WeatherScope: A Comprehensive Weather Monitoring & Analysis System.

Team 17

Team Member 1: Charitha Namireddy

Team Member 2: Kavya Sree Gogadi

Team Member 3: Praval Goud Madhagouni

1. INTRODUCTION TO RTOS and IoT

RTOS stands for Real-Time Operating System, which is a type of operating system designed to run applications that require a high degree of predictability and reliability. RTOS is designed to provide deterministic behaviour, which means that the timing of events is guaranteed. This is especially important for embedded systems, which are devices that have limited computing resources and are often used in real-time applications such as automotive systems, medical devices, and industrial control systems.

IoT stands for Internet of Things, which refers to the network of physical devices, vehicles, home appliances, and other items that are embedded with electronics, software, sensors, and network connectivity. IoT devices are designed to collect and exchange data over the internet, enabling the automation of various tasks and the optimization of processes.

RTOS is an important component of IoT systems because it enables the real-time processing and control of data from IoT devices. By providing a high degree of predictability and reliability, RTOS ensures that IoT devices can respond quickly and accurately to changing conditions and events.

In summary, RTOS and IoT are both essential technologies for the development of real-time and connected systems. RTOS provides the deterministic behaviour required for real-time applications, while IoT enables the collection and exchange of data between devices and the cloud, enabling the optimization of various processes.

Application areas of RTOS:

There are many applications of RTOS and IoT. As technologies continue to develop, we can expect to see them used in a growing number of industries and applications. Here are a few examples:

1. **Industrial Automation:** RTOS and IoT are used extensively in industrial automation to control processes and monitor performance. For example, RTOS can be used to control the timing of robotic arms in a factory, while IoT can be used to collect data on energy consumption and performance.

2. **Smart Home Systems:** RTOS and IoT are also used in smart home systems to control and monitor home appliances and security systems. For example, RTOS can be used to control the timing of lighting and heating systems, while IoT can be used to monitor security cameras and door locks.

3. **Healthcare Monitoring:** RTOS and IoT can be used in healthcare monitoring systems to monitor patient health and track medical equipment. For example, RTOS can be used to control the timing of medical pumps, while IoT can be used to collect data on patient vital signs.

4. **Autonomous Vehicles:** RTOS and IoT are critical components of autonomous vehicles. RTOS is used to control the timing of vehicle operations, while IoT is used to collect data on road conditions and traffic.

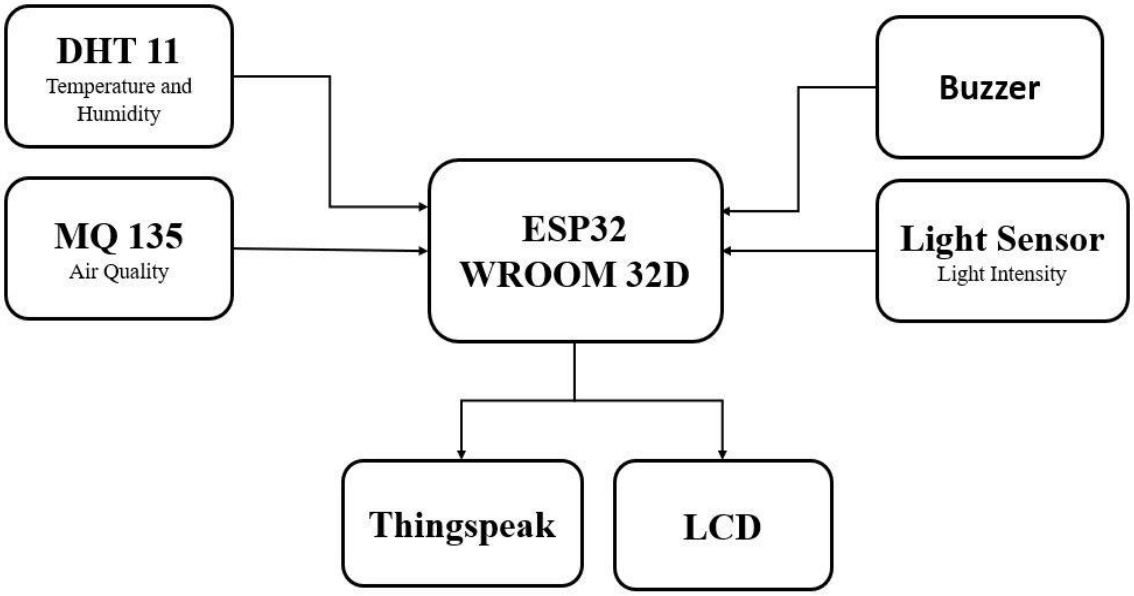
5. **Agriculture:** RTOS and IoT are also used in agriculture to control irrigation systems and monitor crop health. For example, RTOS can be used to control the timing of irrigation systems, while IoT can be used to collect data on soil moisture levels and plant growth.

2. INTRODUCTION TO WEATHERSCOPE

The Weather Scope's objective is to constantly display several real-time weather parameters, including temperature, humidity, air quality and light intensity. This system uses a variety of sensors to identify and track weather characteristics. The information it gathers is then shown on an LCD and transferred to the cloud, where it may be viewed via a website using the internet. To monitor local air conditions and microclimates for weather forecasting and prediction from anywhere in the world. Also, the data can be kept for both short and long periods of time to examine changes in weather patterns and determine how human-induced climate change has impacted your local weather. In addition, we can use graphical depictions to show parameter trends. We have weather forecasting stations in many places in the modern world, but having our own weather monitoring system has benefits, particularly for people who live in rural areas. A DHT11 temperature and humidity sensor, a light diode sensor, an LCD, MQ 135 and an ESP32 WROOM 32D microcontroller with an integrated Wi-Fi module make up this system, which transmits the data it collects to a web server. We are implementing this project with the help of FreeRTOS. FreeRTOS is intended to be small and easy to use. To make it easier to port and maintain, it is primarily written in the C programming language. It also includes a few assembly language functions that are used in architecture-specific scheduler routines when they are required.

