************************************
// Función:                  Validar_Fecha
//
// Descripción:              Verifica que la combinación dia/mes exista
//
// Parámetros:               dia y mes
//
// Valor devuelto: bool: true (fecha valida) false (fecha inexistente)
//
//*********************************************************************
bool Validar_Fecha(int dia,int mes)
{
  if(dia>cant_dias_mes[mes-1])
  {
#ifdef DEBUG_SERIE
    Serial.print("cant dias del mes:");
    Serial.print(cant_dias_mes[mes-1]);
#endif
    return false;
  }
  else
    return true;
}

//*********************************************************************
// Función:                  Efecto_Titilar
//
// Descripción:    Efecto que hace titilar los digitos de pantalla cada 1 seg
//
// Parámetros:        int col (columna inicial)
//                   int fila (fila inicial)
//                   int digitos (digitos a hacer titilar)
//                   int delay_ms (para ajustar el tiempo en off)
//
// Valor devuelto: void
//
//*********************************************************************
```

```
void Efecto_Titilar(int col,int fila,int digitos,int delay_ms)
{
  if(ActualTime.seg != seg_anterior)
  {
    lcd.setCursor(col,fila);

    for(int i=0; i<digitos;i++)
      lcd.print(F(" "));

    delay(delay_ms);

    seg_anterior=ActualTime.seg;
  }
}

//*********************************************************************
// Función:                  Modificar_Variable
//
// Descripción:   Permite modificar una variable con el encoder entre un
//                  mínimo y un máximo
//
// Parámetros:        int variable_a_modificar
//                  int minimo
//                  int maximo
//
// Valor devuelto:  int: devuelve la variable modificada
//
//*********************************************************************
int Modificar_Variable(int variable_a_modificar, int minimo, int maximo)
{
  switch(Leer_Encoder())
  {
    default:
      break;

    case DERECHA:
      variable_a_modificar++;
      if(variable_a_modificar > maximo)
        variable_a_modificar=minimo;
    break;

    case IZQUIERDA:
      variable_a_modificar--;
      if(variable_a_modificar < minimo)
        variable_a_modificar=maximo;
    break;
  }

  return(variable_a_modificar);
}

//*********************************************************************
// Función:                  Menu_Feriados
//
// Descripción:              Muestra los feriados y permite agregar o quitar.
//
```

```
// Parámetros:              void
//
// Valor devuelto: void
//
//*****************************************************************
void Menu_Feriados(void)
{
  char BufferAux[30];

  switch(estado_menu_feriados)
  {
    default:
      estado_menu_feriados=0;
    break;

    case 0:
      //Parto desde la hora actual
      AuxTime = ActualTime;

      estado_menu_feriados++;
    break;

    case 1:

      switch(Leer_Encoder())
      {
        default:
          break;

        case DERECHA:
          AuxTime.dia++;
          if(AuxTime.dia > cant_dias_mes[AuxTime.mes-1])
          {
            AuxTime.dia=1;
            AuxTime.mes++;

            if(AuxTime.mes>12)
              AuxTime.mes=1;
          }
        break;

        case IZQUIERDA:
          AuxTime.dia--;
          if(AuxTime.dia < 1)
          {
            AuxTime.mes--;

            if(AuxTime.mes<1)
              AuxTime.mes=12;

            AuxTime.dia=cant_dias_mes[AuxTime.mes-1];
          }
        break;

        case ENTER:
          estado_menu_feriados++;
```

```
        lcd.setCursor(0, 1);
        lcd.print(F(">"));
        lcd.setCursor(0, 0);
        sprintf(BufferAux,"%02d/%02d",AuxTime.dia,AuxTime.mes);
        lcd.print(BufferAux);
        return;
      break;
    }

    lcd.setCursor(0, 0);
    sprintf(BufferAux,"%02d/%02d",AuxTime.dia,AuxTime.mes);
    lcd.print(BufferAux);

    lcd.setCursor(6, 0);
    if(verificaFeriado(AuxTime.dia,AuxTime.mes) == true)
    {
      lcd.print(F("es feriado"));
      lcd.setCursor(1, 1);
      lcd.print(F("Eliminar"));
      feriado = ES_FERIADO;
    }
    else
    {
      lcd.print(F("          "));
      lcd.setCursor(1, 1);
      lcd.print(F("Agregar "));
      feriado = NO_ES_FERIADO;
    }
    cursor_feriado=0;

    lcd.setCursor(11, 1);
    lcd.print(F("Salir"));

    //Hago titilar a los números
    Efecto_Titilar(0,0,5,100);
    lcd.setCursor(2, 0);
    lcd.print(F("/"));

  break;

  case 2:
    //Elijo entre Agregar/Eliminar o Salir
    switch(Leer_Encoder())
    {
      default:
        break;

      case DERECHA:
        cursor_feriado=1;
        lcd.setCursor(0, 1);
        lcd.print(F(" "));
        lcd.setCursor(10, 1);
        lcd.print(F(">"));
      break;
```

```
        case IZQUIERDA:
          cursor_feriado=0;
          lcd.setCursor(0, 1);
          lcd.print(F(">"));
          lcd.setCursor(10, 1);
          lcd.print(F(" "));
        break;

        case ENTER:
          estado_menu_feriados++;
        break;
      }
    break;

    case 3:
      if(cursor_feriado == 0)
      {
        modificacion_realizada=true;
        lcd.setCursor(0, 0);
        // Modifico el estado actual
        if(feriado== ES_FERIADO)
        {
          seteaFeriado(AuxTime.dia, AuxTime.mes, NO_ES_FERIADO);
          sprintf(BufferAux,"%02d/%02d se quita  ",AuxTime.dia,AuxTime.mes);
          lcd.print(BufferAux);
          lcd.setCursor(0, 1);
          lcd.print(F("de los feriados  "));
        }
        else
        {
          seteaFeriado(AuxTime.dia, AuxTime.mes, ES_FERIADO);
          sprintf(BufferAux,"%02d/%02d se agrega ",AuxTime.dia,AuxTime.mes);
          lcd.print(BufferAux);
          lcd.setCursor(0, 1);
          lcd.print(F("a los feriados   "));
        }
        delay(3000);
        //Me permite seguir eligiendo feriados
        estado_menu_feriados=1;
        lcd.clear();
      }
      else
      {
        //Presionó Salir
        estado_menu_feriados=0;
        estado_menu=MENU_PRINCIPAL;
      }

    break;

