LIVE WEATHER STATION

A PROJECT REPORT

Submitted by

MADESH G	**211420105057**
SANJAI S	**211420105088**
SANJAY KUMAR M	**211420105092**
SANJAY RAJAN J	**211420105093**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL AND ELECTRONICS ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"LIVE WEATHER STATION"** is the bonafide work of **"MADESH G (211420105057), SANJAI S (211420105088), SANJAY KUMAR M (211420105092), SANJAY RAJAN J (211420105093),"** who carried out the project work under my supervision.

SIGNATURE

Dr. S.SELVI, M.E, Ph.D.
HEAD OF THE DEPARTMENT
PROFESSOR
Department of Electrical and
Electronics Engineering,
Panimalar Engineering College,
Chennai-600 123

SIGNATURE

MR.G.PONKUMAR ,
SUPERVISOR
ASSISTANT PROFESSOR
Department of Electrical and
Electronics Engineering,
Panimalar Engineering College,
Chennai-600 123

Submitted for End Semester Project Viva Voce held on .....at Panimalar Engineering College, Chennai.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our respected Chairman and Chancellor, Sathyabama University **Dr. JEPPIAR M.A., B.L., Ph.D.,** for being in the vanguard of scientific progress in our college.

We would like to express our deep gratitude to our Beloved Secretary and Correspondent **Dr. P.CHINNADURAI M.A., Ph.D.,** for extending his never ending support to us.

We extend our sincere thanks to our Professor & Head of the Department, Electrical and Electronics Engineering, **Dr. S. SELVI M.E, Ph.D.** and a heartfelt thanks to our Professor **Dr. D. SILAS STEPHEN M.E, Ph.D.** for his timely guidance in technical front and for instilling immense confidence in us for completing our project successfully.

We are thankful and forever indebted to our Project Supervisor , **MR.K.KIRUBAKARAN, M.E** Assistant Professor for his insightful feedback and prompt assistance in completion our project.

We also extend our thanks to **All Staff Members** of Electrical and Electronics Engineering for their support and technical assistance. On a personal note, we would like to express our heartfelt thanks to our beloved **Parents** and our Friends, for their help and wishes in successfully completing this project. Thanks to **Almighty** for giving us the strength to do this project successfully.

ABSTRACT

The online weather station using ESP32 and BMP280 is a system that provides real-time weather data by utilizing the power of the internet and wireless technology. The system consists of an ESP32 microcontroller and a BMP280 sensor, which collects data on temperature, humidity, and atmospheric pressure. The data is then transmitted wirelessly through Wi-Fi to an online platform, where it can be accessed and analyzed by the user. The project allows for easy and efficient monitoring of weather conditions, with potential applications in agriculture, meteorology, and environmental monitoring.

The ESP32 microcontroller is a low-cost, low-power device that provides Wi-Fi connectivity and processing capabilities, making it an ideal choice for the online weather station. The BMP280 sensor is a highly accurate and reliable sensor that provides precise measurements of temperature, humidity, and atmospheric pressure.

The online platform allows the user to access the weather data from anywhere in the world using a web browser or mobile application. The user can view real-time weather data, historical data, and trends, and receive alerts when weather conditions change significantly.

Overall, the online weather station using ESP32 and BMP280 provides an affordable and accessible solution for weather monitoring, allowing individuals and organizations to make informed decisions based on accurate and up-to-date weather data.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The online weather station using ESP32 and BMP280 is a system that uses the internet and wireless technology to offer real-time weather data. An ESP32 microcontroller and a BMP280 sensor, which measures temperature, humidity, and atmospheric pressure, make up the system. The information is then wirelessly transferred over Wi-Fi to an internet platform where the user can access and analyze it. The project enables quick and effective weather monitoring, with potential uses in meteorology, agriculture, and environmental monitoring. The ESP32 microcontroller is a low-cost, low-power component that offers processing capability and Wi-Fi connection, making it the perfect option for the online weather station. The BMP280 sensor, which delivers accurate readings of temperature, humidity, and air pressure, is a very accurate and dependable sensor. Using a web browser or a mobile application, the user can access the weather data on the internet platform from any location in the world. The user can browse historical data, trends, and real-time weather information, as well as get alerts when the weather substantially changes. Overall, the ESP32 and BMP280 online weather station offers a cost-effective and convenient solution for weather monitoring, enabling people and organizations to make wise decisions based on precise and current weather data.

