KY_035 ANALOG HALL MAGNETIC SENSOR

Presented by
D.Panidhar
SBCS India pvt ltd
20/11/2023

Table of contents:

1.Abstract

2.Hardware Components

2.1 STM32F446RE Microcontroller

2. 2 KY-035 Analog Hall Magnetic Sensor

2.3 Rugged Board-a5d2x

2.4 WE10 Wi-Fi Module

3.Software Components

3.1 STM32 IDE Tool

3.2 Minicom

3.3 MQTT Rightech Cloud

4.Project Stages

4.1 Stage 1: Sending ADC Values to Minicom.

4.1.1 Sample code

4.1.2 Pin Configurations

4.1.3 Explanation

4.1.4 Connections

4.1.5 Output

4.2 Stage 2: Cloud Integration (Rightech Cloud)

4.2.1 Sample code

4.2.2 Pin Configurations

4.2.3 Explanation

4.2.4 Connections

4.2.5 Output

4.3 Stage 3: STM32 to Rugged board-a5d2x

4.3.1 Sample code for STM32

4.3.2 Sample code for Rugged board

4.3.3 Pin Configurations

4..3.4 Explanation

4.3.5 Connections

4.3.6 Output

4.4 Stage 4: Rugged board to Rihjtech cloud

4.4.1 Sample code for Rugged board

4.4.2 Pin Configurations

4.4.3 Explanation

4.4.4 Connections

4.4.5 Output

5.Advantages

6.Dis-advantages

7.Real-Time Applications

8.References

9.Conclusion

1.Abstract:

This project involves the development of a comprehensive system for monitoring and transmitting analog sensor data using the STM32F446RE microcontroller.The system aims to capture analog sensor data, specifically ADC values from a KY-035 analog hall sensor connected to the STM32F446RE microcontroller.

The project is divided into four stages, each addressing a unique aspect of data transmission, involving communication with various devices and platforms.The data is then transmitted through different channels, the system begins by sending Analog-to-Digital Converter (ADC) values from a KY-035 analog hall sensor directly to Minicom. Subsequently, the focus shifts to cloud integration, with the STM32F446RE communicating with the Rightech Cloud through a WE10 Wi-Fi module. The third stage involves the STM32F446RE interacting with a rugged board, establishing an additional layer of data transfer. The final stage sees the rugged board transmitting data to the Rightech Cloud via the WE10 Wi-Fi module.

2.Hardware Components:

2.1 STM32F446RE Microcontroller.

2. 2 KY-035 Analog Hall Magnetic Sensor.

2.3 Rugged Board.

2.4 WE10 Wi-Fi Module.

3.Software Components:

3.1 STM32 IDE Tool.

3.2 Minicom.

3.3 MQTT Rightech Cloud.

Hardware Components:

2.1 STM32F446RE Microcontroller

The STM32F411RE is a high-performance microcontroller based on the ARM Cortex-M4 processor. It has a number of features that make it well-suited for a variety ofapplications, including:

- ARM Cortex-M4 core running at a maximum frequency of 180 Mhz.
- Flash memory: Up to 512 KB for program storage.
- SRAM: Up to 128 KB for data storage.
- A rich set of peripherals, including timers, communication interfaces (USART, SPI, I2C), GPIO pins, and more.
- Advanced peripherals such as DMA (Direct Memory Access), ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), and RTC (Real-Time Clock).
- Typically operates at 3.3V, but some pins are 5V tolerant.

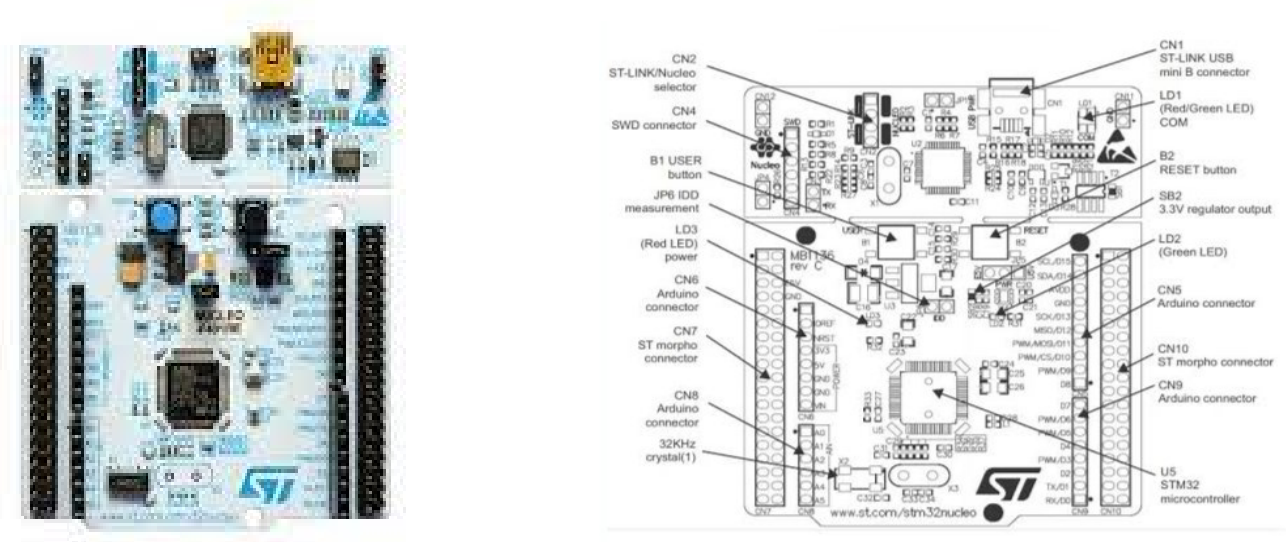


Fig:STM32F446RE Microcontroller.