  }
}
//*********************************************************************
// Función:        verificaFeriado
//
// Descripción:    Chequea en la EEPROM si hoy es feriado
```

```
//
// Parámetros:              int dia, int mes
//
// Valor devuelto: bool: true (es feriado), false (no es feriado)
//
//*********************************************************************
bool verificaFeriado(int dia, int mes)
{
  //Obtenemos el dia del año en formato "M|D"
  int indexMonthEeprom = mes - 1;
  int daysAccum = 0;

  for(int i = 0; i < indexMonthEeprom; i++)
  {
    daysAccum += cant_dias_mes[i];
  }

  daysAccum += ULTIMA_POSICION_EVENTOS + dia;

  if(EEPROM.read(daysAccum) == ES_FERIADO)
  {
#ifdef DEBUG_SERIE
      Serial.print(F("Feriado detectado\n"));
#endif
    return true;
  }
  else
  {
    return false;
  }
}

//*********************************************************************
// Función:                 seteaFeriado
//
// Descripción:  Setea como feriado el conjunto dia/mes recibido guardandolo
//                  en EEPROM
//
// Parámetros:     int dia, int mes, bool feriado (para indicar si es o no es)
//
// Valor devuelto: void
//
//*********************************************************************
void seteaFeriado(int dia, int mes, bool feriado)
{
  int indexMonthEeprom = mes - 1;
  int daysAccum = 0;
  char buff_aux[4];

  for(int i = 0; i < indexMonthEeprom; i++)
  {
    daysAccum += cant_dias_mes[i];
  }

  daysAccum += ULTIMA_POSICION_EVENTOS + dia;
  EEPROM.write(daysAccum, feriado);
```

```
  //Trato de guardar los cambios si hay uSD
  Guardar_Cambios_uSD();

}

//*********************************************************************
// Función:                  Guardar_Cambios_uSD
//
// Descripción: Si está colocada la microSD, guarda las modificaciones
//                  ya sea de feriados o de eventos
//
// Parámetros:                         void
//
// Valor devuelto: void
//
//*********************************************************************
void Guardar_Cambios_uSD(void)
{
  char buff_feriado[4];
  char buff_eventos[6];
  int valor;
  //Estructura para el archivo de la uSD
  File myFile;


  //Borro el archivo actual para crear uno nuevo
  SD.remove("feriados.txt");

  myFile = SD.open("feriados.txt",FILE_WRITE);

  if(myFile)
  {
    //Guardo los feriados
    for(int i=PRIMERA_POSICION_FERIADOS; i<=ULTIMA_POSICION_FERIADOS;i++)
    {
      if(EEPROM.read(i)==1)
      {
        sprintf(buff_feriado,"%d,",i);
        myFile.print(buff_feriado);
#ifdef DEBUG_SERIE
        Serial.print(buff_feriado);
#endif
      }
    }

    //Ya terminé con los feriados
    myFile.print('|');

    //Guardo los eventos
    for(int i=PRIMERA_POSICION_EVENTOS; i<=ULTIMA_POSICION_EVENTOS;i++)
    {
      //Guardo posicion:valor, asegurandome que tanto posicion como valor ocupen 2
dígitos
      valor = EEPROM.read(i);
```

```
      if(valor > MAX_VALOR_HORARIOS)
        valor=0;

      sprintf(buff_eventos, "%02d:%02d,", i, valor);

      myFile.print(buff_eventos);
#ifdef DEBUG_SERIE
      Serial.print(buff_eventos);
#endif

    }

    //Ya terminé con los eventos
    myFile.print('.');

    myFile.close();
  }

}
//*****************************************************************
// Función:        configuraInterrupcionRTC
//
// Descripción:    Configura la Interrupción del RTC
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*****************************************************************
void configuraInterrupcionRTC (void)
{
  // Set alarm 1 to fire at one-second intervals
  RTC.turnOffAlarm(1);
  RTC.setA1Time(0, 0, 0, 0, alarmBits, false, false, false);
  // enable Alarm 1 interrupts
  RTC.turnOnAlarm(1);
  // clear Alarm 1 flag
  RTC.checkIfAlarm(1);

  pinMode(CLINT, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(CLINT), isr_TickTock, FALLING);

  // Use builtin LED to blink
  //pinMode(LED_BUILTIN, OUTPUT);
}

//*****************************************************************
// Función:                 isr_TickTock
//
// Descripción:             Interrupción del RTC
//
// Parámetros:              void
//
// Valor devuelto: void
//
//*****************************************************************
```

```
void isr_TickTock(void) {
  // interrupt signals to loop
  tick = 1;

  return;
}

//*********************************************************************
// Función:                                    loop
//
// Descripción:              Bucle principal del programa
//
//*********************************************************************
void loop(){

#ifdef DEBUG_SERIE
  char BufferAux[50];
#endif
  int min_aux, seg_aux;

         //Reloj();

  if(estado_menu != estado_menu_anterior)
  {
    lcd.clear();
    //proximo_menu = 0;
    estado_menu_anterior = estado_menu;
    delay(300);
  }

         switch(estado_menu)
         {
                  default:
                  case MENU_PRINCIPAL:
       Menu_Principal();
                  break;

                  case MENU_AJUSTE_FECHA_HORA:
     Ajuste_Reloj();
                  break;

                  case MENU_CONFIGURACION:
     Seleccion_Configuracion();
                  break;

                  case MENU_PROGRAMA_SEMANAL:
     Programacion_Semanal();
                  break;

                  case MENU_FERIADOS:
     Menu_Feriados();
                  break;

    case MENU_CAMBIOS_SD:
      lcd.clear();
      digitalWrite(Salida_Backlight,0);//se enciende el backlight
```

