STM32 NRF24L01 Sensor Node with ESP32 NRF24L01 Gateway

1. Objective

In this project, we aim to monitor sensor data wirelessly by integrating the NRF24L01 module with an STM32 microcontroller board and an ESP32 WiFi module. The system is divided into two main components:

1. **Wireless Sensor Node**: This consists of a sensing unit (BME280 Barometric Pressure Sensor), a processing unit (STM32F103C microcontroller), a transceiver unit (NRF24L01 wireless module), and a power supply (3.7V Lithium-Ion Battery).
2. **WiFi Gateway**: This component uses the ESP32 module and NRF24L01 to receive data from multiple sensor nodes and upload it to a cloud server (ThingSpeak) via a WiFi network. The gateway acts as a bridge between the sensor nodes and the cloud, performing critical functions such as protocol translation, data processing, and encryption.

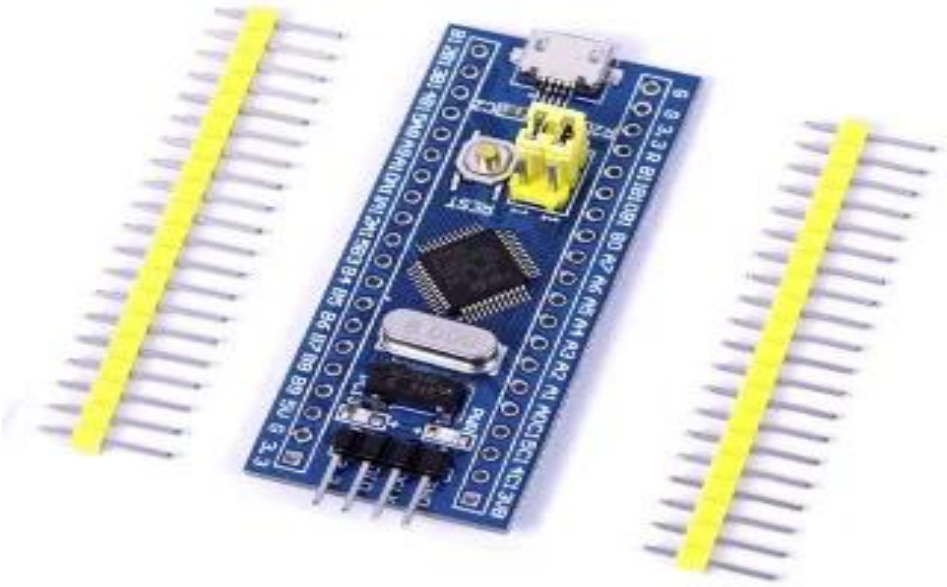
The project demonstrates efficient data transmission and real-time visualization using the IoT platform, ThingSpeak.

2. Components

S.N.	Component Name	Quantity	Description
1	ESP32 Board	1	Microcontroller with WiFi capability
2	STM32 Microcontroller	1	Bluepill board for sensor node
3	NRF24L01 PA+LNA	2	Wireless transceiver module
4	BME280 Barometric Pressure Sensor	1	Sensor for temperature, humidity, and pressure
5	Power Supply (5V)	2	Powering the components
6	Connecting Wires	20	For circuit connections
7	Breadboard	1	For prototyping

2.1 Bluepill Board

The **Blue Pill** is a compact, low-cost development board featuring the **STM32F103C8T6** microcontroller (ARM Cortex-M3, 72 MHz). It offers multiple GPIOs, UART, SPI, I²C, CAN, ADC, and PWM support. Breadboard-friendly, it includes a micro-USB port for programming and can be powered via USB or external sources. Compatible with **Arduino IDE**, **PlatformIO**, and **STM32CubeIDE**, it's ideal for IoT, robotics, and prototyping, with extensive community support.

