University of Sri Jayewardenepura

Faculty of Applied Sciences

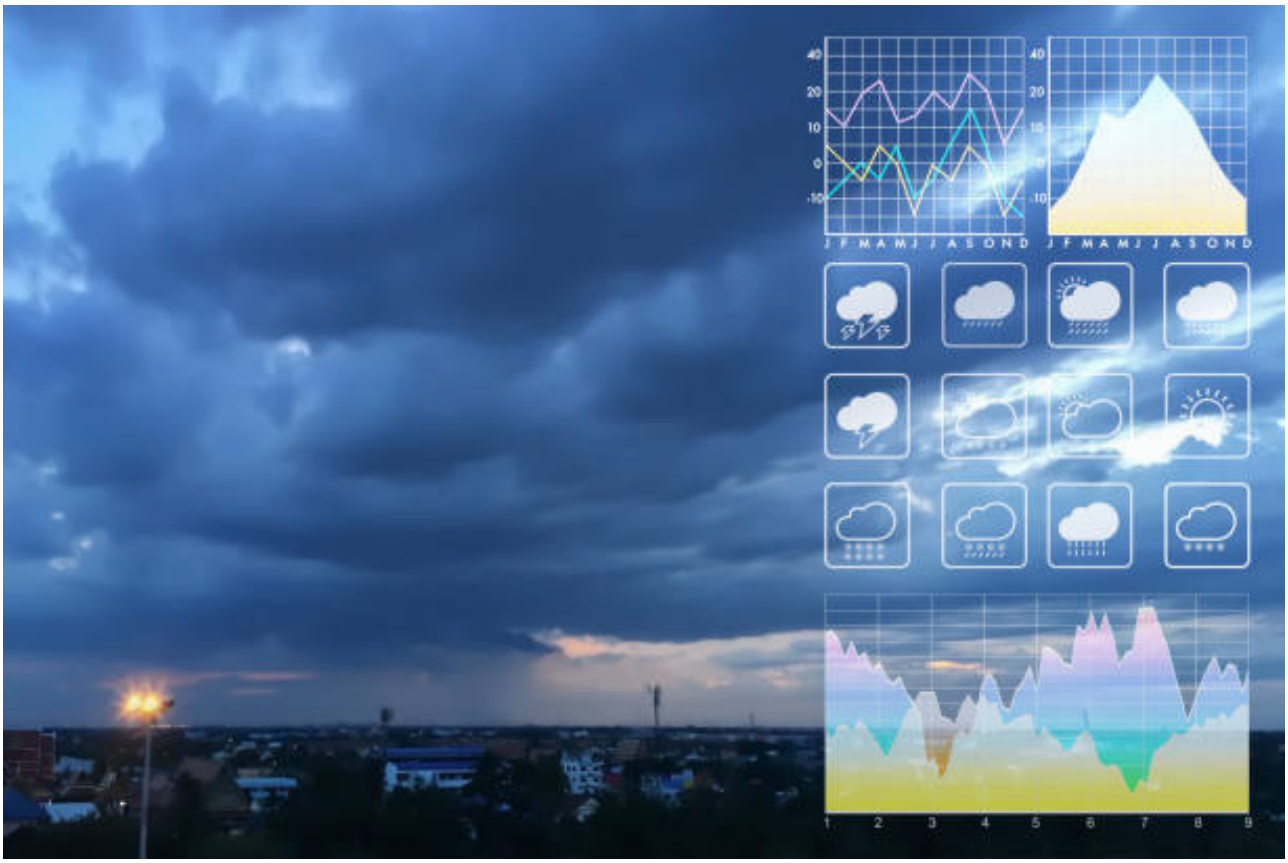
ICT 305 2.0 Embedded System

Project Report

AS2020911

E.A.C.HESHAN

IOT WEATHER MONITORING SYSTEM



Acknowledgments

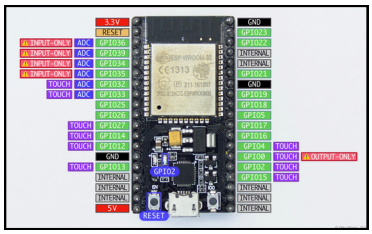
I would really like to place on record my deep sense of gratitude to Dr. M.D.R. Perera, Senior lecturer in Computer Science, Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura, Gangodawila, Nugegoda, Sri Lanka for his generous guidance, assist and beneficial suggestions. I specific my honest assistance to the embedded system project.

Contents

Introduction

This system leverages the Internet of Things (IoT) to revolutionize weather monitoring. By seamlessly integrating sensors, data analytics, and real-time communication, this solution provides a comprehensive and accurate overview of environmental conditions. Stay ahead of the weather with a smart, interconnected solution that transforms how we understand and respond to atmospheric changes.