WEATHER STATION

A weather station is a facility, either on land or sea, with instruments and equipment for measuring atmospheric conditions to provide information for weather forecasts and to study the weather and climate. The measurements taken include temperature, atmospheric pressure, humidity, wind speed, wind direction, and precipitation amounts. Wind measurements are taken with as few other obstructions as possible, while temperature and humidity measurements are kept free from direct solar radiation, or insolation. Manual observations are taken at least once daily, while automated measurements are taken at least once an hour. Weather conditions out at sea are taken by ships and buoys, which measure slightly different meteorological quantities such as sea surface temperature (SST), wave height, and wave period. Drifting weather buoys outnumber their moored versions by a significant amount.

TYPES OF WEATHER STATION SENSORS

Measuring and analyzing the weather has always been relevant to scientists and farmers, but in recent years, understanding what's going on with the weather has become a greater area of concern around the globe. In order to address emerging weather challenges, we need devices that can capture data to inform our planning in a variety of settings and contexts

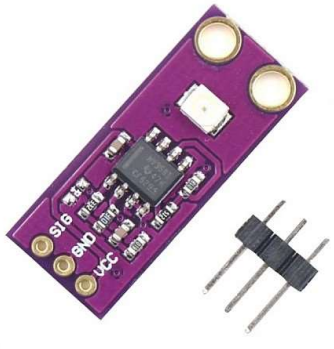
- Temperature & humidity sensors

- Road weather sensors

- Wind speed & quality sensor

- Water quality, level, & flow sensors

- Sunlight & UV sensors



MEASURING PARAMETERS

TEMPERATURE

Temperature is a physical quantity that expresses quantitatively the perceptions of hotness and coldness. Temperature is measured with a thermometer.

Thermometers are calibrated in various temperature scales that historically have relied on various reference points and thermometric substances for definition. The most common scales are the Celsius scale with the unit symbol °C (formerly called *centigrade*), the Fahrenheit scale (°F), and the Kelvin scale (K), the latter being used predominantly for scientific purposes. The kelvin is one of the seven base units in the International System of Units (SI).

PRESSURE

Pressure (symbol: *p* or *P*) is the force applied perpendicular to the surface of an object per unit area over which that force is distributed.

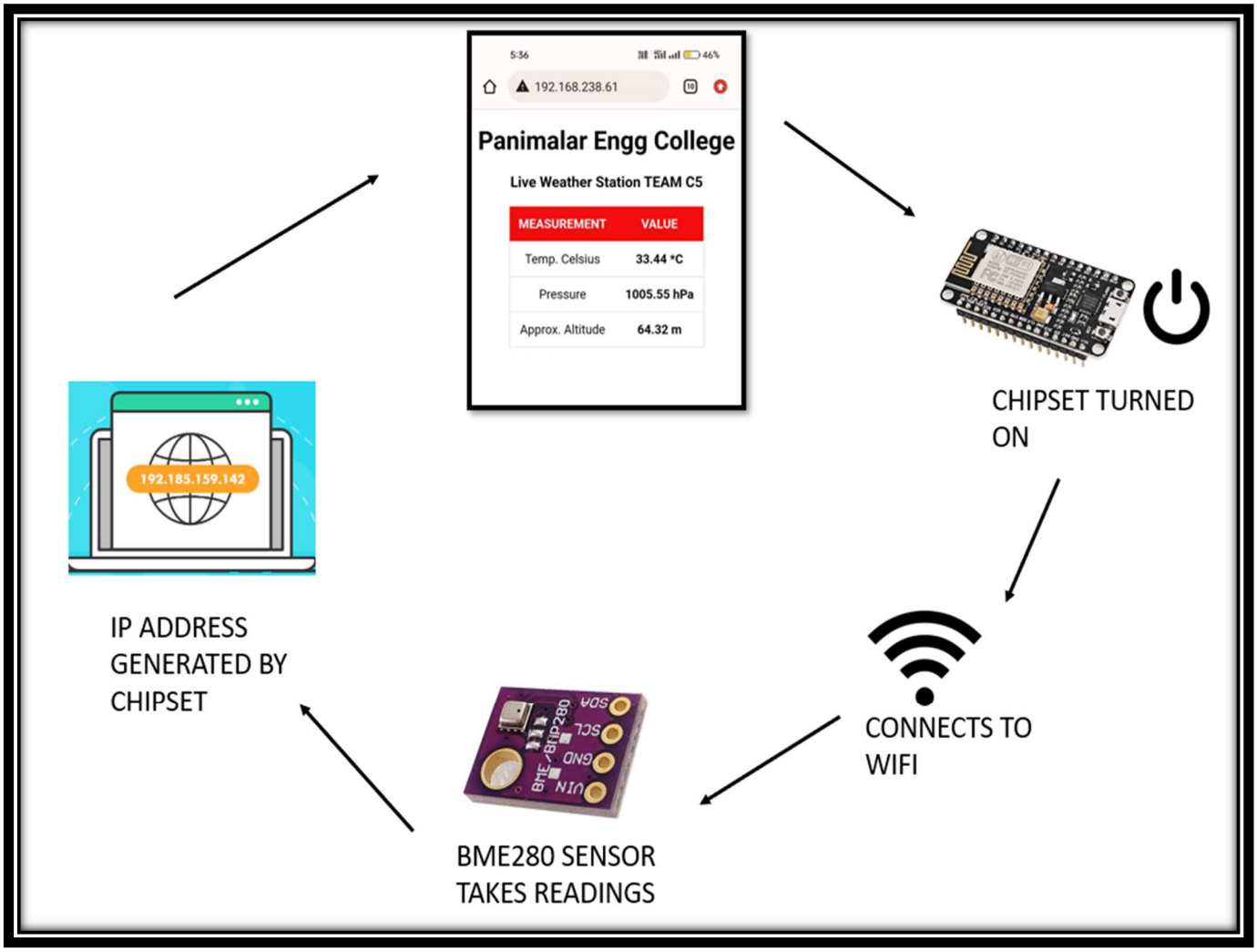
Gauge pressure (also spelled *gage* pressure)[a] is the pressure relative to the ambient pressure. Various units are used to express pressure. Some of these derive from a unit of force divided by a unit of area; the SI unit of pressure, the pascal (Pa), for example, is one newton per square metre (N/m^2); similarly, the pound-force per square inch (psi,

