IOT BASED LIVE WEATHER MONITORING STATION

A PROJECT REPORT

submitted By

SILPA S PILLAI

TVE20MCA-2050

to

the APJ Abdul Kalam Technological University
in partial fullfilment of the requirements for the award of the degree

of

Master of Computer Applications

Department of Computer Applications

College of Engineering

Trivandrum-695016

JULY 2022

Declaration

I undersigned hereby declare that the project report titled **"IOT BASED LIVE WEATHER MONITORING STATION "** submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Smt.Sreerekha V K, Assistant Professor. This submission represents my ideas in my words and where ideas or words of others have been included. I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity as directed in the ethics policy of the college and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and/or University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title.

Place : Trivandrum **SILPA S PILLAI**

Date : 12/07/2022

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING
TRIVANDRUM

CERTIFICATE

This is to certify that the report entitled **IOT BASED LIVE WEATHER MONITORING STATION** submitted by **SILPA S PILLAI(TVE20MCA-2050)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a bonafide record of the project work carried out by her under my guidance and supervision. This report in any form has not been submitted to any University or Institute for any purpose.

Internal Supervisor External Supervisor

Head of the Dept

Acknowledgement

First and for most I thank **GOD** almighty and to my parents for the success of this project. I owe a sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely thankful to **Dr. Suresh Babu V**, Principal, College of Engineering Trivandrum for providing me with the best facilities and atmosphere which was necessary for the successful completion of this project.

I am extremely grateful to **Prof. Deepa S S**, HOD, Dept of Computer Applications, for providing me with best facilities and atmosphere for the creative work guidance and encouragement.

I express our sincere thanks to **Prof. Sreerekha V K**, Department of Computer Applications, College of Engineering Trivandrum for her valuable guidance, support and advice that aided in the successful completion of my project.

I profusely thank other Asst. Professors in the department and all other staffs of CET, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project. No words can express my humble gratitude to my beloved parents and relatives who have been guiding me in all walks of my journey.

SILPA S PILLAI

Abstract

Rapid advances in a variety of fields are having a significant impact on human existence in different ways, but we still need to adapt technology in order to make it easier to evaluate human life. This essay suggests a model for an Internet of Things-based live weather monitoring station. A weather station is a tool or apparatus that gives us information about the weather in our immediate surroundings. It is a tool that uses a variety of sensors to gather information about the meteorological environment. There are two different kinds of weather stations: one that has its own sensors, and the other that gets its information from the servers of meteorological observation posts. Here, we'll create a custom lookout.In essence, this instrument measures the temperature, pressure, humidity, intensity level, and amount of rain. The prototype contains a variety of sensors that can be used to measure all the aforementioned aspects. We can monitor the gas pressure of the space in addition to the temperature and humidity of a particular room or location. We have the ability to keep track of the rain value.

The ESP32 micro controller, a low-cost, low-power system on a chip (SoC) series with Wi-Fi dual-mode Bluetooth capabilities, is one of the components utilised. A simple, inexpensive digital temperature and humidity sensor may be the DHT11. The BMP180 is Bosch Sensortec's brand-new, very effective digital atmospheric pressure sensor. In order to detect water outside the range of a humidity sensor, rain sensors are used. Telegram notifies the appliance's owner to take the necessary action whenever one of these values, such as the rain level, exceeds a chosen threshold limit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent years have seen a lot of interest in environmental monitoring and climatic change. Man wants to be aware of the most recent weather data for any location, such as a campus for a school or any specific structure. There should be weather stations because the world is changing so quickly. For any area, the observation post we present in this study is quite helpful.IoT is required for this weather observation station (internet of things). It has environmental sensors that can measure the environment at any location and upload the results in real time to the cloud. We used the Arduino ESP32 for this as well as a variety of environmental sensors, including the DHT11 sensor, BMP180, and rain drop module.

The sensors frequently collect data and send it through a wi-fi connection to the web server. The collected parameters are transferred to the cloud, which subsequently offers the weather information in real time. This study focuses on the application of IoT in the next generation of environmental information and provides a replacement paradigm for environmental monitoring in the future. The technology was created expressly to provide weather updates for any particular place, such as an office or room, with the aim of developing a smart city.

Chapter 2

Problem Defnition and Motivation

The project puts out the concept of an IoT-based weather station. A weather station is an apparatus or instrument that informs us of the weather conditions in our immediate surroundings. It is a tool that collects data on the environment and the weather using a range of sensors. In essence, this instrument measures humidity, pressure, temperature, and rain. It can be used to track the temperature or humidity of a specific space, as well as the atmospheric pressure and likelihood of precipitation in that space. Every time the likelihood of rain exceeds a set threshold limit for each, a telegram alert will inform the appliance's owner to take the appropriate precautions.

One of the main problems is live environmental condition because it presents an IoT of challenges when it is measured. There are two different kinds of weather stations: one that has its own sensors, and the other that gets its information from the servers of other weather stations. The latter type describes the existing system. The existing system is rather pricey. In contrast to the current system, our suggested system includes devices that monitor meteorological parameters at the lowest possible cost. The system that was designed utilised fewer sensors than the model that was previously in use.

2.1 Existing System

Currently existing system uses more expensive sensors than proposed system.

2.1.1 Limitation of Existing System

- Existing system is highly expensive, High power consumption and requires external voltage.
- Devices that cost more in the current system to monitor weather conditions.
- Use of ESP8266 rather than ESP32.
- ESP8266 is the predecessor of ESP32.

2.2 Proposed System

Proposed system demonstrates an IoT-based weather monitoring system. Sensors used to extract environmental parameters. The suggested system scales numerous factors, including humidity, temperature, pressure, and rain value, using a variety of sensors. Proposes an extra capability for weather monitoring in the form of an alarm system based on the exceeding of certain sensing parameters, such as the amount of rain. The sensor collected data and transmitted it to the MCU controller, where it was loaded and could be seen on the I2C OLED display and serial monitor. The sensed data is uploaded using the Arduino IDE. displays the information on a web server and keeps track of weather data in real time. The main objective of our proposed model is to achieve economic and cost-effectiveness of the system. Therefore, anyone can utilise it without restriction.The proposed system gathers data from numerous sensors and sends it all to a web page.

2.2.1 Advantages of Proposed System

- The Arduino ESP32 IOT weather mentoring system project is entirely automated.
- Hardware used is economical.
- Quality of hardware is not an issue.
- It doesn't need any assistance from people.
- Weather alerts are available in advance.
- In this system, there is a lower cost and less work.

- High accuracy.
- Self Defense.
- Environmental monitoring done in a clever and effective manner.

Chapter 3

Literature Review

3.1 IoT-Based Real-Time Weather Monitoring System

The 2019 article "Real Time Weather Monitoring System Using IoT" introduces an IoT-based weather monitoring system. In this study, sensors can be used to get the ambient parameter. The LDR sensor is utilised by the author to scale several parameters like humidity, temperature, pressure, and rain value. The system also uses the temperature prototype to determine the dew point value. The value of a certain region, room, or location can be determined using the temperature sensor. The author's usage of the light intensity is made possible by the LDR sensor.When the value of the sensed parameters,including the following: temperature, humidity, pressure, light intensity, and amount of rain,exceeds a certain threshold, In this work, the author used an additional feature of the weather monitoring as an SMS alert system. The author also includes a system for email and tweet post alerts. The creator incorporates multiple sensors and a node MCU 8266 into this system.

3.2 A prototype for IoT-based weather information using WeMos

In the 2020 paper "A prototype for IoT-based weather information using WeMos", the author illustrates a live weather monitoring system that is affordable with an OLED display. The creator also demonstrates the various sectors where the IoT has generated distinctive features in the system. The creator discussed a brand-new, ground-breaking system,which gauges the

current weather's state. Everyone, including farmers, business owners, people who labour in daily life, and students, benefits greatly from weather monitoring. Therefore, by creating a live weather surveillance system,the author also decreased the difficulty for industry and farmers. The weather conditions and other information will be displayed on an OLED display by the author in this work. The Microcontroller-based WeMos D1 board using the ESP8266-EX, running on Arduino, is employed by the author in the suggested method to obtain the cloud data. A wifi module called WeMos D1 was created using the ESP-8266EX microprocessor. It features a flash memory of 4MB. It is one of the best and can be programmed using the Arduino IDE and Node MCU. Wemos and OLED are the sole devices the author employs in this paper to measure the weather conditions. After the connection, the data will be stored in the cloud, and a thingspeak website will be utilised to display the weather data. The system uses OLED and thingSpeak cloud to display the data.The creator wants to display real-time weather information on an OLED screen.

3.3 Finding Dynamic Climatic Parameters Using IoT-Based Weather Reporting

In the paper "IOT Based Weather Reporting System to Find Dynamic Climatic Parameters," the creator implements weather monitoring system powered by IoT and explains how IoT may be used to monitor the weather. and which divulge information about the effects of climate change.Through this effort, people can learn about climatic changes. It produces results that are precise and effective, and it uses the swarm method to further increase accuracy. So, the purpose of this project is for the author to develop a weather monitoring system using IoT. This project uses hardware and software, making it simple to implement. The author of the invention employs a separate sensor to gather climate data, which is then saved in the cloud. The internet site www.thingspeak.com frequently used in projects involving the Internet of Things as this storage. Then using an API key, it extracts all of the weather data from the cloud storage area and sends it to the Android app for mobile devices.