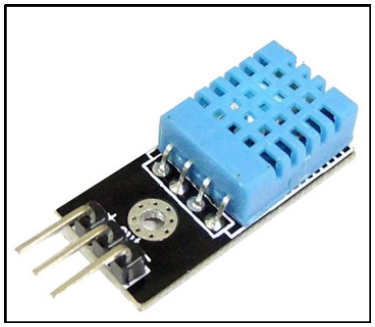
Components



NodeMCU - ESP32S

DataSheet

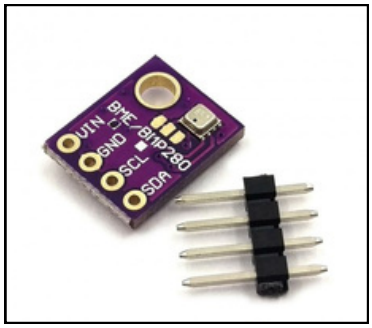
NodeMCU ESP32 is a versatile open-source development board based on the **ESP32 microcontroller**. It combines the power of the ESP32 chip with the ease of NodeMCU programming, making it suitable for a wide range of IoT applications. With built-in **Wi-Fi and Bluetooth** capabilities, it enables seamless connectivity and communication with other devices. Its **compact size** and **low power consumption** further enhance its suitability for **embedded systems and IoT applications**.



DHT11 Humidity and Temperature Sensor

DataSheet

The DHT11 is a low-cost **digital temperature and humidity** sensor. It is commonly used in electronics projects and applications to measure ambient temperature and relative humidity. The sensor is designed to provide accurate and reliable readings with a simple interface, making it suitable for a wide range of DIY projects, **weather stations, and environmental monitoring systems**. The DHT11 communicates over a single-wire digital interface, making it easy to integrate into various microcontroller-based projects. Keep in mind that while the DHT11 is affordable and **easy to use**.



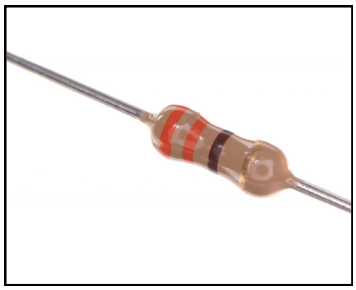
BMP280 Barometric Pressure Sensor

DataSheet

The BMP280 is a digital barometric pressure sensor that also measures temperature. Developed by Bosch Sensortec, this sensor is commonly used in various applications, including **weather stations, altitude tracking in GPS modules**, and **indoor navigation systems**. It provides **accurate pressure and temperature data with high resolution**, making it suitable for both consumer and industrial applications. The BMP280 communicates over **I2C or SPI interfaces**, making it easy to integrate into a variety of electronic systems. Its **compact size, low power consumption**, and precision make it a popular choice for projects requiring atmospheric pressure and temperature monitoring.

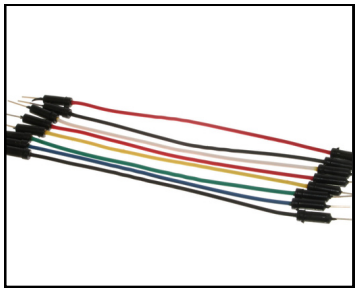
LED

LED stands for Light Emitting Diode. It is a semiconductor device that emits light when an electric current passes through it. LEDs are energy-efficient, durable, and versatile light sources commonly used in various applications, including lighting, displays, indicators, and more. They come in a range of colors and are known for their long lifespan and low power consumption compared to traditional light sources. LEDs have become widely used in everyday electronics, lighting fixtures, automotive lighting, and as a key component in the development of energy-efficient lighting solutions.



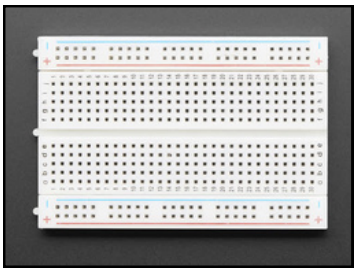
Resistor - 330ohm

A 330-ohm resistor is an electronic component designed to impede the flow of electric current in a circuit. It has a resistance value of 330 ohms, meaning it resists the passage of current and helps control the amount of current flowing through a circuit. Resistors are commonly used in electronics to limit current, divide voltage, or protect components from excessive current. The 330-ohm resistor is a specific value that can be employed in various electronic applications to achieve specific electrical characteristics in a circuit.



Connecting Wires

A connecting wire is a conductor, typically made of metal, used to establish an electrical connection between two or more points in a circuit. It serves to transmit electrical signals, allowing the flow of current between components such as switches, resistors, and electronic devices. Connecting wires come in various lengths, thicknesses, and materials to suit different applications, providing a crucial role in enabling the functioning of electrical and electronic systems.