2.2 KY-035 Analog Hall Magnetic Sensor

The KY-035 is an analog Hall magnetic sensor module that can be used to detect the presence of magnetic fields. Here is some basic information about the KY-035 analog Hall magnetic sensor:

- The sensor consists of a Hall Effect sensor and a comparator.
- When a magnetic field is present, the Hall sensor produces a voltage proportional to the magnetic flux density.
- The detection range of the KY-035 sensor depends on the strength of the applied magnetic field.
- The module provides an analog output that varies with the strength of the magnetic field. The analog signal can be read by an analog-to-digital converter (ADC) on a microcontroller.
- The module typically operates on a low voltage (e.g., 3.3V or 5V) and consumes a small amount of current.

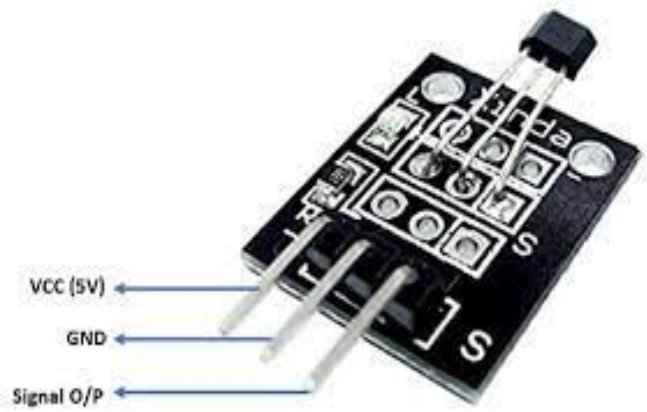


Fig:Analog hall magnetic sensor.

2.3 Rugged Board

Rugged Board is an Open Source Hardware & Software initiative to align with the fast growing Semiconductor technologies with a switch from classic to modern product development strategy & process. The usage of System on Module over a System on Chip is the rapid way to achieve time to market, curtail development risks for product quantities ranging from few to thousands.Rugged Board team targets to combine the Open source (Carrier Boards) community strength with industrial grade some and initiated the first Open Source Hardware "Industrial Pico Computer" which is powered by phyCORE-A5D2 SOM with Microchip A5D2x Cortex-A5 Core @500 Mhz.

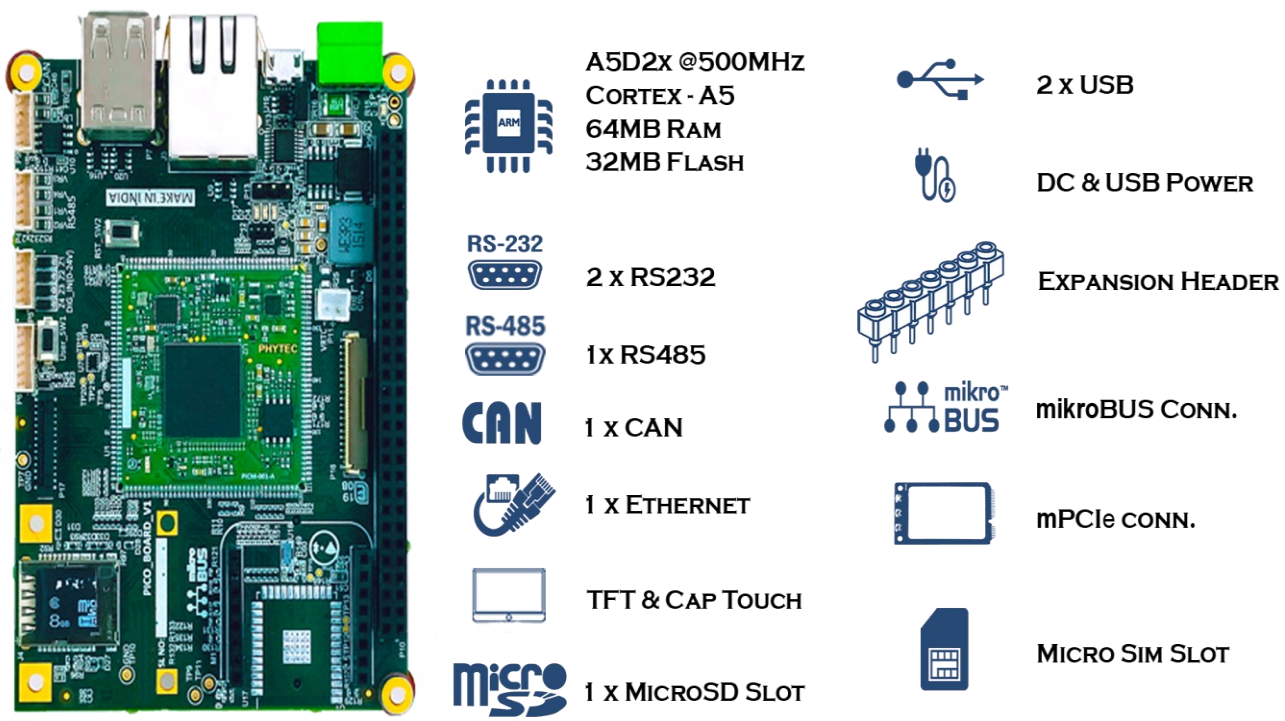


Fig:Rugged board.

2.4 WE10 Wi-Fi Module

The W10 WiFi module is a low-cost, easy-to-use WiFi module that can be used to connect IoT devices to the internet. The module has a built-in TCP/IP stack, so it can be easily connected to a variety of IoT platforms. The module also has a number of other features, such as:

- 100mW transmit power
- 11Mbps data rate
- 802.11 b/g/n compatibility
- Integrated antenna

Software Components:

3.1 STM32 IDE Tool

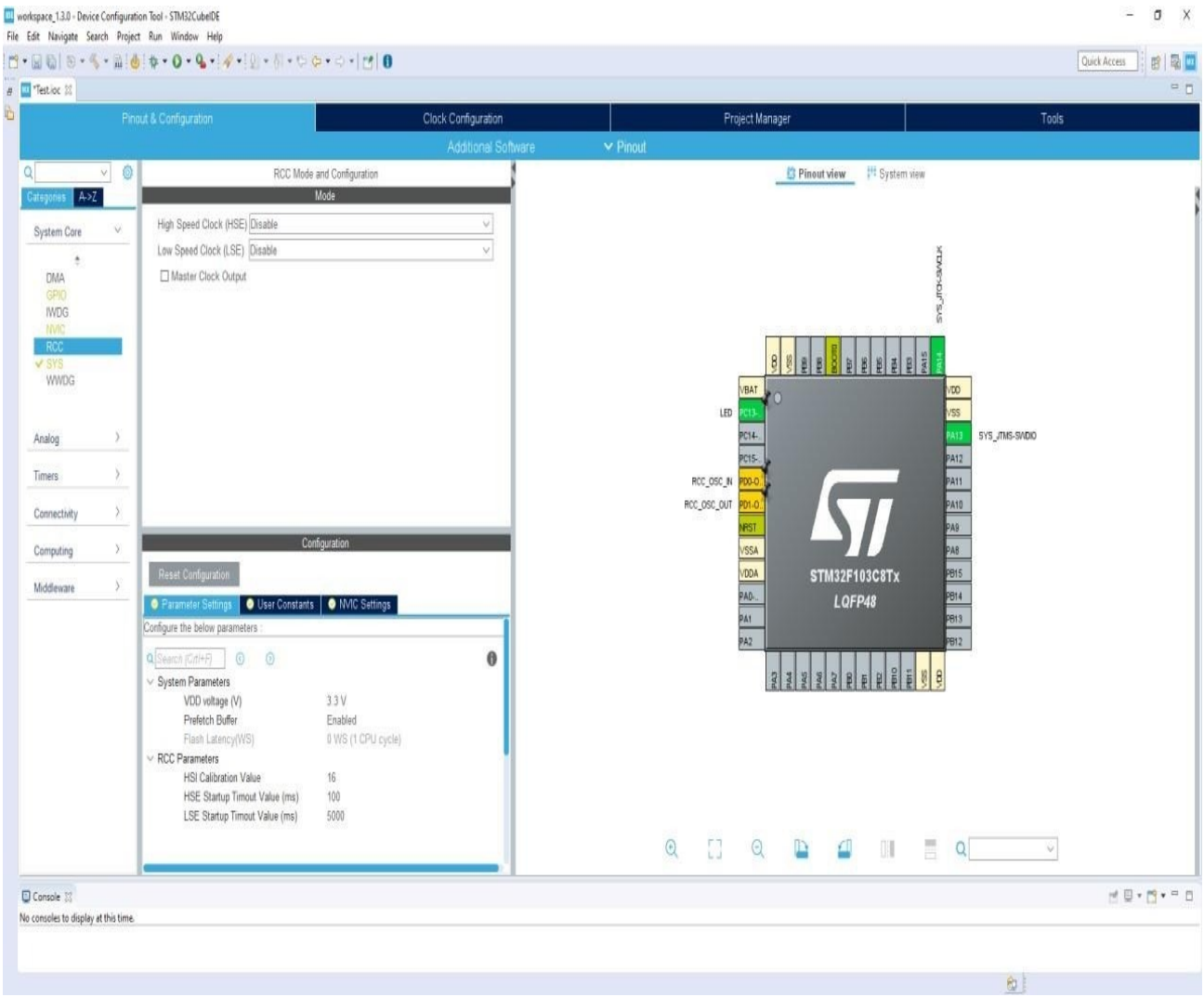
STM32CubeIDE is an integrated development environment from STMicroelectronics that provides a comprehensive set of tools to develop and debug embedded applications for STM32 microcontrollers.

Features:

- Code Editor: Offers features like syntax highlighting, code completion, and intelligent code navigation.
- Project Management: Supports the management of STM32CubeMX configuration files and project settings.
- Debugging: Integrated debugging capabilities with support for various debugging tools.
- STM32CubeMX Integration: Allows the seamless integration of STM32CubeMX, a graphical software configuration tool for STM32 microcontrollers.
- Peripheral Configuration: Provides tools to configure STM32 peripherals and generate initialization code.

Workflow:

- Create a new project in STM32CubeIDE.
- Configure the microcontroller and peripherals using STM32CubeMX.
- Write and edit your C code in the IDE.
- Compile your code to generate the binary file.
- Use the debugger to find and fix issues in your code.
- Flash the compiled code onto the STM32 microcontroller and run the application.



3.2 Minicom

Minicom is a text-based communication program that runs in a terminal window. It is designed for use on Unix-like operating systems, and it provides a simple and efficient way to connect to remote devices using serial communication.

Features:

- Minicom facilitates serial communication with devices connected to the computer through serial ports.
- Users can configure parameters such as baud rate, data bits, stop bits, and parity settings to match the requirements of the connected device.
- Minicom provides basic ASCII terminal emulation, allowing users to interact with devices using a terminal-like interface.
- Minicom supports scripting, enabling users to automate tasks and interactions with connected devices.

Installation:

- Minicom is available for installation on many Unix-like operating systems through package managers. For example, on Debian-based systems, it can be installed using the command

```
<sudo apt-get install minicom>
```

Usage:

- To start Minicom, users typically open a terminal window and run the `MINICOM` command, specifying the serial port and configuration parameters.
- The interface provides a command menu with various options for configuring and managing the communication session.
- Minicom allows users to send and receive text, control signals, and manage the connection to the serial device.

3.3 MQTT Rightech Cloud

MQTT (Message Queuing Telemetry Transport):

- MQTT is a lightweight, publish-subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency, or unreliable networks.
- It follows a client-server architecture, where clients (devices or applications) communicate through a central broker.
- MQTT is commonly used in IoT (Internet of Things) applications for its efficiency and simplicity.

Rightech Cloud:

- Rightech Cloud is a cloud-based platform designed to facilitate the development, deployment, and management of IoT solutions.
- It provides services for data storage, analytics, device management, and integration with various IoT devices and protocols.
- Rightech Cloud supports MQTT as one of the communication protocols for IoT device connectivity.