3. HARDWARE IMPLEMENTATION OF THE PROJECT

Block diagram:



Block diagram of WeatherScope

Components Required:

- ESP32 WROOM
- DHT 11
- MQ135
- Light Sensor BPW34
- Buzzer
- LCD
- Resistors
- Jumper wires and Breadboard

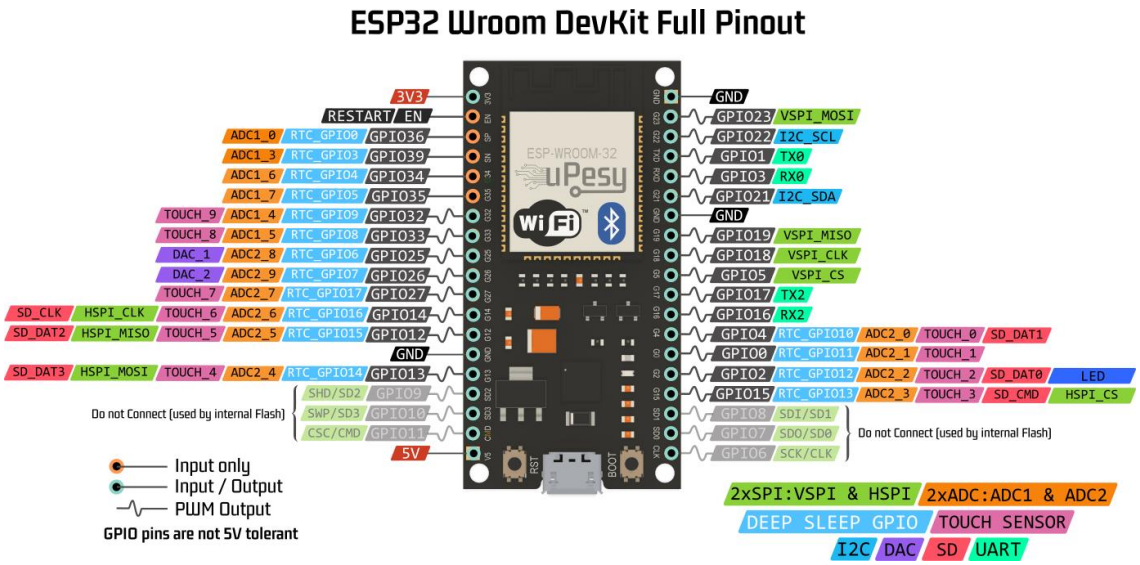
3.2.1 ESP32 WROOM:

Description:

ESP32 WROOM-32D is a highly integrated module that is based on the ESP32-D0WDQ6 chip. It is designed for use in a wide range of IoT applications that require both wireless connectivity and low power consumption. The module supports both Wi-Fi and Bluetooth connectivity, making it suitable for a wide range of applications. It also features an integrated 4MB SPI flash memory, which can be used for storing firmware and other data. The module can be programmed using the ESP-IDF

development framework, which provides a range of tools and libraries for developing applications.

Features of ESP32 WROOM-32D include a Dual-core processor, Wi-Fi and Bluetooth connectivity, Low power consumption, and Integrated SPI flash memory.



3.2.2 DHT 11:

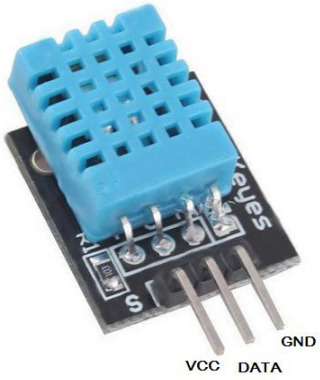
Description:

DHT11 is a low-cost temperature and humidity sensor module that is commonly used in a wide range of applications, including weather stations and environmental monitoring. The module is based on a thermistor and a humidity sensor, and it provides accurate readings of temperature and humidity.

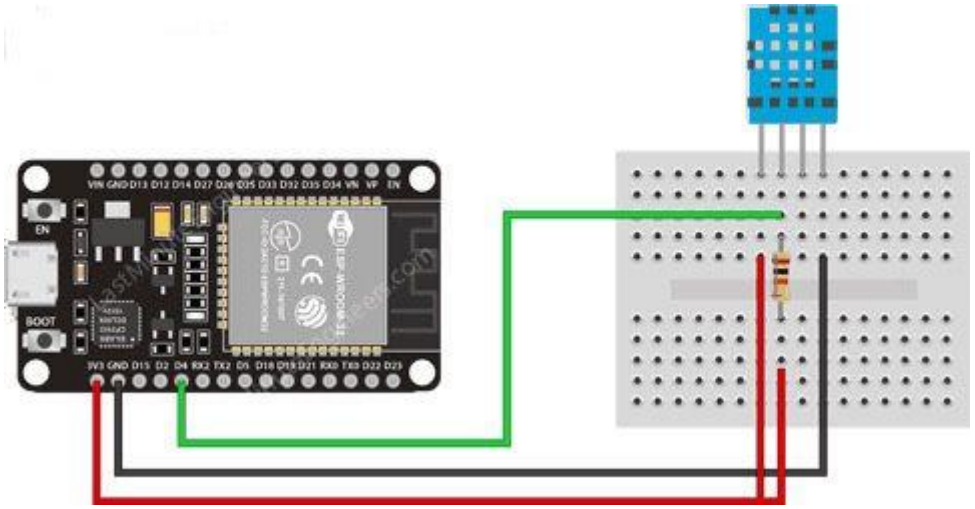
The DHT11 module has a compact design and is easy to use. It features a single-wire digital interface, which makes it easy to connect to microcontrollers and other devices. The module also has a built-in 10k ohm pull-up resistor, which simplifies the wiring process.

