GOVERNMENT OF TAMILNADU

DIRECTORATE OF TECHNICAL EDUCATION, CHENNAI

NAAN MUDHALVAN SCHEME (TNSDC) SPONSORED

STUDENTS DEVELOPMENT PROGRAMME

ON

IoT AND ITS APPLICATIONS

HOST INSTITUTION

xxxxx

COIMBATORE – 04

TRAINING PARTNER

ENTHU TECHNOLOGY SOLUTIONS INDIA PVT LTD

DATE:

Name	Roll No
Name 1	1
Name 2	2
Name 3	3
Name 4	4
Name 5	5

Table Of Contents

Abstract

In the evolving field of healthcare, continuous monitoring of vital signs is crucial for timely intervention and enhancing patient outcomes. This project presents an IoT-Based Health Monitoring System designed to track critical health metrics, including heart rate, blood oxygen levels (SpO2), and body temperature, in real-time. Utilizing the ESP32 microcontroller as the central unit, the system collects and processes data from sensors such as the MAX30105 for heart rate and SpO2, and the LM35 for temperature. Data is displayed locally on an LCD and transmitted via Wi-Fi to the ThingzMate Cloud, facilitating remote access by healthcare providers. The system includes alert mechanisms to notify users of abnormal readings and features secure data transmission to protect sensitive information. This scalable solution integrates traditional health monitoring with IoT capabilities, making it suitable for home healthcare, remote patient monitoring, and emergency applications.

Introduction

Continuous monitoring of vital signs is essential in modern healthcare to ensure patient safety and enable prompt medical responses. Traditional methods often require patients to be physically present at healthcare facilities, which can be challenging for those with chronic conditions or in remote areas. The IoT-Based Health Monitoring System addresses these issues by providing a means for real-time remote monitoring of health metrics. Central to this system is the ESP32 microcontroller, which integrates data from sensors like the MAX30105, measuring heart rate and SpO2, and the LM35, which tracks body temperature. The ESP32 processes and displays this data locally while sending it to the ThingzMate Cloud for remote access. Alerts are generated for any readings that exceed predefined thresholds, ensuring timely intervention. The system also includes security features to safeguard sensitive data. This IoT-enabled approach offers a versatile solution for enhancing patient care in various scenarios, including home healthcare and remote monitoring.

Hardware and Software Requirements

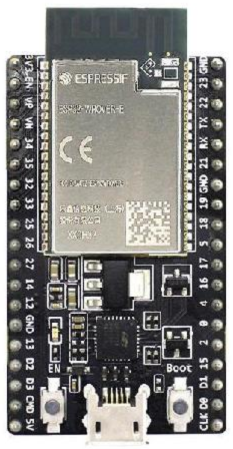
Hardware Requirements

1. ESP32 Microcontroller
2. MQ-2 Gas Sensor
3. DHT11 Sensor
4. LED
5. BreadBoard
6. USB Cable
7. Jumper Wires

Software Requirements

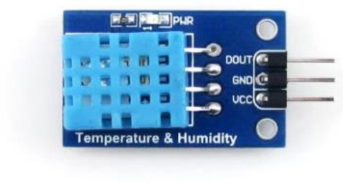
1. Wokwi Simulator
2. Arduino IDE
3. Thingzmate Cloud

ESP32 Microcontroller



The ESP32 microcontroller is a powerful and versatile chip used in this project to control the four-way traffic light system. It features integrated Wi-Fi and Bluetooth capabilities, allowing for seamless communication with the ThingzMate Cloud for remote monitoring and control. With its dual-core processor and multiple GPIO pins, the ESP32 efficiently handles the timing and sequencing of the traffic lights, making it ideal for real-time IoT applications.

DHT11



The DHT11 sensor is employed to measure environmental temperature and humidity. It provides accurate and reliable readings necessary for monitoring environmental conditions. The ESP32 processes the data from the DHT11 and displays it locally or transmits it to ThingzMate Cloud for remote access and analysis.

