D:\CONTENIDO\Ciclo 8\Iot\Proyecto\Avance 5\PROYECTO_RIEGO_AUTOMATICO_DOCUMENTED.ino

```
/*
Estas líneas incluyen las librerías necesarias para:
La comunicación I2C (Wire),
Manejo de gráficos (Adafruit_GFX),
Pantalla OLED (Adafruit_SSD1306),
Y, Reloj en tiempo real (RTClib).
*/
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RTClib.h"

/*
Definición del PIN G32 "outputPin"; este es el pin conectado a un relé que controla la bomba
State es una "bandera booleana" utilizada para rastrear el estado de la bomba (encendido o apagado)
RTC es una instancia de la clase RTC_DS3231 para interactuar con el reloj en tiempo real.
*/
const int outputPin = 32;
bool state = true;
RTC_DS3231 rtc;

/*
Librerías para:
Operaciones de memoria flash NVS (Non-Volatile Storage) (nvs_flash.h)
Almacenamiento de preferencias (Preferences.h).
*/
#include <nvs_flash.h>
#include <Preferences.h>
Preferences preferences;

String daysOfTheWeek[7] = { "Domingo", "Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado" };
String monthsNames[12] = { "Enero", "Febrero", "Marzo", "Abril", "Mayo",  "Junio", "Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre" };


//Libería para la comunicación Bluetooth (BluetoothSerial.h)
#include "BluetoothSerial.h"

// Verficiación del estado del Bluetooth
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth no está habilitado! Ejecute `make menuconfig` y actívelo
#endif

BluetoothSerial SerialBT;

//Librería DHT para interactuar con el sensor de temperatura y humedad DHT11.
#include <DHT.h>  //https://github.com/adafruit/DHT-sensor-library
#define DHTPIN 26
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

// Librería Arduino.h, suele incluirse automáticamente y proporciona funcionalidad básica de Arduino.
#include <Arduino.h>;
```

```
53
54
55  /*
56  ////////////////////////////////////////
57  Variables Globales y Definiciones de Búfer
58  ////////////////////////////////////////
59  */
60
61  //SensorPin representa el pin conectado al sensor de humedad del suelo.
62  const int SensorPin=13;
63
64  //SCREEN_WIDTH y SCREEN_HEIGHT especifican las dimensiones de la pantalla OLED.
65  #define SCREEN_WIDTH 128
66  #define SCREEN_HEIGHT 64
67
68  /* Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
69  Oled es una instancia de la clase Adafruit_SSD1306 para controlar la pantalla.*/
70  Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
71
72
73  /*////////////////////////////////////////
74        Controlar relay bomba de agua
75  ////////////////////////////////////////*/
76
77  // RELAIS es el pin conectado al relé que controla la bomba.
78  #define RELAIS 32
79  // BUF_SIZE define el tamaño de los buffers de caracteres usados para almacenar cadenas.
80  #define BUF_SIZE    32
81
82  /* Estas variables se utilizan para almacenar:
83     Las lecturas de los sensores.
84     El estado de la bomba.
85     La información de temporización.
86  */
87  char estado_bomba[BUF_SIZE] = "Apagada";
88  char clase_riego[BUF_SIZE] = "Automatico";
89
90  char buffer [BUF_SIZE] = "";
91
92  int periodo = 2000;
93  unsigned long TiempoAhora = 0;
94  int tipo_riego=2;   /// riego automatico por defecto
95  int humedad,bomba,sensorValue,t,h,readId,pgAct = 0;
96
97
98  //////************************************************************//////
99  void setup() {
100    //Se inicializa la comunicación serie a una tasa de baudios de 115200.
101   Serial.begin(115200);
102   delay(20);
103   //Se comprueba si el módulo DS3231 RTC está conectado. Si no, entra en un bucle infinito.
104   if (! rtc.begin())
105   {
106     Serial.println("DS3231 RTC Modulo no esta conectado");
107     while (1);
108   }
109   /*Se comprueba si el RTC ha perdido alimentación.