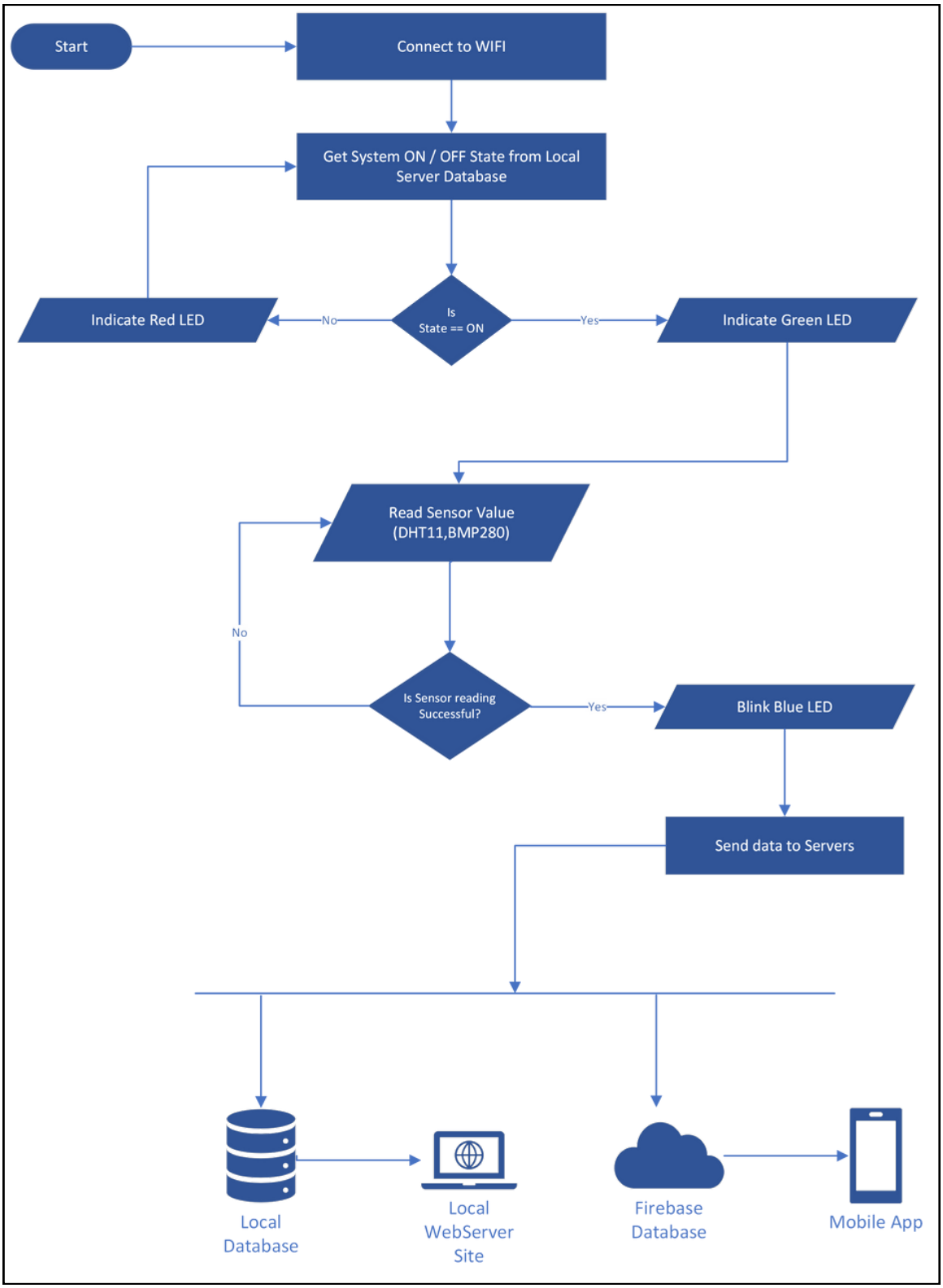


Bread Board

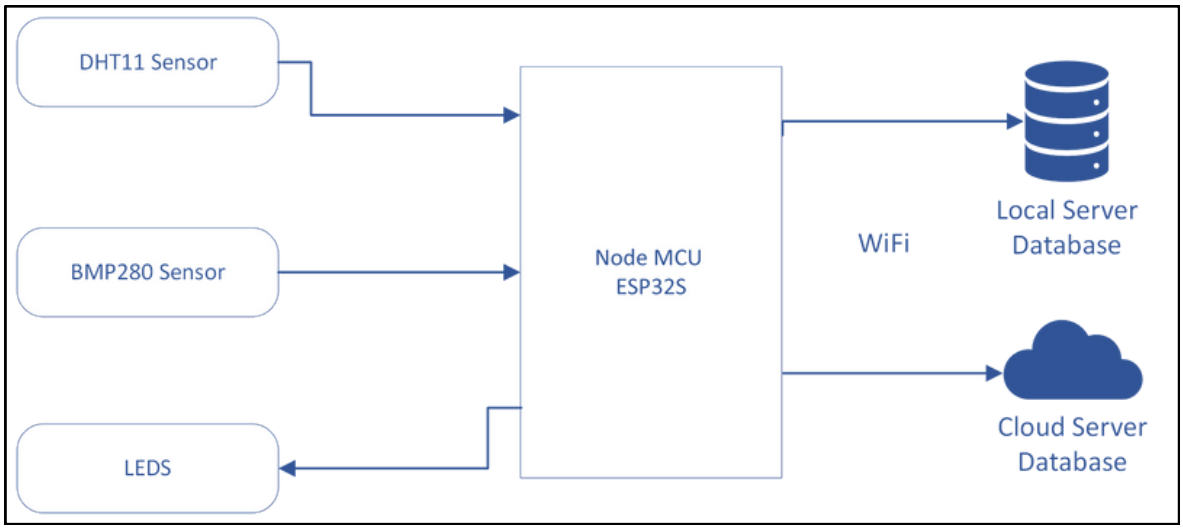
A breadboard is a reusable solderless device used for prototyping electronic circuits. It typically consists of a rectangular board with a grid of holes into which electronic components and wires can be inserted. The holes are connected in groups, allowing users to create temporary connections between components without the need for soldering. This makes it easy to experiment with and test different circuit configurations before committing to a permanent design. Breadboards are widely used by electronics enthusiasts, students, and professionals for rapid and flexible prototyping of electronic projects.