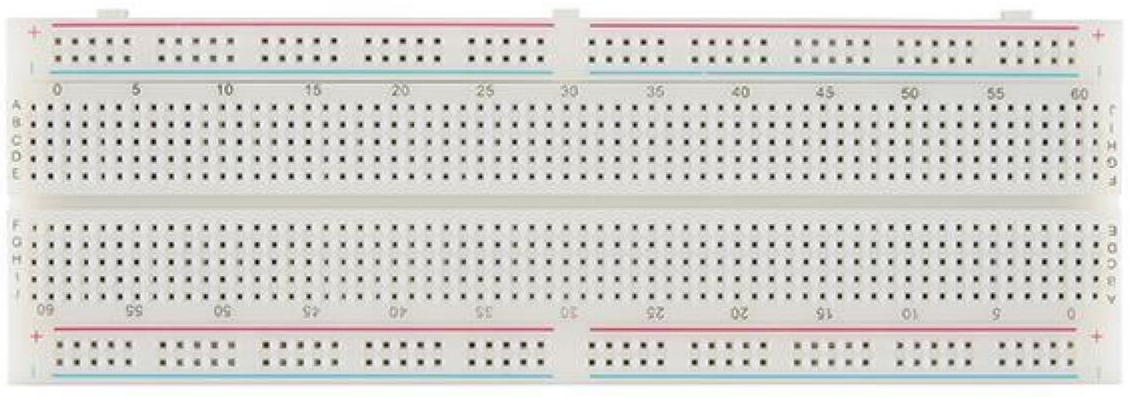
MQ-2 Gas Sensor

The MQ-2 gas sensor detects various gases, including smoke and carbon dioxide (CO2). This sensor plays a crucial role in monitoring air quality by providing real-time data on gas concentrations. The ESP32 interfaces with the MQ-2 sensor to capture and process this data, enabling the system to issue alerts for high gas levels and send the information to the cloud for remote monitoring.

LED

In the Environment Monitoring System, LEDs are used as visual indicators for system status and environmental conditions. For example, a green LED might indicate normal air quality, while a red LED could signal the presence of high gas concentrations or abnormal conditions. These visual cues help users quickly assess the system's status.

BreadBoard



The breadboard is used for prototyping and connecting the ESP32 microcontroller with sensors such as the DHT11 and MQ-2, as well as other components like LEDs. It allows for easy modifications and testing of the circuit design without soldering, making it an essential tool during the development phase.

USB-Cable

The USB cable is a critical tool in this project, used to connect the ESP32 microcontroller to a computer for power supply, programming, and debugging. It enables the transfer of code and data between the development environment and the microcontroller, facilitating the upload of firmware and real-time communication during the development process. The USB connection also allows for serial monitoring, providing valuable insights into the system's performance and behavior.

Jumper Wires

Jumper wires are essential in this project, used to connect the ESP32 microcontroller to the components on the breadboard. These wires provide a flexible and reliable way to link the microcontroller’s GPIO pins to the LEDs, resistors, and other circuit elements, enabling proper signal and power flow. Their ease of use allows for quick modifications and testing during the prototyping stage.

Wokwi Simulator

The Wokwi Simulator is a powerful online tool for virtual prototyping of the Environment Monitoring System. It allows developers to simulate the entire setup, including the ESP32 microcontroller, DHT11 sensor, MQ-2 gas sensor, and other components like LEDs. Using Wokwi, you can visualize real-time data outputs, test various scenarios, and debug the system without physical hardware. This simulation capability helps in validating the design, identifying potential issues, and making necessary adjustments to the circuit and code, thereby saving time and resources during development.