110   Si es así, reinicia la hora utilizando los valores actuales de fecha y hora.*/
111   if (rtc.lostPower())
112   {
```

```
    Serial.println("RTC falla de alimentación, restablecer la hora!");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  rtc.clearAlarm(1);
  rtc.disableAlarm(1);
  rtc.clearAlarm(2);
  rtc.disableAlarm(2);
  DateTime now = rtc.now(); // Obtener hora actual

  // Se Imprimie la hora y fecha actual
  char buff[] = "Hora de inicio hh:mm:ss DDD, DD MMM YYYY";
  Serial.println(now.toString(buff));

  SerialBT.begin("Riego_automatico_ESP32");
  Serial.println("El dispositivo comenzó, ¡ahora puedes emparejarlo con bluetooth!");
  // Se Inicializa el almacenamiento de preferencias con el nombre "mi-app".
  preferences.begin("my-app", false);
  delay(100);
  // Se Inicializa el sensor DHT.
  dht.begin();

  pinMode(SensorPin, INPUT);    ///.......... Pin del sensor de humedad.
  pinMode(RELAIS,OUTPUT);       ///.......... Relay que controla la bomba.


 /* Se inicializa la pantalla OLED usando la función begin().
    Si la inicialización falla, entra en un bucle infinito.*/
  if(!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Dirección 0x3D para 128x64
    Serial.println(F("SSD1306 la asignación falló"));
    for(;;);
  }
}
////*************************************************************////
void loop()
{
 /*
   Comprueba si ha transcurrido un cierto tiempo desde la última lectura del sensor.
   Si es así, lee los valores de humedad y temperatura del sensor DHT
   Y también del valor de humedad del suelo del pin analógico.
 */
 if (millis() > TiempoAhora + periodo) {
  TiempoAhora = millis();
  h = dht.readHumidity();
  t = dht.readTemperature();
  sensorValue = analogRead(SensorPin);
  humedad = ( 100 - ( (sensorValue/4095.00) * 100 ) );
  }
  if (Serial.available()) {       /// Leer informacion puerto serie y la envia al bluetooth
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available())     /// Leer informacion enviada desde bluetooth ///
  {
    //Inicializacion de variables de programacion de riego (Hora y Minutos).
    int h_on, m_on, h_off, m_off = 0;
    int domDay, lunDay, marDay, mierDay, jueDay, vierDay, sabDay=0;
    String datos = SerialBT.readString();     /// Guarda como string lo recibido ///
    tipo_riego = (datos.toInt());
    delay(10);
    //Automatico
    if (datos.startsWith("2"))
```

```
    {
      preferences.begin("my-app", false); // Inicializar el EEPROM en modo lectura y escritu
(false)
       tipo_riego=2;
      preferences.putInt("ModoProg", tipo_riego);    /// Guarda lo recibido en EEProm ///
    }
    //Manual
    if (datos.startsWith("1"))
    {
      preferences.begin("my-app", false);
      tipo_riego=1;
      preferences.putInt("ModoProg", tipo_riego);
    }
    //Manual
    if (datos.startsWith("3"))
    {
      preferences.begin("my-app", false);
      tipo_riego=3;
      preferences.putInt("ModoProg", tipo_riego);
    }
    //Programado
    //Separacion de cadena a diferentes variables para guardar el horario de riego.