System Design & Diagram

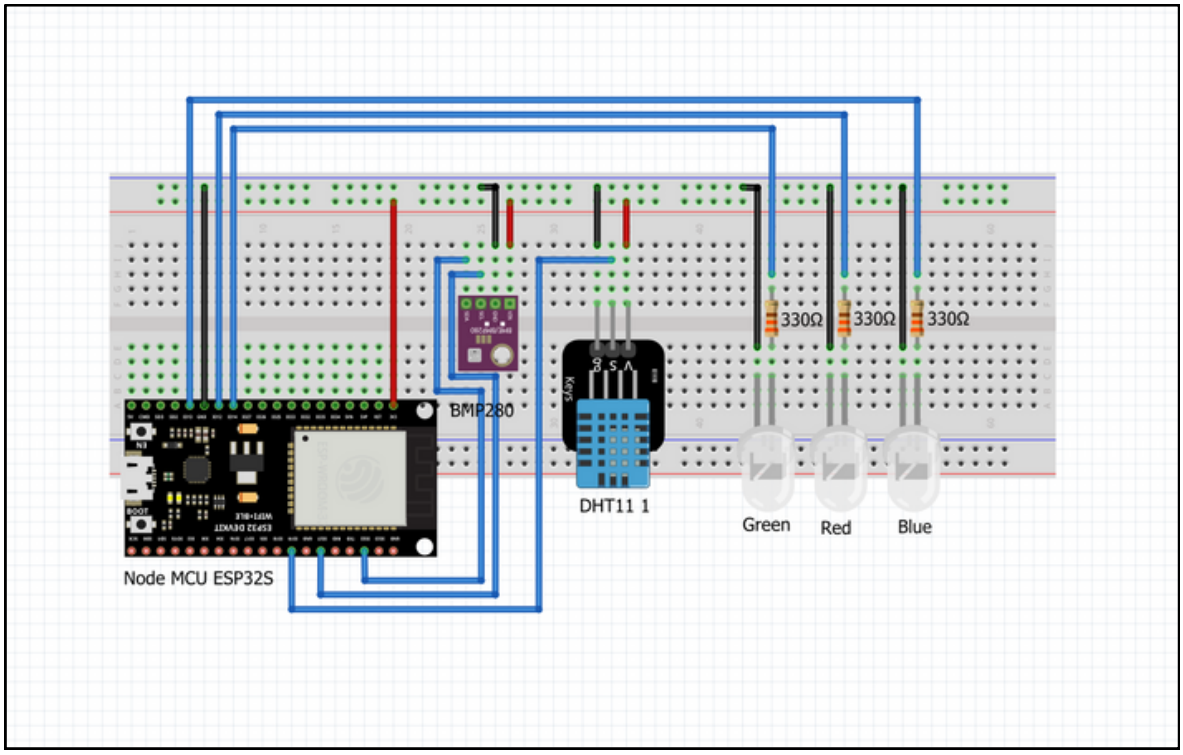
- Data Flow Diagram



• Block Diagram



• Circuit Diagram



• Sofrware and technologies

- Arduino IDE
- XAMPP
- HTML , PHP , JavaScript
- FireBase
- Android Studio

Code

- Arduino

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
#include <Adafruit_BMP280.h>
#include <Arduino_JSON.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

#define DHTPIN 19
#define DHTTYPE DHT11
#define LED_GRN_PIN 12
#define LED_RED_PIN 13
#define LED_BLUE_PIN 14
#define API_KEY ""
#define DATABASE_URL ""
#define SERVER_IP ""
#define WIFI_SSID ""
#define WIFI_PASSWORD ""


DHT dht11(DHTPIN, DHTTYPE);
Adafruit_BMP280 bmp;
String URL = "http://" + String(SERVER_IP) + "/dht11_project/test_data.php";

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
String location = "";

int temperature = 0;
int humidity = 0;
int pressure = 0;
bool sensorOn = false;
unsigned long sendDataPrevMillis = 0;
bool signupOK = false;
```