symbol lbf/in^2) is the traditional unit of pressure in the imperial and US customary systems. Pressure may also be expressed in terms of standard atmospheric pressure; the atmosphere (atm) is equal to this pressure, and the torr is defined as $^1\!/_{760}$ of this. Manometric units such as the centimetre of water, millimetre of mercury, and inch of mercury are used to express pressures in terms of the height of column of a particular fluid in a manometer.

ALTITUDE

Altitude or height is a distance measurement, usually in the vertical or "up" direction, between a reference datum and a point or object. The exact definition and reference datum varies according to the context (e.g., aviation, geometry, geographical survey, sport, or atmospheric pressure). Although the term *altitude* is commonly used to mean the height above sea level of a location, in geography the term elevation is often preferred for this usage. Vertical distance measurements in the "down" direction are commonly referred to as *depth*.

BASIC OVERVIEW



EXISTING METHOD

- Traditionally, weather stations were large and expensive systems that were only accessible to professionals or institutions. These systems required a considerable investment in both equipment and installation, making them out of reach for most individuals or small businesses. Moreover, the data collected by these systems was not easily accessible, limiting its usefulness to a select few.

- Most of the physical weather monitoring devices are ideally placed in one place and the parameters info cannot be viewed through remotely via other gadgets.

- Small weather monitoring devices may have limited data transmission capabilities, which can limit their usefulness for real-time monitoring or remote access.