The DHT11 module has the following specifications:

1. Temperature range: 0°C to 50°C
2. Temperature accuracy: ±2°C
3. Humidity range: 20% to 90% RH
4. Humidity accuracy: ±5% RH
5. Operating voltage: 3.3V to 5V DC
6. Current consumption: 1.5mA



DHT11 Sensor Module



Integration of DHT11 and ESP32

DHT11 Pinout:

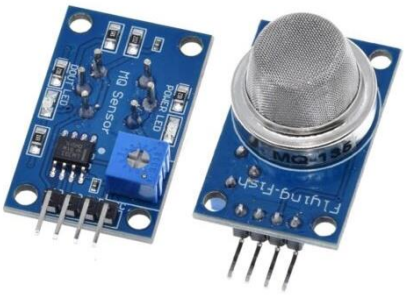
DHT11	ESP32
Vcc	3.3V
GND	GND
Data Pin (S)	D4

3.2.3 MQ135

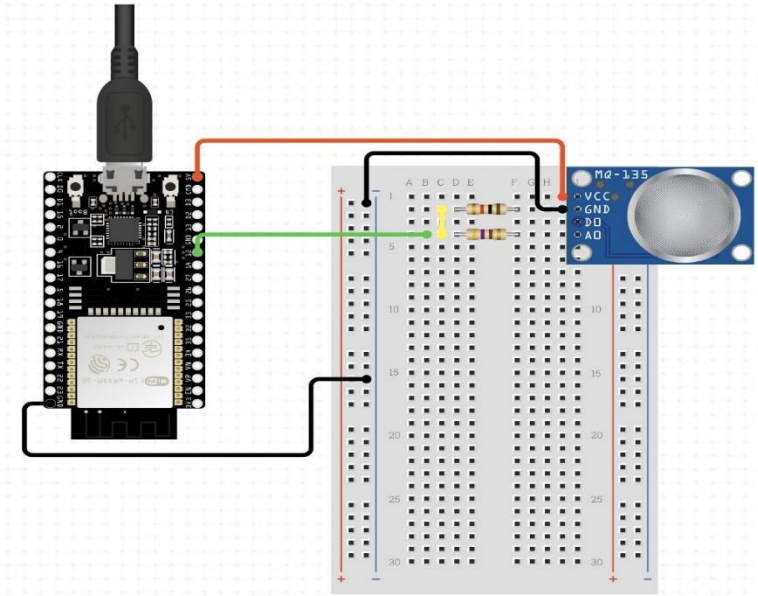
Description:

MQ-135 is a gas sensor that can detect carbon dioxide, alcohol, smoke, and nitrogen dioxide. It is based on the principle of gas absorption, in which the target gas is absorbed onto the surface of a sensing material and changes the resistance of the sensor. The change in resistance is then measured and converted into a gas concentration value.

The MQ-135 sensor is widely used in air quality monitoring applications such as indoor air quality monitoring, automotive emissions monitoring, and industrial gas leakage detection. It is a low-cost and easy-to-use sensor.



MQ135 Sensor



Integration of MQ135 and ESP32

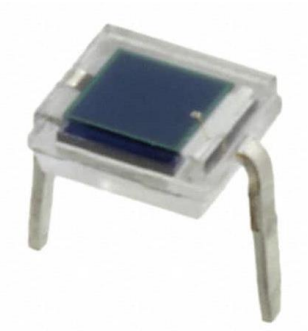
MQ135 Pinout:

MQ135	ESP32
Vcc	3.3V
GND	GND
A0	34

3.2.4 Light Sensor (BPW34)

Description:

BPW34 is a silicon PIN photodiode that is commonly used as a light sensor in various applications. The BPW34 photodiode is known for its high sensitivity and fast response time, making it suitable for use in light detection systems, optical communications, and other applications where accurate and fast light detection is required.



BPW Sensor

BPW34 Pinout:

BPW34	**ESP32**
Anode	35
Cathode	GND

3.2.5 Buzzer:

Description:

A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard.

Buzzer

Buzzer Pinout:

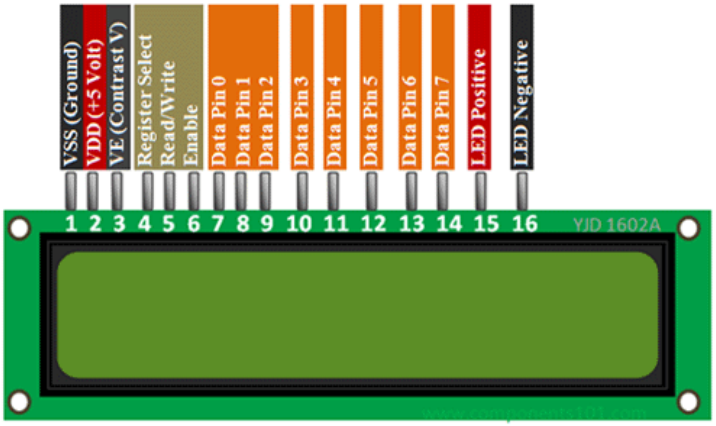
Buzzer	**ESP32**
+	12
GND	GND

3.2.6 LCD:

Description:

A 16x2 LCD display is a type of LCD module that is commonly used in many electronic devices, such as digital clocks, calculators, and measurement instruments. It consists of

16 columns and 2 rows of characters, allowing it to display up to 32 alphanumeric characters at once.