MQTT in Rightech Cloud:

- Rightech Cloud incorporates MQTT to enable seamless communication between IoT devices and the cloud infrastructure.
- MQTT's publish-subscribe model allows devices to publish data to specific topics, and other devices or services can subscribe to those topics to receive the data.

Security:

- MQTT communication can be secured using mechanisms such as TLS/SSL to encrypt data during transmission.
- Rightech Cloud typically provides authentication and authorization mechanisms to secure communication and data access.

4.Project Stages:

4.1 Stage 1: Sending ADC Values to Minicom

- **Objective:** Transmit ADC values from the KY-035 analog hall sensor to Minicom.
- **Implementation:** Direct communication between STM32F446RE and Minicom via UART 2.

4.1.1 Sample code:

```
while (1)
{

 HAL_ADC_Start(&hadc1);
 HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
 adcValue = HAL_ADC_GetValue(&hadc1);
 HAL_ADC_Stop(&hadc1);
 char buffer[50];

 if (adcValue > 3500 || adcValue < 3000)
 {
 sprintf(buffer, "MAGNET detected %d\r\n", adcValue);
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
 }
 else
 {
 sprintf(buffer,"MAGNET not detected %d\r\n",adcValue);
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
 }

 HAL_UART_Transmit(&huart2,(uint8_t *)buffer,strlen(buffer),     HAL_MAX_DELAY);
 HAL_Delay(1000);
  }
```

4.1.2 Pin Configurations:

- Connect the VCC of sensor to the 5V of STM32 MCU.
- Connect the GND of sensor to the GND of STM32 MCU.
- Connect signal pin of sensor to analog pin(A0) of STM32 MCU.

4.1.3 Explanation:

1.ADC Conversion:

- `HAL_ADC_Start(&hadc1)`: Initiates the ADC conversion.
- `HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY)`: Waits for the ADC conversion to complete with a maximum timeout.
- `adcValue = HAL_ADC_GetValue(&hadc1)`: Retrieves the ADC conversion result.

2. **Condition Check:**

- Checks if the `adcValue` is outside the range (3500 to 3000).
  - If true, it means a magnet is detected.
  - If false, it means a magnet is not detected.

3. **LED Control:**

- `HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET)`: Turns ON the LED if a magnet is detected.
- `HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET)`: Turns OFF the LED if a magnet is not detected.

4. **Message Formatting:**

- Uses `sprintf` to format a message string based on whether a magnet is detected or not.

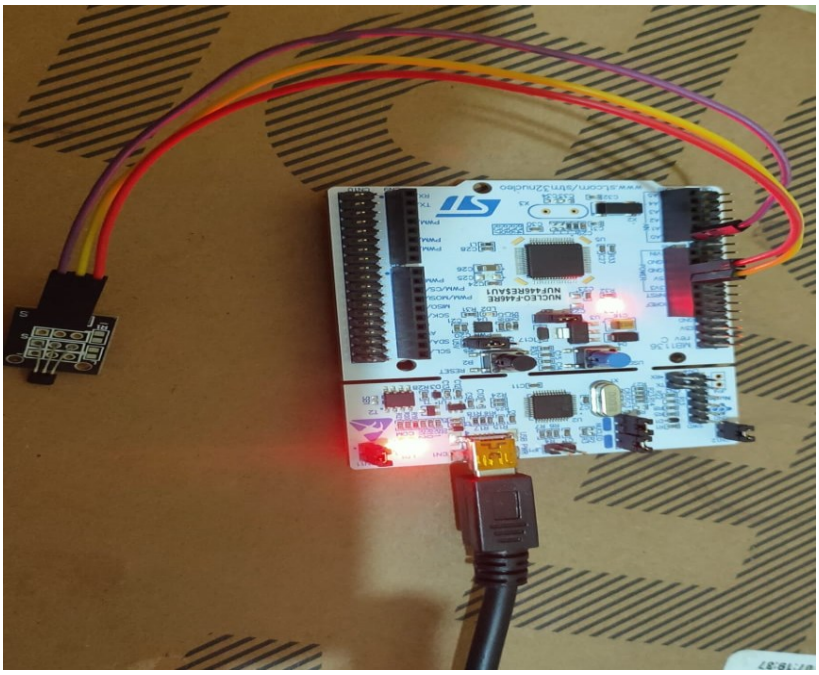
5. **UART Transmission:**

- `HAL_UART_Transmit(&huart2, (uint8_t *)buffer, strlen(buffer), HAL_MAX_DELAY)`: Transmits the formatted message over UART.

