IOT BASED SMART WEATHER SYSTEM USING ESP32

ABSTRACT:

- The IoT-based Smart Weather System using ESP32 is a comprehensive project aimed at harnessing the power of Internet of Things (IoT) technology to provide real-time weather data monitoring and visualization.
- In an era where climate information is of paramount importance, this system integrates a range of sensors, including temperature, humidity, and pressure sensors, with the versatile ESP32 microcontroller.
- This project serves as an accessible and cost-effective solution for weather data collection, processing, and dissemination.
- The ESP32, known for its efficiency and connectivity options, acts as the central processing unit and communication hub, collecting data from sensors, processing it, and transmitting it to a cloud server for storage and visualization.
- The system's architecture ensures robustness and reliability, even in remote or inaccessible locations, enabling users to access up-to-date weather information via web or mobile interfaces.
- Through this project, we aim to contribute to the growing field of IoT applications for environmental monitoring and encourage further innovation in the realm of weather-related solutions.

- The Internet-of-Things (IoT) concept has been largely integrated into heterogeneous systems and it has recently become more relevant to practical implementations because of the growth of ubiquitous communication and remote data access techniques.
- This paper presents an IoT system that basically gathers weather-related data information from a forecast API (Application Programming Interface) for a specific configurable location and displays them on TFT touch screen.
- The implemented system uses a low-cost ESP32 microcontroller and exploits its integrated Wi-Fi to achieve transmitting and receiving communications.
- Moreover, we develop an embedded web server on ESP32 using SPIFFS (Serial Peripheral Interface Flash File System) to monitor and control power consumption mode over a local network and a secured internet connection via a laptop, cellphone, or tablet.
- These functionalities are implemented by using FreeRTOS that provides multitasking support for the ESP32 dual-core microcontroller.

INTRODUCTION:

- In an age marked by rapid technological advancements and an increasing need for real-time environmental data, the integration of Internet of Things (IoT) technology has become pivotal in the domain of weather monitoring.
- Weather-related information influences numerous aspects of our daily lives, from agriculture and transportation to disaster management and everyday decision-making.
- To address this demand for accurate and accessible weather data, our project introduces an innovative solution: the IoT-based Smart Weather System using the ESP32 microcontroller.
- Traditional weather monitoring systems often rely on costly and centralized infrastructure, limiting their reach and accessibility.
- In contrast, our project leverages the capabilities of the ESP32, a versatile and cost-effective microcontroller renowned for its capabilities in IoT applications. By integrating a suite of sensors—measuring parameters such as temperature, humidity, and atmospheric pressure—we create a dynamic and comprehensive weather data collection system.

- The motivation behind this project is twofold.
- Firstly, it addresses the increasing importance of localized and real-time weather information.
- Accurate weather data is invaluable for industries like agriculture, transportation, tourism, and disaster management.
- Secondly, the project serves as an exemplar of how IoT technology can democratize data collection and distribution, making such critical information widely accessible.
- The primary objectives of the IoT-based Smart Weather System are as follows:
- **Data Collection**: Implement a network of sensors to collect essential weather data, including temperature, humidity, and atmospheric pressure.
- **Data Processing**: Develop algorithms to process raw sensor data, ensuring accuracy and reliability.
- **Data Transmission**: Establish a robust communication system to transmit processed weather data to a cloud-based server for storage and analysis.
- **Data Visualization**: Create user-friendly interfaces, including web and mobile applications, to visualize weather data in real-time.
- **Cost-effectiveness**: Design the system to be cost-effective, ensuring its viability for deployment in various settings.

- In this project, we will learn how to create a weather station, which will display reading from a BME280 module and live weather data from OpenWeatherMap API in a webserver.
- The device will get temperature, humidity, barometric pressure, and altitude from the BME280 sensor and external temperature, humidity, weather condition, and sunrise and sunset from OpenWeatherMap API.
- We can see those reading in a web browser
- This project's scope encompasses the design, development, and implementation of a self-contained, IoT-based smart weather monitoring system.
- It includes both the hardware and software components required for data collection, processing, transmission, and visualization.
- While the project focuses on a basic set of weather parameters, it can serve as a foundation for future enhancements and applications.
- The IoT-based Smart Weather System not only contributes to the field of weather monitoring but also highlights the potential of IoT technology to revolutionize data collection in various domains.
- By enabling easy access to localized weather information, it empowers individuals and industries to make informed decisions that can have a significant impact on safety, efficiency, and productivity.
- In the subsequent sections of this report, we delve into the project's components, methodology, implementation, results, and future possibilities, offering a comprehensive exploration of our innovative solution for weather data monitoring and analysis.