16x2 LCD

I2C:

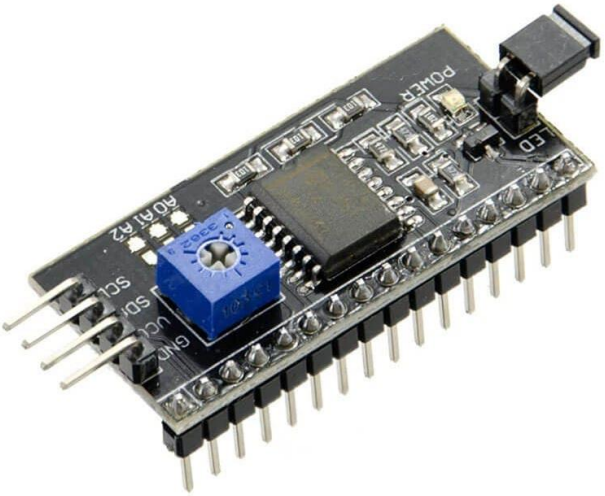
Description:

The I2C (Inter-Integrated Circuit) module is a popular serial communication protocol used to interface microcontrollers with various peripherals such as sensors, EEPROMs, and LCD displays. It is a two-wire protocol that uses a master/slave architecture, where the master device initiates the communication, and the slave device responds to the master's requests.

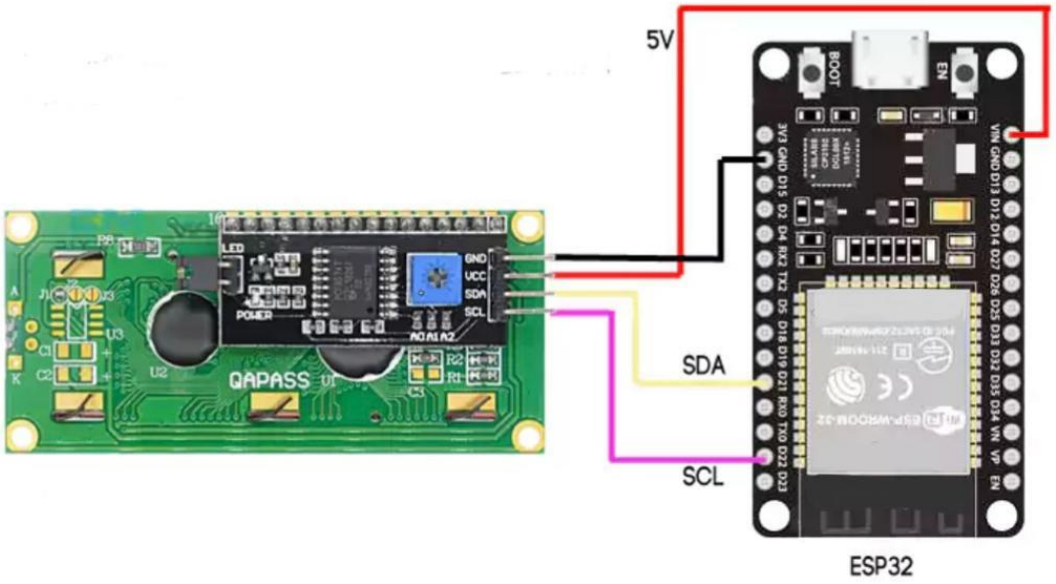
The I2C module consists of two lines: SDA (Serial Data Line) and SCL (Serial Clock Line). The SDA line is used for bidirectional data transfer between the master and slave devices, while the SCL line provides a clock signal that synchronizes the data transfer.

To communicate with an I2C peripheral, the microcontroller must first send a start signal to initiate the communication. This is followed by the slave device's address, which is typically 7 bits long. If the slave device acknowledges its address, the master can then send or receive data from the slave device.

In summary, the I2C module is a versatile and widely used serial communication protocol that enables microcontrollers to interface with various peripherals in an efficient and reliable manner.



I2C Module



Integration of Lcd with I2C on ESP32

LCD Pinout:

LCD	ESP32
Vcc	5V
GND	GND
SDA	21
SCL	22

3.2.7 ThingSpeak:

Description:

ThingSpeak is an open-source IoT platform that allows users to collect, analyze, and visualize data from connected devices. It is a cloud-based service that provides an easy-to-use interface for users to develop and deploy IoT applications and services.

With ThingSpeak, users can easily connect their devices to the internet and collect data in real time. The platform offers a variety of features, including data storage, data visualization, and data analysis tools. Users can also create custom dashboards to monitor and control their devices and set up alerts and notifications based on specific data conditions.

Here are the basic steps to get started with ThingSpeak:

- Sign up for a ThingSpeak account.
- Create a new channel.
- Add fields to your channel.
- Configure your device.
- Send data to your channel.
- Visualize your data.
- Analyze your data.