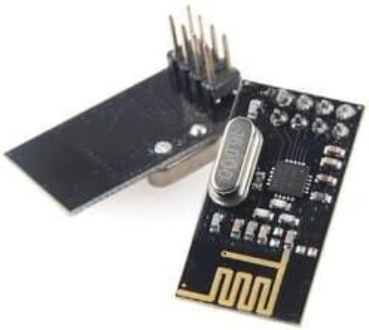


Fig(1):STM 32 Microcontroller

2.2 NRF24L01 Module

The **NRF24L01** is a 2.4GHz wireless transceiver designed for low-power, bidirectional communication. It supports a **SPI interface** (with 5V-tolerant pins) for compatibility with microcontrollers like **STM32** and **Arduino**, and can communicate with up to six modules, enabling **mesh networking**. Key features include:

- **Operating frequency**: 2.4GHz
- **Voltage requirement**: 3.3V
- **Address range**: 125 unique addresses
- **Transmission distance**: Up to 100



Fig(2):NRF24L01 module

2.3 ESP 32

The **ESP32** is a powerful, low-cost microcontroller with built-in **Wi-Fi** and **Bluetooth** capabilities. It is ideal for wireless communication and IoT applications, offering dual-core processing (up to 240 MHz) and a range of peripherals. The **ESP32** supports various interfaces like **SPI**, **I²C**, **UART**, and **PWM**, making it versatile for multiple projects. Key features include:

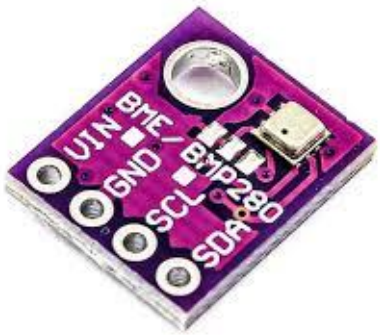
- **Connectivity**: Wi-Fi (802.11) and Bluetooth (Classic and BLE)
- **Processing power**: Dual-core, up to 240 MHz
- **Voltage requirement**: 3.3V
- **Peripherals**: GPIO, ADC, DAC, SPI, I²C, UART
- **Low power consumption**: Ideal for battery-powered devices

Fig(3):ESP 32 Microcontroller

2.4 BME280 Barometric Pressure Sensor

The **BME280** is a compact sensor that measures **barometric pressure**, **temperature**, and **humidity**. It communicates via **I²C** or **SPI** and operates at **3.3V to 5V**. Key features include:

- **Pressure range**: 300 to 1100 hPa
- **Temperature range**: -40°C to +85°C
- **Humidity range**: 0% to 100% RH Ideal for environmental monitoring and weather applications.



