CLASS: B.E. E&TE **SUBJECT: ESRTOS**

EXPT. NO.: 7 **DATE:**

TITLE:

Build Weather Station by utilizing cloud-ready temperature and Humidity sensor (DHT-11/22) with the Node MCU and the any IoT Platform.

PROBLEM STATEMENT:

Write a program to Build Weather Station by utilizing cloud-ready temperature and Humidity sensor (DHT-11/22) with the Node MCU and the any IoT Platform..

OBJECTIVE:

- To study weather station by using DHT-11/22 and ESP32.
- To implement the weather station using an IoT-cloud platform (ThingSpeak).
- To monitor the data collected by the sensor on the platform.
- To study the wireless connectivity of ESP32 (Wi-Fi).

S/W AND H/W PACKAGES USED:

- ESP32 DOIT DEVKIT V1
- Arduino IDE
- MathWorks ThingSpeak

THEORY

Monitoring temperature and humidity in the surroundings has become very crucial for the modern-day humans. One of the everyday applications of this monitoring is weather forecasting which gives an estimate of the weather conditions throughout the day which is helpful in planning tasks to be done accordingly.

A simple weather station can be implemented with the help of an ESP and a DHT-11/22. ESP32 is the preferred microcontroller because it can connect with the internet easily. To monitor the selected parameters, a DHT-11 is used which stands for “Digital Humidity and Temperature” sensor. DHT-11 can measure Temperature across the range of 0°C to 50°C and Humidity across the range of 20% to 80% and can operate between 3V to 5.5V. ESP32 is capable of supplying 3.3V output voltage

which makes direct interfacing of the sensor and microcontroller possible. The output of the sensor is in digital format which allows easier integration of the hardware.

The IoT-cloud platform used to perform this experiment is “ThingSpeak” because it allows 4 free channels and data visualization is possible in it. It provides channel keys which make reading and writing data from and to the platform simpler. ThingSpeak allows to store data for up to 8 different variables in a single channel and 2 of them are utilized for this experiment and named as “Temperature” and “Humidity” respectively.

To send and receive the data from and to the ESP32 board, it is important that it is connected to the internet. In order to do this, internet connectivity to the board and the device on which the output is being visualized must be ensured. A good connection must be established between the terminals of the sensor and the ESP32 board and a stable power supply must be ensured.

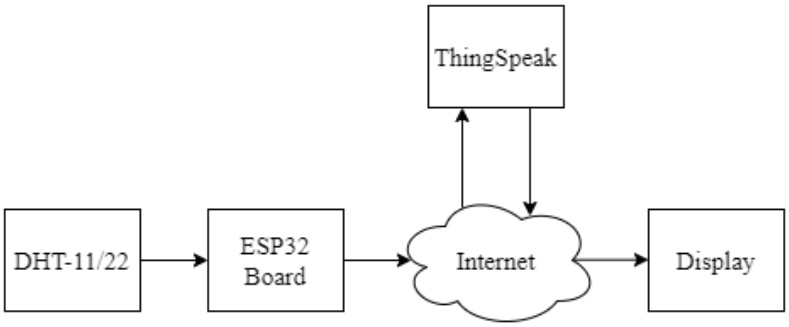
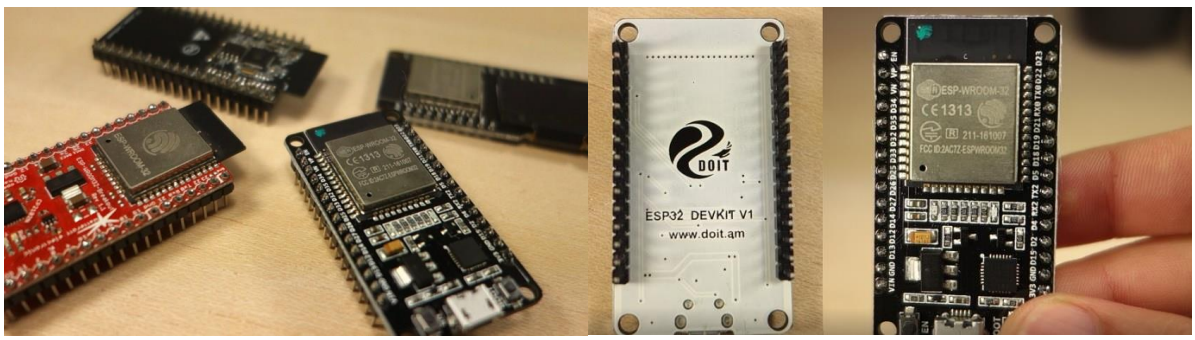