3.2.8 Arduino IDE:

Description:

The Arduino IDE is a software application that is used to program and upload code to Arduino boards. It is a cross-platform application that can be used on Windows, Mac OS X, and Linux operating systems.

The Arduino IDE provides an easy-to-use interface for writing and uploading code to Arduino boards. It includes a code editor with syntax highlighting and auto-complete features, as well as a built-in serial monitor for debugging and communication with the Arduino board.

The Arduino IDE supports a variety of programming languages, including C and C++. It also includes a library of pre-written code examples that can be used as a starting point for your own projects.

Arduino IDE

3.2.9 FreeRTOS:

Description:

FreeRTOS is a real-time operating system (RTOS) that is designed to be small, efficient, and easy to use. It is a popular choice for embedded systems, especially those with limited resources.

FreeRTOS provides a multitasking kernel, which allows multiple tasks to run concurrently on a single processor. Each task has its own stack and execution context, and the kernel provides the necessary synchronization mechanisms to ensure that tasks can communicate and coordinate with each other.

FreeRTOS also includes several other features, such as support for semaphores, mutexes, and message queues, as well as a timer and interrupt management. It is available under a modified GPL license, which allows it to be used in both open-source and commercial projects.

FreeRTOS supports many threads or processes, mutexes, semaphores, and software timers. An option for tickles mode is offered for low-power applications. Thread priorities are supported. While objects in FreeRTOS can be dynamically allocated using one of five memory management (allocation) strategies, applications can only be statically allocated.

- Allocate only.
- Allocate and free with a very simple, fast algorithm.
- Allocate and free with a more complicated but fast algorithm with memory coalescence.

- An alternative to the more complicated scheme that includes memory coalescence allows a heap to be broken across multiple memory areas.
- Allocate and free with some mutual exclusion protection in the C library.

Overall, FreeRTOS is a robust and reliable RTOS that is well-suited for embedded systems that require real-time performance and efficient use of resources.

4. PSEUDOCODE AND FUNCTIONAL DESCRIPTION

Pseudocode:

Include Libraries

Declare Variables

Define WiFi ID and Password

Define Pins for sensors

```
 Void connecttoWiFi(){

//With your credentials, connect to wifi
            If()
              {
              connected = WiFi connected..!
              }
        }
 Void setup(){

 //Initialize Humidity sensor
 //Initialize DHT
sensor
 //Initialize LCD
 //Initialize Wi-Fi connection
 //Initialize thingspeak
 Create tasks to read the values of temperature and humidity from DHT11 --- /*
 ReadTempHumid */
 Create tasks to read the values of light from BPW34 --- /* ReadLight */
 Create tasks to read the values of Air Quality from MQ135--- /* Air_quality */
 Create tasks to display above value on LCD --- /* LCD */
 Create tasks to display above values in Thingspeak --- /* WIFI_Display */
}

Void ReadTempHumid(){
       For( ; ; ){

        //Write a function to read values from sensor
        Delay
        //Print sensor values in serial monitor
             //Taskdelay
              }
       }
```

```
 Void ReadLight(){

      For( ; ; ){
      //Write a function to read values from sensor


      //Print sensor values in serial monitor
      //Taskdelay
      }

}

 Void Air_quality(){

      For( ; ; ){

      //Write a function to read values from sensor
            If (air_quality >= 2000){
                  //Buzzer ON
                  Delay
              }
            If (air_quality < 2000){
                  //Buzzer OFF
                   Delay
             }
//Print sensor values in serial monitor
//Taskdelay
}

}
 Void LCD(){

      For( ; ; )

            {

               // Write functions to display light sensor values on LCD.
                 Delay
                //Write functions to display air quality sensor values on LCD.
                Delay
                //Write functions to display temperature sensor values on LCD.
```

```
            Delay
            //Write functions to display humidity sensor values on LCD.
            Delay
        }

       Taskdelay
    }

Void WIFI_Display  (){

    For( ; ; ){

            // Write a function to display values in thingspeak
             If(x==200)
                {
                  //   Print “Channel update successful”
                 }
             Taskdelay
        }

    }
  Void loop(){

     Taskdelay;

  }
```