```
void setup() {
  pinMode(LED_GRN_PIN, OUTPUT);
  pinMode(LED_RED_PIN, OUTPUT);
  pinMode(LED_BLUE_PIN, OUTPUT);

  dht11.begin();
  bmp.begin(0x76);

  Serial.begin(115200);
  connectWiFi();

  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;

  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("Firebase Signup ok");
    signupOK = true;
  } else {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

void loop() {
  Serial.println("Start ------------------------------------------------------------");
  checkState("OnOff");
  checkState("Configuration");

  if (sensorOn) {
    digitalWrite(LED_GRN_PIN, HIGH);
    digitalWrite(LED_RED_PIN, LOW);

    if (WiFi.status() != WL_CONNECTED) {
      connectWiFi();
    }

    LoadDHT11Data();
    LoadBMP280Data();
    Blink();

    String postData = "temperature=" + String(temperature) +
                      "&humidity=" + String(humidity) +
                      "&pressure=" + String(pressure);

    HTTPClient http;
    http.begin(URL);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    int httpCode = http.POST(postData);
    String payload = http.getString();
```

```
    Serial.print("URL :");
    Serial.println(URL);
    Serial.print("Data:");
    Serial.println(postData);
    Serial.print("httpCode :");
    Serial.println(httpCode);
    Serial.print("payload :");
    Serial.println(payload);
    Serial.println("---------------------------------");

    float fbTemperature = dht11.readTemperature();
    float fbHumidity = dht11.readHumidity();
    float fbPressure = bmp.readPressure() / 100;

    if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 5000 ||
sendDataPrevMillis == 0)) {
      Blink();
      sendDataPrevMillis = millis();

      if (Firebase.RTDB.setInt(&fbdo, "DHT_11/Temperature", fbTemperature) &&
          Firebase.RTDB.setFloat(&fbdo, "DHT_11/Humidity", fbHumidity) &&
          Firebase.RTDB.setFloat(&fbdo, "DHT_11/Pressure", fbPressure) &&
          Firebase.RTDB.setString(&fbdo, "DHT_11/location", location)) {
        Serial.println("Firebase Data Sent Successfully");
      } else {
        Serial.println("Failed to Send Data to Firebase");
        Serial.println("REASON: " + fbdo.errorReason());
      }
    }
    Serial.println("End ------------------------------------------------------------");
  } else {
    digitalWrite(LED_RED_PIN, HIGH);
    digitalWrite(LED_GRN_PIN, LOW);
  }
}

void checkState(String stateType) {
  String postData = "";
  String payload = "";
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    int httpCode;
    postData = "id=1";
    payload = "";

    Serial.println();
    Serial.println("---------------getdata.php");
    if (stateType == "OnOff") {
      http.begin("http://" + String(SERVER_IP) + "/dht11_project/onoffstate.php");
    }
    if (stateType == "Configuration") {
      http.begin("http://" + String(SERVER_IP) + "/dht11_project/configstate.php");
    }
```

```
    httpCode = http.POST(postData);
    payload = http.getString();

    Serial.print("httpCode : ");
    Serial.println(httpCode);
    Serial.print("payload  : ");
    Serial.println(payload);

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

    JSONVar myObject = JSON.parse(payload);

    if (JSON.typeof(myObject) == "undefined") {
      Serial.println("Parsing input failed!");
      Serial.println("---------------");
      return;
    }

    if (stateType == "OnOff") {
      if (myObject.hasOwnProperty("State")) {
        Serial.print("Sensor State = ");
        Serial.println(myObject["State"]);
      }

      if (strcmp(myObject["State"], "1") == 0) {
        sensorOn = true;
        Serial.println("Sensor ON");
      }
      if (strcmp(myObject["State"], "0") == 0) {
        sensorOn = false;
        Serial.println("Sensor OFF");
      }
    }

    if (stateType == "Configuration") {
      if (myObject.hasOwnProperty("Location")) {
        Serial.print("Location State = ");
        Serial.println(myObject["Location"]);
        location = String(static_cast<const char*>(myObject["Location"]));
      }
    }
  }
  delay(1000);
}

void Blink() {
  digitalWrite(LED_BLUE_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_BLUE_PIN, LOW);
  delay(1000);
}
```

```
void connectWiFi() {
  WiFi.mode(WIFI_OFF);
  delay(1000);
  WiFi.mode(WIFI_STA);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.print("Connected to: ");
  Serial.println(WIFI_SSID);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void LoadBMP280Data() {
  pressure = bmp.readPressure() / 100;

  if (isnan(pressure)) {
    Serial.println("Failed to read data from BMP280 sensor!");
    pressure = 0;
  }

  Serial.printf("Pressure: %d hPa\n", pressure);
}

void LoadDHT11Data() {
  temperature = dht11.readTemperature();
  humidity = dht11.readHumidity();

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read data from DHT11 sensor!");
    temperature = 0;
    humidity = 0;
  }

  Serial.printf("Temperature: %d °C\n", temperature);
  Serial.printf("Humidity: %d %%\n", humidity);
}
```

- Web - Admin Page

