IOT HOLIDAY ASSIGNMENT

1) Write a Embeded C program to Create a Weather Reporting System that provides real-time environmental data to users.

```
#include <Wire.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ThingSpeak.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

```
const char* ssid = "Wokwi-GUEST";
const char* password = "";
String APIKEY = "8c9f6eac52a56ea89b8c36162a6d60c7";
String CityID = "1185241"; // Example City ID
WiFiClient client;
char servername[] = "api.openweathermap.org";
String result;
```

```
unsigned long channelID = 2235258;
const char* writeAPIKey = "IU90PCW31HECJ1V5";
```

```
void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  delay(200);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Connecting...");
  display.display();

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    display.print(".");
    display.display();
  }
```

```
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Connected to WiFi");
  display.display();
  delay(1000);
  display.clearDisplay();
}
```

```
void loop() {
```

```
if (client.connect(servername, 80)) {
  client.println("GET /data/2.5/weather?id=" + CityID + "&units=metric&APPID=" + APIKEY);
  client.println("Host: api.openweathermap.org");
  client.println("User-Agent: ArduinoWiFi/1.1");
  client.println("Connection: close");
  client.println();
} else {
  Serial.println("connection failed");
  Serial.println();
}
```

```
while (client.connected() && !client.available())
  delay(1);

while (client.connected() || client.available()) {
  char c = client.read();
  result = result + c;
}
```

```
client.stop();
```

```
// Parse JSON
DynamicJsonDocument doc(1024);
deserializeJson(doc, result);
```

```
String location = doc["name"];
String country = doc["sys"]["country"];
float temperature = doc["main"]["temp"].as<float>();
int humidity = doc["main"]["humidity"];
float windSpeed = doc["wind"]["speed"].as<float>();
```

```
// Send data to ThingSpeak
ThingSpeak.begin(client);
ThingSpeak.setField(1, temperature);
ThingSpeak.setField(2, humidity);
ThingSpeak.setField(3, windSpeed);
int httpCode = ThingSpeak.writeFields(channelID, writeAPIKey);
if (httpCode == 200) {
  Serial.println("Data sent to ThingSpeak successfully");
} else {
  Serial.print("Error sending data to ThingSpeak. HTTP code: ");
  Serial.println(httpCode);
}
```

```
Serial.println();
Serial.print("Country: ");
Serial.println(country);
Serial.print("Location: ");
Serial.println(location);
Serial.print("Location ID: ");
Serial.println(CityID); // Print the City ID you used
Serial.printf("Temperature: %.2f°C\r\n", temperature);
Serial.printf("Humidity: %d %%\r\n", humidity);
Serial.printf("Wind speed: %.2f m/s\r\n", windSpeed);
```

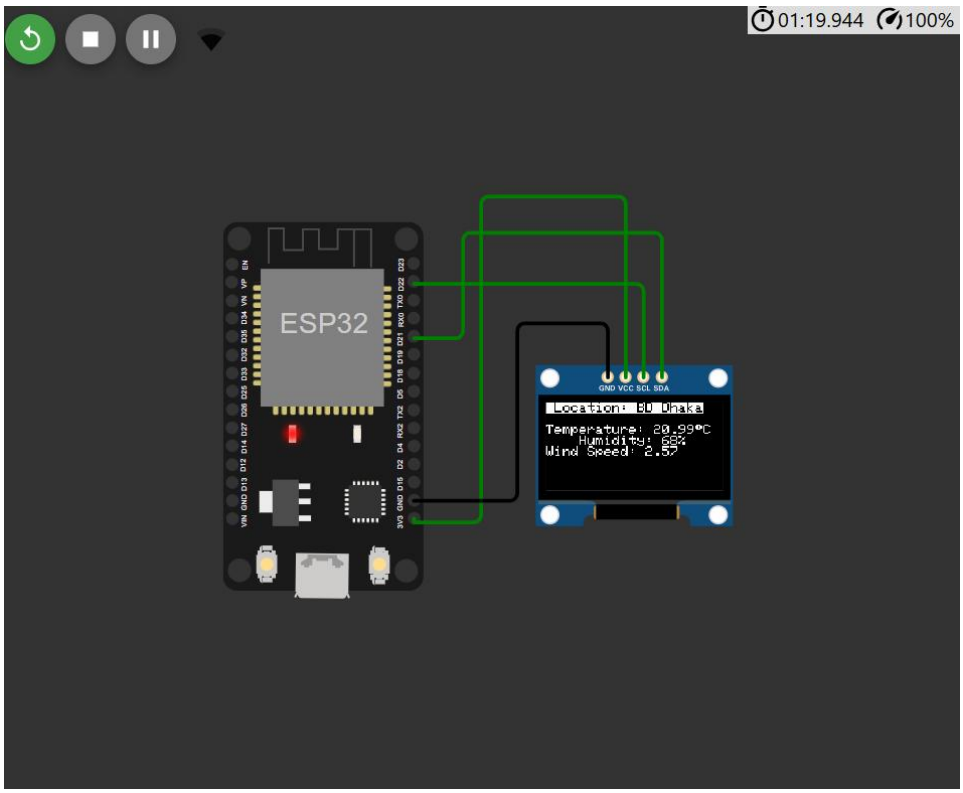
```
display.clearDisplay();
display.setCursor(0, 0);
display.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
display.print(" Location: ");
display.print(country);
display.print(" ");
```