Fig(4):BME280 Barometric Pressure Sensor

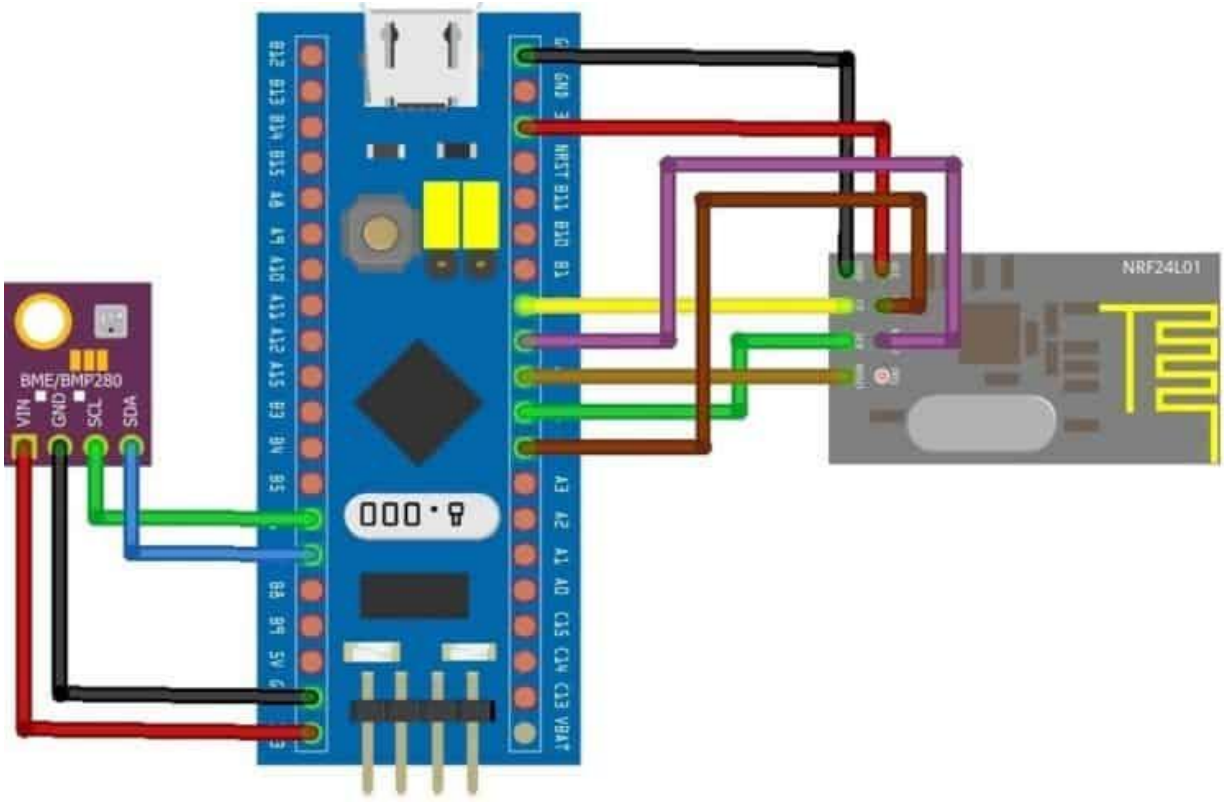
3. Circuit Design

3.1 Sensor Node Using NRF24L01 & STM32F103C

The sensor node is built using the STM32F103C microcontroller, NRF24L01 transceiver, and BME280 sensor. The BME280 measures temperature, humidity, pressure, and altitude. The NRF24L01 transmits this data wirelessly to the gateway.

Connections:

- NRF24L01:
  - VCC ➔ 3.3V (STM32)
  - CSN ➔ PA4 (STM32)
  - MOSI ➔ PA7 (STM32)
  - GND ➔ GND (STM32)
  - CE ➔ PB0 (STM32)
  - SCK ➔ PA5 (STM32)
  - MISO ➔ PA6 (STM32)
- BME280:
  - Interfaced via I2C or SPI with STM32F103C



