2211CS020517

V. AISHWARYA

III AIML SIGMA

INTERNET OF THINGS

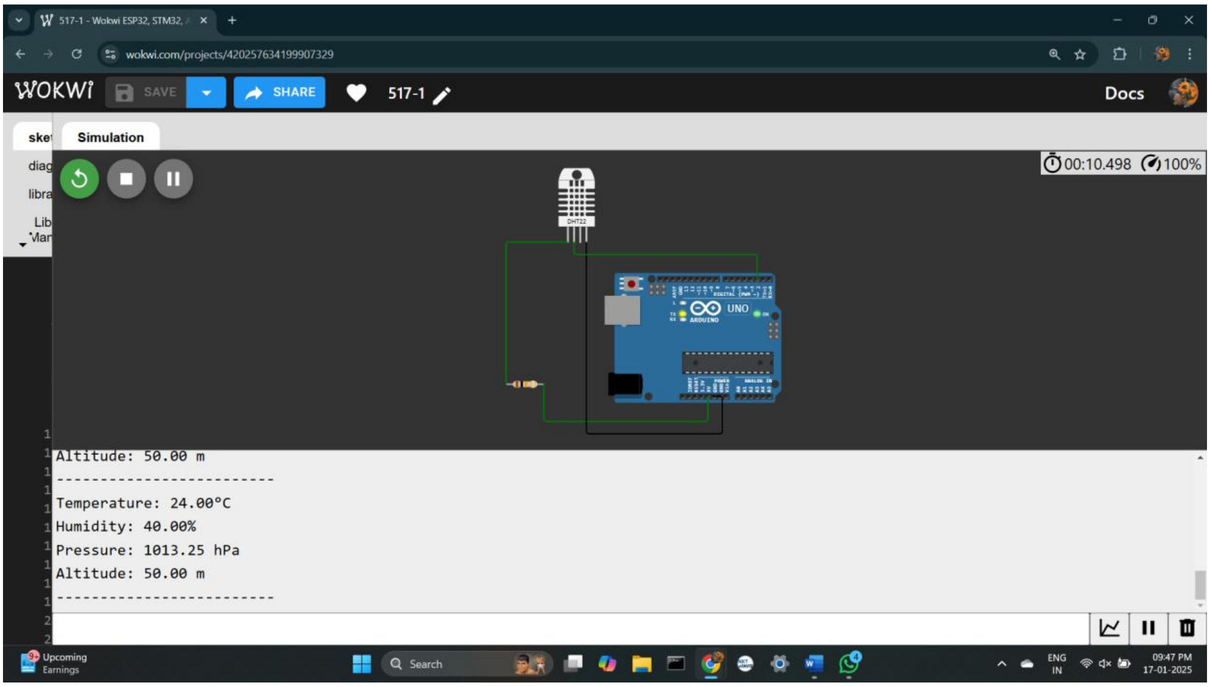
HOLIDAY ASSIGNMENT

1. **Write an Embedded C Program to Create a Weather Reporting System that provides real- time environmental data to users.**

Embedded C Program:

```
#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <DHT_U.h>


// Define DHT sensor pin and type

#define DHTPIN 2      // Pin where the DATA pin is connected

#define DHTTYPE DHT22 // DHT22 sensor type


// Initialize DHT sensor

DHT dht(2, DHT22);


void setup() {

  Serial.begin(9600);

  Serial.println("Weather Report System");


  // Initialize the DHT sensor

  dht.begin();

  Serial.println("DHT22 sensor initialized");
```

```
}

void loop() {
  // Read temperature and humidity from DHT22
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Check if readings are valid
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT22 sensor!");
  } else {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println("°C");

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println("%");
  }

  // Simulate pressure and altitude data (as BMP180 is unavailable)
  float pressure = 1013.25; // Sea level standard atmospheric pressure in hPa
  float altitude = 50.0;    // Simulated altitude in meters

  Serial.print("Pressure: ");
  Serial.print(pressure);
  Serial.println(" hPa");
  Serial.print("Altitude: ");
  Serial.print(altitude);
  Serial.println(" m");

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

  // Delay before the next reading
  delay(2000);
}
```