```
  display.println(location);

  display.println();
  display.setTextColor(SSD1306_WHITE, SSD1306_BLACK);
  display.print("Temperature: ");
  display.print(temperature, 2);
  display.print((char)247);
  display.print("C     ");
  display.print("Humidity: ");
  display.print(humidity);
  display.println("%  ");
  display.print("Wind Speed: ");
  display.print(windSpeed, 2);
```

```
  display.display();

  delay(60000); // 1 minute delay
}
```



2) Write a Embeded C program to Create a Home Automation System that simpllifies daily routines(Any 2 devices) by controlling devices remotely.

```
// Home Automation System

// Thingspeak Server dB Public View: https://thingspeak.com/channels/2052162
```

```
#include <DHT.h>
#define DHTPIN 15
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

```
#include <WiFi.h>
#include "ThingSpeak.h" // always include thingspeak header file after other header files and
custom macros
```

```
char ssid[] = "Wokwi-GUEST";   // your network SSID (name)
char pass[] = "";   // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;
```

```
// Weather station channel details
unsigned long weatherStationChannelNumber = 2052162;
unsigned long myChannelNumber = 2052162;
const char * myWriteAPIKey = "QS963Q0GCOTDY6GY";
```

```
// Timer variables
unsigned long lastTime = 0;
unsigned long timerDelay = 30000;
```

```
int statusCode = 0;
int field[8] = {1,2,3,4};
```

```
int ch1 = 0;
int ch2 = 0;
int ch3 = 0;
int ch4 = 0;
```

```
#define ch1Pin 23
#define ch2Pin 22
#define ch3Pin 21
#define ch4Pin 19
float Prevtemp = 0;
```

```
void setup() {
  Serial.begin(115200);      // Initialize serial
```

```
  // Pin Mode declaration
  pinMode(ch1Pin, OUTPUT);
  pinMode(ch2Pin, OUTPUT);
  pinMode(ch3Pin, OUTPUT);
  pinMode(ch4Pin, OUTPUT);
  dht.begin();
```

```
  while (!Serial) {  ;} // wait for serial port to connect. Needed for Leonardo native USB
port only
//  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
```

```
  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println("Wokwi");
```

```
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open
or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("WiFi Connected");
    delay(1000);
  }
  Serial.println("Welcome at Smart Home");
  delay(1000);
}
```

```
void loop() {
    // use ThingSpeak.readMultipleFields(channelNumber, readAPIKey) for private channels
    statusCode = ThingSpeak.readMultipleFields(weatherStationChannelNumber);

    if(statusCode == 200)
    {
      // Fetch the stored data
      ch1 = ThingSpeak.getFieldAsInt(field[0]); // Field 1
      ch2 = ThingSpeak.getFieldAsInt(field[1]); // Field 2
      ch3 = ThingSpeak.getFieldAsInt(field[2]); // Field 3
      ch4 = ThingSpeak.getFieldAsInt(field[3]); // Field 4
    }
    else{Serial.println("Problem reading channel. HTTP error code " + String(statusCode));}
```

```
    float temp = dht.readTemperature();
    float humidity = dht.readHumidity();
    Serial.print("weather   ");
    if (isnan(temp) || isnan(humidity)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }
    String message = "temp:  " + String(temp) + "    humidity:  " + String(humidity);
    Serial.println(message);
    delay(500);
```

```
    if (temp >= 35){
      ch1 = 1;
    }
    else{
      ch1 = 0;
    }
```

```
    Serial.println("Ch1: " + String(ch1));
    Serial.println("Ch2: " + String(ch2));
    Serial.println("Ch3: " + String(ch3));
    Serial.println("Ch4: " + String(ch4));

    //    Hardware Control
    if (ch1 >= 1){digitalWrite(ch1Pin, HIGH);}
    if (ch1 == 0){digitalWrite(ch1Pin, LOW);}
```

```
    if (ch2 >= 1){digitalWrite(ch2Pin, HIGH);}
    if (ch2 == 0){digitalWrite(ch2Pin, LOW);}
```

```
    if (ch3 >= 1){digitalWrite(ch3Pin, HIGH);}
    if (ch3 == 0){digitalWrite(ch3Pin, LOW);}
```

```
    if (ch4 >= 1){digitalWrite(ch4Pin, HIGH);}
    if (ch4 == 0){digitalWrite(ch4Pin, LOW);}
```