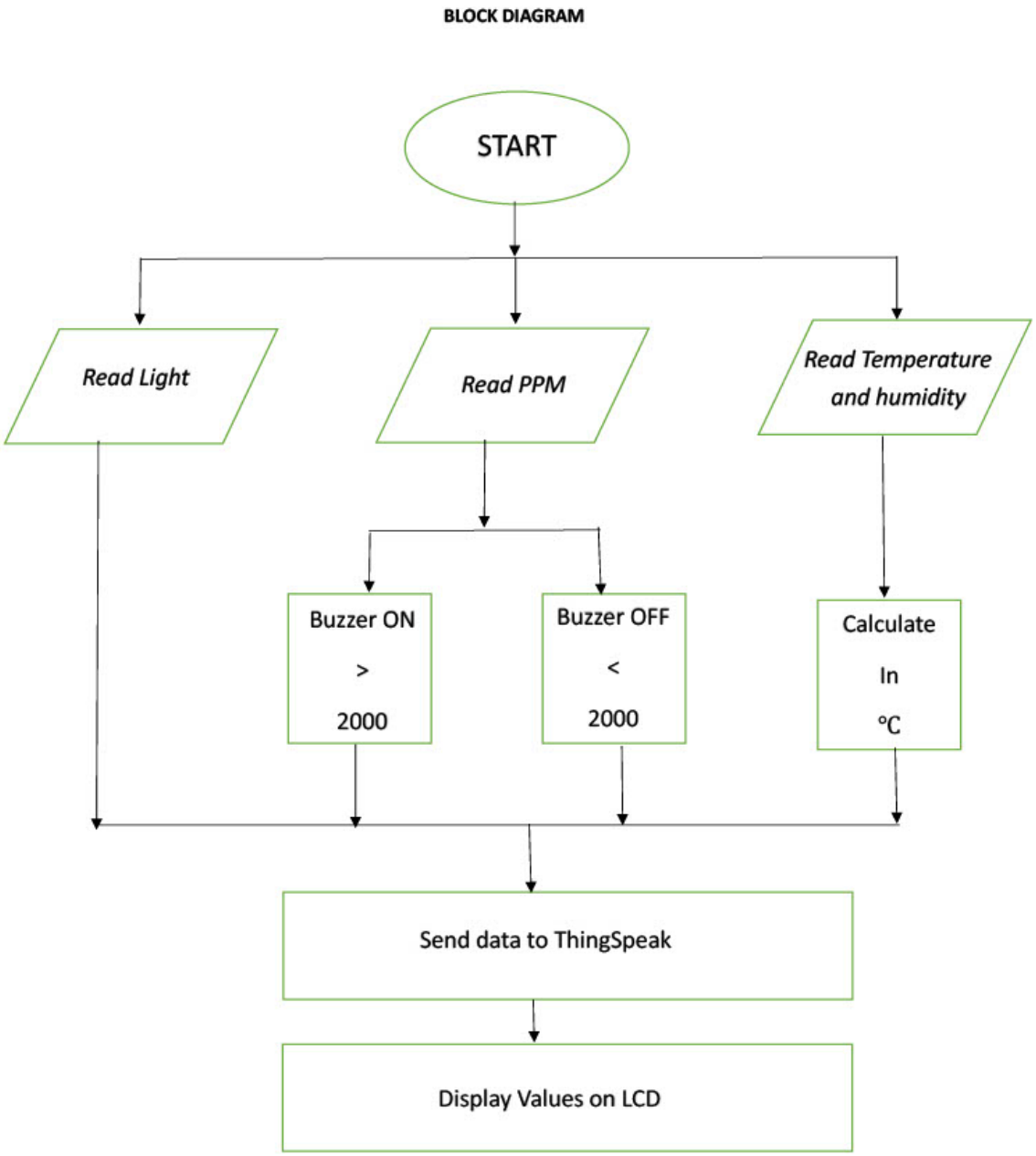
Functional Description:

In our project, we are using ESP-WROOM-32D, which is an MCU module that targets a wide variety of applications ranging from low-power sensor networks to the most demanding tasks. The sensors that we used in our project are DHT11, MQ135, Buzzer, and BPW 34.

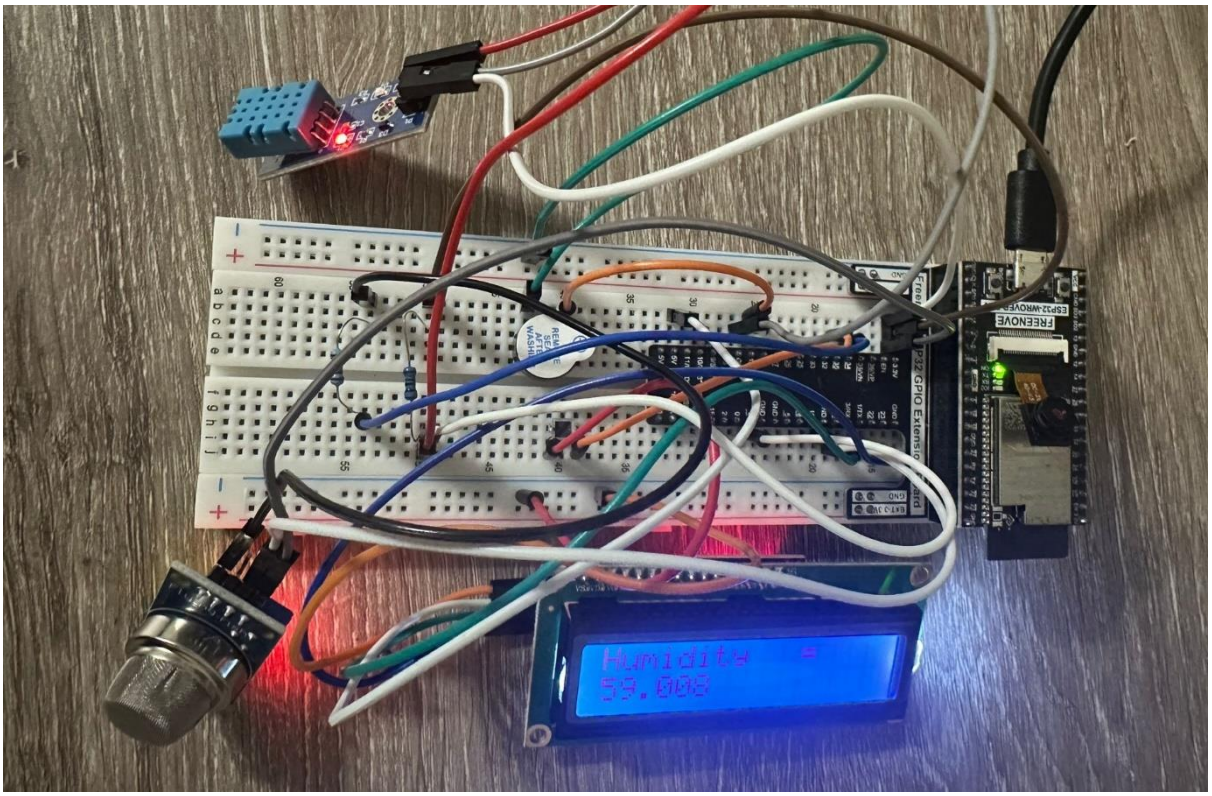
- DHT11 is a temperature and humidity sensor. In the DHT11 sensor Vcc pin is connected to the 3.3V pin of ESP32, GND is connected to GND of ESP32 data pin of DHT11 is connected to GPIO 4.
- MQ135 is a gas sensor, which we are using to check the air quality. AO is the analog output pin that we are connecting to pin 34. We added a 1K resistor between the AO pin of MQ135 and pin34 of ESP32. Vcc (3.3v) and GND are connected to their respective pins.
- The buzzer in our project turns on when the air quality goes above 2000PPM. It has two terminals of which the positive terminal is connected to pin 12 of ESP32 and the negative terminal is connected to GND.
- BPW34 is the light sensor that has two pins namely the anode which is connected to pin 35 of ESP32 and the cathode which is connected to GND.