Rain sensor are devices that can detect raindrops. As soon as the plague shows the rains on the strips, the voltage is taken into account. Water conducts electricity poorly, and the sensor acts like a variable resistance, hence a short circuit condition cannot arise. After the voltage

has been measured, the circuit takes the output. A potentiometer is used to detect the voltage, and an LM393 Comparator is used in the system to convert the analogue signal to the digital signal. The LED turns on to show that the digital output is high while the power supply system is visible and the sensor is dry.Additionally, the sensor specifies that the digital output is actively low whenever the sensor's sensor plate gets damp. Three sensors, each with a chip on the back and TC thermistors, can be utilised to measure humidity. The humidity component, this has two electrodes and measures humidity. The controller is a Node MCU. Typically, the Node MCU is utilised in IoT projects. The Arduino IDE was used to programme this. Programming is carried by using the scripting language LUA. On the ESP8266 WIFI module, NodeMCU is powered. Use of this Android STUDIO was created by Google developers. This application's goal is to simplify processing.

Chapter 4

System Requirements and Specification

4.1 Purpose

One of the main problems is live environmental condition because it presents an IoT of challenges when it is measured. There are two different kinds of weather stations: one that has its own sensors, and the other that gets its information from the servers of other weather stations. The latter type describes the current system. The current system is rather pricey. In contrast to the current system, our suggested system includes devices that monitor meteorological parameters at the lowest possible cost. The system that was designed utilised fewer sensors than the model that was previously in use.

4.2 Product Functions

- Connect all the sensors with ESP32 board.
- Write the code using the Arduino IDE and import the necessary libraries.
- Check to see if Wi-Fi is connected or not after uploading the code to the Arduino IDE and opening the serial monitor.
- Data fetched from devise to the server.
- After getting IP address, copy it and paste it on the web browser ,it will display the real time data on webpage

4.3 Hardware Description

4.3.1 ESP32

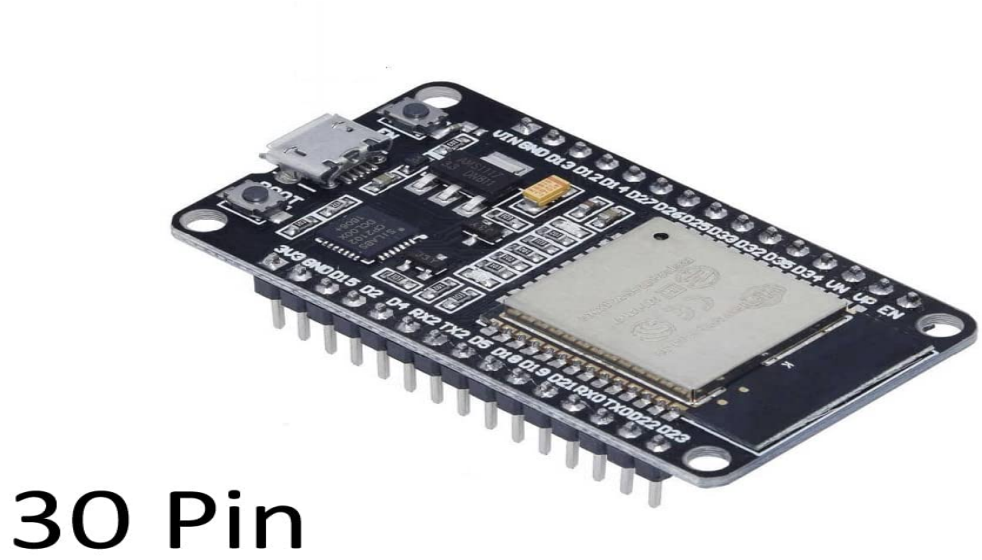


Figure 4.1: ESP32-micro controller chip

The ESP32 line of microcontroller chips is made in Shanghai by Espressif Systems. It is offered in a selection of inexpensive modules. The ESP8266, a chip that "surprise" experimenters in the West in 2014, has been developed into the ESP32. The ESP8266 was first introduced on a module named the ESP-01, which at the time had limited English documentation and few known capabilities. The ESP32 design was enhanced in several ways over the ESP32 design.Only WiFi is supported by the ESP8266, despite having Bluetooth and BLE (Bluetooth Low Energy) . It has a dual-core architecture and is more powerful. Additionally, it features a very low power setting, which makes it appropriate for applications that use batteries.

You may create code for the ESP32 and flash it into the micro controller using other IDEs besides only Arduino. There is the ESP-IDF, which offers significantly greater configuration flexibility and is the official development framework for the ESP32. It is not nearly as simple to use and straightforward as the Arduino IDE, so if you are going to start with the ESP32, the Arduino IDE is the best option. There is also almost any ESP32 capability that can't be implemented with the Arduino IDE thanks to the large developer community's contribution of

the numerous supporting libraries for ESP32. The more skilled and advanced programmers that need to push ESP32 to its boundaries should use ESP-IDF.

4.3.2 DHT11 Temperature and Humidity sensor

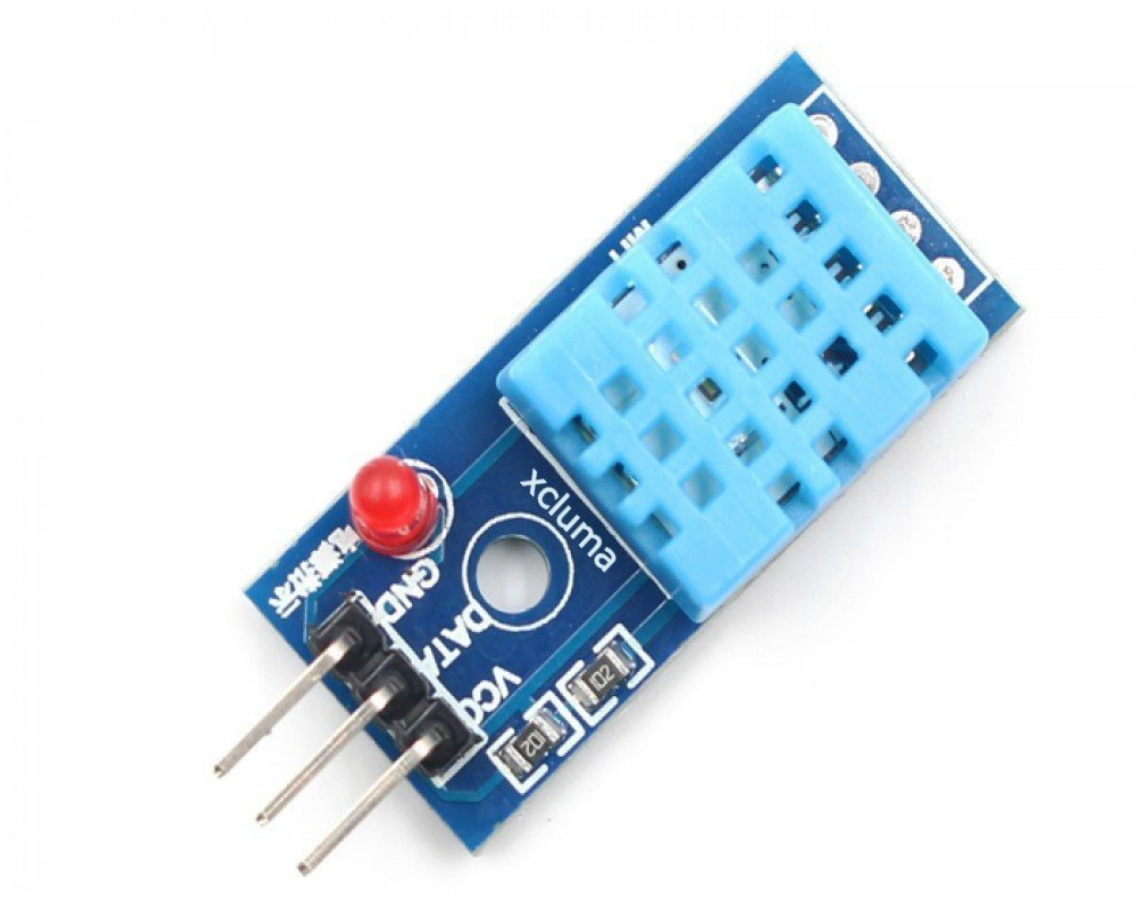


Figure 4.2: DHT11

DHT11 temperature and humidity sensor gives us the value of temperature and humidity where the devise is placed.It gives accurate value while compared to the standard values.Value of temperature is in degree Celsius and humidity is in percentage.