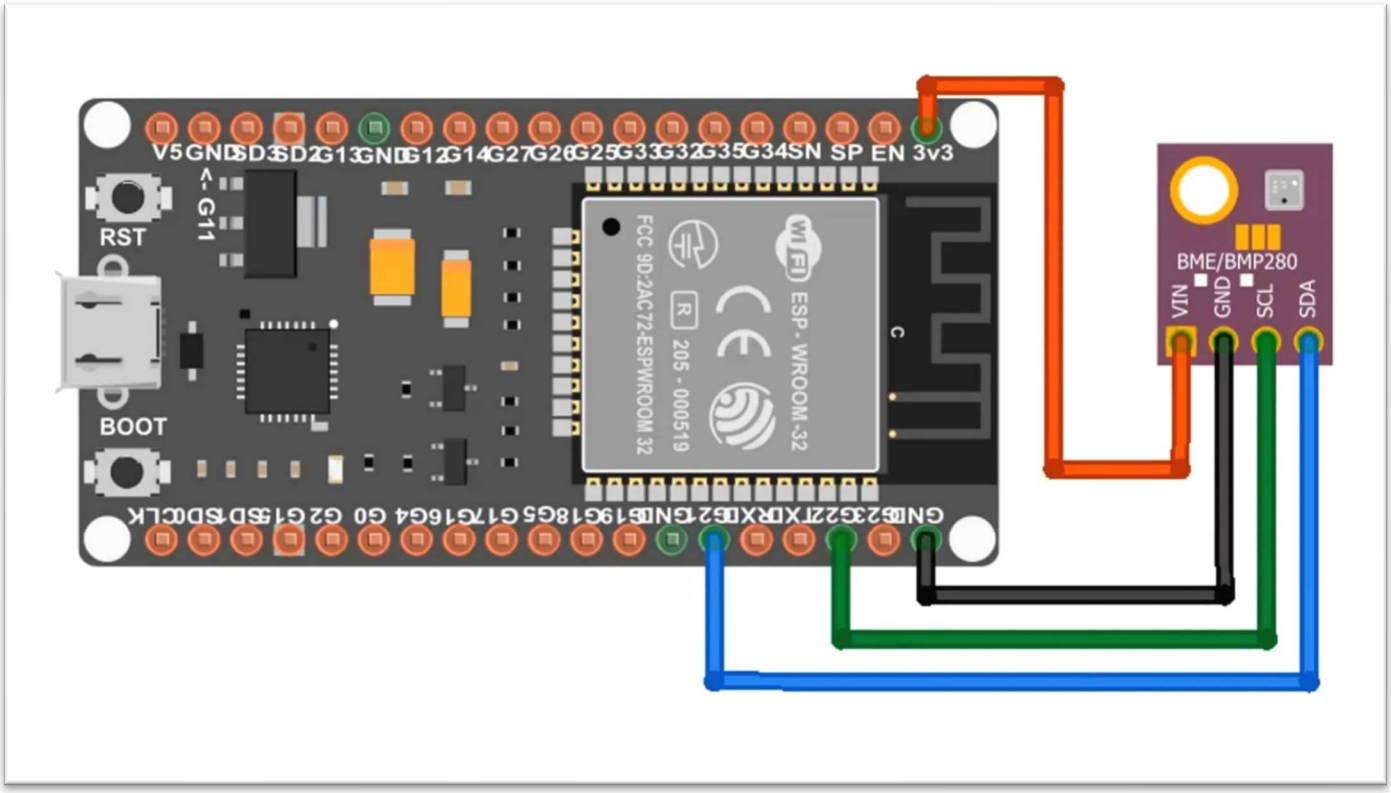
PROPOSED METHOD

- The proposed method involves using low-cost and easy-to-install sensors such as the BME280 sensor, which can measure various weather parameters such as temperature, humidity, and air pressure.
- These sensors can be integrated with a microcontroller board such as the ESP32, which can transmit data to remote servers in real-time using Wi-Fi or cellular networks.
- This data can be stored and processed using cloud computing, making it easily accessible to a wider audience through web-based interfaces or mobile applications.
- The proposed method is cost-effective, easy to install, and can provide accurate and real-time weather information to individuals, small businesses, and institutions.
- This can help in making informed decisions based on the weather conditions, improving our understanding of weather patterns, and better preparing for the future.

CONNECTION DIAGRAM

LIVE WEATHER STATION USING

ESP32 AND BMP280



HARDWARE COMPONENTS

ESP32 MICROCONTROLLER

ESP32 is the SoC (System on Chip) microcontroller which has gained massive popularity recently. Whether the popularity of ESP32 grew because of the growth of IoT or whether IoT grew because of the introduction of ESP32 is debatable. If you know 10 people who have been part of the firmware development for any IoT device, chances are that 7−8 of them would have worked on ESP32 at some point. So what is the hype all about? Why has ESP32 become so popular so quickly? Let's find out.

Before we delve into the actual reasons for the popularity of ESP32, let's take a look at some of its important specifications. The specs listed below belong to the ESP32 WROOM 32 variant.−

- Integrated Crystal− 40 MHz
- Module Interfaces− UART, SPI, I2C, PWM, ADC, DAC, GPIO, pulse counter, capacitive touch sensor
- Integrated SPI flash− 4 MB
- ROM− 448 KB (for booting and core functions)
- SRAM− 520 KB
- Integrated Connectivity Protocols− WiFi, Bluetooth, BLE
- On−chip sensor− Hall sensor
- Operating temperature range− −40 − 85 degrees Celsius
- Operating Voltage− 3.3V
- Operating Current− 80 mA (average)

With the above specifications in front of you, it is very easy to decipher the reasons for ESP32's popularity. Consider the requirements an IoT device would have from its microcontroller (μC). If you've gone through the previous chapter, you'd have realized that the major operational blocks of any IoT device are sensing, processing, storage, and transmitting. Therefore, to begin with, the μC should be able to interface with a variety of sensors. It should support all the common communication protocols required for sensor interface: UART, I2C, SPI. It should have ADC and pulse counting capabilities. ESP32

fulfills all of these requirements. On top of that, it also can interface with capacitive touch sensors. Therefore, most common sensors can interface seamlessly with ESP32.

