FPT UNIVERSITY

[SENSOR STATION - DISPLAY THE VALUE: TEMPERATURE, HUMIDITY, AIR QUALITY]

Member:

1. Phan Thai Khanh Nhi – DE180931
2. Nguyen Le Quang Huy – DE170716
3. Duong Tri Khang – DE170015

DANANG – March 19, 2024

Table of Contents

INTRODUCTION

The project is built to create a system to measure temperature, humidity, and the amount of gas in the air and send notifications to users if the gas level exceeds the permissible limit. Although it is currently only in the basic stage of development, this will be the foundation for an environmental quality measurement system.

PROCESS DIAGRAM

For Arduino UNO part:

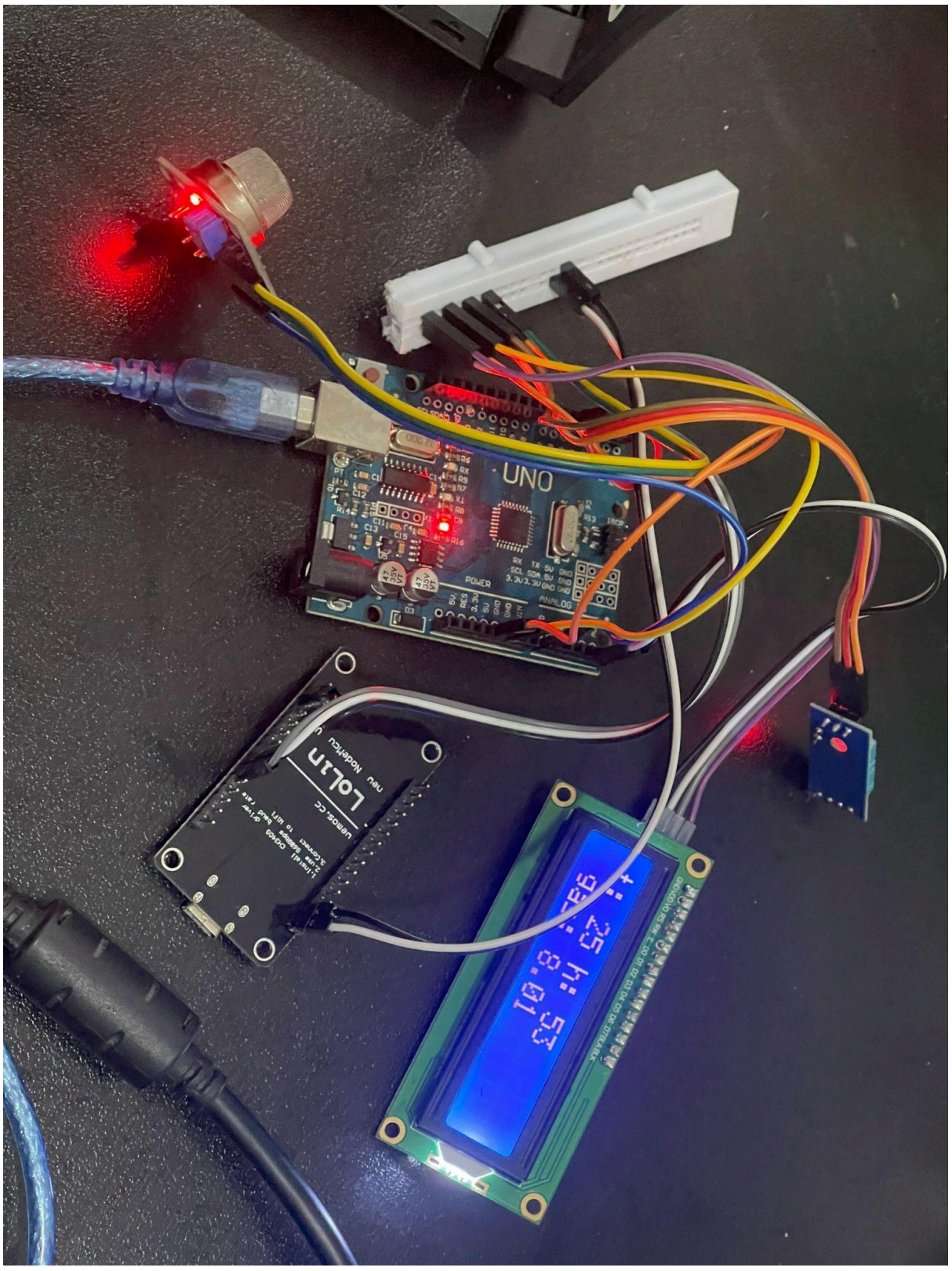
- DHT11 Temperature and Humidity Sensor: Connect the data pin of the DHT11 sensor to digital pin 2 on the Arduino UNO.
- MQ2 Gas Sensor: Connect the analog output pin of the MQ2 sensor to analog pin A0 on the Arduino UNO.
- LCD Display (16x2 I2C): Connect the SDA pin of the LCD to A4 and the SCL pin to A5 on the Arduino UNO.
- LEDs (connected to pins 3, 4, 5): Connect one end of each LED to digital pins 3, 4, and 5 on the Arduino UNO. Connect the other end of each LED to a current-limiting resistor, and then to the ground (GND) pin on the Arduino UNO.