4.3.3 BMP180 Barometric Pressure Sensor

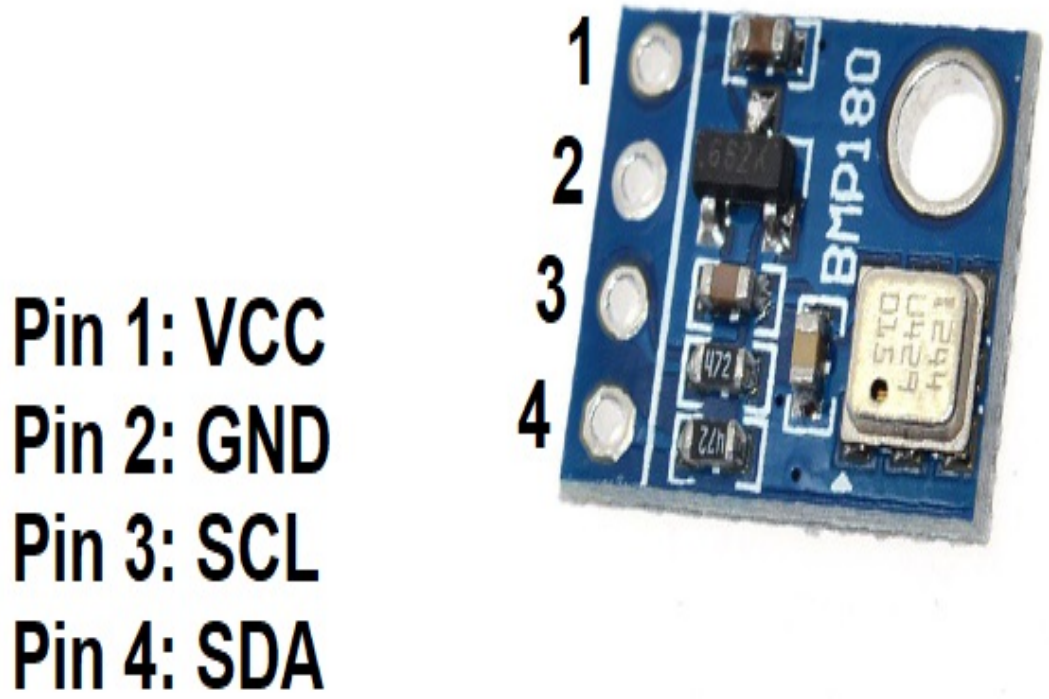


Figure 4.3: BMP180

Bosch's BMP180 Sensor is a highly accurate, reasonably priced sensor that measures temperature and pressure.It can also be used as an altimeter. because pressure varies with height. A 3.3V regulator, I2C level shifter, and pull-up resistors on the I2C pins are attached onto a PCB with the sensor. Software-wise, it is exactly like the BMP085 in every way.

4.3.4 Rain Drop Module

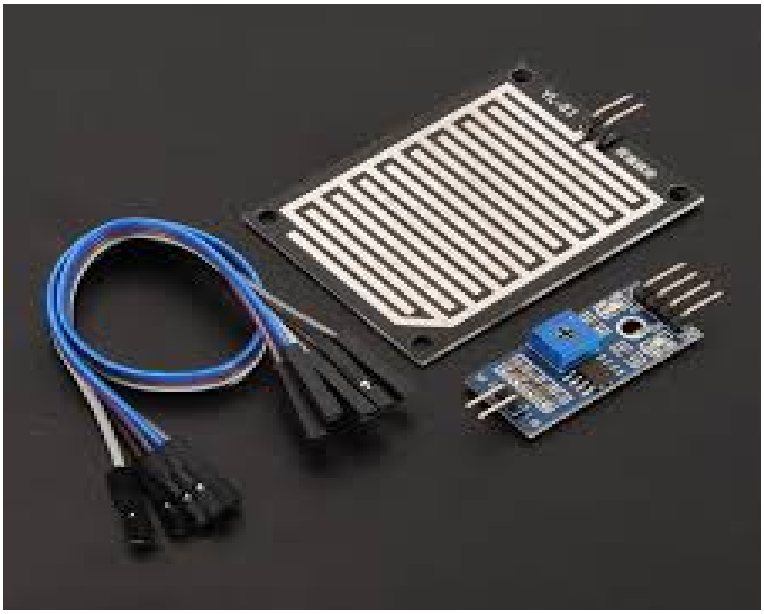


Figure 4.4: Rain Drop Module

It is used to sense the possibility of rain. When rain drops over the module, module shows the value. When water content increases the value will decrease. Default value is 4095.

4.3.5 128*64 OLED I2C Display

Figure 4.5: OLED Display

It is used to display the live data updated from devise. We can see the updation of frequently changing weather parameters through the display.

4.3.6 Jumper Wires

Figure 4.6: Jumper Wires

Components in a circuit are connected by these.

4.4 Software Description

4.4.1 Arduino IDE

An Arduino sketch is a programme developed using the Arduino IDE. The earlier versions of the Arduino Software (IDE) stored sketches as text files with the extension.pde. Only two functions in a straightforward Arduino C/C++ sketch are recognised by the Arduino IDE programmer.

When a sketch starts after being powered on or reset, setup is the first function that is invoked. Variables, input and output pin modes, and other libraries are configured there. The second is loop, which causes the main programme to repeat the function loop after setup has been called. Until it is turned off or reset, it controls the board. On the majority of Arduino boards, pin 13

and ground are connected by an LED and a load resistor, which is a feature that is helpful for many programming and testing activities.

4.4.2 Python Django Framework

Python Django is highy used framework for web application development. It is open source and completely free. Model View Template (MVT) is the design pattern used here. It has high ability of faster development, since it is highly demanding. Application framework takes very short time after collecting all customer requirements. This framework employs the well-known slogan: The web framework for deadline-driven perfectionists.

4.4.3 HTML and CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the fundamental technologies used to create Web pages.

- HTML provides the page structure, while CSS provides the (visual and aural) layout for a variety of devices.
- HTML is simple to learn and apply.
- All browsers support it.
- It is simple to integrate with other languages.
- CSS provides consistent designs
- Provide more website Speed

4.5 Hardware Requirements

- IoT Modules : ESP32 Microcontroller chip,DHT11 temperature and humidity sensor, BMP180 pressure sensor,Rain drop module,i2c 128*64 OLED Display, jumper wires
- Mobile phone connection : Smart phone with internet
- Internet connection : 512 kbps or above

- Processor : Intel Core i3 or above.
- Storage : 512 GB Hard Disk space

4.6 Software Requirements

- Windows 7 or higher is required.
- Code editor : VS code
- Language : Python Django and Processing in Arduino ide
- IDE : Arduino
- Front end : Python Django Framework
- Back end : Arduino
- Web Browser : Firefox/Chrome/Safari/Opera

4.7 Functional Requirements

Functional requirements describe how the framework is intended to operate. This behaviour may be expressed as services, tasks, or abilities that the preset framework must carry out. For this project, the useful prerequisites listed below have been acknowledged.

4.8 Non-Functional Requirements

Global restrictions on a software system are known as non-functional requirements. Costs associated with development, running the business, and factors like as performance, dependability, maintenance, portability, robustness, etc. It is frequently referred to as software characteristics or simply the "ilities." typically unable to be implemented in a single programme module.

4.9 Testing

The framework has to be tested in many scenarios. We test each module's functionality separately. Check the I2c OLED Display's functionality as well. Successfully and accurately updating live weather parameters over the area where the device was installed. Show the most current data that was accessed from the device to the web server and shown on the web page.

4.10 User Interface

The Arduino IDE is used to code the sensors, and HTML and CSS are used to create a website that displays readings from the database to make the proposed framework simple to understand. Additionally, when the rain likelihood value rises above a particular threshold, the user will receive a telegram notice.

Chapter 5

Design And Implementation

Several sensors, including a pressure sensor, a temperature and humidity sensor, and a rain drop module, includes in the proposed system to verify the on-time data. There are two types of weather stations: one has its own sensors, and the other has data centres and servers for remote placement of weather station servers. The proposed system is of the first kind. The suggested system uses a client-side architecture model and operates with the HTTP request protocol. We can download data from the device, upload it to the web server, and then access it via the internet using web pages.

5.1 Weather Monitoring System Development Using Multi Sensors

The project suggests the notion of creating a weather surveillance system that will continually and wirelessly update the state of the weather and monitor live environmental parameters. Numerous sensors are used by the weather monitoring system to measure the variables.

SLNO	MODES	DESCRIPTION
1	Mode 1	If DHT11, i.e., temperature and pressure, is present and turned on, it captures the temperature value and displays it on the web page.
2	Mode 2	If mode 2 is present, BMP 180 displays the pressure of the surrounding environment.
3	Mode 3	If mode 3 present, the Rain sensor module i.e., it indicates the rain value of the surroundings.

Table 5.1: Multiple modes in weather monitoring System

Sensor node-1	Sensor node-2	Sensor node-3
DHT11 (Temperature and pressure sensor)	BMP 180 (Barometric pressure)	Raindrop sensor (Rain sensor)

Table 5.2: Sensor nodes are organised into modes.

5.2 Hardware diagram of the proposed system

The diagram of architecture is shown in the figure below.The sensor is linked to the MCU node. pins using this architecture. ESP32 is used as the node's microcontroller unit (MCU).Power is supplied to the node MCU via USB, which is connected to the PC.