Secondly, the μC should be able to perform basic processing of the incoming sensor data, sometimes at high speeds, and have sufficient memory to store the data. ESP32 has a max operating frequency of 40 MHz, which is sufficiently high. It has two cores, allowing parallel processing, which is a further add-on. Finally, its 520 KB SRAM is sufficiently large for processing a large array of data onboard. Many popular processes and transforms, like FFT, peak detection, RMS calculation, etc. can be performed onboard ESP32. On the storage front, ESP32 goes a step ahead of the conventional microcontrollers and provides a file system within the flash. Out of the 4 MB of onboard flash, by default, 1.5 MB is reserved as SPIFFS (SPI Flash File System). Think of it as a mini−SD Card that lies within the chip itself. You can not only store data, but also text files, images, HTML and CSS files, and a lot more within SPIFFS. People have displayed beautiful Webpages on WiFi servers created using ESP32, by storing HTML files within SPIFFS.

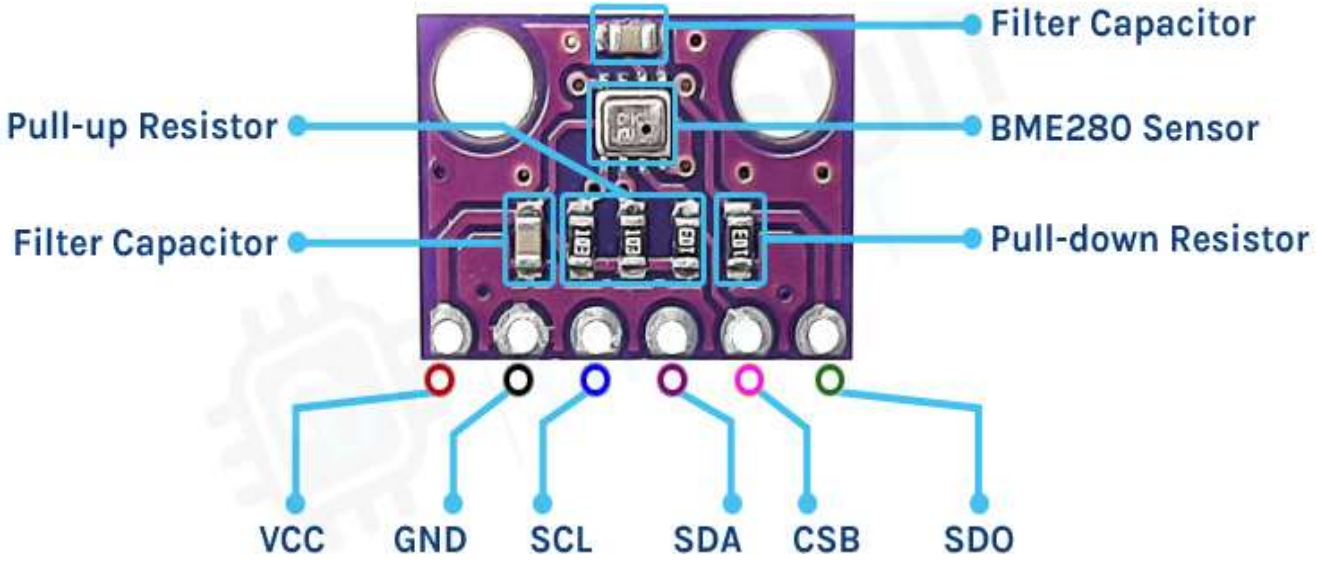
Finally, for transmitting data, ESP32 has integrated WiFi and Bluetooth stacks, which have proven to be a game-changer. No need to connect a separate module (like a GSM module or an LTE module) for testing cloud communication. Just have the ESP32 board and a running WiFi, and you can get started. ESP32 allows you to use WiFi in Access Point as well as Station Mode. While it supports TCP/IP, HTTP, MQTT, and other traditional communication protocols, it also supports HTTPS. Yep, you heard that right. It has a crypto−core or a crypto-accelerator, a dedicated piece of hardware whose job is to

accelerate the encryption process. So you cannot only communicate with your web server, you can do so securely. BLE support is also critical for several applications. Of course, you can interface LTE or GSM or LoRa modules with ESP32. Therefore, on the 'transmitting data' front as well, ESP32 exceeds expectations.

With so many features, ESP32 would be costing a fortune, right? That's the best part. ESP32 dev modules cost in the ballpark of ₹ 500. Not only that, the chip dimensions are quite small (25 mm x 18 mm, including the antenna area), allowing its use in devices requiring a very small form factor.