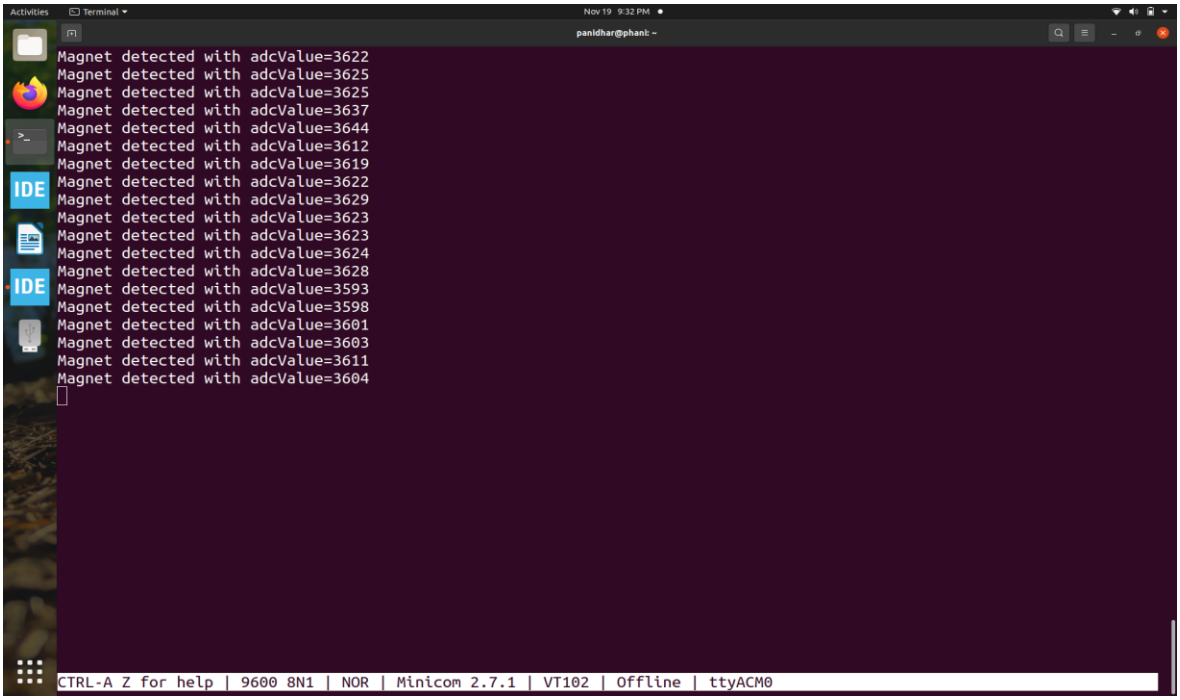
6. **Delay:**

- `HAL_Delay(1000)`: Introduces a delay of 1 second before the next iteration of the loop.

4.1.4 Connections:



4.1.5 Output



4.2 Stage 2: Cloud Integration (Rightech Cloud)

- **Objective:** Connect STM32F446RE to the Rightech Cloud via the WE10 Wi-Fi module.
- **Implementation:** Establish a secure connection, enabling data transfer to the cloud.

4.2.1 Sample Code:

```
int main(void)
{

 HAL_Init();


 SystemClock_Config();


 MX_GPIO_Init();
 MX_USART2_UART_Init();
 MX_ADC1_Init();
 MX_USART1_UART_Init();

 char buffer[128];

 sprintf (&buffer[0], "CMD+RESET\r\n");
 HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
 HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

 HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
 HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);



 sprintf (&buffer[0], "CMD+WIFIMODE=1\r\n");
 HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
 HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

 HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
 HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);



 sprintf (&buffer[0],"CMD+CONTOAP=Phani,123456789\r\n");
 HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
 HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);
 HAL_Delay(2000);
 HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
 HAL_Delay(500);
```

```
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);


sprintf (&buffer[0], "CMD?WIFI\r\n");
HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_Delay(500);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);


char buffer[128];


sprintf (&buffer[0], "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n");
HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 10000);
HAL_Delay(500);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 10000);



sprintf (&buffer[0], "CMD+MQTTCONCFG=3,mqtt-panidharece2023-vt8h5q,,,,,,,,\r\n");
HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_Delay(500);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);




sprintf (&buffer[0], "CMD+MQTTSTART=1\r\n");
HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_Delay(5000);
HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_Delay(500);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);


sprintf (&buffer[0], "CMD+MQTTSUB=base/relay/led1\r\n");
HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_Delay(500);
HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

while (1)
{

HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
adcValue = HAL_ADC_GetValue(&hadc1);
```

```
HAL_ADC_Stop(&hadc1);
char buffer[50];

if (adcValue > 3500 || adcValue < 3000)
{
sprintf(buffer, "MAGNET detected %d\r\n", adcValue);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
sprintf(&buffer[0], "CMD+MQTTPUB=reading/adcValue,%d\r\n", n);
HAL_UART_Transmit(&huart1, (uint16_t *)buffer, strlen(buffer),1000);
HAL_Delay(100);
}
else
{
sprintf(buffer, "MAGNET not detected %d\r\n", adcValue);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
sprintf(&buffer[0], "CMD+MQTTPUB=reading/adcValue,%d\r\n", n);
HAL_UART_Transmit(&huart1, (uint16_t *)buffer, strlen(buffer), 1000);
HAL_Delay(100);
}
HAL_UART_Transmit(&huart2,(uint8_t *)buffer,strlen(buffer),
HAL_MAX_DELAY);
HAL_Delay(1000);
}
}
```

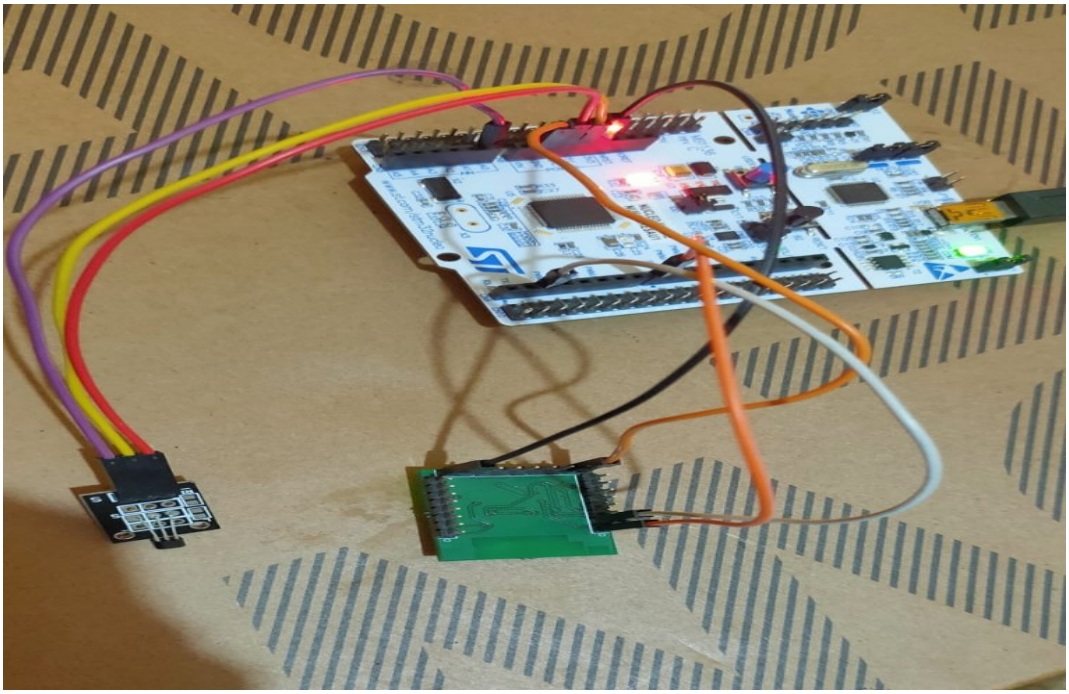
4.2.2 Pin Configurations:

- Connect the VCC of sensor to the 5V of STM32 MCU.
- Connect the GND of sensor to the GND of STM32 MCU.
- Connect signal pin of sensor to analog pin(A0) of STM32 MCU.
- Connect WE10 module RX to STM UART1 TX(D8) and WE10 TX to STM UART1 RX(D2).
- Connect WE10 GND to STM32 GND.
- Connect WE10 VCC to STM VCC(3.3V).

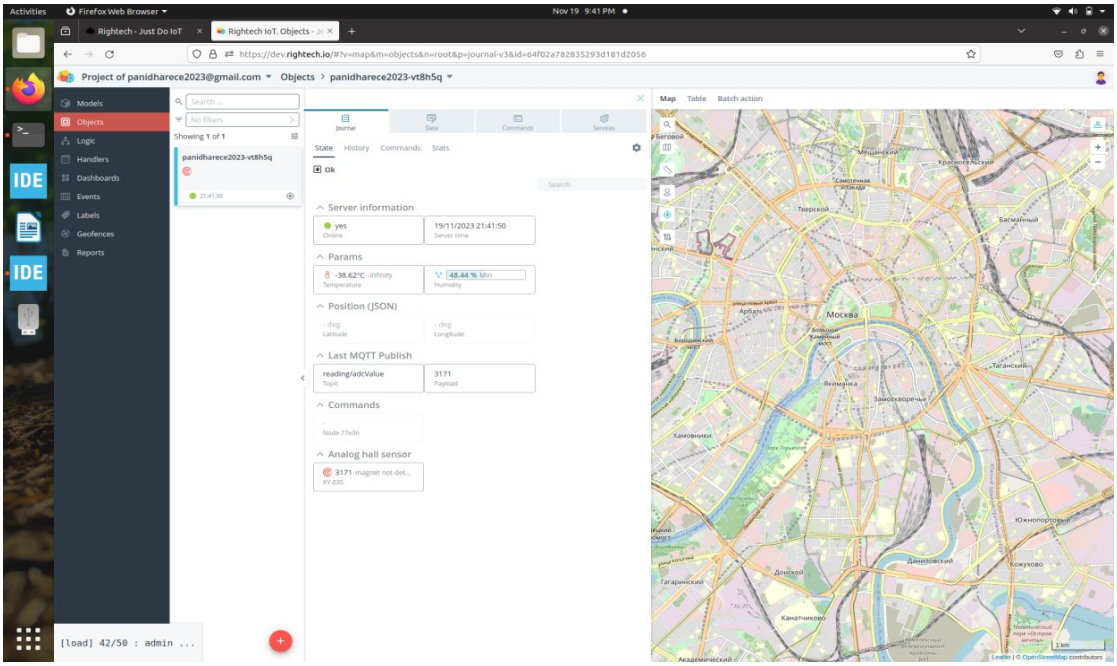
4.2.3 Explanation:

- To initializes the microcontroller, configures the system clock, and initializes GPIO pins, UART peripherals (USART2 and USART1), and the ADC module.
- It Reads ADC values from the sensor.
- To sends various commands to the WE10 module using UART1 and . Each command is transmitted, and the microcontroller waits for a response from the WE10 module.
- Publishes the ADC value to MQTT using the `CMD+MQTTPUB` command.

4.2.4 Connections:



4.2.5 Output:



4.3 Stage 3: STM32 to Rugged board-a5d2x

- **Objective:** Enhance the system by introducing a rugged board for data processing from stm32.
- **Implementation:** STM32F446RE communicates with the rugged board, expanding the project's capabilities.Similar to Stage 2, the rugged board communicates with the WE10 module to set up Wi-Fi and MQTT configurations.

4.3.1 Sample code for STM32:

```
while (1)
{
HAL_ADC_Start(&hadc1);
 HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
 adcValue = HAL_ADC_GetValue(&hadc1);
 HAL_ADC_Stop(&hadc1);

 char buffer[50];

 if (adcValue > 3500 || adcValue < 3000)
 {
sprintf(buffer, "MAGNET detected %d\r\n", adcValue);
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
 }
 else
 {
 sprintf(buffer,"MAGNET not detected %d\r\n",adcValue);
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
 }
HAL_UART_Transmit(&huart2,(uint8_t *)buffer,strlen(buffer),
HAL_MAX_DELAY);
HAL_Delay(1000);
 }
```

4.3.2 Sample code for Rugged board:

```
int main()
{
const char *portname = "/dev/ttyS3";  // Replace with the actual UART device file
int fd = open(portname, O_RDWR | O_NOCTTY | O_SYNC);
if (fd < 0){
        error("Error opening UART");}
        struct termios tty;
        if (tcgetattr(fd, &tty) < 0) {
        error("Error from tcgetattr");
```

```
        }
cfsetospeed(&tty, B9600);  // Set the baud rate
        cfsetispeed(&tty, B9600);
tty.c_cflag |= (CLOCAL | CREAD); // Ignore modem control lines, enable receiver
        tty.c_cflag &= ~CSIZE;   // Clear data size bits
        tty.c_cflag |= CS8       // 8-bit data
        tty.c_cflag &= ~PARENB;  // No parity bit
        tty.c_cflag &= ~CSTOPB;  // 1 stop bit
        tty.c_cflag &= ~CRTSCTS; // No hardware flow control
 tty.c_lflag = 0;               // Non-canonical mode
tty.c_cc[VMIN] = 1;             // Minimum number of characters to read
        tty.c_cc[VTIME] = 1;     // Time to wait for data (in tenths of a second)
if (tcsetattr(fd, TCSANOW, &tty) != 0)
{
        error("Error from tcsetattr");
 }
char buf[50];
        memset(buf, 0, sizeof(buf));
        int n = read(fd, buf, sizeof(buf));
        if (n < 0)
{
        error("Error reading");
  }
printf("Received: %s\n", buf);
close(fd);
return 0;

}
```