```
      tiempo_sin_pulsar=TIEMPO_PARA_APAGAR_LCD;
      lcd.setCursor(0, 0);
      lcd.print(F("Leyendo memoria"));

      delay(2000);//2seg

      lcd.setCursor(0, 0);
      lcd.print(F(" Modificaciones "));
      lcd.setCursor(0, 1);
      lcd.print(F("   realizadas   "));

      delay(4000);//4seg

      modificacion_realizada=1;
      estado_menu = MENU_PRINCIPAL;
    break;

          }

  min_aux=RTC.getMinute();

  seg_aux=RTC.getSecond();

  //El tick no lo estamos utilizando
  if(seg_aux != seg_ant)
  {
    //Aca entra cada 1seg
    ActualTime.seg++;

    //Cada 5 segundos me fijo si metieron una uSD para cargar los feriados
    if(ActualTime.seg%10==0)
      leerFeriadosDeSd();

    if(tiempo_sin_pulsar>0)
      tiempo_sin_pulsar--;

    if((min_aux != min_ant)||(modificacion_realizada))
    {
      //Aca entra cada un minuto o cuando se haga una modificación en el Reloj o en los
Rangos

      Leer_Fecha_Hora_RTC(ActualTime);

      //Si cambia el día, leo los rangos permitidos
      if((ultimo_dia != ActualTime.DoW)||(modificacion_realizada))
      {
        Cargar_Rangos();
        ultimo_dia=ActualTime.DoW;
      }

      modificacion_realizada=0;

      codigo_hora_actual = (ActualTime.hora*4)+(ActualTime.min/15);

      if(codigo_hora_actual != codigo_hora_actual_anterior)
      {
```

```
#ifdef DEBUG_SERIE
          sprintf(BufferAux,"Codigo actual: %d\n",codigo_hora_actual);
          Serial.print(BufferAux);
#endif
        codigo_hora_actual_anterior=codigo_hora_actual;
      }

      //Sólo con esto manejo la salida

if((((codigo_hora_actual>=Rangos_hoy[INICIO_1])&&(codigo_hora_actual<Rangos_hoy[FIN_1]))
||
((codigo_hora_actual>=Rangos_hoy[INICIO_2])&&(codigo_hora_actual<Rangos_hoy[FIN_2]))||
((codigo_hora_actual>=Rangos_hoy[INICIO_3])&&(codigo_hora_actual<Rangos_hoy[FIN_3]))||
((codigo_hora_actual>=Rangos_hoy[INICIO_4])&&(codigo_hora_actual<Rangos_hoy[FIN_4])))&&
(!verificaFeriado(ActualTime.dia,ActualTime.mes)))
      {
        //Si no es feriado y se encuentra dentro de alguno de los rangos programados,
alimento la carga
        digitalWrite(Salida_Rele,1);
      }
      else
      {
        //Si se encuentra fuera de los rangos dejo de alimentar la carga
        digitalWrite(Salida_Rele,0);
      }

      // Clear Alarm 1 flag
      RTC.checkIfAlarm(1);
    }
  }

  min_ant = min_aux;
  seg_ant = seg_aux;

  //Controlo el backlight del lcd
  if(!tiempo_sin_pulsar)
    digitalWrite(Salida_Backlight,1);//Si pasa cierto tiempo en el que no se tocó el
encoder, se apaga el display
  else
    digitalWrite(Salida_Backlight,0);//Si se toca el encoder, se enciende

}
```

4.2. Guardado de datos en memoria

Para guardar los datos relacionados a la programación del equipo se ha utilizado la memoria EEPROM interna del ATMEGA328P (1 KB de memoria). Las primeras 56 posiciones (0 a 55) se utilizaron para la programación de los eventos semanales, mientras que las siguientes 365 posiciones se utilizaron para indicar los días feriados.

- *Eventos*

Se establecieron los siguientes parámetros:

$Días$:

LUNES	0
MARTES	1
MIÉRCOLES	2
JUEVES	3
VIERNES	4
SÁBADO	5
DOMINGO	6

$Rangos$:

ENCENDIDO_1	0
APAGADO_1	1
ENCENDIDO_2	2
APAGADO_2	3
ENCENDIDO_3	4
APAGADO_3	5
ENCDENDIDO_4	6
APAGADO_4	7

La posición de memoria que representa cada combinación se obtiene con la siguiente fórmula:

$$posición = \big((día + 1) * 8\big) + rango$$

Además, para cada horario del día se genera un código, el cual se guarda en esa posición de memoria. Este código puede ser un valor entre 0 y 95, y se obtiene de la siguiente fórmula:

$$código = hora * 4 - \frac{min}{15}$$

Hay que recordar que los lapsos de horarios son cada 15 minutos.

Ejemplo:

Si se quiere guardar que el tercer encendido del jueves ocurra a las 19:45, el código realizará lo siguiente:

$$posición = \big((\text{JUEVES} + 1) * 8\big) + \text{ENCENDIDO\_3} = \big((3 + 1) * 8\big) + 4 = \mathbf{28}$$

$$código = hora * 4 - \frac{min}{15} = 19 * 4 - \frac{45}{15} = \mathbf{79}$$

Se obtiene en la EEPROM:

Posición	Dato
28 [0x1C]	79 [0x4F]

- *Feriados*

  Cada día del año, incluyendo al 29 de febrero, tiene asignada una posición de memoria, comenzando desde el 01/01 (posición 56 [0x38]) hasta el 31/12 (posición 421 [0x1A5]).

  Si el día es considerado feriado, guardará un 1 en su respectiva posición. En cambio, si no lo es, guardará un 0.

- *Archivo "feriados.txt"*

  El archivo "feriados.txt" es un archivo de texto que puede ser interpretado y escrito, tanto por el dispositivo como por el software desarrollado.

  Para la correcta interpretación de los datos generados, se debe tener en cuenta que el formato en el que se guardan los datos es el siguiente:

posicion_feriado_1,posicion_feriado_2,…,posicion_feriado_N,|posicion_rango_1:codigo_horario_rango1,
posicion_rango_2: codigo_horario _rango2,…, posicion_rango_N: codigo_horario _rango_N,.

  Será fundamental que se respete el orden y los símbolos separadores para la correcta interpretación de los datos.

  Se interpretará como feriado a todos los días cuya posición se encuentre en el archivo, mientras que las posiciones que no aparezcan, se las tomará como días no feriados. Para su correcta interpretación, estas posiciones deben estar ordenadas de menor a mayor.

  En cuanto a los rangos de eventos, se cargarán en el dispositivo los rangos que aparezcan en el archivo siempre y cuando tengan un código horario valido. Si algún rango de eventos no aparece en el archivo, mantiene el valor previo sin ninguna modificación.

  A continuación, se muestra el contenido del archivo Default. El mismo contiene todos los feriados del año 2023, y tiene como rangos de encendido los horarios en los que la facultad permanece abierta.