Finally, ESP32 can be programmed using the Arduino IDE, making the learning curve much less steep. Isn't that great? Are you excited to get your hands dirty with ESP32? Then let's start by installing the ESP32 board in the Arduino IDE in the next chapter. See you there.

BMP280 SENSOR



This BMP280 is a cheapest and tiny Atmospheric Sensor Breakout to measure barometric pressure, and temperature readings all without taking up too much space. Basically, anything you need to know about atmospheric conditions you can find out from this tiny breakout. The BMP280 Breakout has been design to be used in indoor/outdoor navigation, weather forecasting, home automation, and even personal health and wellness monitoring.

This module uses an environmental sensor manufactured by Bosch with temperature, barometric pressure sensor that is the next generation upgrade to the popular BMP085/BMP180/BMP183 Sensor. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI! This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute accuraccy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter accuracy.

The BMP280 is the next-generation of sensors, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C or SPI. For simple easy wiring, go with I2C. If you want to connect a bunch of sensors without worrying about I2C address collisions, go with SPI.

Features:-

- Get more precise temperature, atmospheric pressure values, and approximate altitude data
- Enhancement of GPS navigation (e.g. time-tofirst-fix improvement, dead-reckoning, slope detection)
- Indoor navigation (floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Health care applications (e.g. spirometry)
- Vertical velocity indication (e.g. rise/sink speed)
- Libraries and tutorials available for Arduino and other micro controllers
- Operating Voltage: 1.71V to 3.6V – would typically be operated from 3.3V
- Operating Temperature: -40 to +85 deg. Celsius (full accuracy between 0 and +65 deg. C)
- Operating Pressure: 300 hPa to 1100 hPa
- Peak current: 1.12mA
- Accuracy between 700 to 900hPa and 25 to 40 deg. C: ±0.12hPa and ±1.0m

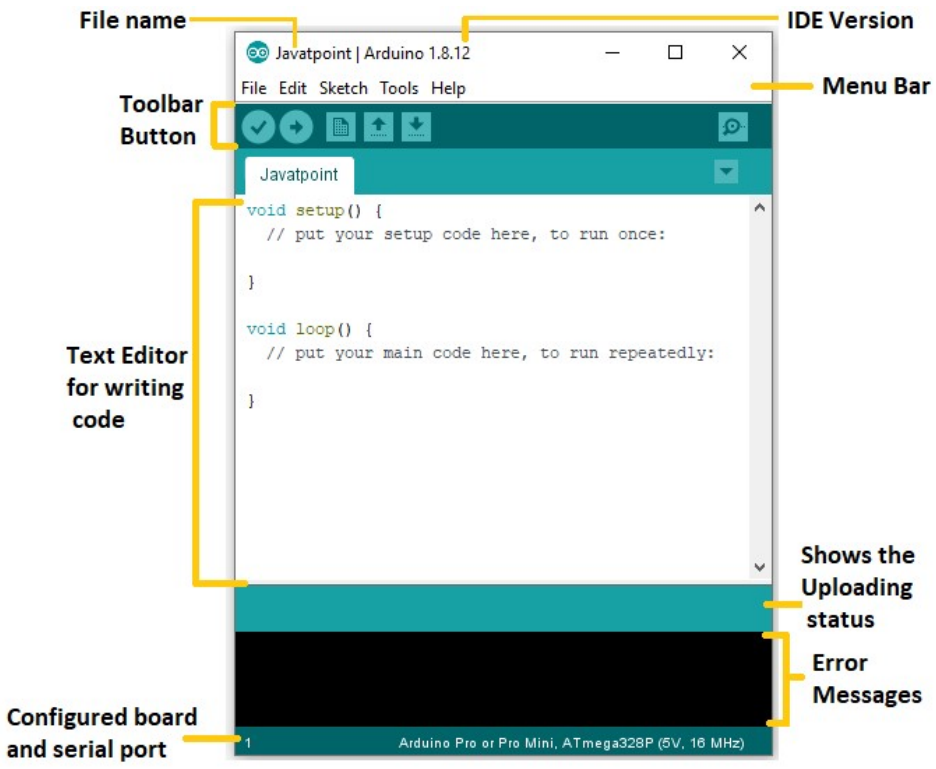
SOFTWARE

ARDUINO IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Arduino IDE will appear as:



VISUAL STUDIO CODE EDITOR

Visual Studio Code is a free, lightweight but powerful source code editor that runs on your desktop and on the web and is available for Windows, macOS, Linux, and Raspberry Pi OS. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages (such as C++, C#, Java, Python, PHP, and Go), runtimes (such as .NET and Unity), environments (such as Docker and Kubernetes), and clouds (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

Aside from the whole idea of being lightweight and starting quickly, Visual Studio Code has IntelliSense code completion for variables, methods, and imported modules;

graphical debugging; linting, multi-cursor editing, parameter hints, and other powerful editing features; snazzy code navigation and refactoring; and built-in source code control including Git support. Much of this was adapted from Visual Studio technology.