```
<!DOCTYPE html>
<html>
<head>
  <title>Sensor Data</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <link rel="stylesheet" href="Style.css">
</head>
<body>
  <h1 class="display-4">Weather Station 5.0 - Admin</h1>
  <hr class="hr hr-blurry" />

  <div class="container chart-container">
    <div class="chart">
      <h2 class="mb-4">Temperature Chart</h2>
      <canvas id="temperatureChart" width="400" height="200"></canvas>
    </div>

    <div class="chart">
      <h2 class="mb-4">Humidity Chart</h2>
      <canvas id="humidityChart" width="400" height="200"></canvas>
    </div>

    <div class="chart">
      <h2 class="mb-4">Pressure Chart</h2>
      <canvas id="pressureChart" width="400" height="200"></canvas>
    </div>
    <!-- Sidebar -->
    <div class="container">
      <div class="card">
        <div class="card header">
          <h3 style="font-size: 1rem;text-align: center;">System Controlling</h3>
        </div>
        <br>
        <label class="switch">
              <input type="checkbox" id="chkbtn" data-record-id="<?php echo $row['id']; ?>"
onclick="updateSystemState(this)">
          <div class="sliderTS"></div>
        </label>

      </div>
    </div>
  </div>
  <div class="container">
    <h2 class="mb-4">Weather Station Configuration</h2>
    <form action="updateConfig.php" method="post">
      <div class="form-group">
        <label for="Location">Location:</label>
        <input type="text" class="form-control" name="Location" id="Location" required>
      </div>
      <button type="submit" class="btn btn-primary">Save Configuration</button>
    </form>
  </div>
```

```
  <div class="container">
    <h2 class="mb-4">Sensor Data</h2>
    <table class="table table-striped table-bordered">
      <thead class="thead-dark">
        <tr>
          <th>ID</th>
          <th>Temperature</th>
          <th>Humidity</th>
          <th>Pressure</th>
          <th>Date/Time</th>
        </tr>
      </thead>
      <tbody>
        <?php

        $hostname = "localhost";
        $username = "root";
        $password = "";
        $database = "sensor_db";
        // Connect to the MySQL database
        $db = new mysqli($hostname,$username,$password,$database);

        // Check for connection errors
        if ($db->connect_error) {
        die("Connection failed: " . $db->connect_error);
        }

        // Fetch data from the DHT11 table and BMP280 table
            $query = "SELECT dht11.*, bmp280.presure FROM dht11 JOIN bmp280 ON dht11.id =
bmp280.ID ORDER BY id DESC LIMIT 20";
        $result = $db->query($query);

        if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
          echo "<tr>";
          echo "<td>" . $row['id'] . "</td>";
          echo "<td>" . $row['temperature'] . "</td>";
          echo "<td>" . $row['humidity'] . "</td>";
          echo "<td>". $row["presure"] . "</td>";
          echo "<td>" . $row['date/time'] . "</td>";
          echo "</tr>";
        }
        }

        // Close the database connection
        $db->close();
        ?>
      </tbody>
    </table>
  </div>
     <script   src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js">
</script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <script src="Script.js"></script>
</body>
</html>
```

- Android App

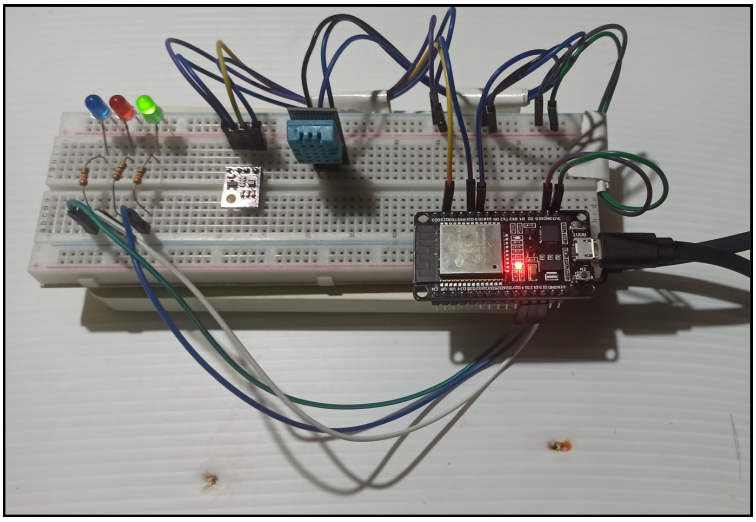
```
package com.example.weatherapp30;

import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {

    private DatabaseReference mDatabase;
    private TextView humidityTextView, temperatureTextView, pressureTextView , maxminTextView
, RealFellTextView , LocationTextView ,WStaetement;

    private double fltTempMax =0.0;
    private double fltTempMin = 0.0;
    private double fltTempMid = 0.0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize Firebase database reference
        mDatabase = FirebaseDatabase.getInstance().getReference().child("DHT_11");