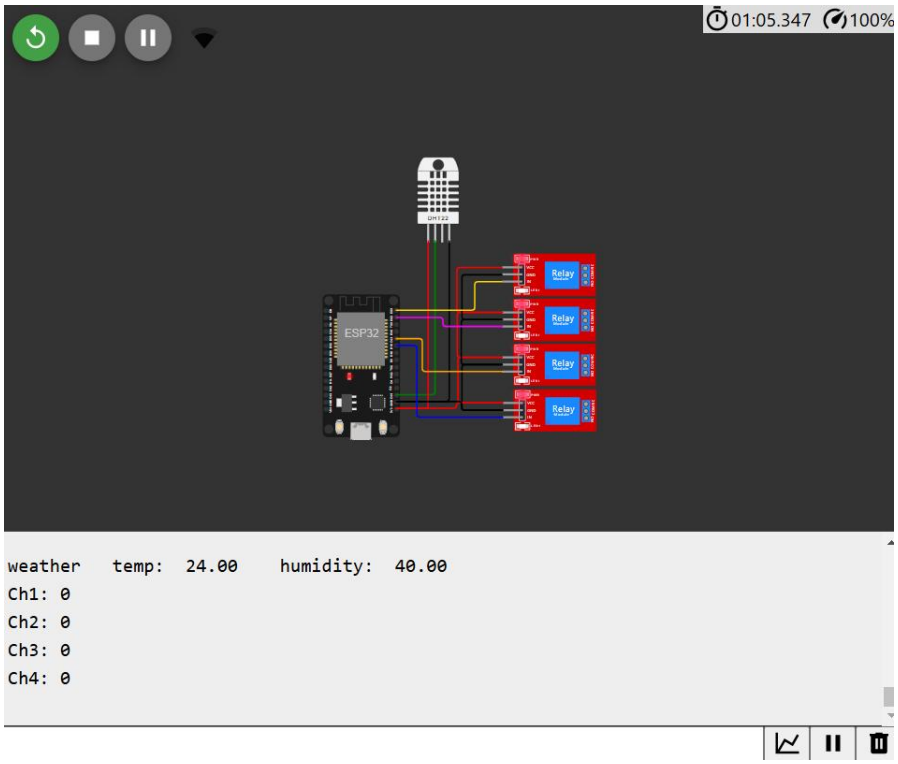
```
    if (temp != Prevtemp){
          Prevtemp = temp;
          // Serial.println(temp);
          // Serial.println(Prevtemp);
          //    upload data:
          ThingSpeak.setField(1, ch1);
          ThingSpeak.setField(2, ch2);
          ThingSpeak.setField(3, ch3);
          ThingSpeak.setField(4, ch4);
          ThingSpeak.setField(5, temp);
          ThingSpeak.setField(6, humidity);

          // Write to ThingSpeak.
          int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

          if(x == 200){
            Serial.println("Channel update successful.");
          }
          else{
            Serial.println("Problem updating channel. HTTP error code " + String(x));
          }
        }

        Serial.println();
        delay(6000); // no need to fetch too often

}
```