Visual Studio Code proper is built using the Electron shell, Node.js, TypeScript, and the Language Server Protocol, and is updated on a monthly basis. The many extensions are updated as often as needed. The richness of support varies across the different programming languages and their extensions, ranging from simple syntax highlighting and bracket matching to debugging and refactoring. You can add basic support for your favorite language through TextMate colorizers if no language server is available.

LANGUAGE USED

HTML

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting

structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> and </**p**> surround and provide information about document text and may include sub-element tags. Browsers do not display the HTML tags but use them to interpret the content of the page.

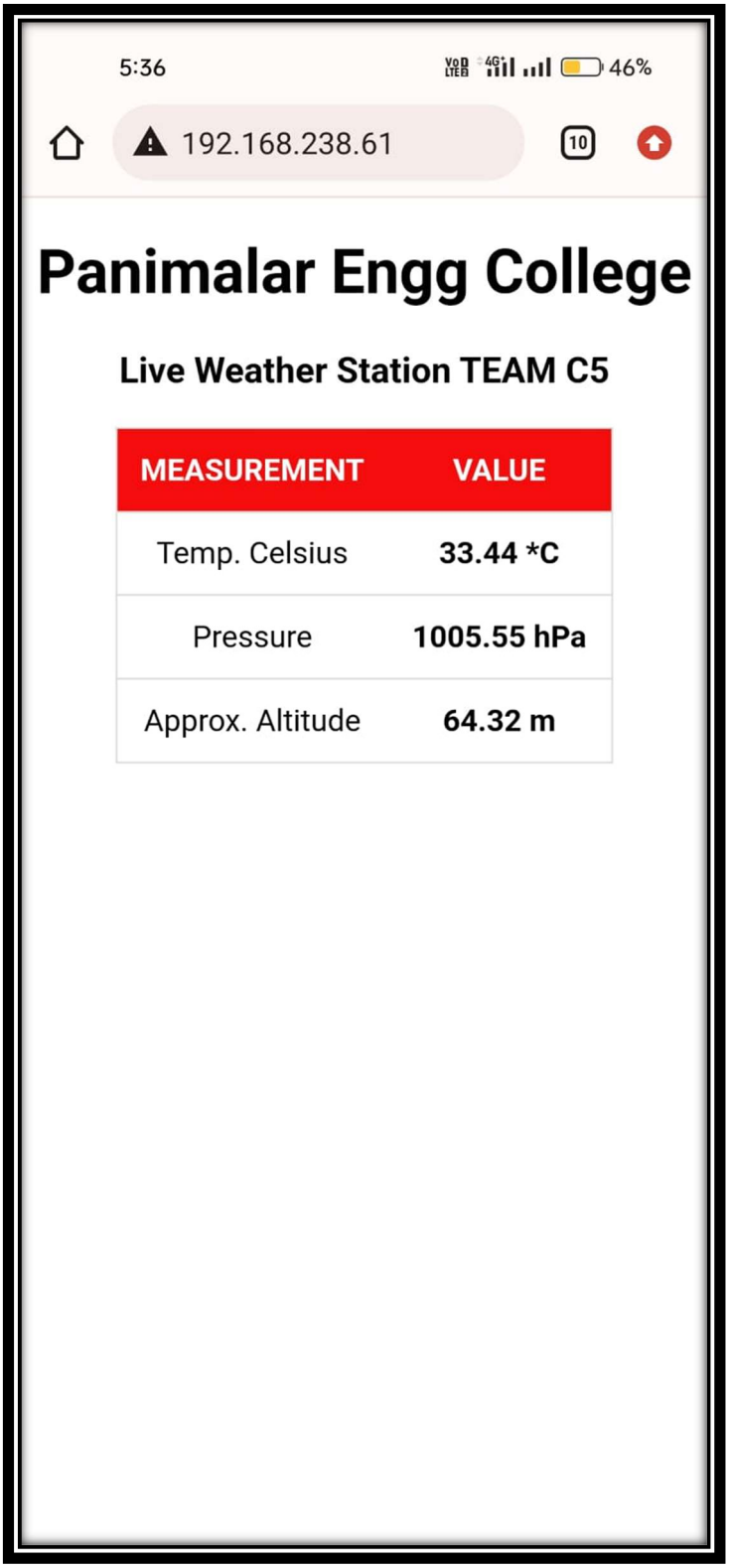
HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the <**canvas**> element, together with JavaScript.