ESP32 MICROCONTROLLER:

- ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth.
- The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules.
- ESP32 is created and developed by Espressif Systems,a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.
- It is a successor to the ESP8266 Microcontroller.
- The ESP32 microcontroller is a highly versatile and popular microcontroller developed by Espressif Systems.
- It is well-known for its powerful processing capabilities, built-in Wi-Fi and Bluetooth connectivity, and a wide range of GPIO (General Purpose Input/Output) pins.
- The ESP32's combination of processing power, wireless connectivity, and extensive peripheral support has made it a go-to choice for IoT projects and a wide range of embedded systems applications.
- Its open-source nature and affordability have contributed to its popularity in the maker and development communities.

BME280 Temperature, Humidity & Pressure Sensor:

- The BME280 is an integrated environmental sensor developed specifically for mobile applications and wearables where size and low power consumption are key design parameters.
- The unit combines high linearity and high accuracy sensors and is perfectly feasible for low current consumption and long-term stability.
- The BME280 sensor offers an extremely fast response time and therefore supports performance requirements for emerging applications such as context awareness, and high accuracy over a wide temperature range.
- This sensor can measure relative humidity from 0 to 100% with ±3% accuracy, barometric pressure from 300Pa to 1100 hPa with ±1 hPa absolute accuracy, and temperature from -40°C to 85°C with ±1.0°C accuracy.
- The BME280 supports I2C and SPI interfaces.
- The module we are using supports a voltage range of 1.7 – 3.3V.
- The BME280 has an average current consumption of 3.6µA when measuring all the parameters at a refresh rate of 1Hz.
- The sensor can be operated in three power modes: Sleep, normal, and forced mode.

- One thing to keep in mind while buying a BME280 module is that a similar-looking module also comes with a BMP280 sensor which lacks the humidity measurement capability, which makes it way cheaper than the BME280.
- The best way to identify between BME280 and BMP280 sensors is to look at the product code.
- BME280 sensor will have a product code starting with ‘U’, e.g., UP, UN, etc.
- While the BMP280 will have a product code starting with “K”, e.g., KN, KP, etc.
- Another visual difference is that BME280 is somewhat square while BMP280 is rectangular
- The BME280 sensor is highly versatile and capable of providing essential environmental data for a wide range of applications.
- Its ability to measure temperature, humidity, and pressure in a compact and energy-efficient package makes it a popular choice among makers and engineers for various projects.

WEATHER MAP AI:

- A Weather Map AI (Artificial Intelligence) refers to the use of AI and machine learning techniques to analyze and visualize weather data on maps.
- It involves the integration of weather forecasting models, real-time sensor data, and historical weather information to provide accurate and actionable weather insights.
- OpenWeatherMap is an online service, that provides global weather data via API, including current weather data, forecasts, nowcasts, and historical weather data for any geographical location.
- Through their API we can access current weather data for any location on Earth including over 200,000 cities!
- They collect and process weather data from different sources such as global and local weather models, satellites, radars, and a vast network of weather stations.
- Data is available in JSON, XML, or HTML format. We will be using HTTP GET to request the JSON data for this project.