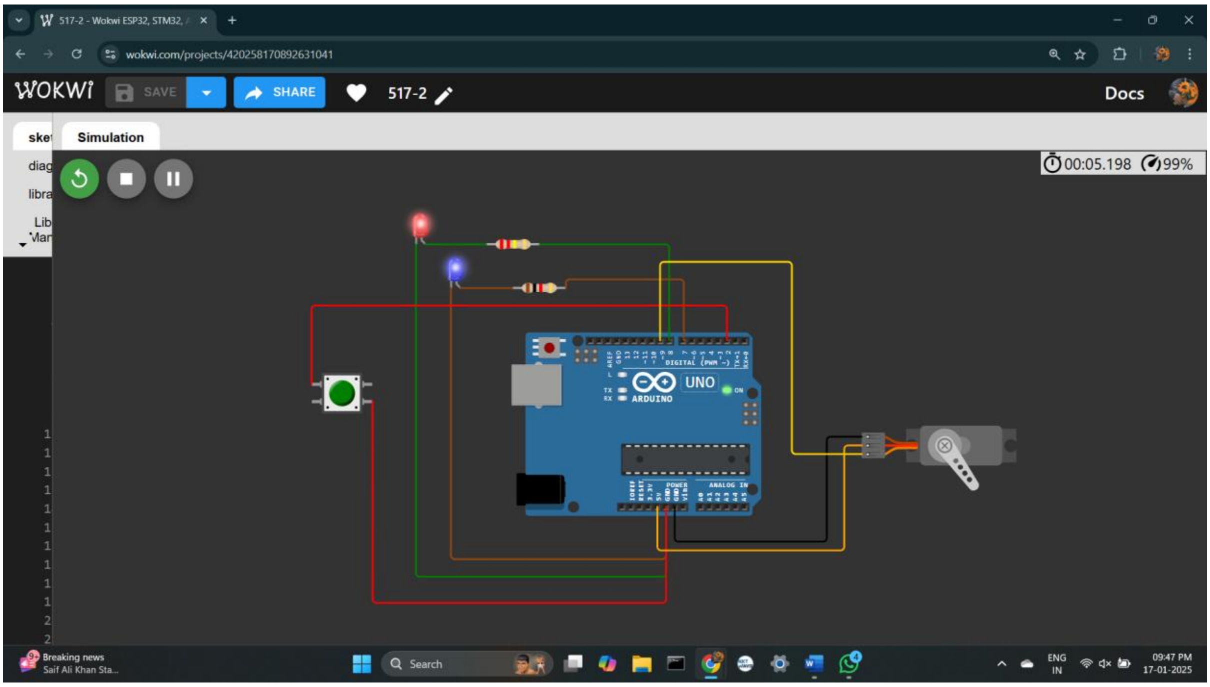
```
Altitude: 50.00 m
------------------------
Temperature: 24.00°C
Humidity: 40.00%
Pressure: 1013.25 hPa
Altitude: 50.00 m
------------------------
```

2. **Write a Embedded C Program to Create a Home Automation System that simplifies daily routines (Any 2 Devices) by controlling devices remotely.**