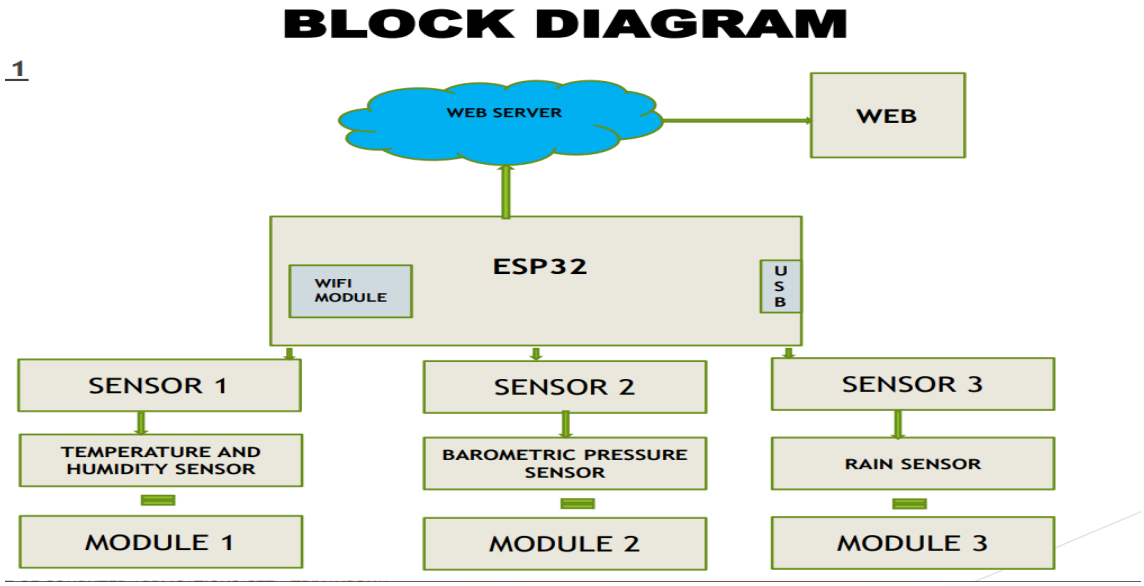


Figure 5.1: Architecture of the Proposed System

5.3 Circuit Diagram

A USB cable connects the hardware to the system, and the temperature sensor, pressure sensor, and rain sensor are all connected to the node MCU pins. The pictures below show what the prototype model looks like. For the connections to work properly, they must all be made in the same way.