        // Initialize UI elements
        humidityTextView = findViewById(R.id.humidityTextView);
        temperatureTextView = findViewById(R.id.TemperatureView);
        pressureTextView = findViewById(R.id.pressureTextView);
        maxminTextView = findViewById(R.id.maxmintextview);
        RealFellTextView = findViewById(R.id.temperatureTextView);
        LocationTextView = findViewById(R.id.LocationTextView);
        WStaetement = findViewById(R.id.wstatementtextview);

        // Attach a ValueEventListener to listen for changes in data
        mDatabase.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    // Retrieve and display data
                    Long humidityLong = dataSnapshot.child("Humidity").getValue(Long.class);
                                                                    Long    temperatureLong    =
dataSnapshot.child("Temperature").getValue(Long.class);
                    Long pressureLong = dataSnapshot.child("Pressure").getValue(Long.class);
                                                                  String    StringLocation    =
dataSnapshot.child("location").getValue(String.class);
```

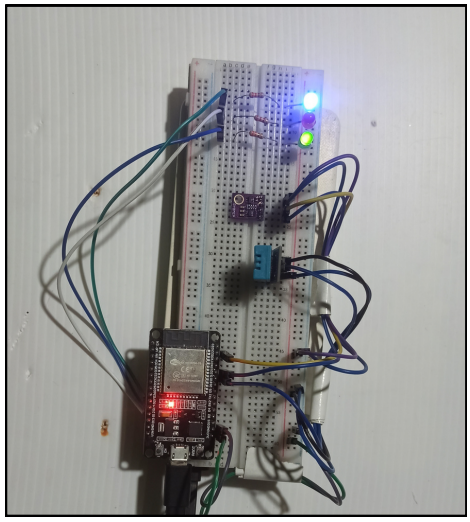
```
                if (humidityLong != null) {
                    String humidity = String.valueOf(humidityLong);
                    humidityTextView.setText(humidity + "%");
                } else {
                    humidityTextView.setText("N/A");
                }

                if (temperatureLong != null) {
                    String temperature = String.valueOf(temperatureLong);
                    temperatureTextView.setText(temperature + "°C");
                } else {
                    temperatureTextView.setText("N/A");
                }

                if (pressureLong != null) {
                    String pressure = String.valueOf(pressureLong);
                    pressureTextView.setText(pressure + "mbar");
                } else {
                    pressureTextView.setText("N/A");
                }

                if (StringLocation != null) {
                    LocationTextView.setText(StringLocation);
                } else {
                    LocationTextView.setText("N/A");
                }

                if(fltTempMin == 0.0 && fltTempMin == 0.0 ){
                    fltTempMin = temperatureLong;
                    fltTempMax = temperatureLong;
                }else {
                    if (temperatureLong >= fltTempMax) {
                        fltTempMax = temperatureLong;
                    }
                    if (temperatureLong <= fltTempMin) {
                        fltTempMin = temperatureLong;
                    }
                }

                String strTempMax = String.valueOf(fltTempMax);
                String strTempMin = String.valueOf(fltTempMin);
                maxminTextView.setText(strTempMax + "/" + strTempMin);

                fltTempMid = (fltTempMax + fltTempMin)/2;
                String strTempMid = String.valueOf(fltTempMid);
                RealFellTextView.setText(strTempMid + "°C");

                // Generate weather statement
                          String  weatherStatement  =  getWeatherStatement(temperatureLong,  humidityLong,
pressureLong);

                // Display weather statement
                WStaetement.setText(weatherStatement);


            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            // Handle any errors
        }
    });
}
```

```
    private String getWeatherStatement(long temperature, long humidity, long pressure) {
        // Define your conditions and thresholds here
        // You can use the provided getWeatherStatement function from the previous response

        // Example conditions
        boolean isHighTemperature = temperature > 30;
        boolean isLowHumidity = humidity < 30;
        boolean isHighPressure = pressure > 1015;