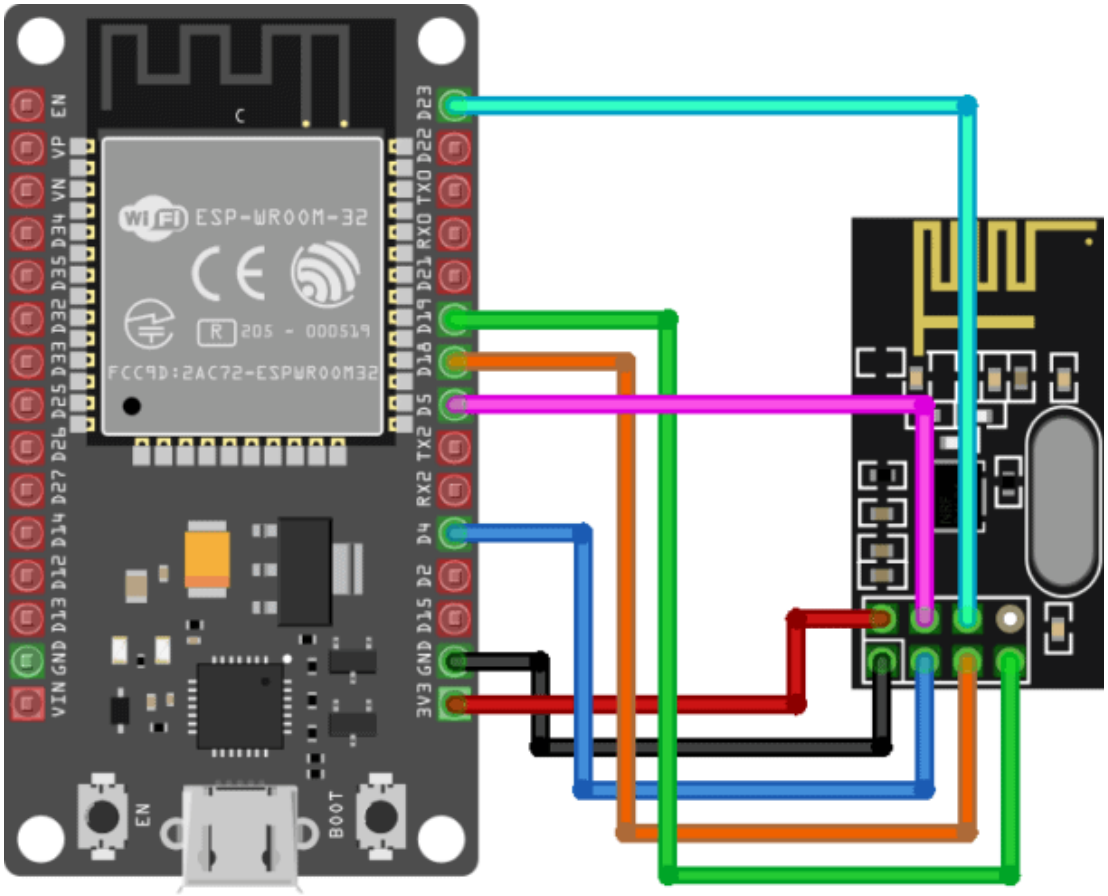
Fig(5): Schematic Diagram for Sensor Node Using NRF24L01 & STM32F103C

3.2 WiFi Gateway Using NRF24L01 & ESP32

The gateway collects data from sensor nodes via the NRF24L01 and uploads it to the ThingSpeak server using the ESP32 WiFi module. This component bridges the sensor network and the cloud, ensuring seamless data transmission and visualization.

Connections:

- NRF24L01:
  - VCC ➔ 3.3V (ESP32)
  - CSN ➔ D5 (ESP32)
  - MOSI ➔ D23 (ESP32)
  - GND ➔ GND (ESP32)
  - CE ➔ D4 (ESP32)
  - SCK ➔ D18 (ESP32)
  - MISO ➔ D19 (ESP32)



Fig(6): Schematic Diagram for WiFi Gateway Using NRF24L01 & ESP32

4. Program

4.1 Source Code/Program for Sensor Node

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(PB0, PA4); // CE, CSN on Blue Pill
const uint64_t address = 0xF0F0F0F0E1LL;
int counter = 0;


float temperature;
float humidity;
float altitude;
float pressure;

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme;

struct MyData
{
  int counter;
  float temperature;
  float humidity;
  float altitude;
  float pressure;
};
MyData data;

void setup()
{
Serial.begin(115200);
radio.begin();                  //Starting the Wireless communication
radio.openWritingPipe(address); //Setting the address where we will send the data
radio.setPALevel(RF24_PA_MIN);  //You can set it as minimum or maximum depending on the
distance between the transmitter and receiver.
radio.stopListening();          //This sets the module as transmitter

if (!bme.begin(0x76))
{
   Serial.println("Could not find a valid BME280 sensor, check wiring!");
   while (1);
  }
```

```
}

void loop()
{
  data.counter = counter;
  data.temperature = bme.readTemperature();
  data.pressure = bme.readPressure() / 100.0F;
  data.altitude = bme.readAltitude(SEALEVELPRESSURE_HPA);
  data.humidity = bme.readHumidity();

  Serial.print("Packet No. = ");
  Serial.println(data.counter);

  Serial.print("Temperature = ");
  Serial.print(data.temperature);
  Serial.println("*C");

  Serial.print("Pressure = ");
  Serial.print(data.pressure);
  Serial.println("hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(data.altitude);
  Serial.println("m");

  Serial.print("Humidity = ");
  Serial.print(data.humidity);
  Serial.println("%");