- LCD is the module through which we are going to display our output. SDA and SCL pins of LCD are connected to GPIO pins 21 and 22 of ESP32. We have to find I2C address of our LCD in order to initialize it.

Flow Chart:



The following is the final Integration of all the sesnors to ESP32



Discussion:

Based on the requirements of our project, we implemented several tasks using FreeRTOS and scheduling using the Arduino IDE. The sensor readings were sent to the thingspeak cloud for continuous weather monitoring. The DHT-11 sensor reading and photodiode reading were given the same priority. The tasks with the same priority were given an equal amount of processor time using the round-robin scheduling algorithm. The second priority is given to the MQ135 sensor. This ensures that the weather monitoring system will always receive data and that the data displayed showed the latest updates on the LCD and on the website.

The following are the outputs displayed on ESP32.

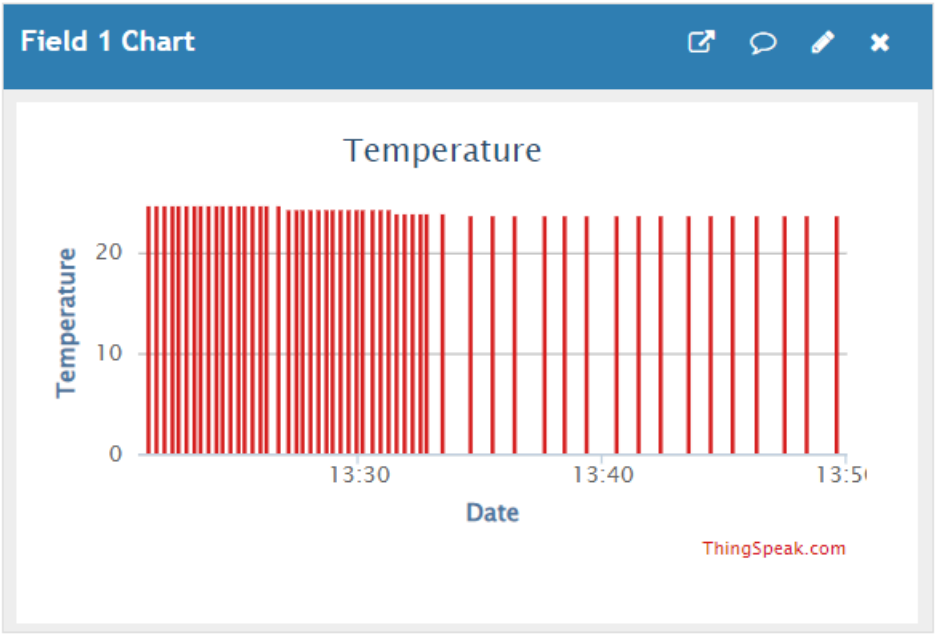
Display of temperature sensor in ℃

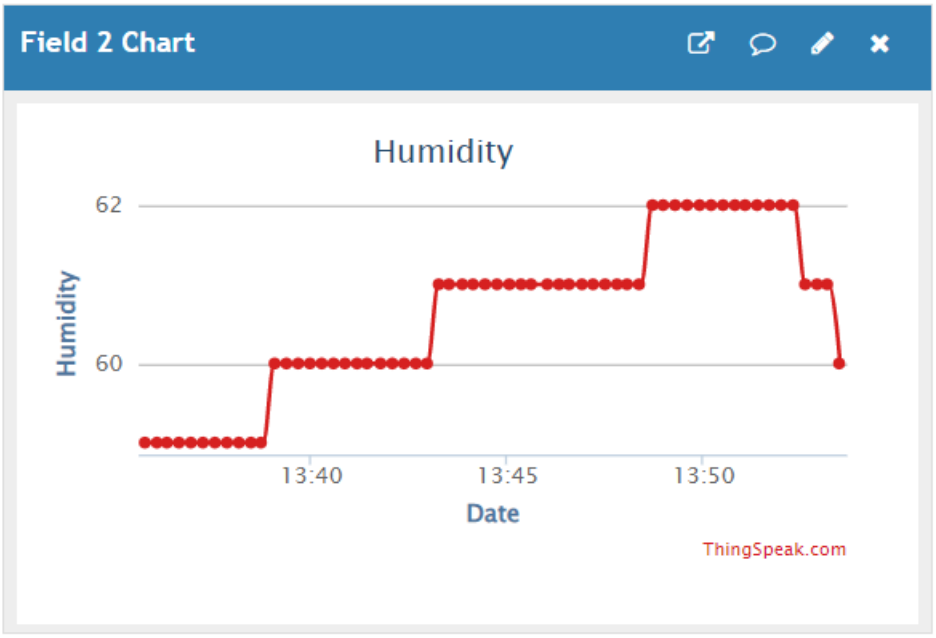
Display of Humidity in percentage

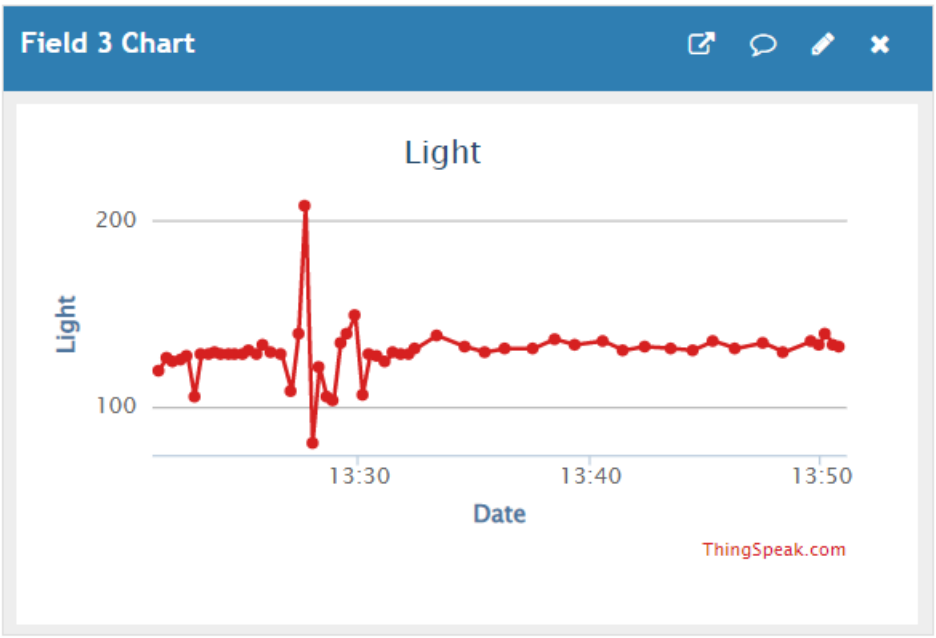