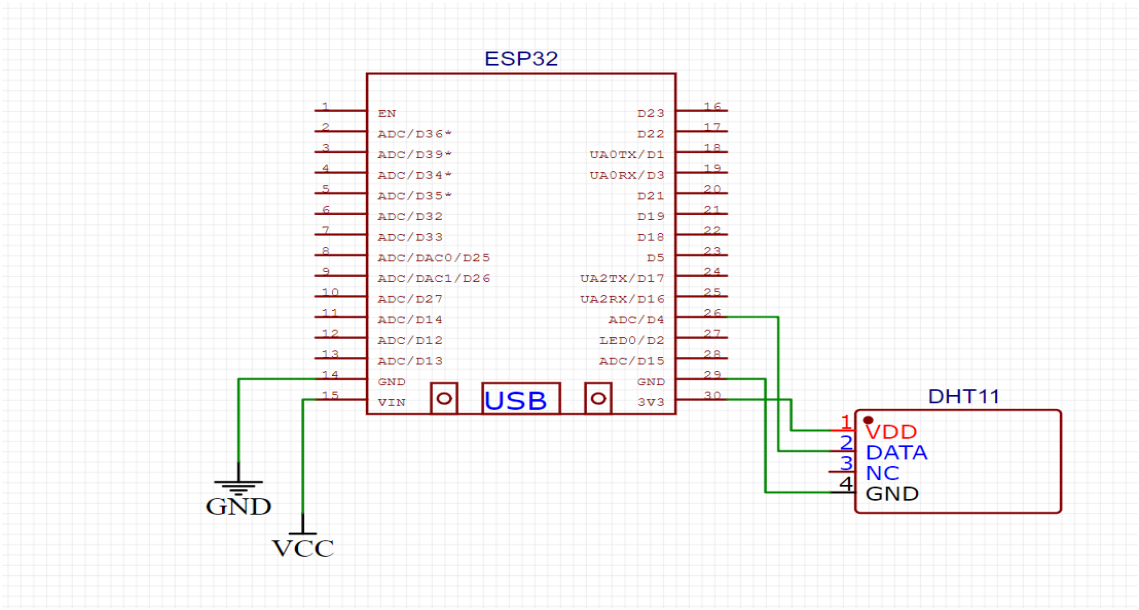


Figure 5.2: Connection of DHT11 with Node MCU

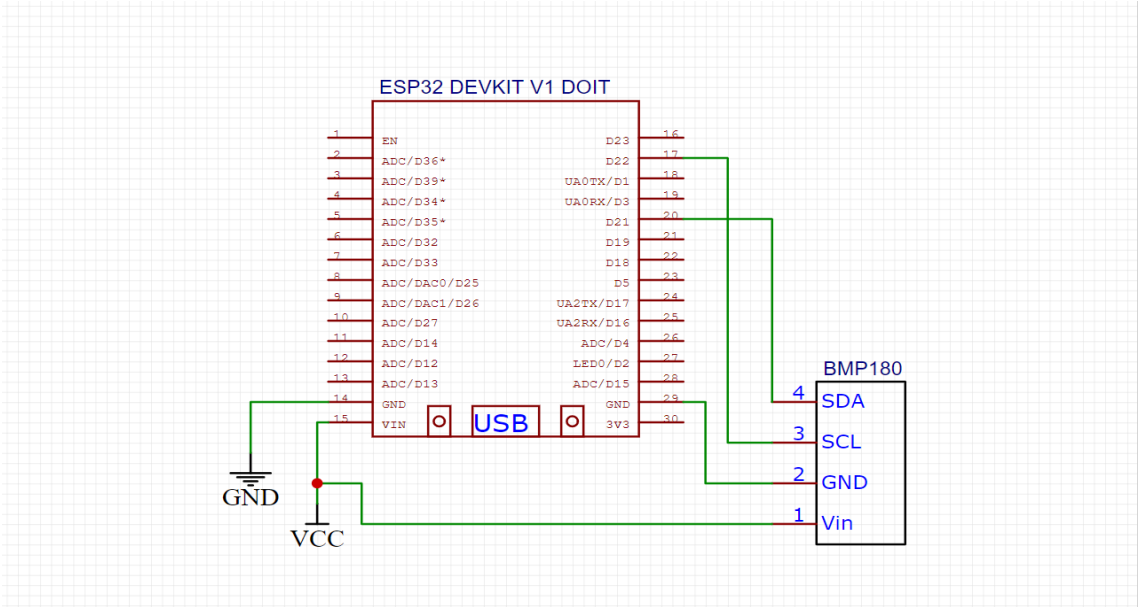


Figure 5.3: Connection of BMP180 with Node MCU

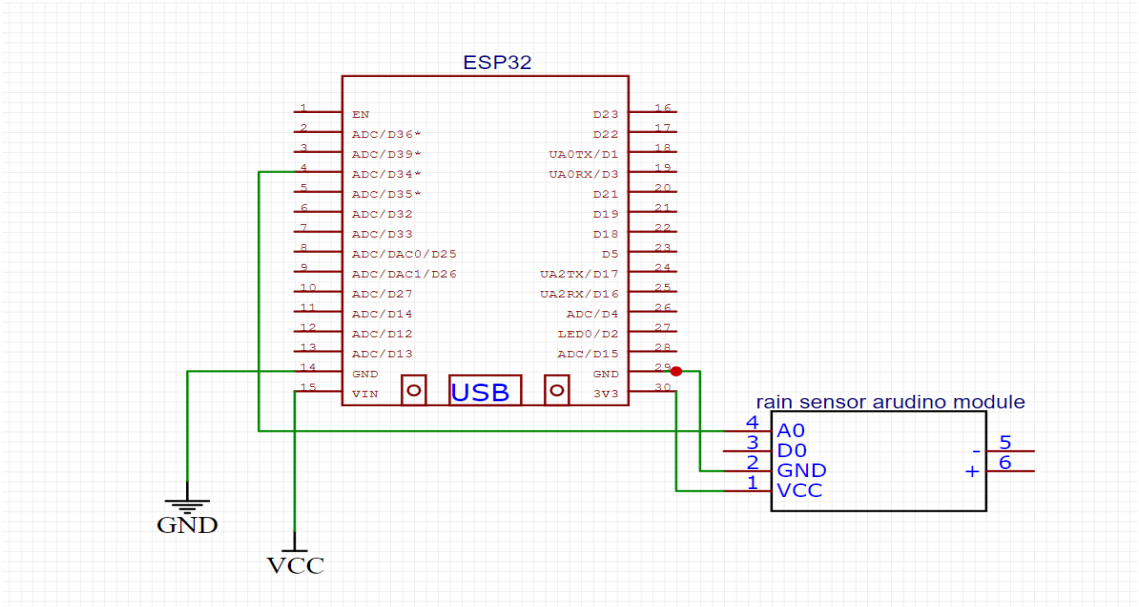


Figure 5.4: Connection of Rain Sensor with Node MCU

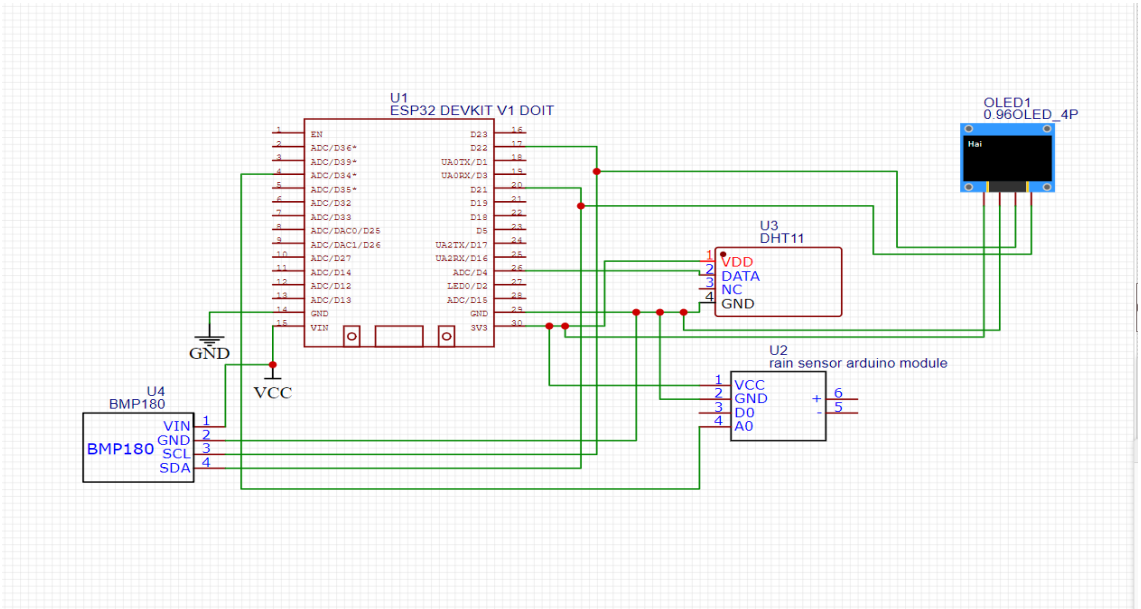


Figure 5.5: Complete Circuit of all sensors along OLED Display with Node MCU

5.4 Methodology

System is a term used to describe how our suggested model works. To help people like farmers, this proposed technology is being created like a weather forecasting system. This system uses the HTTP request protocol to show the results on a web page using only three sensors.

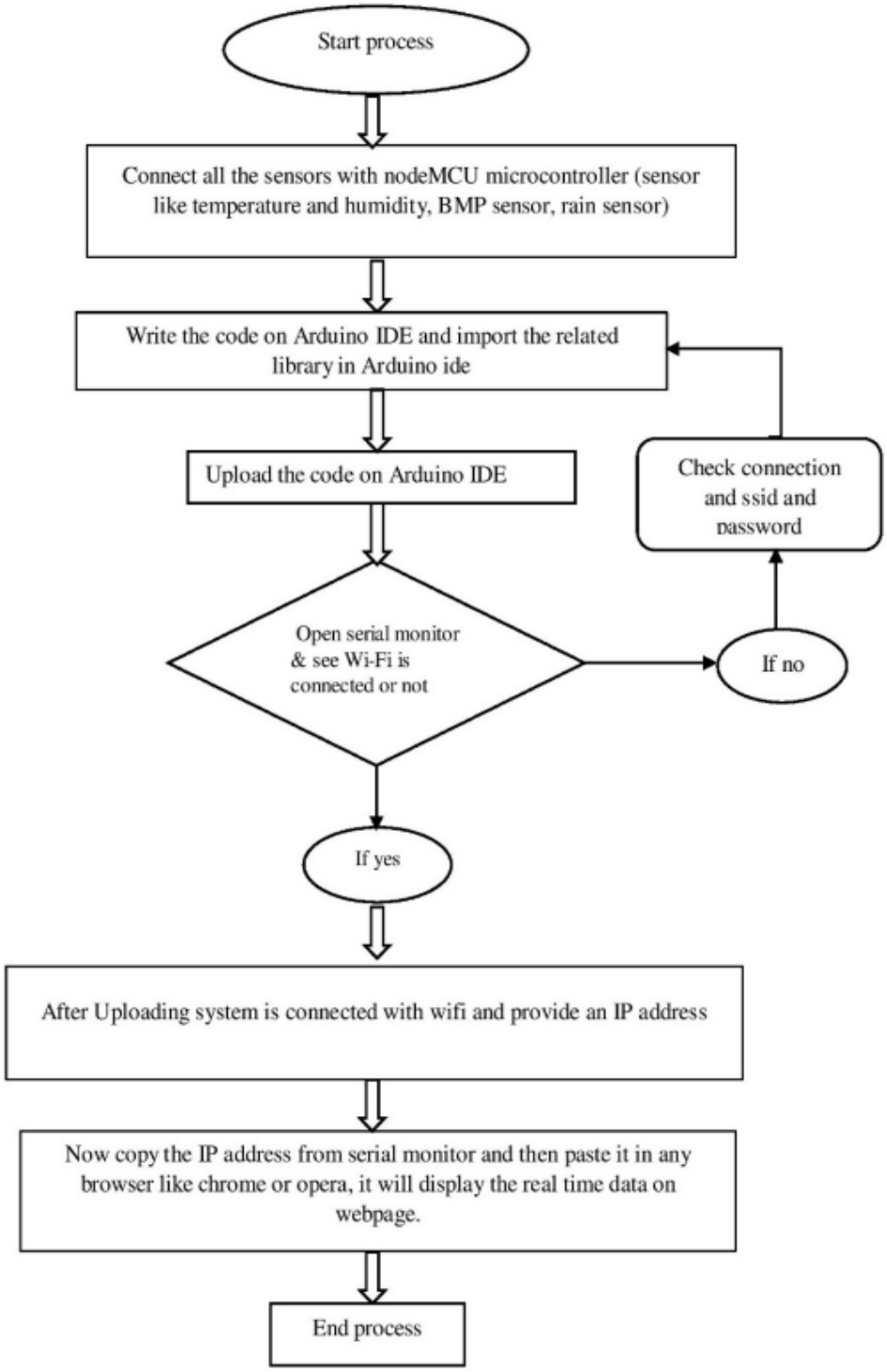


Figure 5.6: Flow Chart

5.5 Working Procedure

Following the completion of all connections, the node MCU is powered using a USB cable that is attached to the personal computer. Two code modules have been created by us. The first step is to link the various sensors in the Arduino code, establish wi-fi, retrieve data from the device to the web server, operate the OLED display unit, and warn the user to make the right choice using the Telegram application based on the rain value. The second code module is for the HTML, CSS, and Django-written web page.

After establishing a hardware connection. then put the code we created online. However, it must first validate before uploading code. So, open the Arduino IDE, enter your code there, select "Verify," then wait for about 1-2 minutes while the verification process is completed, after which you may choose to upload the code once it has successfully completed compiling. Open the serial monitor after that. There, it indicates that the system is Wi-Fi connected. Upon hardware connection being made, HTTP request processing begins. The IP address of the connected Wi-Fi should be copied and pasted into a web browser such as Chrome, Internet Explorer, Opera, etc. Any browser we choose will display the matching temperature, humidity, pressure, and rainfall values.