  Serial.println();

  radio.write(&data, sizeof(MyData));

  Serial.println("Data Packet Sent");
  Serial.println("");

  counter++;
  delay(5000);
}
```

4.2 Source Code/Program for ESP32 Wifi Gateway

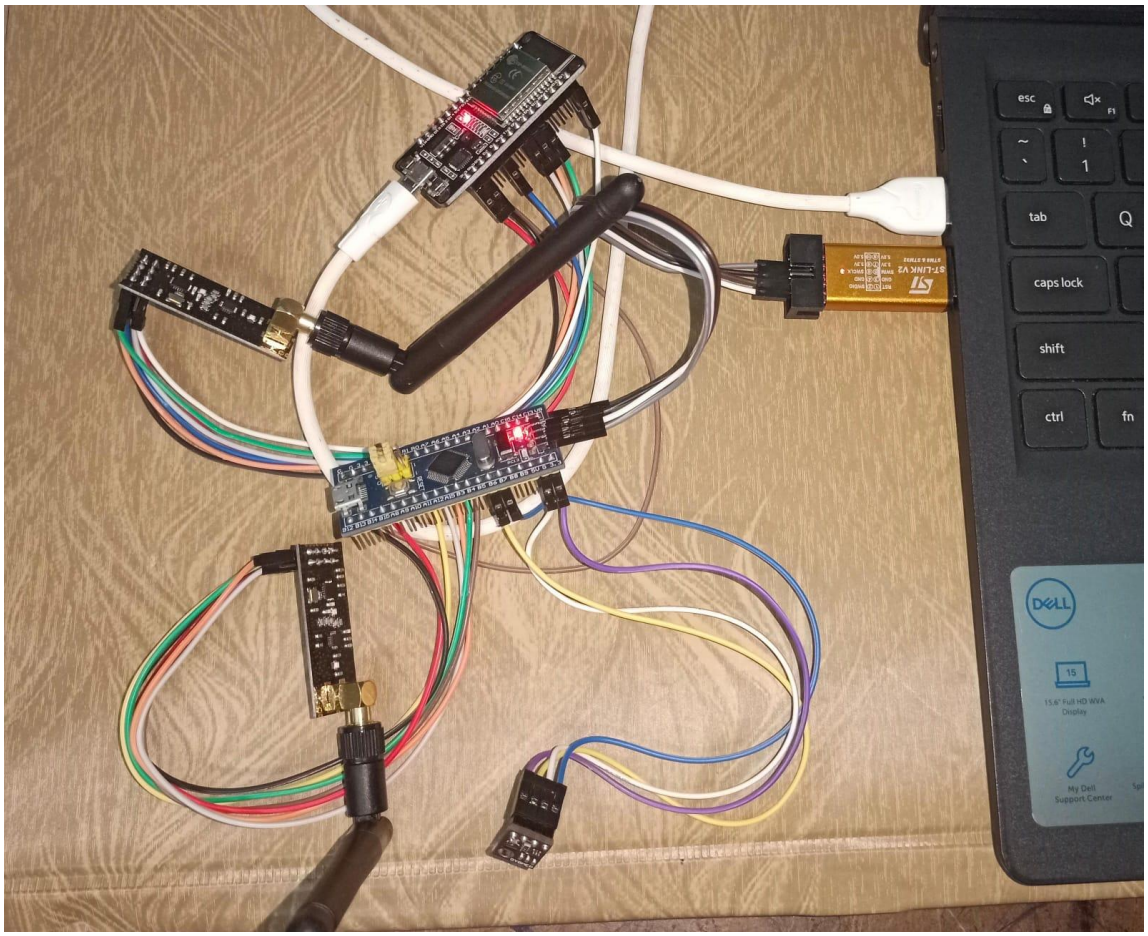
```
#include <WiFi.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

String apiKey = "C25ICK6FHOR7PST4";
```

```
const char* ssid = "Alexahome";
const char* password = "loranthus";

const char* server = "api.thingspeak.com";