3) Write a Embedded C program to Create an Air Pollution Monitoring System that tracks air quality levels in real-time to ensure a healthier environment.

```
//Air Pollution Monitoring System
#define name value#define BLYNK_TEMPLATE_ID "TMPL6kWN92xgM"
#define BLYNK_TEMPLATE_NAME "Automated Air purifier"
#define BLYNK_AUTH_TOKEN "29-TfEOHXuD37x_ERtbiYVxHfZMiodqj"

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <BlynkSimpleEsp32.h>
#include <WiFi.h>
```

```
// Define the pins for the DHT22 sensor
#define DHTPIN 2  // Replace with the actual pin connected to DHT22
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);  // 0x27 is the I2C address of the LCD
const int potPin = 34;  // Replace with the actual pin connected to the potentiometer
const int ledPin = 4;   // Replace with the actual pin connected to the LED
```

```
char ssid[] = "Wokwi-GUEST";
char pass[] = "";
```

```
BlynkTimer timer;
```

```
void sendData() {
  // Read temperature and humidity from the DHT22 sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
```

```
  // Read gas value from the potentiometer
  int gasValue = analogRead(potPin);
```

```
  // Send data to Blynk
  Blynk.virtualWrite(V1, temperature);
  Blynk.virtualWrite(V2, humidity);
  Blynk.virtualWrite(V3, gasValue);
}
```

```
void displayMessage(String line1, String line2, int delayTime = 2000) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
  delay(delayTime);
}
```

```
void setup() {
  // Initialize the LCD
  lcd.init();
  lcd.backlight();
```

```
  // Initialize DHT sensor
  dht.begin();
```

```
  // Initialize the LED pin
  pinMode(ledPin, OUTPUT);
```

```
  // Connect to Wi-Fi
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
  }

  // Initialize Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  // Map virtual pins
  Blynk.virtualWrite(V1, 0); // Initialize with 0
  Blynk.virtualWrite(V2, 0); // Initialize with 0
  Blynk.virtualWrite(V3, 0); // Initialize with 0

  // Setup a function to be called every 5 seconds
  timer.setInterval(3000L, sendData);
}

void loop() {
  Blynk.run();
  timer.run();

  // Read temperature and humidity from the DHT22 sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read gas value from the potentiometer
  int gasValue = analogRead(potPin);

  // Determine air level based on the specified conditions
  String airLevel;

  // Check temperature and humidity conditions
  if ((temperature >= 22 && temperature <= 30) && (humidity > 30 && humidity < 60)) {
    airLevel = "Good";
  } else if ((temperature >= 30 && temperature <= 40) && (humidity >= 60 && humidity <= 70)) {
    airLevel = "Normal";
  } else {
    airLevel = "Bad";
  }

  // Determine gas level based on the criteria
  String gasLevel;

  if (gasValue >= 0 && gasValue <= 1364) {
    gasLevel = "Good";
  } else if (gasValue >= 1365 && gasValue <= 2730) {
    gasLevel = "Normal";
  } else {
    gasLevel = "Bad";
  }

  // Determine air quality based on the criteria
  String airQuality;

  if ((airLevel == "Good" || airLevel == "Normal") && (gasLevel == "Good" || gasLevel ==
"Normal")) {
    airQuality = "Good Air Quality";
  } else {
    airQuality = "Bad Air Quality";
  }
```

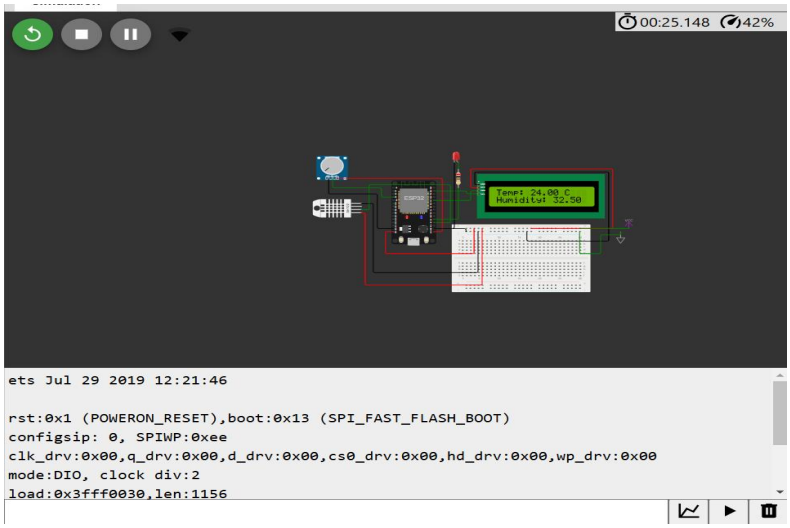
```
  // Display temperature and humidity on the LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: " + String(temperature) + " C");
  lcd.setCursor(0, 1);
  lcd.print("Humidity: " + String(humidity) + " %");
  delay(2000); // Display temperature and humidity for 2 seconds
```

```
  // Display air level on the LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Air Level: " + airLevel);
  delay(2000); // Display air level for 2 seconds
```

```
  // Display gas level and gas value on the LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Gas Level: " + gasLevel);
  lcd.setCursor(0, 1);
  lcd.print("Gas Value: " + String(gasValue));
  delay(2000); // Display gas level and value for 2 seconds
```

```
  // Display air quality on the LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Air Quality: ");
  lcd.setCursor(0, 1);
  lcd.print(airQuality);
  delay(2000); // Display air quality for 2 seconds
```

```
  // Control the LED based on air quality
  if (airQuality == "Bad Air Quality") {
    digitalWrite(ledPin, HIGH); // Turn on the LED
  } else {
    digitalWrite(ledPin, LOW); // Turn off the LED
  }
}
```