Chapter 6

Screenshots

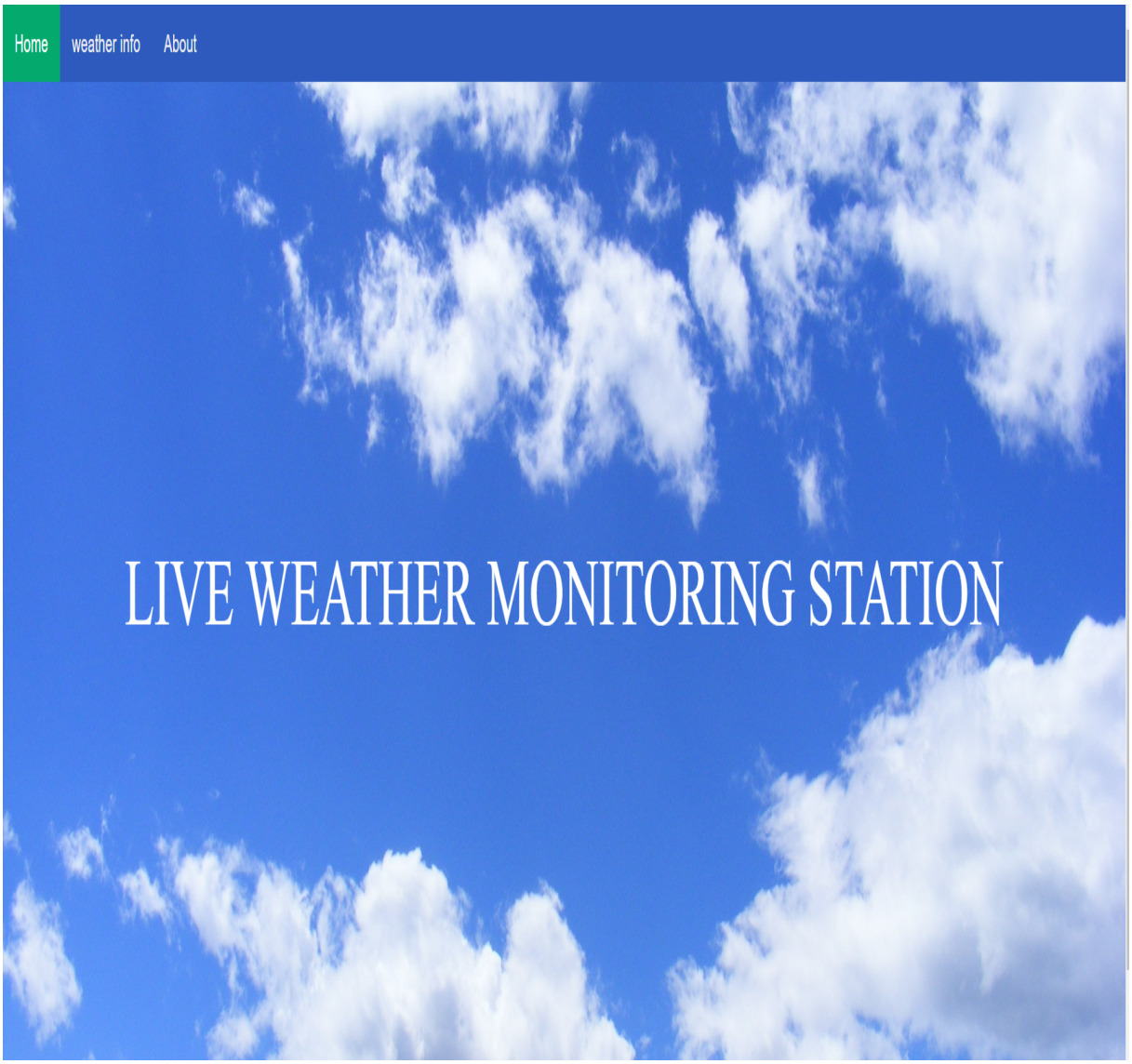


Figure 6.1: Home Page

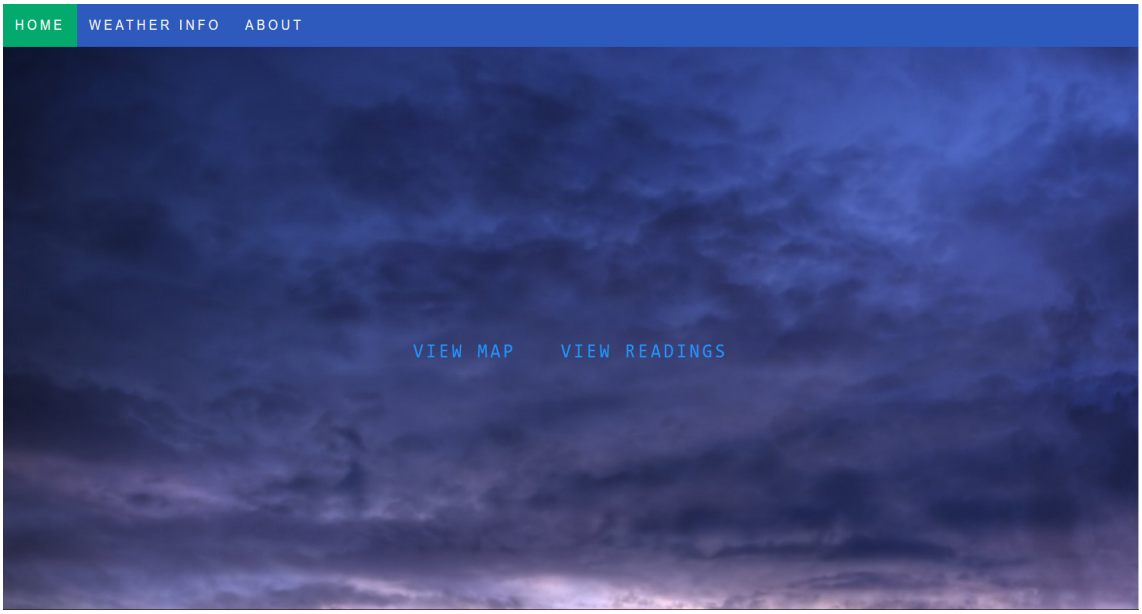


Figure 6.2: Weather Information Page

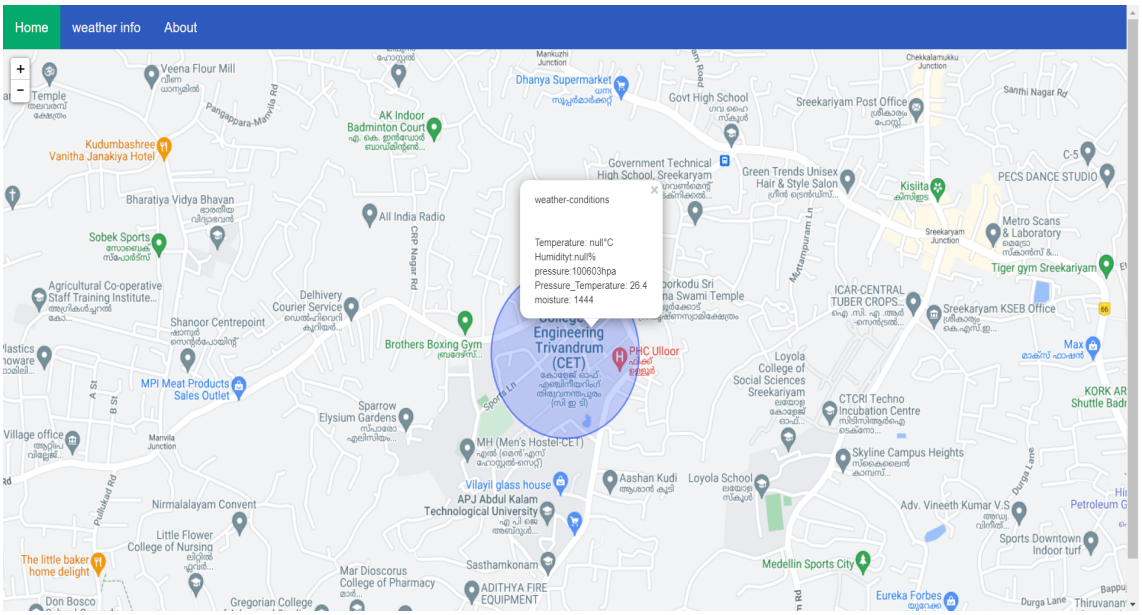


Figure 6.3: Illustration of Map Page

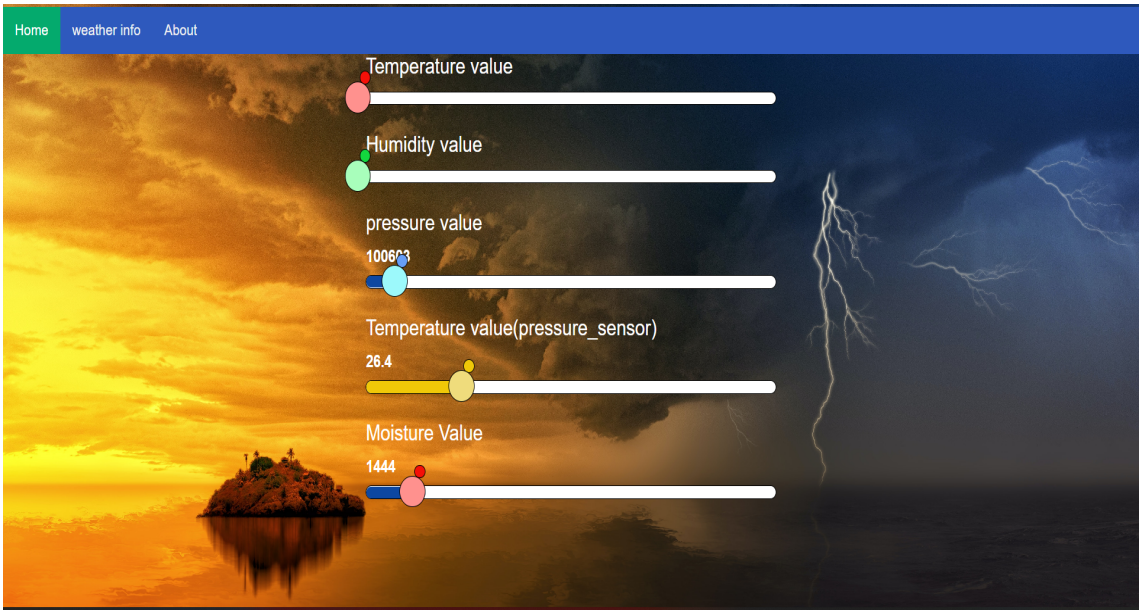


Figure 6.4: Display recently accessed weather data

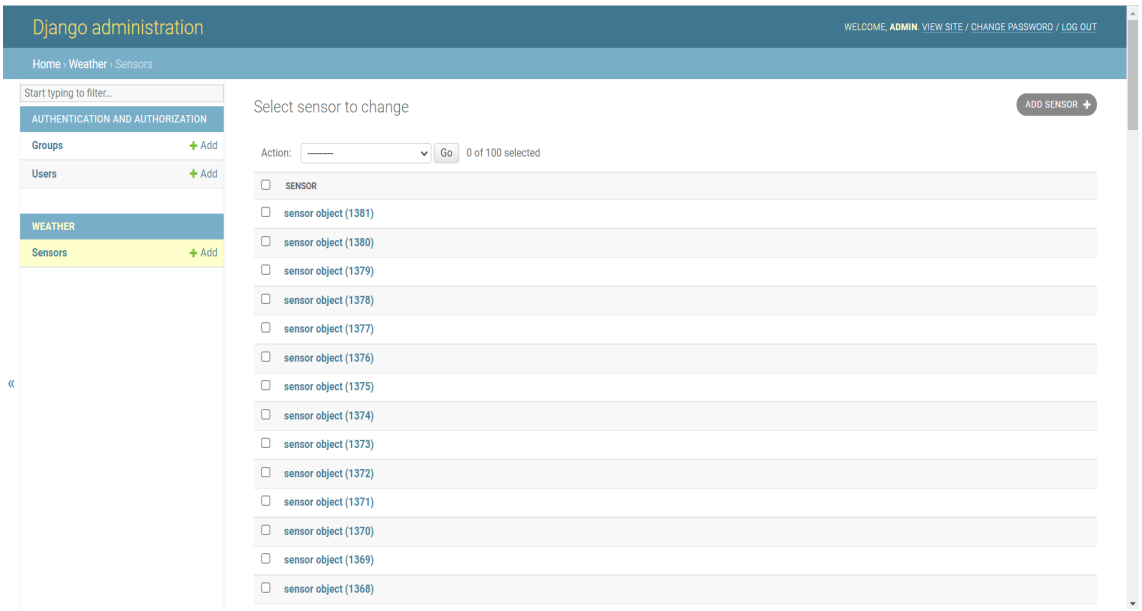


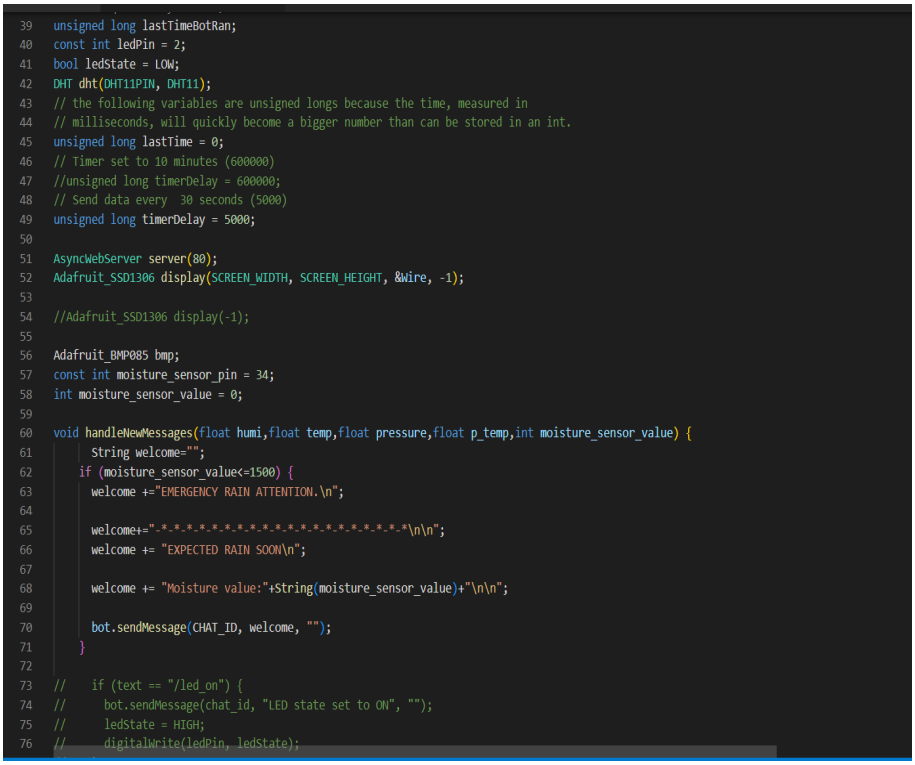
Figure 6.5: DataBase having weather parameters

Chapter 7

Coding

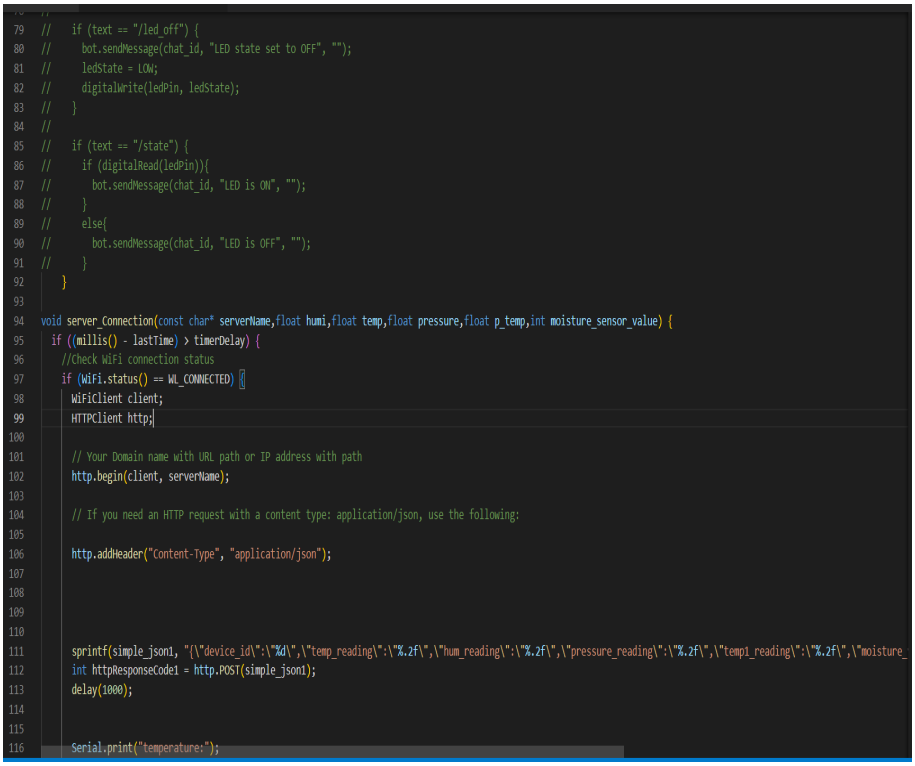
```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>   // Universal Telegram Bot Library written by Brian Lough: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
#include <ArduinoJson.h>
#include <HTTPClient.h>
#include <string.h>
#include "DHT.h"
#include <SPI.h>
#include <Wire.h>
#include <DHT.h>

#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085.h>//DHT library from Adafruit
#define DHT11PIN 4 //Adafruit Unified Sensor library.
#define DHTTYPE    DHT11
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
//#define OLED_RESET -1
#define NUMFLAKES 10
#define LOGO_HEIGHT 16
#define LOGO_WIDTH 16
const char* ssid     = "Silpa";
const char* password = "dhanajjaisilpa";
char simple_json1[200];
char json[200];
#define BOTtoken "5486061774:AAHSipsJOKggNKu9FGnj4asCyaCJmQ7c7tQ"  // your Bot Token (Get from Botfather)


#define CHAT_ID "872187501"
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
//Your Domain name with URL path or IP address with path
const char* serverName = "http://192.168.43.193:8000/readings";
int botRequestDelay = 1000;
```

Figure 7.1: arduino code 1



```
unsigned long lastTimeBotRan;
const int ledPin = 2;
bool ledState = LOW;
DHT dht(DHT11PIN, DHT11);
// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
unsigned long lastTime = 0;
// Timer set to 10 minutes (600000)
//unsigned long timerDelay = 600000;
// Send data every  30 seconds (5000)
unsigned long timerDelay = 5000;

AsyncWebServer server(80);