```
56,106,107,139,148,153,177,201,202,224,226,227,246,289,342,345,380,398,415,|
00:32,01:48,02:56,03:72,04:76,05:92,06:00,07:00,
08:32,09:48,10:56,11:72,12:76,13:92,14:00,15:00,
16:32,17:48,18:56,19:72,20:76,21:92,22:00,23:00,
24:32,25:48,26:56,27:72,28:76,29:92,30:00,31:00,
32:32,33:48,34:56,35:72,36:76,37:92,38:00,39:00,
40:32,41:48,42:56,43:72,44:00,45:00,46:00,47:00,
48:00,49:00,50:00,51:00,52:00,53:00,54:00,55:00,.
```

5. Software

A continuación, se documenta el software para la creación del archivo *feriados.txt*, al cual se le puede cargar feriados y rangos para luego ser leídos por el dispositivo.

El Proyecto en Java 8 se compone de una clase ProyectoPotencia.java que contiene la clase **main** y se crean 3 clases (Dia, Evento y Mes) para crear instancias de estas y usarlas en la clase principal.

Para ejecutar el programa se requiere tener instalado java 8 o una versión superior y ejecutar en consola el comando:

java -jar Proyecto_Potencia_2023.jar

Se adjunta el detalle de las clases descritas anteriormente:

Clase ProyectoPotencia.java

```
package proyecto.potencia;

import java.util.ArrayList;
import java.util.List;
import java.util.OptionalInt;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Collections;
import java.util.HashMap;


/**
 *
 * @author ELECTRONICA DE POTENCIA - SOFTWARE PARA CARGA DE FERIADOS Y RANGOS EN
 * SD
 *
 * Clase ProyectoPotencia con metodo Main
 *
 */
public class ProyectoPotencia {

    //Listado de Meses
    private static List<Mes> meses = new ArrayList<>();

    //Se inicializan los meses con un id
    private static Mes enero = new Mes(1, "Enero");
    private static Mes febrero = new Mes(2, "Febrero");
    private static Mes marzo = new Mes(3, "Marzo");
    private static Mes abril = new Mes(4, "Abril");
    private static Mes mayo = new Mes(5, "Mayo");
    private static Mes junio = new Mes(6, "Junio");
    private static Mes julio = new Mes(7, "Julio");
    private static Mes agosto = new Mes(8, "Agosto");
    private static Mes septiembre = new Mes(9, "Septiembre");
    private static Mes octubre = new Mes(10, "Octubre");
    private static Mes noviembre = new Mes(11, "Noviembre");
    private static Mes diciembre = new Mes(12, "Diciembre");

    //Listado de dias de la semana
    private static List<Dia> dias = new ArrayList<>();

    //Se inicializan los dias de la semana con un id
    private static Dia lunes = new Dia(1, "Lunes");
    private static Dia martes = new Dia(2, "Martes");
    private static Dia miercoles = new Dia(3, "Miercoles");
    private static Dia jueves = new Dia(4, "Jueves");
    private static Dia viernes = new Dia(5, "Viernes");
    private static Dia sabado = new Dia(6, "Sábado");
    private static Dia domingo = new Dia(0, "Domingo");

    private static HashMap<Integer, Integer> decoMeses = new HashMap<>();
    private static HashMap<Integer, Integer> decoDias = new HashMap<>();

    public static void main(String[] args) {

        String logo = " \n"
                + "
\n"
                + "
\n"
                + "                                          ..:::^^^^^^:::..
\n"
                + "                                    .:^~!77777777777777!~^:.
\n"
                + "                                .^!77??JJY55PPPPPPP55YJJ??77????77!^.
\n"
```