    if (datos.startsWith("<")){
      //Horas
      datos.remove(0,1);
       h_on = (datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
       m_on = (datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
       h_off = (datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
       m_off = (datos.toInt());
      datos.remove(0,((datos.indexOf(">"))+1));
      //Dias
      domDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      lunDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      marDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      mierDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      jueDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      vierDay=(datos.toInt());
      datos.remove(0,((datos.indexOf(","))+1));
      sabDay=(datos.toInt());
      datos.remove(0,((datos.indexOf("."))+1));
      tipo_riego=4;

guardar_en_EEPROM(4,h_on,h_off,m_on,m_off,domDay,lunDay,marDay,mierDay,jueDay,vierDay,sabDay
      /// Guardar programa o horario de riego en EEprom ///
    }
    delay(10);
  }
  switch (tipo_riego) {
    case 1:           // riego manual activado manualmente
        encender_bomba(true);
      break;
    case 2:           // automatico segun humedad del suelo
```

```
231         if (humedad <= 35) {
232             encender_bomba(true);   /// Si la humedad del suelo es baja encender bomba ///
233         }
234           else {
235             encender_bomba(false);  /// Si la humedad del suelo es baja encender bomba ///
236           }
237       break;
238     case 3: // riego manual activado manualmente
239           encender_bomba(false);
240       break;
241     default:
242 
243       break;
244   }
245   mostrar_datos_x_oled();
246   enviar_datos_x_bluettoth();
247   Horario_de_riego();
248 }
249 ////**********END LOOP*****************////
250 ////*************************************************************************////
251 void Horario_de_riego()
252 {
253   //isScheduledON retorna un Booleano, si se cumple la hora y dia de riego
254     DateTime now = rtc.now();
255     if (state == false && isScheduledON(now))
256       {
257         encender_bomba(true);
258         state = true;
259         Serial.print("Riego programado iniciado: ");
260         Serial.print(now.hour());
261         Serial.print(":");
262         Serial.println(now.minute());
263       }
264       else if (state == true && !isScheduledON(now))
265       {
266         encender_bomba(false);
267         state = false;
268         Serial.print("Riego programado finalizado: ");
269         Serial.print(now.hour());
270         Serial.print(":");
271         Serial.println(now.minute());
272       }
273       delay(100);
274 }
275 ////*************************************************************************////
276 void enviar_datos_x_bluettoth()  {
277   DateTime ahora = rtc.now();
278   int pgAct=int(isScheduledON(ahora));
279 
280    int h_i = preferences.getInt("HoraI", 0); // Asignar dato por defecto (0)
281    int h_f = preferences.getInt("HoraF", 0);
282    int m_i = preferences.getInt("MinutoI", 0);
283    int m_f = preferences.getInt("MinutoF", 0);
284 
285   sprintf(buffer,
"%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,",humedad,t,h,tipo_riego,bomba,pgAct,h_i,m_i,h_f,m_f);  // Ur
los datos en una cadena para enviar por bluetooth
286   SerialBT.println(buffer);
287   delay(5);
288 
```

```
289 }
290 ////************************************************////
291 void encender_bomba(bool interruptor)
292 {
293   digitalWrite(RELAIS,!interruptor);  //// Envia true o false al pueto digital para encender
    apagar la bomba ////
294   if (interruptor){
295     strcpy(estado_bomba, "Encendida");
296     bomba=1;
297   }
298   else
299   {
300     strcpy(estado_bomba, "Apagada");
301     bomba=0;
302   }
303   delay(10);
304 }
305 ////************************************************////