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

//Adafruit_SSD1306 display(-1);

Adafruit_BMP085 bmp;
const int moisture_sensor_pin = 34;
int moisture_sensor_value = 0;

void handleNewMessages(float humi,float temp,float pressure,float p_temp,int moisture_sensor_value) {
      String welcome="";
    if (moisture_sensor_value<=1500) {
      welcome +="EMERGENCY RAIN ATTENTION.\n";

      welcome+="-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n\n";
      welcome += "EXPECTED RAIN SOON\n";

      welcome += "Moisture value:"+String(moisture_sensor_value)+"\n\n";

      bot.sendMessage(CHAT_ID, welcome, "");
    }

//    if (text == "/led_on") {
//      bot.sendMessage(chat_id, "LED state set to ON", "");
//      ledState = HIGH;
//      digitalWrite(ledPin, ledState);
```

Figure 7.2: arduino code 2



```
//    if (text == "/led_off") {
//      bot.sendMessage(chat_id, "LED state set to OFF", "");
//      ledState = LOW;
//      digitalWrite(ledPin, ledState);
//    }
//
//    if (text == "/state") {
//      if (digitalRead(ledPin)){
//        bot.sendMessage(chat_id, "LED is ON", "");
//      }
//      else{
//        bot.sendMessage(chat_id, "LED is OFF", "");
//      }
    }

void server_Connection(const char* serverName,float humi,float temp,float pressure,float p_temp,int moisture_sensor_value) {
  if ((millis() - lastTime) > timerDelay) {
    //Check WiFi connection status
    if (WiFi.status() == WL_CONNECTED) {
      WiFiClient client;
      HTTPClient http;

      // Your Domain name with URL path or IP address with path
      http.begin(client, serverName);

      // If you need an HTTP request with a content type: application/json, use the following:

      http.addHeader("Content-Type", "application/json");




      sprintf(simple_json1, "{\"device_id\":\"%d\",\"temp_reading\":\"%.2f\",\"hum_reading\":\"%.2f\",\"pressure_reading\":\"%.2f\",\"temp1_reading\":\"%.2f\",\"moisture_
      int httpResponseCode1 = http.POST(simple_json1);
      delay(1000);


      Serial.print("temperature:");
```