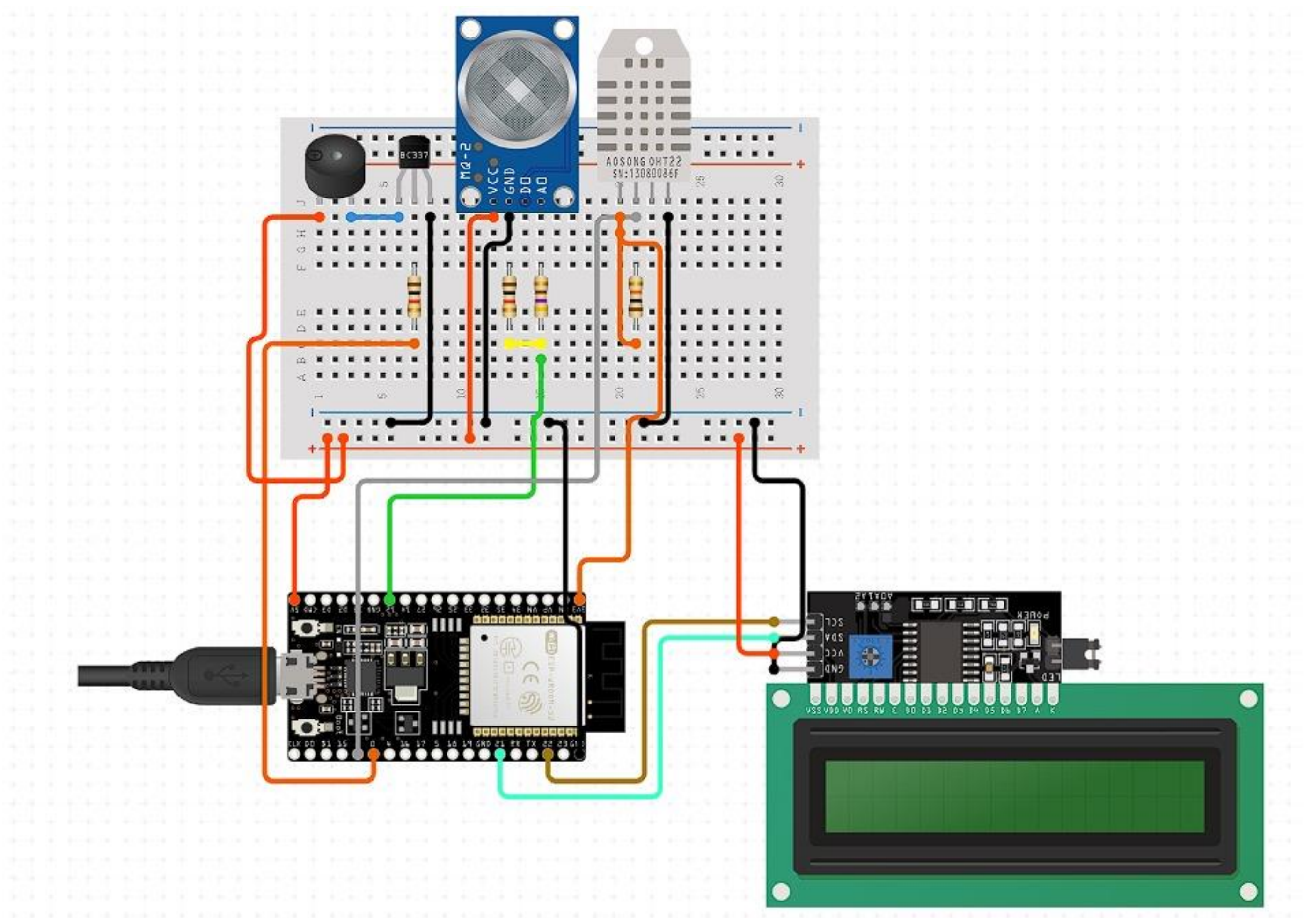
Arduino IDE

The Arduino IDE serves as the main development environment for the Environment Monitoring System. It provides an intuitive interface for writing, compiling, and uploading code to the ESP32 microcontroller. The IDE supports a wide range of libraries and offers seamless integration with the ESP32, streamlining the coding process. It facilitates efficient code iteration, debugging, and deployment, making it essential for managing the sensor interactions, data processing, and communication aspects of the system.

Thingzmate Cloud

ThingzMate Cloud is integral to the Environment Monitoring System, offering real-time data monitoring and management through cloud connectivity. It enables the secure transmission of data from the ESP32 to the cloud, where it can be visualized and analyzed. ThingzMate Cloud provides features for configuring alerts based on sensor readings, storing environmental data securely, and generating insights into air quality trends. This cloud platform ensures that users can remotely monitor environmental conditions, respond promptly to alerts, and maintain effective oversight of the monitoring system.