        // Generate weather statement based on conditions
        if (isHighTemperature && isLowHumidity && isHighPressure) {
            return "Hot and dry weather with high pressure.";
        } else if (isHighTemperature) {
            return "Hot weather.";
        } else if (isLowHumidity) {
            return "Dry weather.";
        } else if (isHighPressure) {
            return "High-pressure system.";
        } else {
            return "Normal weather conditions.";
        }
    }
}
```

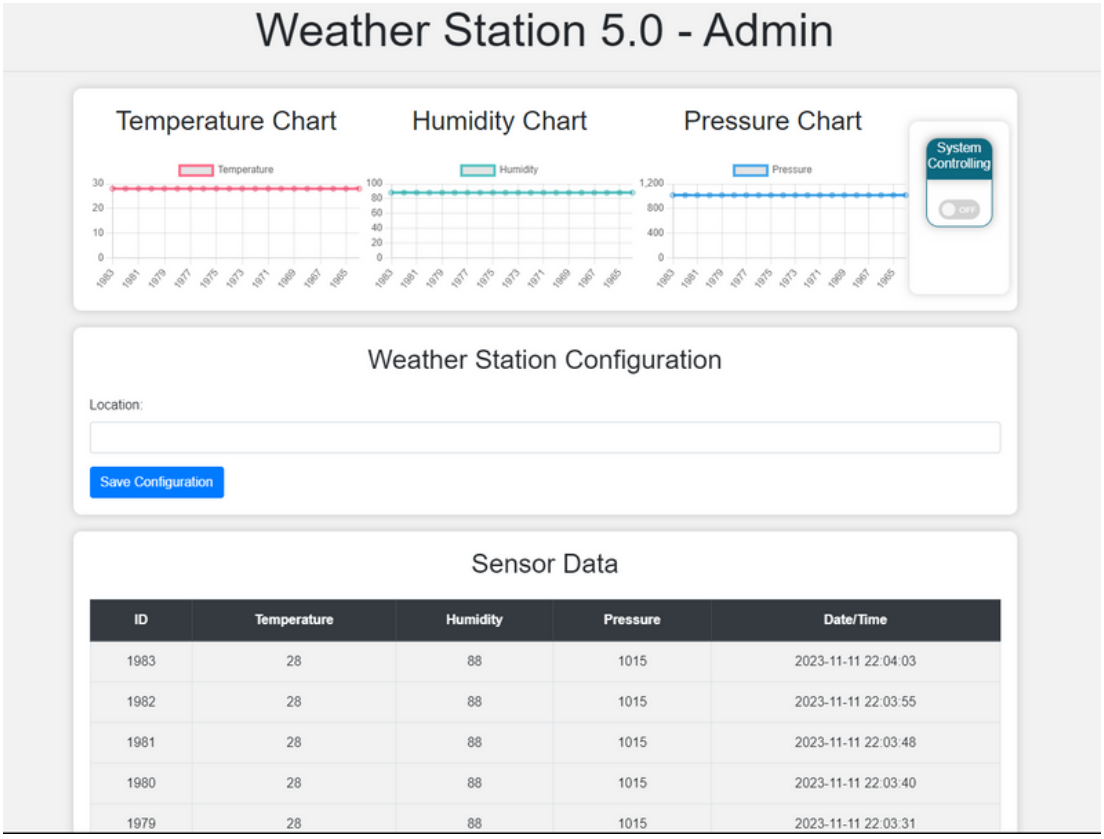
Structure of Prototype

Circuit Prototype





Web-Admin page Prototype



Android App Prototype

Budget

	Component	Cost
1	NodeMCU ESP-32S	LKR 1,500.00
2	DHT11 Temperature and Relative Humidity Sensor	LKR 290.00
3	BMP280 4-pin 5V Digital Barometric Pressure Sensor I2C	LKR 340.00
4	400TP Project Board Breadboard Solderless	LKR 150.00
5	LED * 3	LKR 60.00
6	Resistor - 330ohm * 3	LKR 180.00
7	Connecting Wires	LKR 200.00
	Toatal Cost:	LKR 2,720.00

***NOTE**

This is a prototype and costless version. So that power for circuit setup is given by potable power-bank. But future implementation this setup must be add Battery pack or solar power solution.It can be more cost than this prototype.

Issues & Solutions

- When establish connection between NodeMCU and Laptop , there was a problem in connection because Laptop haven't physical COM port for connection. To solve this issue , USB to UART bridge making service software was used.
- For the purpose optimize the arduino code , library (WeatherStation_V2) was made but it has some errors , I couldn't able to fix the errors.(Library file can be found in code folder)
- To make fully-functional weather station , this setup want another sensors (wind , rain , etc..) But for this prototype version I used temperature , humidity and pressure sensors only .
- To make rain prediction ,application use historical data-sets and machine learning model. It couldn't archive because of lacking of data-sets and possible machine learning model.

Future Implementation

- Add Solar panel and battery pack for power supply.
- Add other weather detecting component (wind , precipitation).
- Made Admin web page with more options.
- Update Mobile application.

Conclution

- the implementation of the simple weather station project has proven to be a valuable endeavor in understanding and monitoring local weather conditions. Through the integration of various sensors and data collection techniques, we have successfully gathered real-time information on key meteorological parameters such as temperature, humidity, and atmospheric pressure.

- The project aimed to provide a cost-effective and accessible solution for individuals or small communities interested in tracking weather patterns at a local level. The results indicate that our weather station is capable of delivering reliable and accurate data, contributing to a better understanding of the surrounding environment.)

- Furthermore, the user-friendly interface and data visualization tools implemented in the project enhance the accessibility of weather information for a broader audience. This contributes to raising awareness about the importance of monitoring weather conditions for various applications, including agriculture, urban planning, and environmental research.

Discussion

- Accuracy and Reliability:

The project successfully demonstrated the accuracy and reliability of the sensor data. However, ongoing calibration and maintenance should be considered to ensure the continued precision of the weather station over time.

- Data Interpretation:

The interpretation of collected data reveals interesting patterns and trends in local weather. Exploring these patterns can provide valuable insights into climate variations, aiding in decision-making processes for various sectors.