RF24 radio(4, 5);
const uint64_t address = 0xF0F0F0F0E1LL;

struct MyData
{
  int counter;
  float temperature;
  float humidity;
  float altitude;
  float pressure;
};
MyData data;

WiFiClient client;

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


  Serial.println("Receiver Started....");
  Serial.print("Connecting to ");
  Serial.println(ssid);
  Serial.println();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  radio.openReadingPipe(0, address);   //Setting the address at which we will receive the data
  radio.setPALevel(RF24_PA_MIN);       //You can set this as minimum or maximum depending
on the distance between the transmitter and receiver.
  radio.startListening();              //This sets the module as receiver
}

int recvData()
{
  if ( radio.available() )
  {
    radio.read(&data, sizeof(MyData));
```

```
   return 1;
   }
   return 0;
}


void loop()
{
 if(recvData())
 {

 Serial.println("Data Received:");
 Serial.print("Packet No. = ");
 Serial.println(data.counter);

 Serial.print("Temperature = ");
 Serial.print(data.temperature);
 Serial.println("*C");

 Serial.print("Pressure = ");
 Serial.print(data.pressure);
 Serial.println("hPa");

 Serial.print("Approx. Altitude = ");
 Serial.print(data.altitude);
 Serial.println("m");

 Serial.print("Humidity = ");
 Serial.print(data.humidity);
 Serial.println("%");

 Serial.println();

 if (client.connect(server, 80))
 {
      String postStr = apiKey;
      postStr += "&field1=";
      postStr += String(data.temperature);
      postStr += "&field2=";
      postStr += String(data.pressure);
      postStr += "&field3=";
      postStr += String(data.altitude);
      postStr += "&field4=";
      postStr += String(data.humidity);
      postStr += "\r\n\r\n\r\n\r\n";

      client.print("POST /update HTTP/1.1\n");
      client.print("Host: api.thingspeak.com\n");
      client.print("Connection: close\n");
      client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
```