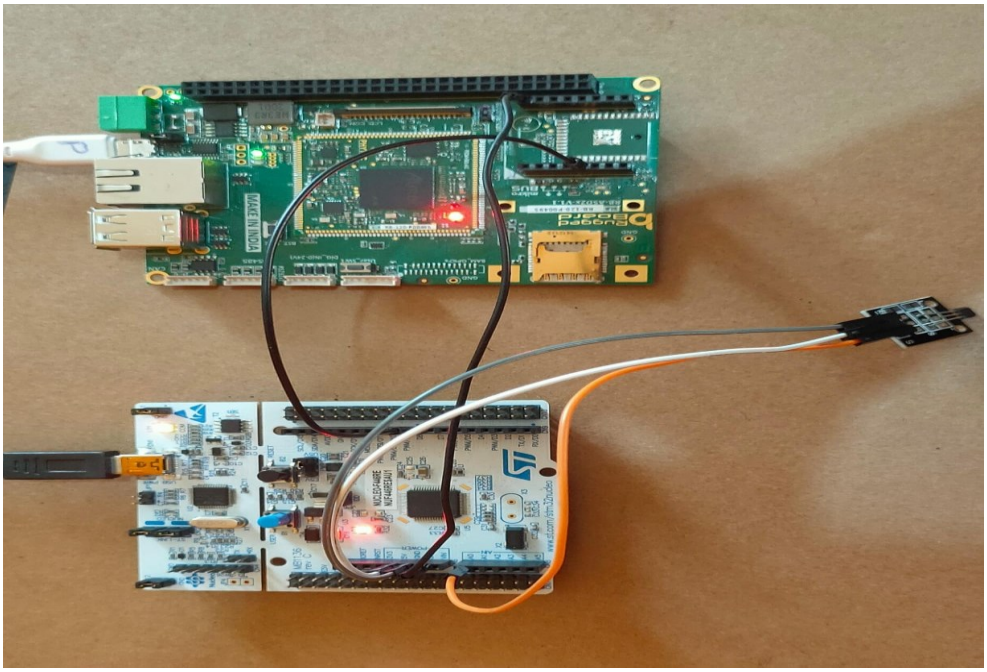
4.3.3 Pin Configurations:

- Connect the VCC of sensor to the 5V of STM32 MCU.
- Connect the GND of sensor to the GND of STM32 MCU.
- Connect signal pin of sensor to analog pin(A0) of STM32 MCU.
- Connect the STM32 UART TX to RX (UART3) of rugged board-a5d2x.
- Connect STM GND TO rugged board –a5d2x GND.

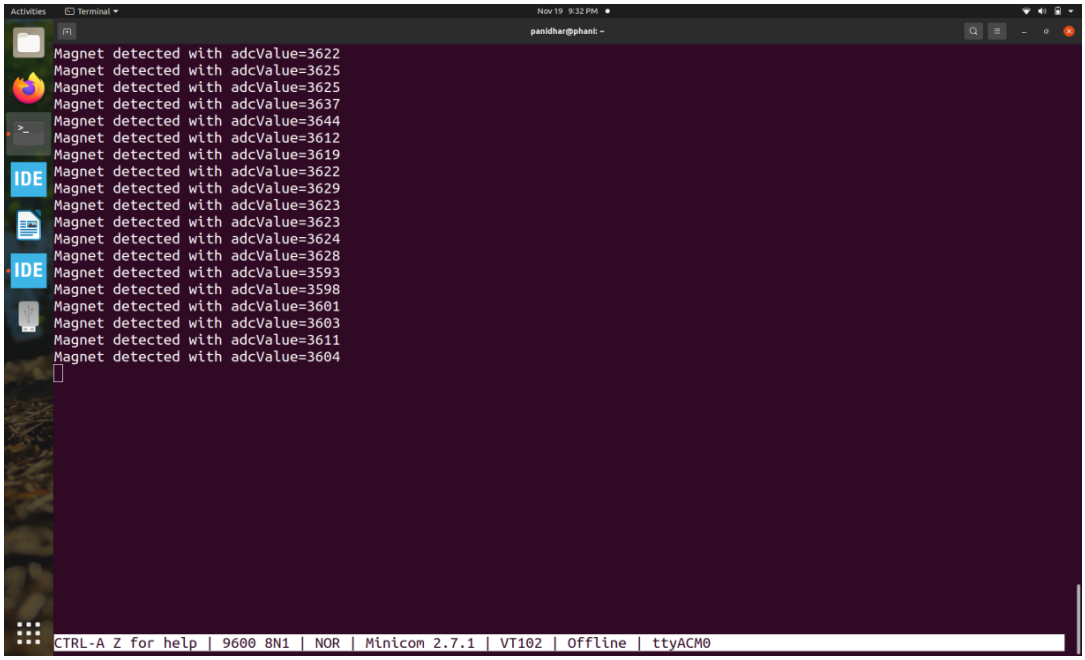
4.3.4 Explanation

- In STM32 using the HAL (Hardware Abstraction Layer) library. It appears to read values from an ADC (Analog-to-Digital Converter), monitor those values, and send corresponding messages via UART (Universal Asynchronous Receiver/Transmitter) .
- In rugged board code that opens a UART (Universal Asynchronous Receiver/Transmitter) port, configures its settings, reads data from the port, and prints the received data to the console.
- It opens the UART device file (`/dev/ttyS3` in this case) using the `open` system call.
- The flags `O_RDWR`, `O_NOCTTY`, and `O_SYNC` are used to open the port for reading and writing, not to become the controlling terminal, and to synchronize read and write operations.
- Sets the baud rate to 9600, 8 data bits, no parity, 1 stop bit, and disables hardware flow control.
- It reads data from the UART port into the `buf` array.It prints the received data to the console.