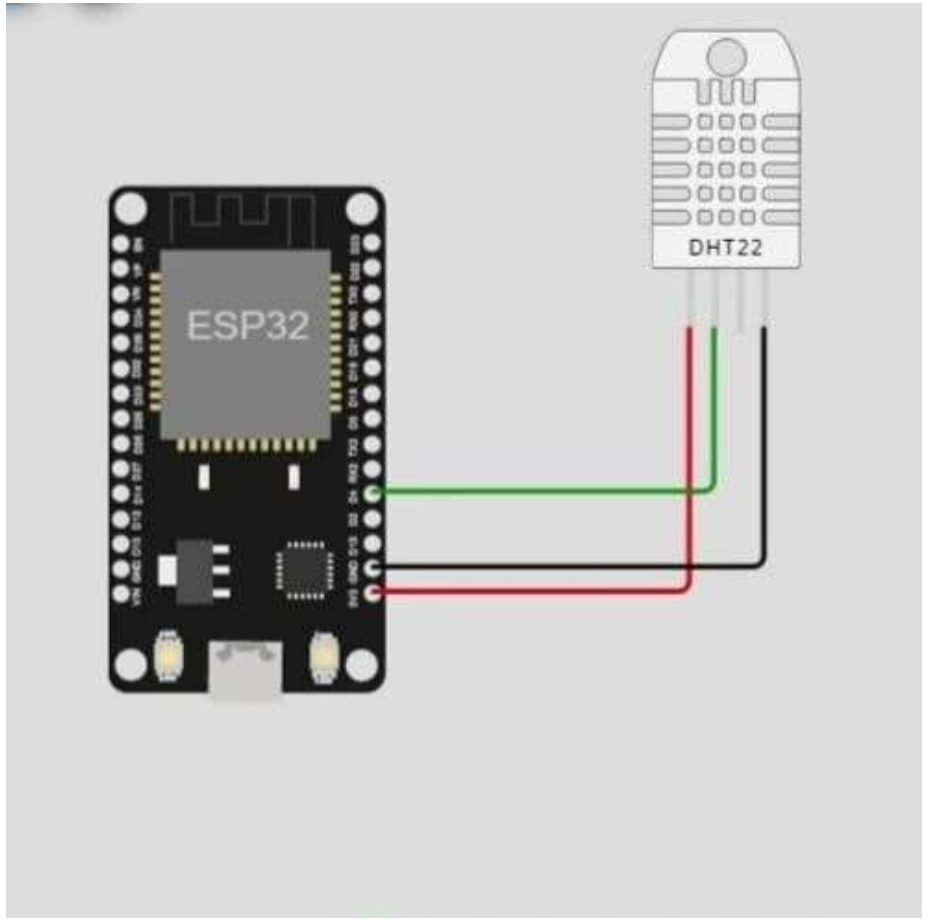
- Weather Map AI systems collect and integrate various sources of weather data, including satellite imagery, radar data, weather station reports, and atmospheric models.
- This data is then processed to generate accurate weather predictions.
- AI algorithms are used to analyze historical weather patterns and current conditions.
- Machine learning models can identify trends, anomalies, and correlations in the data, improving the accuracy of weather forecasts.
- Weather Map AI systems provide real-time updates on weather conditions, including temperature, precipitation, wind speed, humidity, and more.
- These updates are often displayed on interactive maps, making it easy for users to access the information they need.
- Weather Map AI can be used to monitor environmental conditions such as air quality, pollution levels, and climate change.
- It aids in understanding the impact of weather on the environment.
- Weather Map AI plays a crucial role in providing timely and accurate weather information for a wide range of applications, from everyday weather forecasts to critical decision-making in agriculture, transportation, emergency management, and more.
- The use of AI and machine learning continues to enhance the accuracy and usability of weather predictions and data visualization.

METHODOLOGY:

- This project will focus on development of the ThingSpeak an IoT platform that to show the data of the sensor.
- The method divided into two parts which are hardware and software development part.
- The hardware development involves the circuit construction and develops the prototype.
- Meanwhile, the software part involves the IoT coding, circuit schematic diagram, circuit simulation and data acquisition
- Selection of appropriate sensors, including the BME280 for temperature, humidity, and pressure measurements.
- Ensured compatibility with the ESP32 microcontroller.
- Calibration of the sensors to enhance data accuracy.
- Development of calibration procedures to adjust sensor readings.
- Implementation of data sampling routines to continuously collect sensor data.
- Selection of suitable sampling intervals for real-time monitoring.
- Application of filters and smoothing techniques to reduce noise in sensor data.
- Preprocessing to eliminate outliers and anomalies.