Fig. 1: Block Diagram of the System

As seen in the block diagram mentioned above, the role of internet is critical to this experiment because it serves as a link between the Microcontroller, IoT-cloud platform and the Display Device. The entire setup is connected on a breadboard and the data is visualized on a personal computer. The readings of the DHT can be tampered which is reflected in the visualization.

ESP 32 DOIT DEVKIT V1



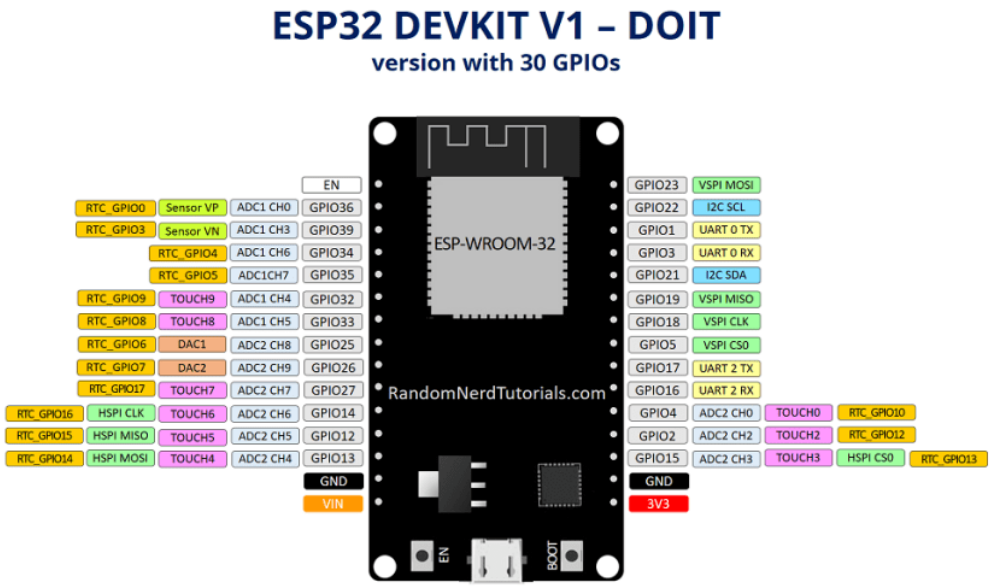
The ESP32 is dual core, this means it has 2 processors.

- It has Wi-Fi and bluetooth built-in.
- It runs 32-bit programs.
- The clock frequency can go up to 240MHz and it has a 512 kB RAM.
- This particular board has 30 or 36 pins, 15 in each row.
- It also has wide variety of peripherals available, like: capacitive touch, ADCs, DACs, UART, SPI, I2C and much more.
- It comes with built-in hall effect sensor and built-in temperature sensor.

Specifications

Number of cores	:2 (dual core)
Wi-Fi	:2.4 GHz up to 150 Mbits/s
Bluetooth	:BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	:32 bits
Clock frequency	:Up to 240 MHz
RAM	:512 KB
Pins	:30 or 36 (depends on the model)
Peripherals	:Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Netwokr), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

Pinout of ESP 32 DOIT DEVKIT V1



DHT 11:

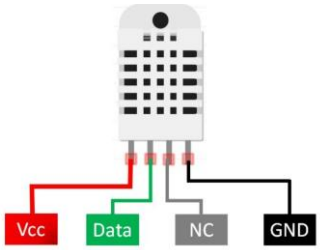
The DHT11/DHT22 is a sensor which measures relative humidity and temperature sensor. It provides a calibrated digital output with a 1-wire protocol. Both sensors are inexpensive. They are quite similar to each other with some differences of specifications.