Embedded C Program:

```
#include <Servo.h>
```

```
#define LIGHT1_PIN 7  // Pin for Light 1 (LED 1)

#define LIGHT2_PIN 8  // Pin for Light 2 (LED 2)

#define BUTTON_PIN 2  // Pin for the push button

#define FAN_SERVO_PIN 9  // Pin for the servo motor (Fan)


Servo fanServo;        // Servo object for fan simulation

bool fanRunning = false; // State of the fan (false = OFF, true = ON)

int currentAngle = 90;   // Current angle of the servo

int step = 1;          // Step size for continuous movement


void setup() {

  // Pin modes for LEDs

  pinMode(7, OUTPUT);

  pinMode(8, OUTPUT);


  // Pin mode for button with internal pull-up

  pinMode(2, INPUT_PULLUP);


  // Attach the servo and set initial position

  fanServo.attach(9);

  fanServo.write(currentAngle); // Start fan at 90° (OFF position)


  // Turn off LEDs initially

  digitalWrite(7, LOW);

  digitalWrite(8, LOW);

}


void loop() {
  static bool buttonPressed = false;

  // Check if the button is pressed and toggle the fan state
  if (digitalRead(2) == LOW && !buttonPressed) {
    fanRunning = !fanRunning; // Toggle fan and lights state
    buttonPressed = true;

    // Toggle lights
    if (fanRunning) {
      digitalWrite(7, HIGH); // Turn Light 1 ON
      digitalWrite(8, HIGH); // Turn Light 2 ON
    } else {
      digitalWrite(LIGHT1_PIN, LOW);  // Turn Light 1 OFF
      digitalWrite(LIGHT2_PIN, LOW);  // Turn Light 2 OFF
      fanServo.write(90);           // Reset fan to 90° (OFF position)
    }
    delay(200); // Debounce delay
  } else if (digitalRead(BUTTON_PIN) == HIGH) {
    buttonPressed = false;
  }

  // Move the servo continuously if the fan is ON
  if (fanRunning) {
    currentAngle += step;

    // Reverse direction when reaching bounds
    if (currentAngle >= 180 || currentAngle <= 90) {
      step = -step;
    }

    fanServo.write(currentAngle);
    delay(10); // Delay for smooth movement
  }
}
```