4) Write a Embedded C program to Create an IOT-based Smart Irrigation System for Agriculture that Automates Watering based on weather and Soil Conditions.

```
// IoT-based Irrigation System for ThingSpeak
// Based on ESP32 WOKWI Simulator by ThinkIOT
// ThingSpeak channel can be found here: https://thingspeak.com/channels/2383114

#include <WiFi.h>
#include "ThingSpeak.h"
#include "DHTesp.h"

const int SOIL_MOISTURE_PIN = 34;
const int SPRINKLER_CONTROL_PIN = 5;
const int DHT_PIN = 15;
DHTesp dhtSensor;

int MOISTURE_THRESHOLD_LOW = 15;            // Set Activation threshold in percentage
int MOISTURE_THRESHOLD_HIGH = 55;           // Set Deactivation threshold in percentage
bool SPRINKLER_ACTIVATION_STATUS = false;

char* WIFI_NAME = "Wokwi-GUEST";
char* WIFI_PASSWORD = "";
int myChannelNumber = 2546422;              // ThingSpeak channel ID
char* myApiKey = "54NGG6QX49UBG6O1";        // ThingSpeak channel write API key
WiFiClient client;

void setup()
{
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  Serial.println("Connecting...");
  Serial.println("Wi-Fi connected");
  Serial.println("Local IP: " + String(WiFi.localIP()));
  Serial.println("-------------");
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);

  pinMode(SPRINKLER_CONTROL_PIN, OUTPUT);
}

void loop()
{
  int soilMoisturePercentage = map(analogRead(SOIL_MOISTURE_PIN), 0, 4095, 0, 100);
  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  ThingSpeak.setField(2,data.temperature);
  ThingSpeak.setField(3,data.humidity);

  if ( soilMoisturePercentage < MOISTURE_THRESHOLD_LOW){
    SPRINKLER_ACTIVATION_STATUS = true;
    digitalWrite(SPRINKLER_CONTROL_PIN, HIGH); //
  }else{
    SPRINKLER_ACTIVATION_STATUS = false;
    digitalWrite(SPRINKLER_CONTROL_PIN, LOW); // Turn off sprinkler and LED
  }

  // Print status
  Serial.print("Soil Moisture Percentage: ");
  Serial.print(soilMoisturePercentage);
  Serial.println("%");

  Serial.println("Temp: " + String(data.temperature, 2) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");
```