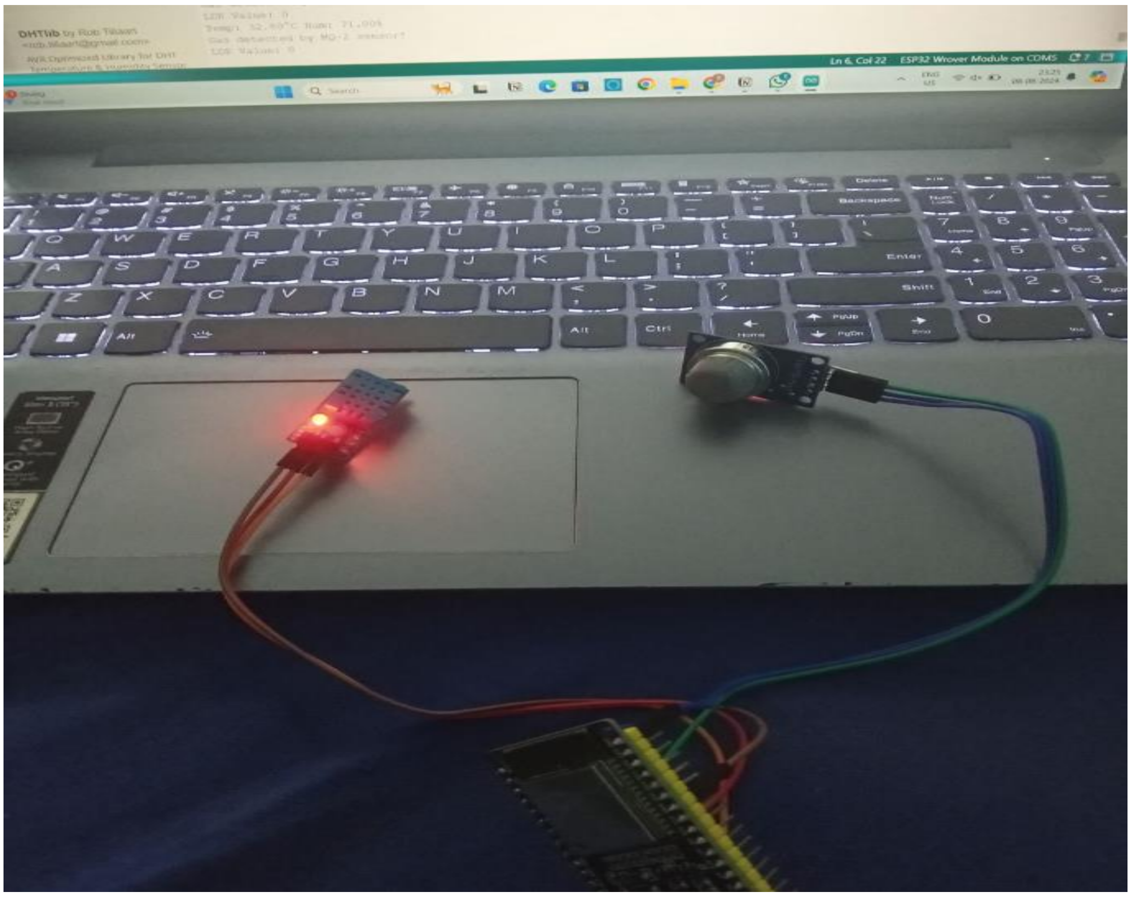
Block Diagram



Code

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>

#define WIFI_SSID "vivo"
#define WIFI_PASSWORD "8983245"

#define DHTPIN 4      // Define the DHT11 data pin
#define DHTTYPE DHT11 // Define the type of DHT sensor (DHT11)
#define MQ2PIN 5      // Define the MQ-2 sensor digital output pin

DHT dht(DHTPIN, DHTTYPE);

const char *serverUrl = "https://console.thingzmate.com/api/v1/device-
types/golden123/devices/golden123/uplink"; // Replace with your server endpoint
String AuthorizationToken = "Bearer b3d88ee5e5c361c619e2459e06304251";

void setup() {
  Serial.begin(115200);
  delay(4000); // Delay to let serial settle

  // Connect to WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi");
  // Initialize the DHT sensor
  dht.begin();

  // Initialize the MQ-2 sensor pin as input
  pinMode(MQ2PIN, INPUT);
}

void loop() {
  float Temperature = dht.readTemperature();
  float Humidity = dht.readHumidity();
  int mq2Value = digitalRead(MQ2PIN); // Read the digital output of the MQ-2 sensor
  int value = 100; // Example value, replace with your sensor reading if needed

  // Check if any reads failed and exit early (to try again).
  if (isnan(Temperature) || isnan(Humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  HTTPClient http;
  http.begin(serverUrl);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", AuthorizationToken); // Authorization token

  // Create JSON payload
  String payload = "{\"temperature\":" + String(Temperature) + ",\"humidity\":" + String(Humidity)
+ ",\"mq2Value\":" + String(mq2Value) + ",\"Raw_Value\":" + String(value) +  "}";

  // Send POST request
  int httpResponseCode = http.POST(payload);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("HTTP Response code: " + String(httpResponseCode));
    Serial.println(response);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }

  http.end(); // Free resources
  delay(10000); // Wait for 10 seconds before sending the next request