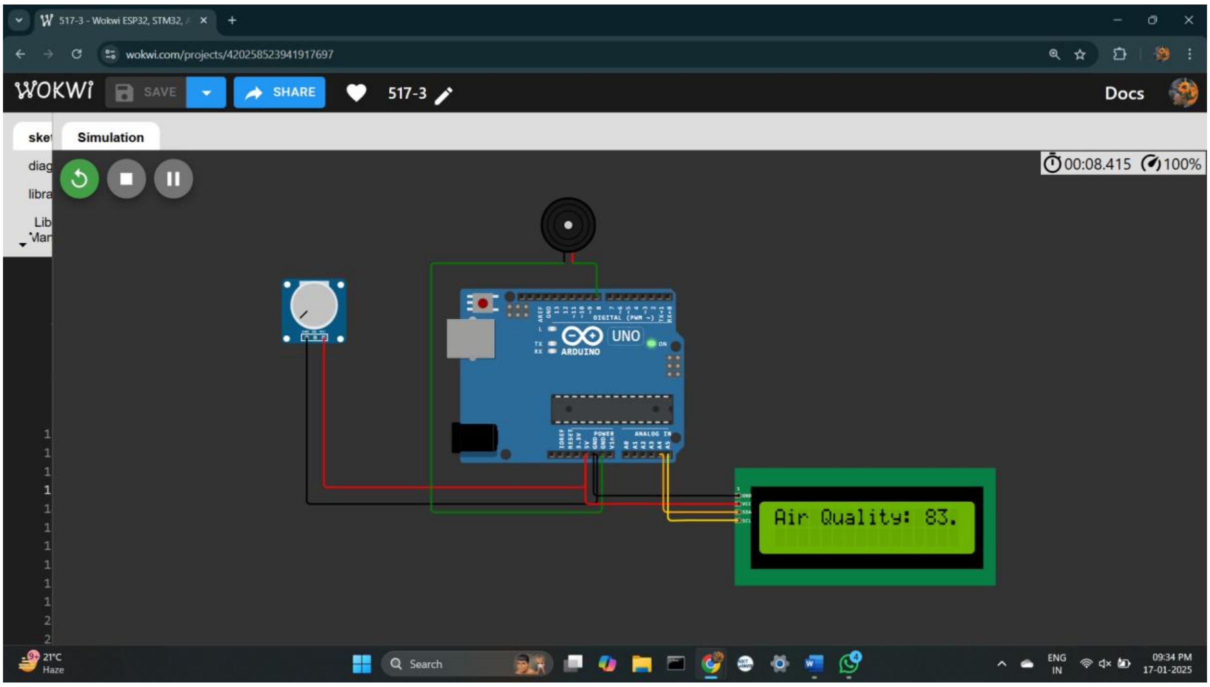


3. Write a Embedded C Program to Create an Air Pollution Monitoring System that tracks air quality levels in real-time to ensure a healthier environment.

Embedded C Program:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Define Pin Assignments
#define AIR_SENSOR_PIN A0  // Analog pin for Air Quality Sensor (use potentiometer for simulation)
```

```
#define BUZZER_PIN 8  // Pin for the Buzzer
#define LIGHT_PIN 9  // Pin for Light (LED)

// LCD I2C Setup (use address 0x27, but try 0x3F if not working)
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Initialize LCD with I2C address 0x27 and 16 columns, 2 rows

// Thresholds for air quality levels
#define GOOD_AIR_QUALITY 700
#define POOR_AIR_QUALITY 300

void setup() {
  // Start Serial Communication
  Serial.begin(9600);

  // Initialize Buzzer and Light pins
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);

  // Initialize the LCD
  lcd.begin(16, 2); // Initialize LCD with 16 columns, 2 rows
  delay(1000);  // Wait for 1 second for the LCD to initialize properly
  lcd.backlight();  // Turn on the LCD backlight
  lcd.setCursor(0, 0);  // Set cursor to the first column of the first row
  lcd.print("Air Quality Monitor");  // Display the title
  delay(2000);  // Wait for 2 seconds

  // Test if LCD is working by printing a test message
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hello World");
  delay(2000);  // Wait for 2 seconds
}

void loop() {
  // Read the air quality sensor value (simulated by potentiometer)
  int airSensorValue = analogRead(A0);

  // Map the sensor value to a percentage (0-100% for display)
  float airQualityPercentage = map(airSensorValue, 0, 1023, 0, 100);

  // Display the air quality on the LCD
  lcd.clear();  // Clear the screen
  lcd.setCursor(0, 0);  // Set cursor to the first column of the first row
  lcd.print("Air Quality: ");
  lcd.print(airQualityPercentage);
  lcd.print("%");

  // Buzzer and Light activation based on air quality
  if (airSensorValue > 700) {
    digitalWrite(8, LOW);  // Turn off Buzzer
    digitalWrite(9, HIGH);  // Turn on Light (Good air quality)
  } else if (airSensorValue < 300) {
    digitalWrite(8, HIGH);  // Turn on Buzzer
    digitalWrite(9, LOW);   // Turn off Light (Poor air quality)
  } else {
   digitalWrite(8, LOW);  // Turn off Buzzer
   digitalWrite(9, LOW);   // Turn off Light (Moderate air quality)
  }

  // Delay for a short time
  delay(500);
}
```


517-3 - Wokwi ESP32, STM32,
wokwi.com/projects/420258523941917697
WOKWi
SAVE
SHARE
517-3
Docs
Simulation
00:08.415
100%
UNO
ARDUINO