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Tim Berners-Lee in late 1991. It describes 18 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house Standard Generalized Markup Language (SGML)-based documentation format at CERN. Eleven of these elements still exist in HTML 4.

HTML is a markup language that web browsers use to interpret and compose text, images, and other material into visible or audible web pages. Default characteristics for

every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS. Many of the text elements are mentioned in the 1988 ISO technical report TR 9537 *Techniques for using SGML*, which describes the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system. These formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with separate structure and markup. HTML has been progressively moved in this direction with CSS.

OUR WEBPAGE DESIGN



CODE

```
// Load Wi-Fi library

#include <WiFi.h>

#include <Wire.h>

#include <Adafruit_BMP280.h>

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BMP280 bmp; // I2C

// Replace with your network credentials

const char* ssid     = "Sanjay X7";

const char* password = "spiderman";

// Set web server port number to 80

WiFiServer server(80);

// Variable to store the HTTP request

String header;

// Current time

unsigned long currentTime = millis();
```

```
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
void setup() {
  Serial.begin(115200);
  bool status;
// default settings
  // (you can also pass in a Wire library object like &Wire2)
  //status = bmp.begin();
  if (!bmp.begin(0x76)) {
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
    while (1);
  }
// Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients
if (client) {                             // If a new client connects,
    currentTime = millis();
```

```
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from the
client
    while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop
while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
```

```
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the table
            client.println("<style>body { text-align: center; font-family: Helvetica}");
            client.println("table { border-collapse: collapse; margin-left:auto; margin-
right:auto; }");
            client.println("th { padding: 12px; background-color: #f50c0c; color: white; }");
            client.println("tr { border: 1px solid #ddd; padding: 12px; }");
            client.println("tr:hover { background-color: #ffe4e3; }");
```

```
        client.println("td { border: none; padding: 12px; }");
        client.println(".sensor { color:black; font-weight: bold; padding: 1px; }");
// Web Page Heading
        client.println("</style></head><body><h1>Panimalar Engg College</h1>");
        client.println("</style></head><body><h3>Live Weather Station TEAM
C5</h3>");
        client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");
        client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");
        client.println(bmp.readTemperature());
        client.println(" *C</span></td></tr>");
        client.println("<tr><td>Pressure</td><td><span class=\"sensor\">");
        client.println(bmp.readPressure() / 100.0F);
        client.println(" hPa</span></td></tr>");
        client.println("<tr><td>Approx. Altitude</td><td><span class=\"sensor\">");
        client.println(bmp.readAltitude(SEALEVELPRESSURE_HPA));
        client.println(" m</span></td></tr>");
        client.println("</body></html>");
```

```
        // The HTTP response ends with another blank line
        client.println();
        // Break out of the while loop
        break;
      } else { // if you got a newline, then clear currentLine
        currentLine = "";
      }
    } else if (c != '\r') {  // if you got anything else but a carriage return character,
      currentLine += c;      // add it to the end of the currentLine
    }
  }
}
// Clear the header variable
header = "";
// Close the connection
client.stop();
```

```
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

CONCLUSION

In conclusion, the use of online weather stations using ESP32 and BME280 sensors has revolutionized the field of weather monitoring. The traditional method of using expensive and complex weather stations is no longer necessary, thanks to the development of low-cost and easy-to-install sensors that can be integrated with microcontroller boards such as the ESP32. The proposed method is cost-effective, easy to install, and can provide accurate and real-time weather information to individuals, small businesses, and institutions. The use of cloud computing to store and process data allows for easy access to weather information from anywhere in the world through web-based interfaces or mobile applications. The proposed method is highly customizable and can be adapted to suit specific needs, making it a versatile solution for weather monitoring. Overall, the online weather station using ESP32 and BME280 sensors is an important advancement in the field of weather monitoring that can help improve our understanding of weather patterns and prepare us for the future.

REFERENCE

1. IOT is a revolutionary approach

   https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0268-2

2. https://ieeexplore.ieee.org/document/7945539

3. https://www.researchgate.net/publication/320532203_Internet_of_Things_IoT_Definitions_Challenges_and_Recent_Research_Directions

4. https://projecthub.arduino.cc/woutvdr/9dd87f80-4b0e-483a-8bdc-3681868441cd

5. ESP32 Chipset IOT USAGE

   http://esp32.net/