Figure 7.3: arduino code 3

```
      Serial.print("temperature:");
      Serial.println(temp);
      Serial.print("Humidity:");
      Serial.println(humi);
      Serial.print("Pressure:");
      Serial.println(pressure);
      Serial.print("Temperature:");
      Serial.println(p_temp);
      Serial.print("Moisture_value:");
      Serial.println(moisture_sensor_value);

      Serial.print("HTTP Response code1: ");
      Serial.println(httpResponseCode1);


      // Free resources
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
  }

}

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

  dht.begin();
  WiFi.begin(ssid, password);
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

Figure 7.4: arduino code 4

```
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());



  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.display();
  delay(2000);
  display.clearDisplay();
  Serial.print("start");
  display.setTextColor(WHITE);
  Serial.println("ing.....");

 /* Start the DHT11 Sensor */

  if (!bmp.begin())
  {
    Serial.println("BMP180 Sensor not found ! ! !");
    while (1)
    {

    }
  }
}

void loop(){

//delay(5000);

  //read temperature and humidity
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
```

Figure 7.5: arduino code 5

```
187   //read temperature and humidity
188   float t = dht.readTemperature();
189   float h = dht.readHumidity();
190   if (isnan(h) || isnan(t)) {
191     Serial.println("Failed to read from DHT sensor!");
192   }
193
194   Serial.print("Temperature: ");
195   Serial.print(t);
196   Serial.print("ºC ");
197   Serial.print("Humidity: ");
198   Serial.println(h);
199   Serial.print("Pressure = ");
200   float pressure=bmp.readPressure();
201   Serial.print(pressure);
202
203   Serial.print(" hPa");
204   Serial.print("  Temp = ");
205    float pressure_temp=bmp.readTemperature();
206   Serial.print(pressure_temp);
207   Serial.print("ºC");
208   moisture_sensor_value = analogRead(moisture_sensor_pin);
209   Serial.print("Moisture value:");
210   Serial.println(moisture_sensor_value);
211
212   // clear display
213   display.clearDisplay();
214
215   // display temperature
216   display.setTextSize(1);
217   display.setCursor(0,0);
218   display.print("Temperature: ");
219   display.setTextSize(2);
220   display.setCursor(0,10);
221   display.print(t);
222   display.print(" ");
223   display.setTextSize(1);
224   display.cp437(true);
```

Figure 7.6: arduino code 6

```
225   display.write(167);
226   display.setTextSize(2);
227   display.print("C");
228   display.display();
229
230   // display humidity
231   display.setTextSize(1);
232   display.setCursor(0, 35);
233   display.print("Humidity: ");
234   display.setTextSize(2);
235   display.setCursor(0, 45);
236   display.print(h);
237   display.print(" %");
238   display.display();
239
240   delay(2000);
241
242   display.clearDisplay();
243
244   //display bmp180 and moisture_sensor
245   display.setTextSize(1);
246   display.setTextColor(WHITE);
247   display.setCursor(0,0);
248   display.print("Temp=");
249   display.print(bmp.readTemperature());
250   display.setCursor(0,10);
251   display.println("*C");
252   //display.setTextSize(1);
253   //display.setTextColor(WHITE);
254   display.setCursor(0,20);
255   display.print("Pressure=");
256   display.print(bmp.readPressure());
257   display.setCursor(0,25);
258   display.println("Pa");
259
260   display.setCursor(0,30);
261   display.print("Moisture value=");
262   display.print(moisture_sensor_value);
```

Figure 7.7: arduino code 7

Chapter 8

Testing and Implementation

System testing is the process that is used to confirm that the structure functions correctly and effectively before live development starts. It is the process of running the programme along with the intention of finding errors and missing steps as well as doing a thorough audit to determine whether the goals have been attained and the needs of the client have been met. Each module is subjected to a test plan employing thorough testing procedures. The unit, integration, and system testing methodologies are defined in the test plan.

By the end of the project development cycle, the user must learn that the project exceeded all the expectations. The functional specification, design specification, or requirements document may not be changed, added to, or deleted without first being documented and tested to the greatest standard attainable given the remaining time provided for the project and the test team's expertise. Application system testing's secondary goal is to ensure that all known issues are fixed before release while also identifying any flaws or potential hazards.

8.0.1 Unit Evaluation

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	Connect all sensors with ESP32	Succesfull completion of circuit	Same as expected	pass
2	Uploading of Arduino Code	Successful uploading of code	Same as expected	pass
3	Checking of Serial Monitor and see Wi-Fi is connected or not	Wi-Fi is connected and display of live weather parameters	Same as expected	pass
4	Transferring data from devise to web server	Successful data transferring	Same as expected	pass
5	Display of web pages	Displaying recently transferred weather data in web pages	Same as expected	pass

Table 8.1: Unit test cases and outcomes

8.0.2 Testing for Integration

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	Integrate the ESP32 and the database to store data.	The ESP32 and the database integrated.	Same as expected	pass
2	Integrate website and the database to retrieve data	website integrated with the database.	Same as expected	pass

Table 8.2: Integration test cases and outcomes

8.0.3 System Evaluation

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	Arduino starts compiling to activate the ESP32.	The ESP32 activates and the data is stored in database.	Same as expected	pass
2	The website updates in real time weather data	Display the live readings from the sensors.	Same as expected	pass

Table 8.3: System test cases and outcomes

Chapter 9

Results and Discussion

The "Live weather monitoring system utilising IOT" focuses to create a system which is used by internet of things to monitor meteorological conditions as well as environmental variables. The built-in weather monitoring system keeps track of numerous variables, such as humidity, pressure, temperature, rain value, etc. After all connections have been made, the web server presents the data from the temperature, humidity, pressure, and rain sensors. The observed data is stored on the web server via HTTP requests on the web page.

9.1 Advantages and Limitations

There are many benefits offered by the proposed system over the current one. It fixes the issues with the current system. In some aspects, the suggested method is more beneficial than the current system.

9.1.1 Advantages

- The suggested system scales numerous factors, including humidity, temperature, pressure, and rain value, using a variety of sensors.
- Hardware used is economical.
- Quality of hardware is not an issue.
- Proposes an extra capability for weather monitoring in the form of an alarm system based on the exceeding of certain parameters, like amount of rain.

- The sensor collected data and transmitted it to the MCU controller, where it was loaded and could be seen on the I2C OLED display and serial monitor.
- The sensed data is uploaded using the Arduino IDE.
- Displays the data on the web server and monitor the real-time data of weather.
- Our suggested model's primary goal is to create a system both economical and cost friendly. to allow unrestricted use by all. The system we propose gathers data from various sensors and sends it all to a web page.

9.1.2 Limitations

- Requirement of consistent Wi-Fi connection.
- Chance of getting hardware complaints.
- Telegram alerts upon rain value only, not considering other weather parameters.

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

Our suggested solution includes a device that monitors weather conditions for the least amount of money. The client-side architecture model underlies how the proposed system operates. The suggested method made use of numerous sensors to observe different environmental data. The system that was designed utilised fewer sensors than the model that was previously in use. Our suggested model's primary goal is to make the system economical and cost-effective. to allow unrestricted use by all. Our solution involves collecting data from a variety of sensors and sending it all to a web page using the HTTP request protocol on the web server.

Once data has been collected from various sensors, it will automatically transmitted to a web server, Once a good connection has been established with the server device, the web server page will enable system monitoring and control. We can access a similar website by entering the server address that is used for monitoring. The website provides details about the weather conditions in the location of the surveillance system.

10.2 Future Scope

Future updates to the planned system could include adding new functionality as well as reducing some of the restrictions in the current model.

- The real-time applications will be suggested the IOT-based weather information system.

- Since we are building a data server, it doesn't physically need any data centres and won't need any in the future.
- So It Reduces the Cost of Equipment.
- During some weather dangers, it may be difficult in some areas to examine and monitor the crucial weather parameter using wires and analogue instruments.
- We can resolve the problem by using Weather parameters are reviewed and monitored using electronic sensors.
- We can extend proposed system to a higher level by detecting more weather parameters.
- We can establish the system into more rural areas

Bibliography

[1] Puja Sharma, Shiva Prakash "Real Time Weather Monitoring System Using Iot"ITM Web of Conferences 40, 01006 (2021) Department of Information Technology and Computer Application ,Madan Mohan Malaviya University of Technology, Gorakhpur.

[2] Ravi Kishore Kodali and Archana Sahu "An IoT based Weather Information Prototype Using WeMos" , 2020 2nd International Conference on Contemporary Computing and Informatics (ic3i), 978-1-5090-5256- 1/16/31.00, IEEE, (2020).

[3] Kavya Ladi, A V S N Manoj, G V N Deepak, "IOT Based Weather Reporting System to Find Dynamic Climatic Parameters", International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2020)

[4] Roopa M L, Varsharani, Prathima S M, Rashmi H C" IOT based Real Time Weather Predication System using NODEMCU 12-E ESP266 and Lab" International Journal of Engineering Research Technology (IJERT), ISSN: 2278-0181,NCCDS – (2021) Conference Proceedings.

[5] Jamal Mabrouki , Mourade Azrour, Driss Dhiba, Yousef Farhaoui,Souad El Hajjaji" IoT-Based Data Logger for Weather Monitoring Using Arduino-Based Wireless Sensor Networks with Remote Graphical Application and Alerts" ISSN 2096-0654 04/07 pp25–32 Volume 4, Number 1, March 2021 DOI: 10.26599/BDMA.2020.9020018.