- Conversion of sensor data into engineering units (e.g., temperature in Celsius, humidity in percentage, and pressure in hPa).
- Data normalization and scaling.
- Selection of communication protocols for data transmission, including Wi-Fi connectivity.
- Configuration of ESP32 to connect to a local network.
- Integration with a cloud platform (e.g., AWS IoT, Azure IoT, or Google Cloud IoT) to securely transmit data.
- Implementation of secure data transmission protocols (e.g., HTTPS or MQTT).
- Development of a web-based user interface for real-time weather data visualization.
- Creation of user-friendly dashboards to display temperature, humidity, and pressure data.
- Development of a mobile application for remote monitoring and alerts.
- Compatibility with Android and iOS platforms.
- Rigorous testing of the ESP32 microcontroller and sensor components for functionality and accuracy.
- Verification of sensor calibration.
- Thorough testing of software modules, including data acquisition, processing, and transmission.
- Validation of data visualization on web and mobile interfaces.

- Installation of the weather monitoring system in the target location.
- Ensuring adequate power supply and environmental considerations.
- Final configuration of cloud-based data storage and analysis tools.
- Periodic analysis of collected weather data for trends and patterns.
- Generation of reports or alerts based on specific weather conditions.
- Establishing a maintenance schedule for sensor calibration and system checks.
- Updating firmware or software as needed.
- Identifying areas for improvement and expansion, such as additional sensors or features.
- This methodology guided the development and implementation of the IoT-based Smart Weather System using the ESP32, ensuring a systematic approach to creating a reliable and efficient weather monitoring solution.


ESP32
DHT22

CODE :

```
#include “WiFi.h”
WiFIClient client;
#include <DHTesp.h>
Const int DHT_PIN = 15;
Dhtesp dhtsensor;
String thingSpeakAddress = “api.thingspeak.com”;
Atring request_string;
Void setup()
{
Serial.begin(115200);
WiFi.disconnect();
WiFi.begin(“Wokwi-GUEST”, “”);
While(WiFi.status() != WL_CONNECTED)
{ Delay(300);
Serial.print(“,”);
}
```

```
Serial.println(“”);
Serial.println(“WiFi connected);
Serial.println(“IP address: “);
Serial.println(WiFi.localIP() );
dhtSensor.setup(DHT_PIN, DHTesp :: DHT22);
}
Void loop ( )
{
Delay(2000);
TempAndHumidity data = dhtSensor.getTempAndHumidity( );
 Float t = data.temperature;
 Float h = data.humidity;
 Kirin_thingspeak(t, h);
 If (isnan(h))          | |  isnan(t) ) {
 Serial.println(“Failed to read from DHT sensor”); Return;
 }
 }
```

```
Void kirim_thingspeak(float t, float hum)
{
If (client.connect(“api.thingspeak.com” , 80))
{ request_string = “/update?”;
request_string += “key=”;
request_string += “2RRDNPNLDSMWVUO6”;
request_string += “&”;
request_string += “field1”;
request_string += “=”;
request_string += t;
Serial.println(String(“GET”) + request_string + “HTTP/1.1\r\n” + “Host: “ + thingSpeakAddress + “\r\n + “Connection :
close\r\n\r\n”);
Client.println(String(“GET “) + request_string + “HTTP/1.1\r\n” + “Host: “ + thingSpeakAddress + “\r\n” + “Connection:
close\r\n\r\n”);
Delay(5000);
Unsigned long timeout = millis ( );
While (client.available ( ) == 0) {
If (millis ( ) = timeout > 5000) { Serial.println(“>>> Client Timeout !”); Client.stop ( );
Return;
}
}
```

```
 While (client.available ( ) )
{
 String line = client. readStringuntil (‘\r’);
 Serial.print(line);
 }
 Serial.println ( );
 Serial.println (“closing connection”);
 }
 }
```

CONCLUSION:

- As the conclusion this project have cleared the objective that to build a system that can monitored weather parameter by wireless system and IoT.
- The Sensor station and Weather station will be communicated by hotspot Wi-Fi and it is limited in areas covered but still better in communication via wireless.
- The value that been recorded from google sheet and Table 1, 2 and 3 it seen that the weather at particular place has different condition from the exact condition with the accuracy of weather reporting system and forecast system data has been compared.
- It says that weather reporting system is more accurate than forecast system.
- This weather reporting system will display the sensor data to ThingSpeak and IFTTT to save the data into google sheet.
- It also can be checked in Blynk app that can be installed in google play store or Appstore.