```
      client.print("Content-Type: application/x-www-form-urlencoded\n");
      client.print("Content-Length: ");
      client.print(postStr.length());
      client.print("\n\n");
      client.print(postStr);
      delay(1000);
      Serial.println("Data Sent to Server");
    }
      client.stop();
  }
}
```

5. Hardware Connections



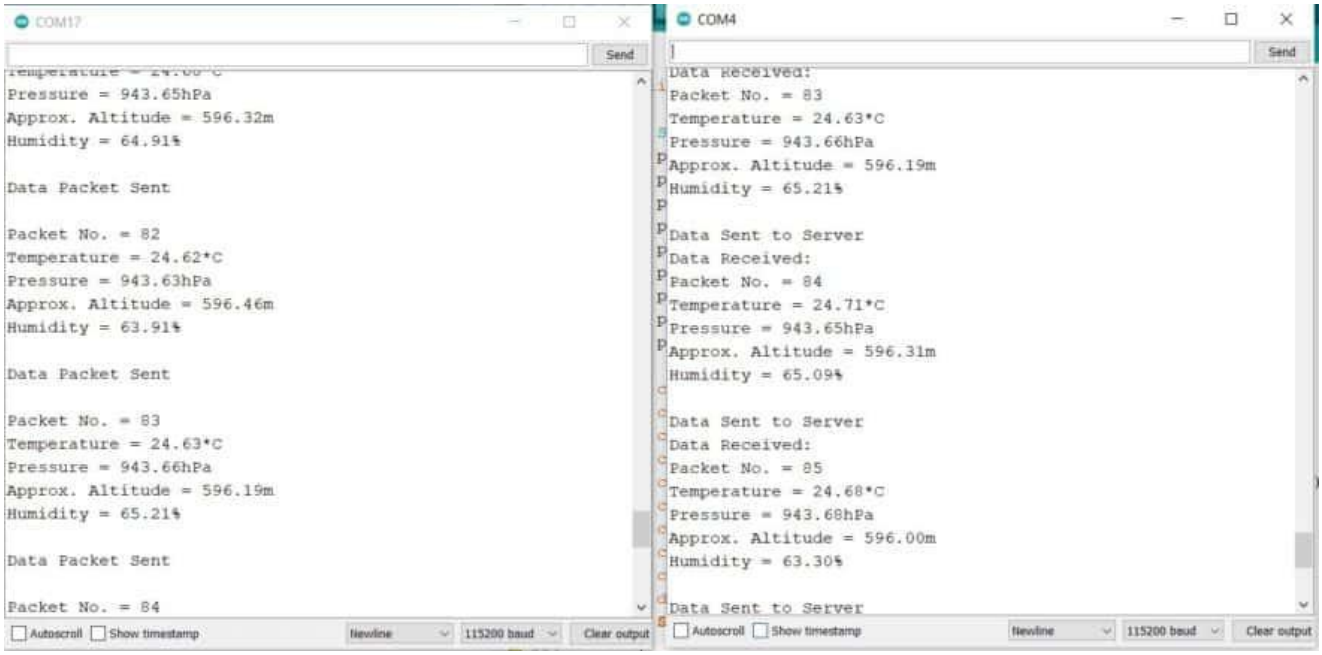
Fig(7): Hardware Setup of the Project

Figure 6 depicts the practical hardware arrangement for the STM32 NRF24L01 Sensor Node with ESP32 Gateway. The **STM32F103C (Blue Pill)** serves as the primary controller in the sensor node, interfaced with the **BME280 sensor** for environmental data collection via **I2C** and the **NRF24L01**

transceiver for wireless communication through **SPI**. The sensor node is powered by a battery or external power supply.

On the gateway side, the **ESP32 microcontroller** is connected to another **NRF24L01 module** via **SPI** to receive data from multiple STM32 nodes. The ESP32 processes the received sensor data and transmits it to the cloud (such as **ThingSpeak or an MQTT broker**) over **WiFi**. The system is set up on a breadboard for testing and validation

6. Serial Monitor



Fig(8): Output on Serial Monitor

Figure 7 shows that, after assembling the Sensor Node & Gateway on a breadboard,the code is uploadedSoon after the code is uploaded, the respective Serial Monitor is opened.The BME280 Sensor Data from Transmitter is received by Receiver/Gateway. The Gateway displays the received data like temperature, humidity, pressure, and altitude on the Serial Monitor .

7.Output in Thingspeak

Fig(9): Output on Thingspeak

Figure 8 showss that, Similarly we can monitor the data online on Thingspeak Server. For that, go to the private view of Thingspeak Server. The data will be logged on after the interval of 15 seconds. To make the device more better, we can implement the Sleep Mode feature and change Data Sending Interval.

8. Applications

Here are 5 applications for a **STM32 NRF24L01 Sensor Node with ESP32 NRF24L01 Gateway** project:

1. **Wireless Environmental Monitoring**: Use sensor nodes with various sensors (e.g., temperature, humidity, air quality) to collect data wirelessly and send it to a central ESP32 gateway for processing or cloud storage.
2. **Home Automation**: Sensor nodes can monitor environmental parameters (e.g., temperature, motion) and send data to an ESP32 gateway to trigger actions like controlling lights, fans, or HVAC systems.
3. **Industrial IoT (IIoT)**: Deploy sensor nodes in factories or warehouses to monitor equipment health (e.g., temperature, vibration) and transmit the data to the ESP32 gateway for predictive maintenance and real-time alerts.
4. **Smart Agriculture**: Use sensor nodes in fields or greenhouses to monitor soil moisture, temperature, and humidity, sending data to the ESP32 gateway to optimize irrigation systems and environmental control.

5. **Wearable Health Monitoring**: Integrate sensor nodes with health sensors (e.g., heart rate, temperature) worn by individuals, sending real-time data to an ESP32 gateway for healthcare monitoring or emergency alerts.

9. Conclusion

This project successfully demonstrates how to use NRF24L01 and ESP32 for wireless sensor data monitoring. The integration of ThingSpeak enables real-time data visualization, making this system an efficient IoT solution for environmental monitoring and data analysis.