```
Air Quality: 83.
```

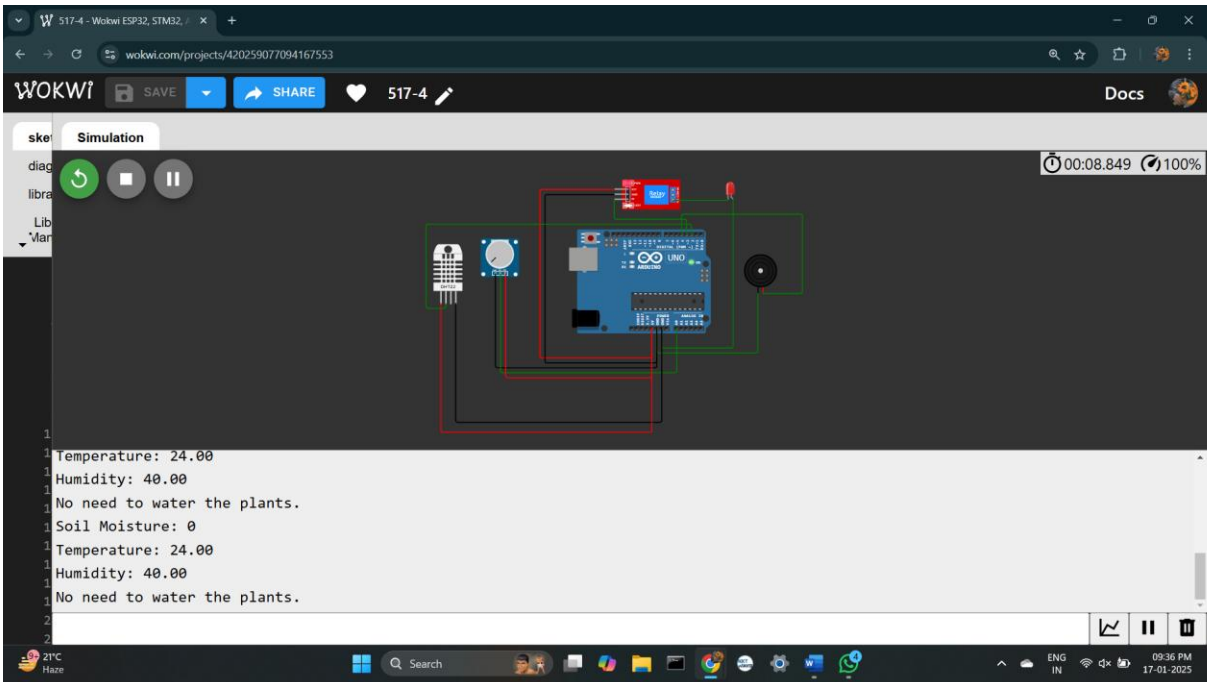
4. Write a Embedded C Program to Create an IoT-based Smart Irrigation System for Agriculture that automates watering based on weather and soil conditions

Embedded C Program:

```
#include <DHT.h>


#define DHTPIN 2       // Pin connected to the DHT sensor
#define DHTTYPE DHT22   // DHT 22 type
```

```
#define POT_PIN A0      // Potentiometer pin (simulating soil moisture sensor)
#define RELAY_PIN 3     // Relay module pin
#define BUZZER_PIN 4    // Buzzer pin (optional)

// Thresholds
#define SOIL_THRESHOLD 400  // Soil moisture threshold (adjustable)
#define TEMP_THRESHOLD 35   // Temperature threshold in Celsius
#define HUMIDITY_THRESHOLD 30  // Humidity threshold (%)

DHT dht(2, DHT22);

void setup() {
  Serial.begin(9600);
  dht.begin();

  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);

  digitalWrite(3, LOW);  // Ensure pump/relay is OFF initially
  digitalWrite(4, LOW);
}

void loop() {
  int soilValue = analogRead(A0);  // Read potentiometer value
  float temperature = dht.readTemperature(); // Read temperature
  float humidity = dht.readHumidity();       // Read humidity
  // Check if any reading failed
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Soil Moisture: ");
  Serial.println(soilValue);
  Serial.print("Temperature: ");
  Serial.println(temperature);
  Serial.print("Humidity: ");
  Serial.println(humidity);