306 void mostrar_datos_x_oled()
307 {
308   oled.cp437(true);   //Activar página de código 437
309   oled.setTextSize(1);
310   oled.setTextColor(WHITE);
311   oled.setCursor(0, 1);
312   oled.println("--RIEGO AUTOMATICO--");
313   oled.print("Humedad suelo:");
314   oled.print(humedad);
315   oled.println("% ");
316   oled.print("Temperatura:");
317   oled.print(t);
318   oled.write(248);
319   oled.println("C");
320   oled.print("Humedad aire:");
321   oled.print(h);
322   oled.println("%");
323   oled.print("Modo:");
324   tipo_riego=leer_de_EEPROM();
325
326   if (tipo_riego==2) strcpy(clase_riego, "Automatico");
327   if (tipo_riego==1) strcpy(clase_riego, "Manual");
328   if (tipo_riego==3) strcpy(clase_riego, "Manual");
329   if (tipo_riego==4) strcpy(clase_riego, "Programado");
330
331   oled.println(clase_riego);
332   oled.print("Bomba:");
333   oled.println(estado_bomba);
334   DateTime now = rtc.now(); // Obtener hora actual
335   char buff[] = "Hora hh:mm:ss";
336   oled.println(now.toString(buff));
337   char buff2[] = "Fecha: DD MMM YYYY";
338   oled.print(now.toString(buff2));
339   oled.display();
340   oled.clearDisplay();
341   delay(50);
342 }
343 /////-----------------------------------------------------------------/////
344 int leer_de_EEPROM()
345 {
346   int dato = preferences.getInt("ModoProg", 0);
347   delay(10);
```

```
  return dato;
}
///////--------------------------------------------------------------------///////
void guardar_en_EEPROM(int Modo, int HoraInicio, int HoraFin, int MinutoInicio, int MinutoFi
int domDay,int lunDay,int marDay,int mierDay,int jueDay,int vierDay, int sabDay)
{
  preferences.begin("my-app", false);
  preferences.putInt("ModoProg", Modo);
  preferences.putInt("HoraI", HoraInicio);
  preferences.putInt("HoraF", HoraFin);
  preferences.putInt("MinutoI", MinutoInicio);
  preferences.putInt("MinutoF", MinutoFin);

  preferences.putInt("domDia", domDay);
  preferences.putInt("lunDia", lunDay);
  preferences.putInt("marDia", marDay);
  preferences.putInt("mierDia", mierDay);
  preferences.putInt("jueDia", jueDay);
  preferences.putInt("vierDia", vierDay);
  preferences.putInt("sabDia", sabDay);

  Serial.println("Horario de riego se guardo en la EEPROM");
  delay(10);
}
///////--------------------------------------------------------------------///////
// Comprobar si esta programado el encendido
bool isScheduledON(DateTime date)
{
  int weekDay = date.dayOfTheWeek();
  // float hours = date.hour() + date.minute() / 60.0;
  float hora = date.hour();
  float minuto = date.minute();
  int pgAct = preferences.getInt("ModoProg", 0);
   int H_i = preferences.getInt("HoraI", 0);
   int H_f = preferences.getInt("HoraF", 0);
   int M_i = preferences.getInt("MinutoI", 0);
   int M_f = preferences.getInt("MinutoF", 0);

  int D1 = preferences.getInt("domDia", 10);
  int D2 = preferences.getInt("lunDia", 20);
  int D3 = preferences.getInt("marDia", 30);
  int D4 = preferences.getInt("mierDia", 40);
  int D5 = preferences.getInt("jueDia", 50);
  int D6 = preferences.getInt("vierDia",60);
  int D7 = preferences.getInt("sabDia", 70);

  if (D1==11) {D1=0;}
  if (D2==21) {D2=1;}
  if (D3==31) {D3=2;}
  if (D4==41) {D4=3;}
  if (D5==51) {D5=4;}
  if (D6==61) {D6=5;}
  if (D7==71) {D7=6;}

  if (D1==10) {D1=8;}
  if (D2==20) {D2=8;}
  if (D3==30) {D3=8;}
  if (D4==40) {D4=8;}
  if (D5==50) {D5=8;}
  if (D6==60) {D6=8;}
```

```
407   if (D7==70) {D7=8;}
408 
409   bool hourCondition = (hora >= H_i && minuto >= M_i && hora <= H_f && minuto < M_f);
410   bool dayCondition = (weekDay == D1 || weekDay == D2 ||  weekDay == D3 || weekDay == D4 ||
    weekDay == D5 || weekDay == D6 || weekDay == D7 );
411   if (hourCondition && dayCondition)
412   {
413     return true;
414   }
415   return false;
416 }
417 ///_____________________________________________________________////
418 
419 
```