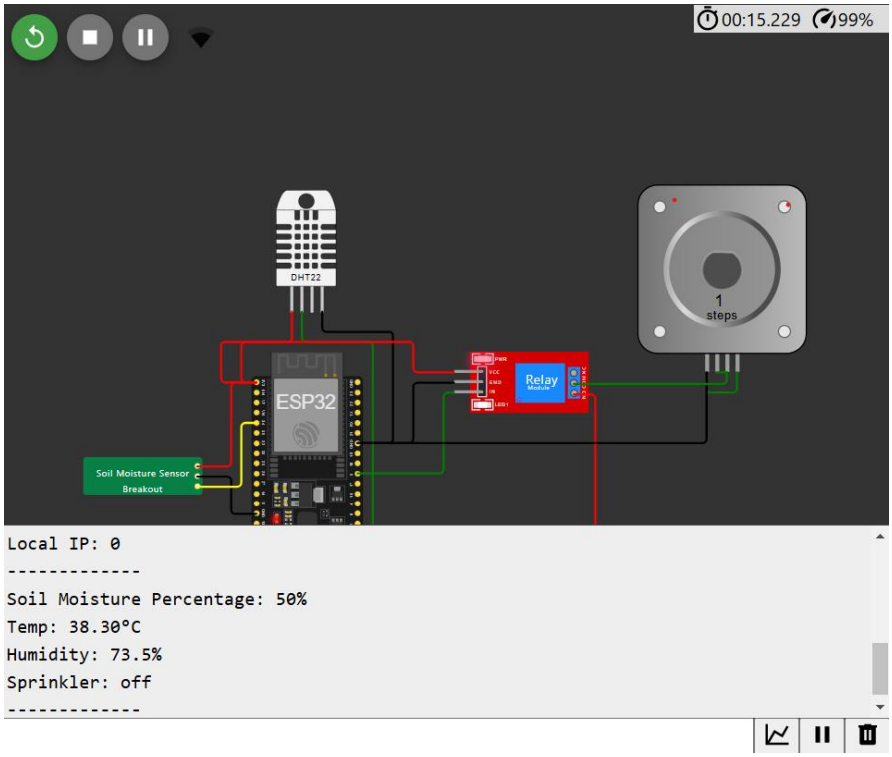
```
  Serial.print("Sprinkler: ");
  Serial.println(SPRINKLER_ACTIVATION_STATUS ? "on" : "off");

  // Send data to ThingSpeak
  ThingSpeak.setField(1, soilMoisturePercentage);
  ThingSpeak.setField(4, SPRINKLER_ACTIVATION_STATUS);

  int x = ThingSpeak.writeFields(myChannelNumber, myApiKey);

  Serial.println("-------------");

  delay(15000); // Thingspeak allows for an update every 15 seconds
}
```



5) Write a Emedded C Program to Create a Smart Alarm Clock that adjusts to your schedule and Environment,Waking you up intelligently.

/* ----- C Program for Arduino based Alarm Clock ---- */

```
#include <Wire.h>

#include<EEPROM.h>

#include <RTClib.h>

#include <LiquidCrystal.h>

const int rs = 8;
const int en = 9;
const int d4 = 10;
const int d5 = 11; //DISPLAY
const int d6 = 12;
const int d7 = 13;

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

```
RTC_DS1307 RTC;
int temp,inc,hours1,minut,add=11;
int next=7;
int INC=6;
int set_mad=5;
#define buzzer 3
int HOUR,MINUT,SECOND;

void setup()
{
 Wire.begin();
 RTC.begin();
 lcd.begin(16,2);
 pinMode(INC, INPUT);
 pinMode(next, INPUT);
 pinMode(set_mad, INPUT);
 pinMode(buzzer, OUTPUT);
 digitalWrite(next, HIGH);
 digitalWrite(set_mad, HIGH);
 digitalWrite(INC, HIGH);

  lcd.setCursor(0,0);
  lcd.print("Real Time Clock");
  lcd.setCursor(0,1);
  lcd.print("Circuit Digest ");
   delay(2000);

 if(!RTC.isrunning())
 {
 RTC.adjust(DateTime(__DATE__,__TIME__));
```

```
 }
}

void loop()
{
  int temp=0,val=1,temp4;
  DateTime now = RTC.now();
  if(digitalRead(set_mad) == 0)      //set Alarm time
  {
    lcd.setCursor(0,0);
    lcd.print("  Set Alarm  ");
   delay(2000);
   defualt();
   time();
   delay(1000);
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("  Alarm time ");
   lcd.setCursor(0,1);
   lcd.print(" has been set  ");
   delay(2000);
 }
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Time:");
 lcd.setCursor(6,0);
 lcd.print(HOUR=now.hour(),DEC);
 lcd.print(":");
 lcd.print(MINUT=now.minute(),DEC);
 lcd.print(":");
```