DHT22 is almost similar to the DHT11 but the former measures temperature and humidity with higher accuracy and supports wider range.

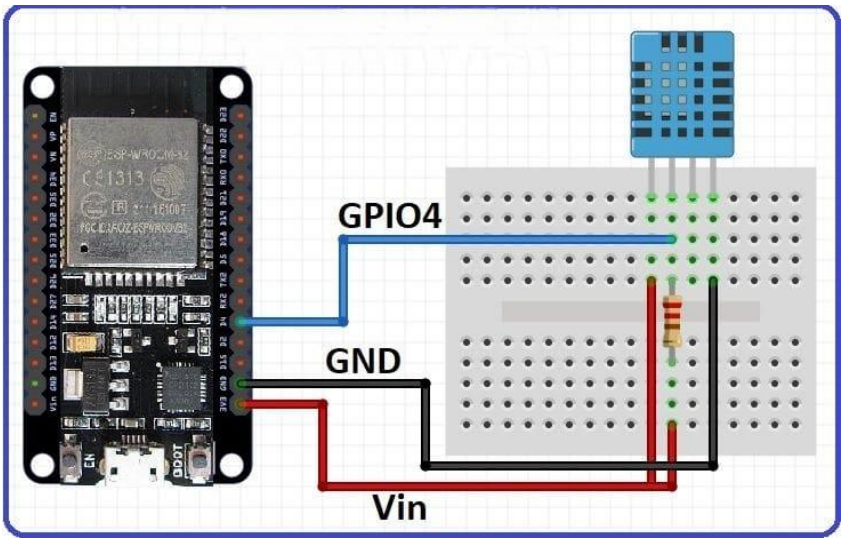
DHT11 SPECIFICATIONS:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

DHT 11 PINOUT:



INTERFACING DIAGRAM:



PROCEDURE:-

1. Make necessary connections as per Interfacing diagram.
2. Write code in Arduino IDE.
3. Verify the Output

CONCLUSION:

REFERENCES:

a) *https://thingspeak.com/*
b) *https://randomnerdtutorials.com/*
c) *https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf*
d) *https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf*
e) *https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json*

COURSE TEACHER

A. A. Gawari

ROLL NO: **DATE**

SOURCE CODE

```
#include <WiFi.h>
#include <ThingSpeak.h>
#include <DHT.h>

const char* ssid = "Enter The Name of Your Wi-Fi Here" ;
const char* password = "Enter The Password of Your Wi-Fi Here" ;

WiFiClient client ;

unsigned long myChannelNumber = 1;//Enter Your Channel Number and change it if it is not 1
const char * myWriteAPIKey = "Enter the Write API key of your Channel Here";

float temp ;
float humid ;

DHT dht(5, DHT11);

void setup() {
  // put your setup code here, to run once:
  pinMode (5, INPUT) ;
  Serial.begin (9600) ;
  dht.begin ();

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);

}

void loop() {
```

```
// put your main code here, to run repeatedly:

//Code to connect/reconnect to Wi-Fi
if (WiFi.status() != WL_CONNECTED) {
  Serial.print("Attempting to connect");
  while (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(ssid, password);
    delay(5000);
  }
  Serial.println("\nConnected.");
}

//Code to monitor the data
temp = dht.readTemperature();
humid = dht.readHumidity();
delay(100);

//Code to Write the Data
Serial.println("Temperature  : ") ;
Serial.print (temp) ;
Serial.print("ºC") ;
Serial.println("Humidity     : ") ;
Serial.print (humid) ;
Serial.print("%") ;
int x = ThingSpeak.writeField(myChannelNumber, 1, temp, myWriteAPIKey);
if (x == 200) {
  Serial.println("Channel update successful.");
}
else {
  Serial.println("Problem updating channel. HTTP error code " + String(x));
}

x = ThingSpeak.writeField(myChannelNumber, 2, humid, myWriteAPIKey);
```

```
 if (x == 200) {
   Serial.println("Channel update successful.");
 }
 else {
   Serial.println("Problem updating channel. HTTP error code " + String(x));
 }
}
```