```
                + "                                :~77???JY555PPPPGGBB#&@@@@@&#GPY?777??77~:
\n"
                + "                             .^!???777777777777777777??J5PB&@@@@&#PY?77???7^.
\n"
                + "                           .~7??77?JY5PGB######BBGP5J?77777?JP#@@@@@&GJ777??7~.
\n"
                + "                         .~7????Y5PPGPPPPPPGBB#&@@@@@&#G5J77777JP&@@@@@BY777??7~.
\n"
                + "                        :7??77????777777777777?JYPB&@@@@@#5?7777?5#@@@@@BJ777??7:
\n"
                + "                       ~??777777?JY5PGGGGP55J?777777?5B&@@@@&G?777?5&@@@@@5777??~
\n"
                + "                     ~??777?YPB&@@@@@@@@@@@@&#GY?77777JG&@@@@&577777?B@@@@@G7777??~
\n"
                + "                   ~????7P#@@@@@@@@@@@@@@@@@@@@&PJ77777JB@@@@@B?77777P@@@@@G77777?~
\n"
                + "                 ^???7?P&@@@@@@@@@@@@@@@@@@@@@@@@&P?77777P@@@@@B?77777G@@@@@5777???^
\n"
                + "               .7?7J#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@B?77777P@@@@@G77777?&@@@@&?7777?7.
\n"
                + "              ~?7?#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@B?7777?#@@@@@J77777G@@@@@577777?~
\n"
                + "              7?7G@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@5777775@@@@@P777775@@@@@G7777?7.
\n"
                + "             .77J@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#77777J@@@@@B77777J@@@@@B7777?7.
\n"
                + "             :?7JGGGP#@@@@@@@@@@@@@@@@@@GGGGGG@@@@@&PPPP5P@@@@@#55555G@@@@@&555PJ7?:
\n"
                + "             .7?77777P@@@@@@@@@@@@@@@@&?7777?&@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@Y7?:
\n"
                + "             .7?77777Y@@@@@@@@@@@@@@@@@@Y77777P@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@B??7.
\n"
                + "              ~?77777?#@@@@@@@@@@@@@@@@B77777?B@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@&J7?!
\n"
                + "              :7?77777Y@@@@@@@@@@@@@@@@@@577777?G@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#J7??:
\n"
                + "               ~?777777P@@@@@@@@@@@@@@@@@@Y777777Y#@@@@@@@@@@@@@@@@@@@@@@@@@@@&P?77?~
\n"
                + "                !?777777P@@@@@@@@@@@@@@@@@@P?777777YB&@@@@@@@@@@@@@@@@@@@@@@@#P?777?!.
\n"
                + "               .!??777775&@@@@@@@@@@@@@@@@@@#Y?777777?YG#&@@@@@@@@@@@@@@&BPJ?7777?!.
\n"
                + "                 ~??77777JB@@@@@@@@@@@@@@@@@@#5J77777777?JY55PPP55YJ?77777??!.
\n"
                + "                  ^7??77777Y#@@@@@@@@@@@@@@@@@@&BPY?77777777777777777??77??7^
\n"
                + "                .!???77777YB&@@@@@@@@@@@@@@@@@@@@&#BP5YJJJJJJJYY5GG5?7??!.
\n"
                + "                  :!???77777?5B&@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@&#PJ7??!:
\n"
                + "                    :~7???77777?YPB&@@@@@@@@@@@@@@@@@@@@@@&BGY?7??7~:
\n"
                + "                      .^!7????77777?JYPGBB##&&&&&&##BGPYJ?77???7!^.
\n"
                + "                         .:~!7?????777777?????????77777??7!~:.
\n"
                + "                              .:^~!77??????????????????77!~^:.
\n"
                + "                                     .::^^^~~~~~~^^^::.
\n"
                + "
\n"
                + " ------------------------------------------------------------------------------------------------
\n"
                + " UNLAM - Electronica de Potencia - 1° Cuatrimestre 2023
\n"
                + " ------------------------------------------------------------------------------------------------
\n"
                + "
";

        System.out.println(logo);
        System.out.println("MENU: ");
        System.out.println("");

        //INICIALIZACIONES
        meses.add(enero);
        meses.add(febrero);
        meses.add(marzo);
        meses.add(abril);
        meses.add(mayo);
        meses.add(junio);
        meses.add(julio);
        meses.add(agosto);
        meses.add(septiembre);
        meses.add(octubre);
        meses.add(noviembre);
        meses.add(diciembre);

        decoMeses.put(1, 31);
        decoMeses.put(2, 29);
        decoMeses.put(3, 31);
        decoMeses.put(4, 30);
        decoMeses.put(5, 31);
        decoMeses.put(6, 30);
        decoMeses.put(7, 31);
        decoMeses.put(8, 31);
        decoMeses.put(9, 30);
        decoMeses.put(10, 31);
        decoMeses.put(11, 30);
        decoMeses.put(12, 31);
```

```
        dias.add(lunes);
        dias.add(martes);
        dias.add(miercoles);
        dias.add(jueves);
        dias.add(viernes);
        dias.add(sabado);
        dias.add(domingo);

        decoDias.put(0, 0);
        decoDias.put(1, 8);
        decoDias.put(2, 16);
        decoDias.put(3, 24);
        decoDias.put(4, 32);
        decoDias.put(5, 40);
        decoDias.put(6, 48);

        //Se cargan en una lista los items del menu
        List<String> lista = new ArrayList<>();
        lista.add("Cargar Feriados");
        lista.add("Ver Feriados Cargados");
        lista.add("Eliminar todos los feriados");
        lista.add("Eliminar un Feriado");
        lista.add("Cargar Rangos");
        lista.add("Ver Rangos Cargados");
        lista.add("Eliminar todos los rangos");
        lista.add("Eliminar un Rango");
        lista.add("Guardar Feriados en SD y Dias");
        lista.add("Leer SD");
        lista.add("Salir");

        Scanner scanner = new Scanner(System.in);

        //Se despliega el menu en consola
        while (true) {
            System.out.println("Selecciona un elemento de la lista (ingresa el índice):");

            for (int i = 0; i < lista.size(); i++) {
                System.out.println(i + ": " + lista.get(i));
            }

            int indiceSeleccionado = scanner.nextInt();

            if (indiceSeleccionado >= 0 && indiceSeleccionado < lista.size()) {
                switch (indiceSeleccionado) {
                    case 0:
                        cargarFeriados(scanner);
                        break;
                    case 1:
                        verFeriadosCargados();
                        break;
                    case 2:
                        eliminarTodoLosFeriados();
                        break;
                    case 3:
                        eliminarUnFeriado(scanner);
                        break;
                    case 4:
                        cargarEventos(scanner);
                        break;
                    case 5:
                        verEentosCargados();
                        break;
                    case 6:
                        eliminarTodoLosEventos();
                        break;
                    case 7:
                        eliminarUnEvento(scanner);
                        break;
                    case 8:
                        guardarCalendarioEnSD(scanner);
                        break;
                    case 9:
                        leersd(scanner);
                        break;
                    case 10:
                        System.exit(0);
                }

            } else {
                System.out.println("Índice inválido. No se seleccionó ningún elemento.");
            }

        }
    }

    //Este Metodo permite cargar feriados en el formato especificado
    private static void cargarFeriados(Scanner scanner) {
        while (true) {
            clearScreen();
            System.out.println("Escriba el feriado en formato dd/mm: ");
            String feriado = scanner.next();
            int mes = Integer.parseInt(feriado.split("/")[1]);
            int dia = Integer.parseInt(feriado.split("/")[0]);

            //Busco el mes en el array
            Mes mesBuscado = meses.stream()
                    .filter(m -> m.indice == mes)
                    .collect(Collectors.toList()).get(0);