For Dust Sensor part:

- Dust Sensor: Connect the output pin of the dust sensor to analog pin A5 on the Arduino UNO. Connect the LED control pin of the dust sensor to digital pin 12 on the Arduino UNO.

For ESP8266 with Blynk part:

- ESP8266 Module: Connect the PIR sensor to pin D1 on the ESP8266 module. Make sure the ESP8266 module is powered properly (refer to its datasheet or documentation).
- Blynk IoT Platform: Make sure the ESP8266 module is connected to the internet. Use the Blynk app to create virtual pins that correspond to the sensors and LEDs you want to control or monitor. Update the Blynk authentication token (BLYNK_AUTH_TOKEN) in the code with the token generated for the Blynk project. Connect the ESP8266 module to the same Wi-Fi network as the Blynk app.


UNO
POWER
ANALOG

FLOWCHART

- **Code for reading temperature and humidity data**

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT11.h>
DHT11 dht11(2);

LiquidCrystal_I2C lcd(0x27,16,2);

unsigned long count = 0;
unsigned long count1 = 0;
unsigned long count2 = 0;

void setup() {
  lcd.init();
  Serial.begin(9600);

  lcd.backlight();
  lcd.print("Hello world");
  lcd.setCursor(0,1);
  lcd.print("I love Arduino !");
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
}

void loop() {
    int temperature = 0;
```

```
    int humidity = 0;
    double gas = analogRead(A0);
    double percentGas = gas / 1024.0 * 100.0;

    int result = dht11.readTemperatureHumidity(temperature, humidity);

    if (result == 0) {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.print(" °C\tHumidity: ");
        Serial.print(humidity);
        Serial.println(" %");
    }
        Serial.print(" °C\tGas Percentage: ");
        Serial.print(percentGas);
        Serial.println(" %");
    if(percentGas >= 25) {
      count = millis();
    }
    if(count + 6000 > millis()) {
      digitalWrite(3, LOW);
    } else {
      digitalWrite(3, HIGH);
    }

    if(temperature >= 30) {
      count1 = millis();
    }
    if(count1 + 6000 > millis()) {
      digitalWrite(4, LOW);
    } else {
      digitalWrite(4, HIGH);
    }
```

```
    if(humidity >= 95) {
      count2 = millis();
    }
    if(count2 + 6000 > millis()) {
      digitalWrite(5, LOW);
    } else {
      digitalWrite(5, HIGH);
    }
    lcd.clear();
    lcd.print("t: " + String(temperature) + " h: " + String(humidity));
    lcd.setCursor(1, 1);
    lcd.print("gas: " + String(percentGas));
}
```

- **Code for reading data from a dust sensor**

```
#include <DHT11.h>
DHT11 dht11(2);

int measurePin = A5;
int ledPower = 12;

unsigned int samplingTime = 280;
unsigned int deltaTime = 40;
unsigned int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

int noise = 3;

void setup(){
```

```
  Serial.begin(9600);
  pinMode(ledPower,OUTPUT);
  pinMode(noise,INPUT);
}

void loop(){
  // DHT
  int temperature = 0;
  int humidity = 0;
  int result = dht11.readTemperatureHumidity(temperature, humidity);

  if (result == 0) {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print(" °C\tHumidity: ");
    Serial.print(humidity);
    Serial.println(" %");
  }

  // Noise
  if(digitalRead(noise) == 0) {

  }

  // Dust
  digitalWrite(ledPower,LOW);
  delayMicroseconds(samplingTime);

  voMeasured = analogRead(measurePin);

  delayMicroseconds(deltaTime);
  digitalWrite(ledPower,HIGH);
  delayMicroseconds(sleepTime);
```

```
  calcVoltage = voMeasured*(5.0/1024);
  dustDensity = 0.17*calcVoltage-0.1;

  if ( dustDensity < 0)
  {
    dustDensity = 0.00;
  }

  Serial.println("Raw Signal Value (0-1023):");
  Serial.println(voMeasured);

  Serial.println("Voltage:");
  Serial.println(calcVoltage);

  Serial.println("Dust Density:");
  Serial.println(dustDensity);