```
 lcd.print(SECOND=now.second(),DEC);
 lcd.setCursor(0,1);
 lcd.print("Date: ");
 lcd.print(now.day(),DEC);
 lcd.print("/");
 lcd.print(now.month(),DEC);
 lcd.print("/");
 lcd.print(now.year(),DEC);
 match();
 delay(200);
}
void defualt()
{
  lcd.setCursor(0,1);
  lcd.print(HOUR);
  lcd.print(":");
  lcd.print(MINUT);
  lcd.print(":");
  lcd.print(SECOND);
}
/*Function to set alarm time and feed time into Internal eeprom*/
void time()
{
  int temp=1,minuts=0,hours=0,seconds=0;
    while(temp==1)
    {
     if(digitalRead(INC)==0)
     {
      HOUR++;
      if(HOUR==24)
```

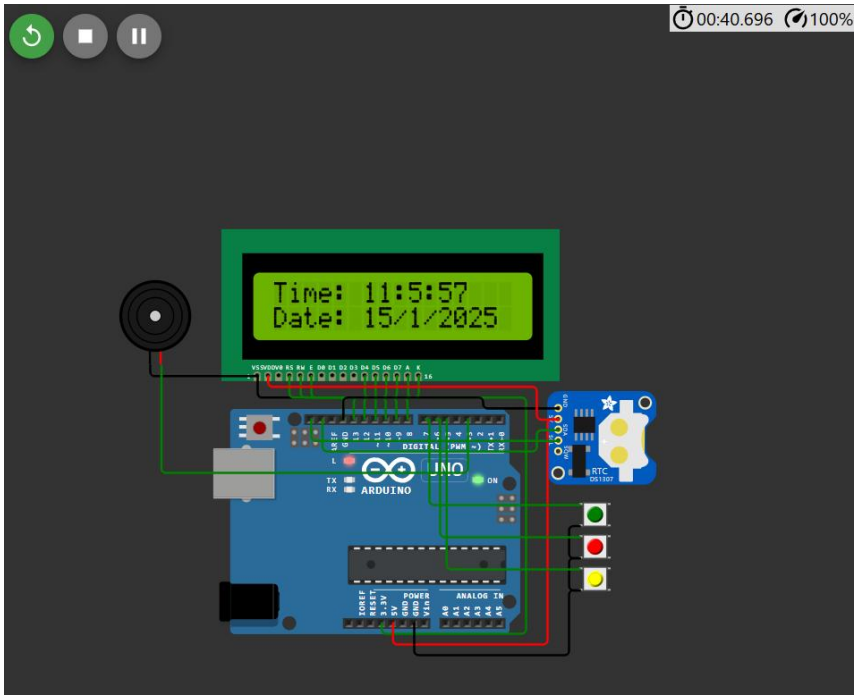
```
  {
   HOUR=0;
  }
  while(digitalRead(INC)==0);
 }
 lcd.clear();
  lcd.setCursor(0,0);
lcd.print("Set Alarm Time ");
//lcd.print(x);
lcd.setCursor(0,1);
lcd.print(HOUR);
lcd.print(":");
lcd.print(MINUT);
lcd.print(":");
lcd.print(SECOND);
delay(100);
if(digitalRead(next)==0)
{
  hours1=HOUR;
  EEPROM.write(add++,hours1);
 temp=2;
 while(digitalRead(next)==0);
}
}

while(temp==2)
{
 if(digitalRead(INC)==0)
 {
  MINUT++;
```

```
      if(MINUT==60)
      {MINUT=0;}
      while(digitalRead(INC)==0);
     }
     // lcd.clear();
    lcd.setCursor(0,1);
    lcd.print(HOUR);
    lcd.print(":");
    lcd.print(MINUT);
    lcd.print(":");
    lcd.print(SECOND);
    delay(100);
      if(digitalRead(next)==0)
      {
       minut=MINUT;
       EEPROM.write(add++, minut);
       temp=0;
       while(digitalRead(next)==0);
      }
    }
    delay(1000);
}

/* Function to chack medication time */
void match()
{
  int tem[17];
  for(int i=11;i<17;i++)
  {
    tem[i]=EEPROM.read(i);
```

```
  }
  if(HOUR == tem[11] && MINUT == tem[12])
  {
   beep();
   beep();
   beep();
   beep();
   lcd.clear();
   lcd.print("Wake Up.....");
   lcd.setCursor(0,1);
   lcd.print("Wake Up.....");
   beep();
   beep();
   beep();
   beep();
  }
}
/* function to buzzer indication */
void beep()
{
   digitalWrite(buzzer,HIGH);
   delay(500);
   digitalWrite(buzzer, LOW);
   delay(500);
}
```



DONE BY:
G.Sai Deekshitha
2211CS020175
AIML-THETA