            //Cargo el feriado al array de dias en el mes
            mesBuscado.feriados.add(dia);
```

```
            //Obtengo el indice del mes del array
            OptionalInt indexMeses = IntStream.range(0, meses.size())
                    .filter(i -> mes == meses.get(i).indice)
                    .findFirst();

            meses.set(indexMeses.getAsInt(), mesBuscado);

            System.out.println("Se agregó el feriado: " + dia + " de " + mesBuscado.name);
            System.out.println("¿Desea agregar otro feriado? (s/n)");
            String agregarOtroferiado = scanner.next();
            if (!agregarOtroferiado.equalsIgnoreCase("s")) {
                break;
            }
        }
    }

    //Permite visualizar el listado de feriados
    private static void verFeriadosCargados() {
        clearScreen();
        List<String> listadoFeriados = new ArrayList<>();
        meses.stream().filter((mes) -> (!mes.feriados.isEmpty())).forEachOrdered((mes) -> {
            mes.feriados.forEach((feriado) -> {
                listadoFeriados.add(feriado + "/" + mes.indice);
            });
        });
        System.out.println("Feriados Cargados:");
        listadoFeriados.forEach((feriado) -> {
            System.out.println(feriado);
        });

        System.out.println("");

    }

    //Permite vizualisar los rangos cargados en formato de lista
    private static void verEentosCargados() {
        clearScreen();
        dias.stream().filter((dia) -> (!dia.eventos.isEmpty())).forEachOrdered((dia) -> {
            System.out.println("[" + dia.name + "]");
            System.out.println("|");
            dia.eventos.forEach((evento) -> {
                System.out.println("|-> Inicio:  " + evento.horaInicio + " Fin: " + evento.horaFin);
            });
            System.out.println("");
        });

        System.out.println("");

    }

    //Elimina todos los feriados de la variable meses
    private static void eliminarTodoLosFeriados() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
        for (int i = 0; i < meses.size(); i++) {
            meses.get(i).feriados.clear();
        }
        System.out.println("Se eliminaron todos los feriados.");

        System.out.println("");
        System.out.println("");
    }

    //Elimina todos los Rangos de la variable dias
    private static void eliminarTodoLosEventos() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
        for (int i = 0; i < dias.size(); i++) {
            dias.get(i).eventos.clear();
        }
        System.out.println("Se eliminaron todos los rangos.");

        System.out.println("");
        System.out.println("");
    }

    //Permite eliminar un feriado en especifico de la variable meses
    private static void eliminarUnFeriado(Scanner scanner) {
        clearScreen();
        System.out.println("Escriba el feriado a eliminar en formato dd/mm: ");
        String feriado = scanner.next();
        int mes = Integer.parseInt(feriado.split("/")[1]);
        int dia = Integer.parseInt(feriado.split("/")[0]);

        //Obtengo el indice del mes del array
        OptionalInt indexMeses = IntStream.range(0, meses.size())
                .filter(i -> mes == meses.get(i).indice)
                .findFirst();

        //Obtengo el indice del mes del array
        OptionalInt indexDia = IntStream.range(0, meses.get(indexMeses.getAsInt()).feriados.size())
                .filter(i -> dia == meses.get(indexMeses.getAsInt()).feriados.get(i))
                .findFirst();

        meses.get(indexMeses.getAsInt()).feriados.remove(indexDia.getAsInt());

        System.out.println("Se eliminó el feriado " + dia + "/" + mes);
        System.out.println("");
        System.out.println("");

    }
```

```
    //Metodo eliminar un rango en especifico de la variable dias
    private static void eliminarUnEvento(Scanner scanner) {
        clearScreen();
        System.out.println("Seleccione el dia del rango que desea eliminar: ");

        System.out.println("0: Lunes");
        System.out.println("1: Martes");
        System.out.println("2: Miercoles");
        System.out.println("3: Jueves");
        System.out.println("4: Viernes");
        System.out.println("5: Sabado");
        System.out.println("6: Domingo");

        String diaDelEvento = scanner.next();
        int diaDelEventoInt = Integer.parseInt(diaDelEvento);

        //Busco el dia en el array
        Dia diabuscado = dias.stream()
                .filter(d -> d.indice == diaDelEventoInt)
                .collect(Collectors.toList()).get(0);

        //Obtengo el indice del mes del array
        OptionalInt indexDia = IntStream.range(0, dias.size())
                .filter(i -> diaDelEventoInt == dias.get(i).indice)
                .findFirst();

        dias.get(indexDia.getAsInt()).eventos.remove(indexDia.getAsInt());

        System.out.println("Se eliminó el Rango del dia " + diabuscado.name);
        System.out.println("");
        System.out.println("");

    }

    //Metodo que permite cargar rangos y los almacena en la variable dias
    private static void cargarEventos(Scanner scanner) {
        while (true) {
            clearScreen();
            System.out.println("Seleccione un día de la semana: ");

            System.out.println("0: Domingo");
            System.out.println("1: Lunes");
            System.out.println("2: Martes");
            System.out.println("3: Miercoles");
            System.out.println("4: Jueves");
            System.out.println("5: Viernes");
            System.out.println("6: Sabado");

            String diaDelEvento = scanner.next();
            int diaDelEventoInt = Integer.parseInt(diaDelEvento);

            //Busco el dia en el array
            Dia diabuscado = dias.stream()
                    .filter(d -> d.indice == diaDelEventoInt)
                    .collect(Collectors.toList()).get(0);

            int eventoInt = -1;
            if (diabuscado.eventos.size() == 4) {
                System.out.println("Los Eventos de este día estan completos, por favor seleccione un rango a
sobrescribir:");

                System.out.println("0: Rango 1");
                System.out.println("1: Rango 2");
                System.out.println("2: Rango 3");
                System.out.println("3: Rango 4");

                String eventoSeleccionado = scanner.next();
                eventoInt = Integer.parseInt(eventoSeleccionado);

            }

            System.out.println("Seleccione el horario inicial del rango en formato hh:mm");
            System.out.println("(Los minutos deben ser múltiplo de 15)");

            String horarioInicial = scanner.next();