  delay(1000);
}
```

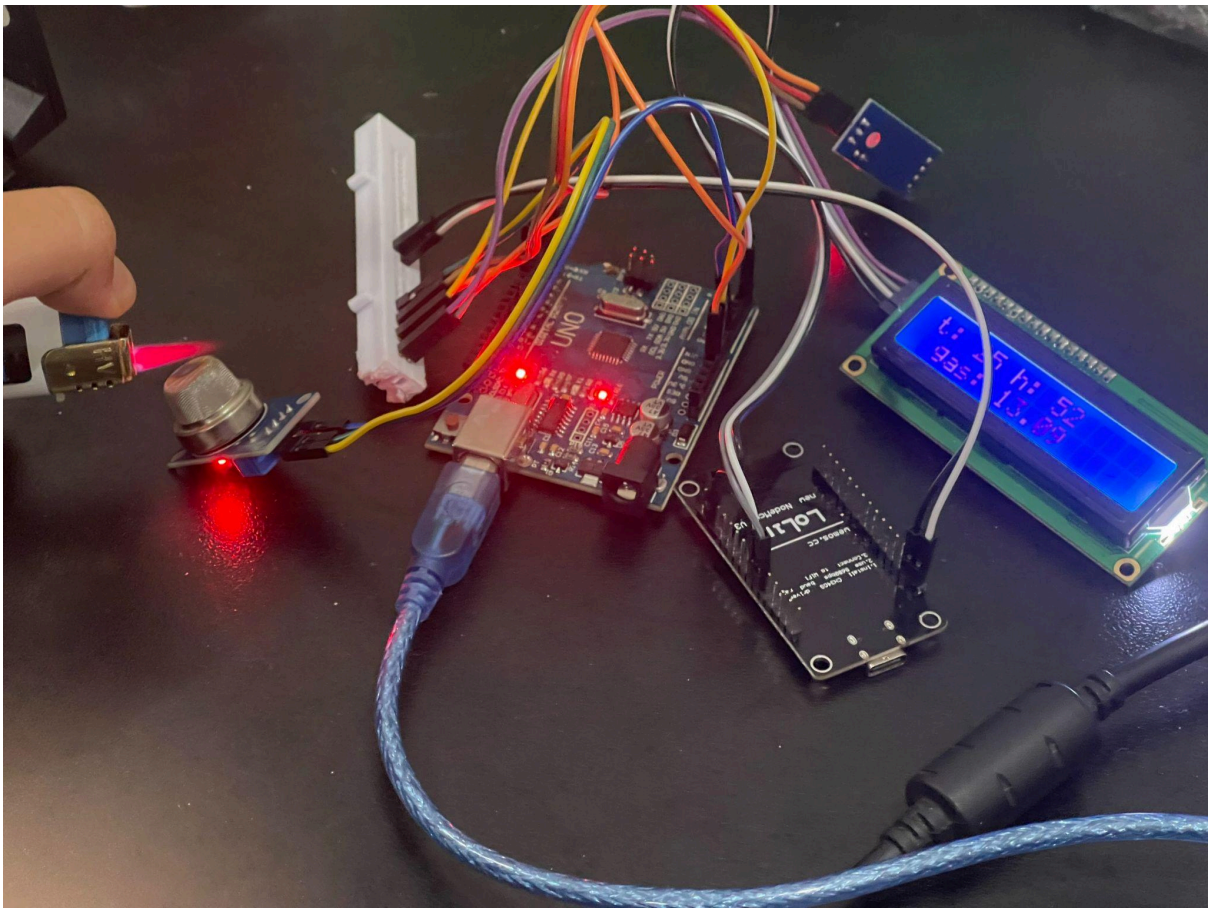
- **Code for utilizing the Blynk IoT platform**

```
#define BLYNK_TEMPLATE_ID "TMPL6W8Hmxd4b"
#define BLYNK_TEMPLATE_NAME "Alert when thing go far"
#define BLYNK_AUTH_TOKEN "ImjkyUW16D86kbfMfjK6QhcbWDWwQTE6"

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "Biibeoo";  // type your wifi name
```

```
char pass[] = "quanghuy";  // type your wifi password

#define PIR_SENSOR D1
BlynkTimer timer;
//int flag=0;
void notifyOnPer()
{
  int percentGas = digitalRead(D7);
  if (percentGas==0) {
    Blynk.logEvent("warn_gas","Gas detected");
  }
}

void notifyOnHu()
{
  int humidity = digitalRead(D1);
  if (humidity==0) {
    Blynk.logEvent("warn_humid","Humidity higher than normal");
  }
}

void notifyOnTemp()
{
  int temperature = digitalRead(D6);
  if (temperature==0) {
    Blynk.logEvent("warn_temper","temperature higher than normal");
  }
}

void setup(){
  pinMode(D1, INPUT_PULLUP);
  pinMode(D2, INPUT_PULLUP);
  pinMode(D3, INPUT_PULLUP);
```

```
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("-");
  }
  //dht.begin();
  timer.setInterval(5000L , notifyOnPer);
  timer.setInterval(5000L , notifyOnHu);
  timer.setInterval(5000L , notifyOnTemp);
}

void loop(){
  Serial.println(digitalRead(PIR_SENSOR));
  Blynk.run();
  timer.run();
}
```

RESULT



CONCLUSION

The project aims to develop a system for measuring temperature, humidity, and gas levels in the air, with the added functionality of sending notifications to users if the gas levels exceed permissible limits. Although currently in its basic stage of development, this project lays the foundation for an environmental quality monitoring system.

By focusing on these key environmental parameters, the project addresses critical aspects of air quality monitoring, paving the way for potential expansion into a comprehensive environmental monitoring network. Despite its simplicity in its current form, the project holds significant potential for further development and enhancement.

As society becomes increasingly conscious of environmental issues, systems like these play a crucial role in promoting awareness and enabling proactive measures to safeguard environmental health. Therefore, this project not only serves as a practical tool for monitoring air quality but also contributes to broader efforts aimed at preserving and improving environmental conditions.