Output

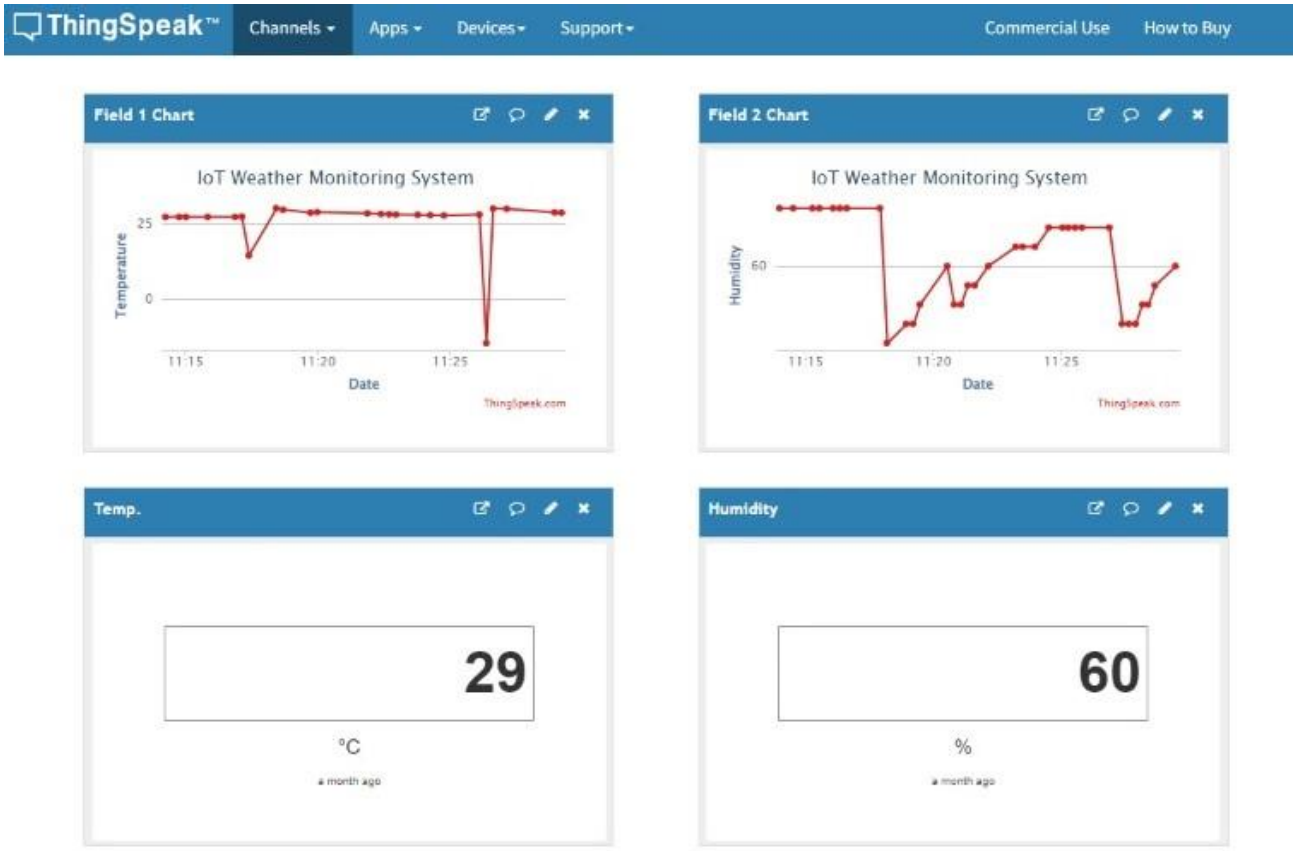


Fig. 2: ThingSpeak Platform Output