            System.out.println("Seleccione el horario final del rango en formato hh:mm");
            System.out.println("(Los minutos deben ser múltiplo de 15)");

            String horarioFinal = scanner.next();

            Evento evento = new Evento(horarioInicial, horarioFinal);

            if (eventoInt < 0) {
                diabuscado.eventos.add(evento);
            } else {
                diabuscado.eventos.get(eventoInt).horaInicio = evento.horaInicio;
                diabuscado.eventos.get(eventoInt).horaFin = evento.horaFin;
                diabuscado.eventos.get(eventoInt).valorCalculadoInicio = evento.valorCalculadoInicio;
                diabuscado.eventos.get(eventoInt).valorCalculadoFinal = evento.valorCalculadoFinal;
            }

            //Obtengo el indice del mes del array
            OptionalInt indexDias = IntStream.range(0, dias.size())
                    .filter(i -> diaDelEventoInt == dias.get(i).indice)
                    .findFirst();

            dias.set(indexDias.getAsInt(), diabuscado);

            System.out.println("Se agregó el rango al día " + diabuscado.name + " [Hora Inicial: " + horarioInicial +
" | Hora Final: " + horarioFinal + "]");
```

```
            System.out.println("¿Desea agregar otro rango? (s/n)");
            String agregarOtroferiado = scanner.next();
            if (!agregarOtroferiado.equalsIgnoreCase("s")) {
                break;
            }
        }
    }

    //Hace una lectura de los feriados y los rangos cargados en la SD y los mergea con los preexistentes
    //durante la ejecución del programa
    private static void leersd(Scanner scanner) {
        clearScreen();

        List<String> lista = new ArrayList<>();

        for (File sysDrive : File.listRoots()) {
            lista.add(sysDrive.toString());
        }

        System.out.println("Seleccione la unidad donde se encuentra la SD (ingresa el índice):");

        for (int i = 0; i < lista.size(); i++) {
            System.out.println(i + ": " + lista.get(i));
        }

        int indiceSeleccionado = scanner.nextInt();
        String elementoSeleccionado = lista.get(indiceSeleccionado);

        /*
        GUARDA EN SD
         */
        String nombreArchivo = "feriados.txt";

        // Tarjeta SD accesible
        File directorioSD = new File(elementoSeleccionado);
        if (!directorioSD.exists() || !directorioSD.isDirectory()) {
            System.out.println("La tarjeta SD no está disponible.");
            return;
        }

        int mes;
        int dia;

        try {

            File archivo = new File(directorioSD, nombreArchivo);
            Scanner myReader = new Scanner(archivo);
            String data = "";
            while (myReader.hasNextLine()) {
                data = myReader.nextLine();
            }

            myReader.close();

            //Carga de feriados
            String feriadosLeidos = data.split("\\|")[0];
            String eventosLeidos = data.split("\\|")[1];

            String[] feriadosLeidosArray = feriadosLeidos.split(",");
            String[] eventosLeidosArray = eventosLeidos.split("[.]")[0].substring(0, eventosLeidos.length() -
1).split(",");

            //Obtengo Feriados Leidos Array
            for (int i = 0; i < feriadosLeidos.split(",").length; i++) {
                int accum = 55;
                mes = 0;
                dia = 0;

                for (int j = 1; j <= decoMeses.size(); j++) {
                    accum = accum + decoMeses.get(j);
                    if (Integer.parseInt(feriadosLeidosArray[i]) < accum) {
                        mes = j;
                        dia = (accum - decoMeses.get(j) - Integer.parseInt(feriadosLeidosArray[i])) * -1;
                        break;
                    }
                }

                cargarFeriadosForSdMethod(mes, dia);

            }

            //Carga de Eventos
            ArrayList<ArrayList<Integer>> eventosHorarios = new ArrayList<>();

            for (String posicionHorario : eventosLeidosArray) {
                ArrayList<Integer> e = new ArrayList();
                e.add(Integer.parseInt(posicionHorario.split(":")[0]));
                e.add(Integer.parseInt(posicionHorario.split(":")[1]));
                eventosHorarios.add(e);
            }

            Collections.sort(eventosHorarios, (ArrayList<Integer> o1, ArrayList<Integer> o2) ->
o1.get(0).compareTo(o2.get(0)));

            //Eventos ordenados
            for (int i = 0; i < eventosHorarios.size(); i = i + 2) {
                ArrayList<Integer> ev = new ArrayList<>();
                ArrayList<Integer> evSiguiente = new ArrayList<>();
                ev = eventosHorarios.get(i);
                evSiguiente = eventosHorarios.get(i + 1);

                //determina dia
```

```
                int diaEvento = 0;
                for (int j = 0; j < decoDias.size(); j++) {
                    if (decoDias.get(j) > ev.get(0)) {
                        diaEvento = j;
                        break;
                    }
                }

                for (int z = 0; z < decoDias.size(); z++) {
                    if (decoDias.get(z) > ev.get(0)) {
                        diaEvento = z - 1;
                        break;
                    }
                }

                Dia diabuscado = diaBuscado(diaEvento);
                int eventoAOcupar = diabuscado.eventos.size();

                if (eventoAOcupar <= 4) {
                    //Determina Horario Inicial y Final

                    String horarioInicial = formatNumber(ev.get(1) / 4) + ":" + formatNumber((ev.get(1) % 4) * 15);
                    String horarioFinal = formatNumber(evSiguiente.get(1) / 4) + ":" +
formatNumber((evSiguiente.get(1) % 4) * 15);

                    Evento evento = new Evento(horarioInicial, horarioFinal);

                    diabuscado.eventos.add(evento);

                    //Obtengo el indice del mes del array
                    OptionalInt indexDias = indexDias(diaEvento);

                    dias.set(indexDias.getAsInt(), diabuscado);

                    System.out.println("Feriados y Rangos Leidos correctamente de SD");

                } else {
                    System.out.println("Todos los rangos estan ocupados para el dia " + diabuscado.name);
                }

            }

        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error al leer el archivo en la tarjeta SD.");
        }

        System.out.println("Feriados y Rangos Leidos correctamente de SD");

    }

    /*

        Este Metodo guarda los feriados y los rangos en la SD:

        Codifica los feriados a la posición de memoria correspondiente a la EEPROM del
        Atmega328, luego el firmware del dispositivo se encarga de setear un 1 en
        dichas posiciones de memoria indicando que corresponde un feriado para esas fechas.