Display of MQ135 sensor data in “Parts Per Million (PPM)”

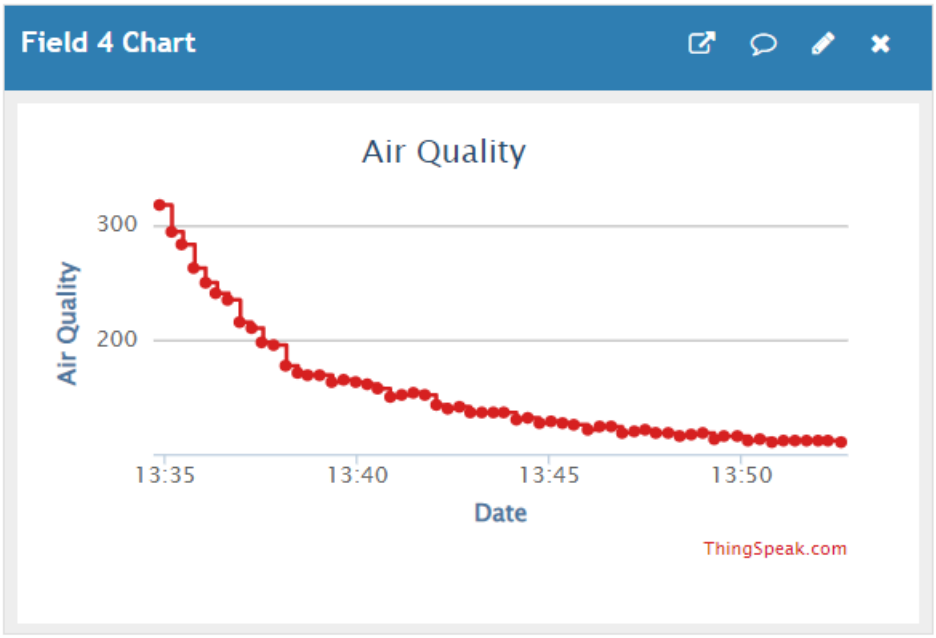
Display of photodiode sensor data in “Lux”

The following are the outputs displayed on Thingspeak






Field 3 Chart
Light
200
100
Light
13:30
13:40
13:50
Date
ThingSpeak.com


Field 4 Chart
Air Quality
300
200
Air Quality
13:35
13:40
13:45
13:50
Date
ThingSpeak.com

Conclusion:

In brief, this project collects weather and ecological sensor information, which provides accurate and reliable data on conditions such as temperature, humidity, light intensity, and air quality and transfers it to thingspeak through Wi-Fi and scheduled tasks using FreeRTOS for real-time weather monitoring. This data can be used to provide up-to-date information for forecasting, planning and emergency response. We can also store this data for later analysis and can use it to determine weather patterns. We can also automate this data to systems such as irrigation systems and building automation, to optimize resource usage and reduce waste. This system is affordable and can easily be integrated, making it a cost-effective solution for monitoring and analyzing environmental data.