}
```

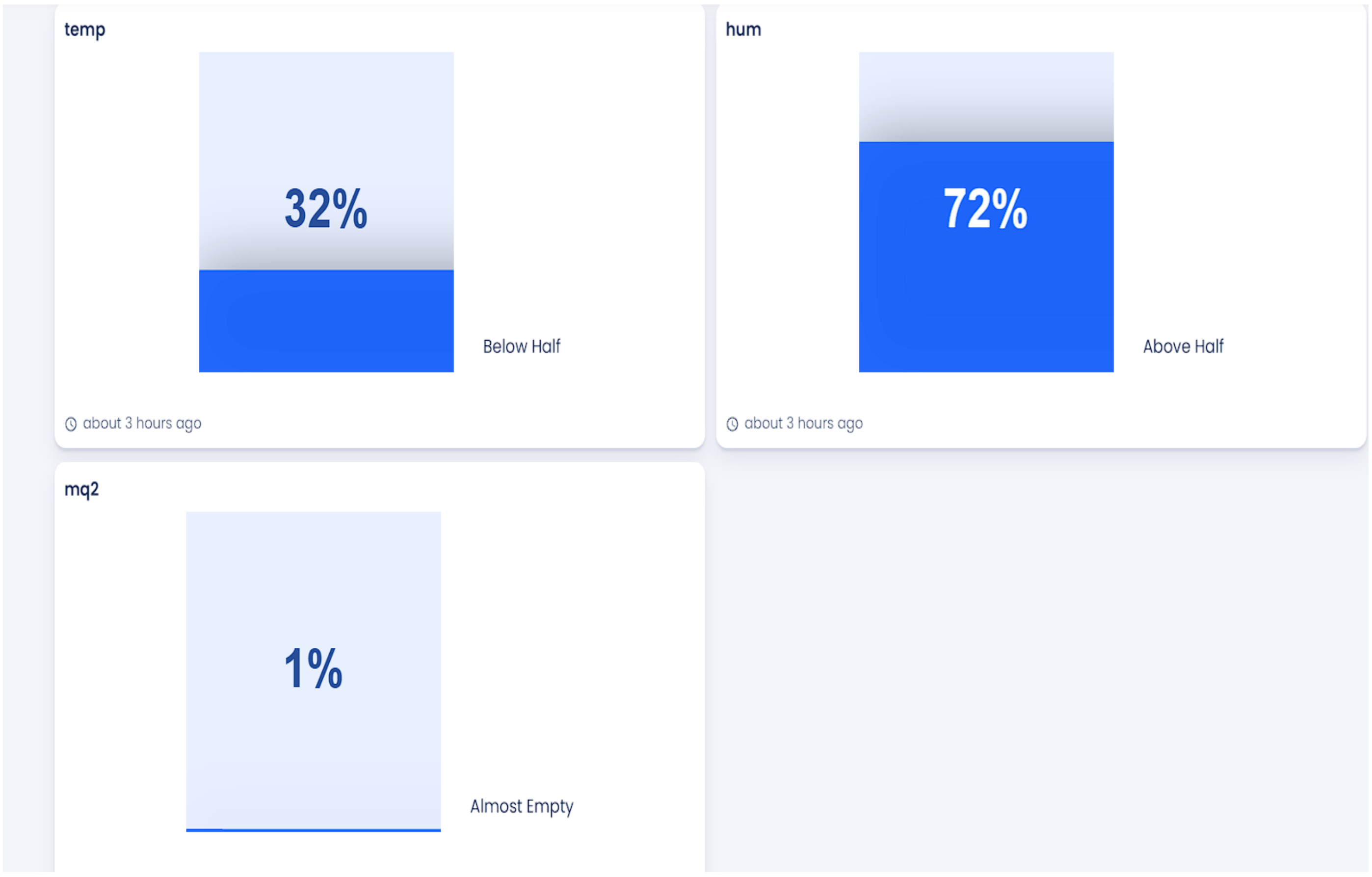
Output Results



```
LDR Value: 0
Temp: 32.40°C Humi: 71.00%
Gas detected by MQ-2 sensor!
LDR Value: 0
```

DHTlib by Rob Tillaart
<rob.tillaart@gmail.com>
AVR Optimized Library for DHT Temperature & Humidity Sensor
Ln 6, Col 22 ESP32 Wrover Module on COM5
Search

Cloud Output



Conclusion

The Environment Monitoring System demonstrates the effective integration of IoT technology for real-time environmental tracking and management. By leveraging the ESP32 microcontroller, DHT11 sensor, MQ-2 gas sensor, and supporting tools like Wokwi Simulator, Arduino IDE, and ThingzMate Cloud, the system provides comprehensive monitoring of key environmental parameters such as temperature, humidity, and air quality.

The use of Wokwi Simulator facilitated virtual testing and debugging, ensuring that the system's functionality and performance were validated before physical implementation. The Arduino IDE streamlined the development process, enabling efficient coding and deployment. ThingzMate Cloud enhanced the system by offering real-time data visualization, alert configuration, and secure data storage, making remote monitoring and management straightforward.

This project not only highlights the potential of combining advanced sensor technology with cloud-based platforms but also underscores the benefits of integrating IoT solutions for effective environmental monitoring. The system's capability to provide timely alerts and insights into environmental conditions can significantly impact various applications, from home automation to industrial monitoring, ultimately contributing to better management and response to environmental changes.