        Para los Rangos se utiliza el formato XX:YY, siendo XX la posición de memoria
        de dentro de la EEPROM e YY el horario a alamacenenar en dicha posicion de memoria.

        Los feriados y los rangos estan separados por un PIPE (|) para facilitar el parseo
        y se detecta el fin de trama con un punto.

    */


    private static void guardarCalendarioEnSD(Scanner scanner) {
        clearScreen();

        List<String> lista = new ArrayList<>();

        for (File sysDrive : File.listRoots()) {
            lista.add(sysDrive.toString());
        }

        System.out.println("Seleccione la unidad donde se encuentra la SD (ingresa el índice):");

        for (int i = 0; i < lista.size(); i++) {
            System.out.println(i + ": " + lista.get(i));
        }

        int indiceSeleccionado = scanner.nextInt();
        String elementoSeleccionado = lista.get(indiceSeleccionado);

        /*
        GUARDA EN SD
         */
        String nombreArchivo = "feriados.txt";

        // Tarjeta SD accesible
        File directorioSD = new File(elementoSeleccionado);
        if (!directorioSD.exists() || !directorioSD.isDirectory()) {
            System.out.println("La tarjeta SD no está disponible.");
            return;
        }

        File archivo = new File(directorioSD, nombreArchivo);

        try {
            String fileToSd = "";
```

```
            for (Mes mes : meses) {
                if (!mes.feriados.isEmpty()) {
                    for (Integer feriado : mes.feriados) {
                        Integer posicionEnMemoria = 55 + feriado;
                        for (int i = 1; i < mes.indice; i++) {
                            posicionEnMemoria = posicionEnMemoria + decoMeses.get(i);
                        }
                        fileToSd += posicionEnMemoria + ",";
                    }
                }
            }

            fileToSd += "|";
            //Dias
            for (Dia dia : dias) {
                if (!dia.eventos.isEmpty()) {
                    int posicionInicial = decoDias.get(dia.indice);
                    for (int i = 0; i < dia.eventos.size(); i++) {
                        fileToSd += formatNumber(posicionInicial) + ":" +
formatNumber(dia.eventos.get(i).valorCalculadoInicio) + ",";
                        posicionInicial = posicionInicial + 1;
                        fileToSd += formatNumber(posicionInicial) + ":" +
formatNumber(dia.eventos.get(i).valorCalculadoFinal) + ",";
                        posicionInicial = posicionInicial + 1;
                    }
                }
            }

            fileToSd += ".";

            // Escribir el contenido en el archivo
            FileWriter escritor = new FileWriter(archivo);
            escritor.write(fileToSd);
            escritor.close();

            System.out.println("Los feriados se guardaron correctamente en la tarjeta SD.");
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error al guardar el archivo en la tarjeta SD.");
        }

    }

    //Metodo para obtener el Objeto Dia por un indice
    private static Dia diaBuscado(int diaEvento) {
        return dias.stream()
                .filter(d -> d.indice == diaEvento)
                .collect(Collectors.toList()).get(0);
    }

    //Metodo para obtener indice en la variable dias
    private static OptionalInt indexDias(int diaEvento) {
        return IntStream.range(0, dias.size())
                .filter(d -> diaEvento == dias.get(d).indice)
                .findFirst();
    }

    //Metodo para cargar feriados dentro de la variable meses
    private static void cargarFeriadosForSdMethod(int mes, int dia) {
        //Busco el mes en el array
        Mes mesBuscado = meses.stream()
                .filter(m -> m.indice == mes)
                .collect(Collectors.toList()).get(0);

        //Cargo el feriado al array de dias en el mes
        mesBuscado.feriados.add(dia);

        //Obtengo el indice del mes del array
        OptionalInt indexMeses = IntStream.range(0, meses.size())
                .filter(i -> mes == meses.get(i).indice)
                .findFirst();

        meses.set(indexMeses.getAsInt(), mesBuscado);
    }

    //Metodo para borrar pantalla
    private static void clearScreen() {
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }

    //Formatea numero en string para siempre manejar 2 decimales
    private static String formatNumber(int valor) {
        if (valor < 10) {
            return "0" + valor;
        } else {
            return "" + valor;
        }
    }

}
```

Clase Dia.java

```
package proyecto.potencia;

import java.util.ArrayList;
import java.util.List;

public class Dia {
        public Dia(int indice, String name) {
        this.indice = indice;
        this.name = name;
    }

    int indice;
    String name;
    List<Evento> eventos = new ArrayList<>();
}
```

Clase Evento.java

```
package proyecto.potencia;


public class Evento {

    String horaInicio;
    String horaFin;
    int valorCalculadoInicio;
    int valorCalculadoFinal;

    public Evento(String horaInicio, String horaFin) {
        this.horaInicio = horaInicio;
        this.horaFin = horaFin;

        this.valorCalculadoInicio =
(Integer.parseInt(horaInicio.split(":")[0]) * 4) +
(Integer.parseInt(horaInicio.split(":")[1]) / 15);
        this.valorCalculadoFinal =
(Integer.parseInt(horaFin.split(":")[0]) * 4) +
(Integer.parseInt(horaFin.split(":")[1]) / 15);
    }

}
```

Clase Mes.java

```
package proyecto.potencia;
import java.util.ArrayList;
import java.util.List;

public class Mes {

    public Mes(int indice, String name) {
        this.indice = indice;
        this.name = name;
    }

    int indice;
    String name;
    List<Integer> feriados = new ArrayList<>();
}
```

6. Foto producto

Ilustración 16: Vista superior del equipo

Ilustración 17: Vista Lateral del equipo

7. Video de Funcionamiento

A continuación, encontrarás el enlace al video donde podrás observar el dispositivo en pleno funcionamiento.

https://www.youtube.com/watch?v=BYdCcWpDvu8