  // Check conditions to water plants
  if (soilValue > SOIL_THRESHOLD && temperature < TEMP_THRESHOLD && humidity > HUMIDITY_THRESHOLD) {
    Serial.println("Watering the plants...");
    digitalWrite(3, HIGH);  // Turn on the relay
    digitalWrite(4, HIGH); // Optional alert
    delay(5000);  // Simulate watering duration (5 seconds)
    digitalWrite(3, LOW);  // Turn off the relay
    digitalWrite(4, LOW);
  } else {
    Serial.println("No need to water the plants.");
    digitalWrite(3, LOW);  // Ensure relay is off
    digitalWrite(4, LOW);
  }
```

```
 delay(2000);  // Wait before the next reading

}
```



```
Temperature: 24.00
Humidity: 40.00
No need to water the plants.
Soil Moisture: 0
Temperature: 24.00
Humidity: 40.00
No need to water the plants.
```

5. Write a Embedded C Program to Create a Smart Alarm Clock that adjusts to your schedule and environment, waking you up intelligently.

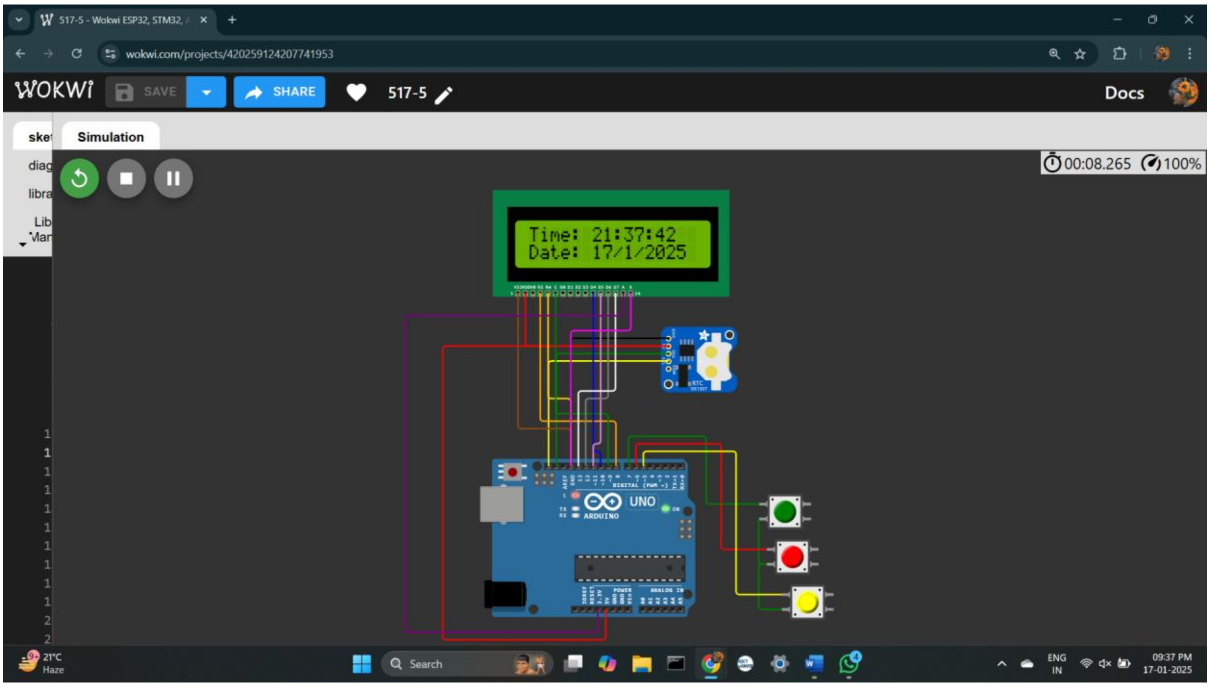
Embedded C Program:

```
#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);
}
```



```
Time: 21:37:42
Date: 17/1/2025
```