4.3.5 Connections



4.3.6 Output



4.4 Stage 4: Rugged Board to Rightech Cloud

- **Objective:** Enable the rugged board to transmit data to the Rightech Cloud via the WE10 Wi-Fi module.
- **Implementation:** Rugged board acts as a relay, forwarding data to the cloud for comprehensive monitoring.

4.4.1 Sample code

```
int main()
{
   char *portname = "/dev/ttyS3";
   int fd;
   int wlen;
   int rdlen;
   int ret;

   char res[5];
   char arr1[] = "CMD+RESET\r\n";
   char arr2[] = "CMD+WIFIMODE=1\r\n";
   char arr[] = "CMD+CONTOAP=\"Phani\",\"123456789\"\r\n";
   char arr3[] = "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n";
   char arr4[] = "CMD+MQTTCONCFG=3,mqtt-panidharece2023-vt8h5q,,,,,,,,,\r\n";
   char arr5[] = "CMD+MQTTSTART=1\r\n";
   char arr6[] = "CMD+MQTTSUB=base/relay/led1\r\n";

   unsigned char buf[100];

   fd = open(portname, O_RDWR | O_NOCTTY | O_SYNC);
   if (fd < 0)
   {
      printf("Error opening %s: %s\n", portname, strerror(errno));
```

```
        return -1;
    }
    set_interface_attribs(fd, B38400);

    printf("%s", arr1);
    wlen = write(fd, arr1, sizeof(arr1) - 1);
    sleep(3);

 // Send CMD+WIFIMODE=
   printf("%s", arr2);
    wlen = write(fd, arr2, sizeof(arr2) - 1);
    sleep(3);
    // Send CMD+CONTOAP
    printf("%s", arr);
    wlen = write(fd, arr, sizeof(arr) - 1);
    sleep(3);
    printf("%s", arr3);
    wlen = write(fd, arr3, sizeof(arr3) - 1);
    sleep(3);
    printf("%s", arr4);
    wlen = write(fd, arr4, sizeof(arr4) - 1);
    sleep(3);
    printf("%s", arr5);
    wlen = write(fd, arr5, sizeof(arr5) - 1);
    sleep(3);
   printf("%s", arr6);
    wlen = write(fd, arr6, sizeof(arr6) - 1);
    sleep(3);
 char buffer[100]; // Create a buffer to hold the formatted message
while(1){
 rdlen = read(fd, buf, sizeof(buf) - 1);
```

```
if (rdlen > 0) {

 buf[rdlen] = '\0'; // Null-terminate the received data
        printf("%s\n", buf);
   int ret = snprintf(buffer, sizeof(buffer), "CMD+MQTTPUB=reading/adcValue,%s\r\n", buf);
    if (ret < 0) {
     } else {
     ssize_t wlen = write(fd, buffer, ret);
        sleep(3);
        if (wlen == -1) {
       }
    }
}
}
close(fd);
    return 0;
}
```

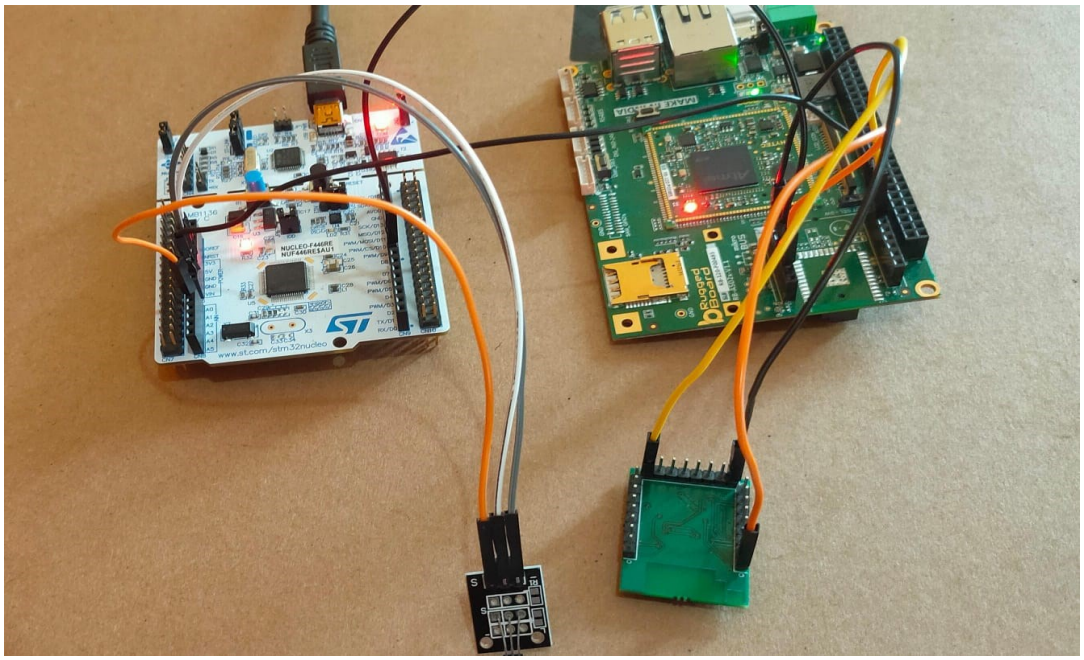
4.4.2 Pin Configurations

- Connect the VCC of sensor to the 5V of STM32 MCU.
- Connect the GND of sensor to the GND of STM32 MCU.
- Connect signal pin of sensor to analog pin(A0) of STM32 MCU.
- Connect WE10 RX to rugged board-a5d2x TX(UART3).
- Connect WE10 GND to rugged board-a5d2x GND.
- Connect WE10 VCC to rugged board-a5d2x (3.3v).

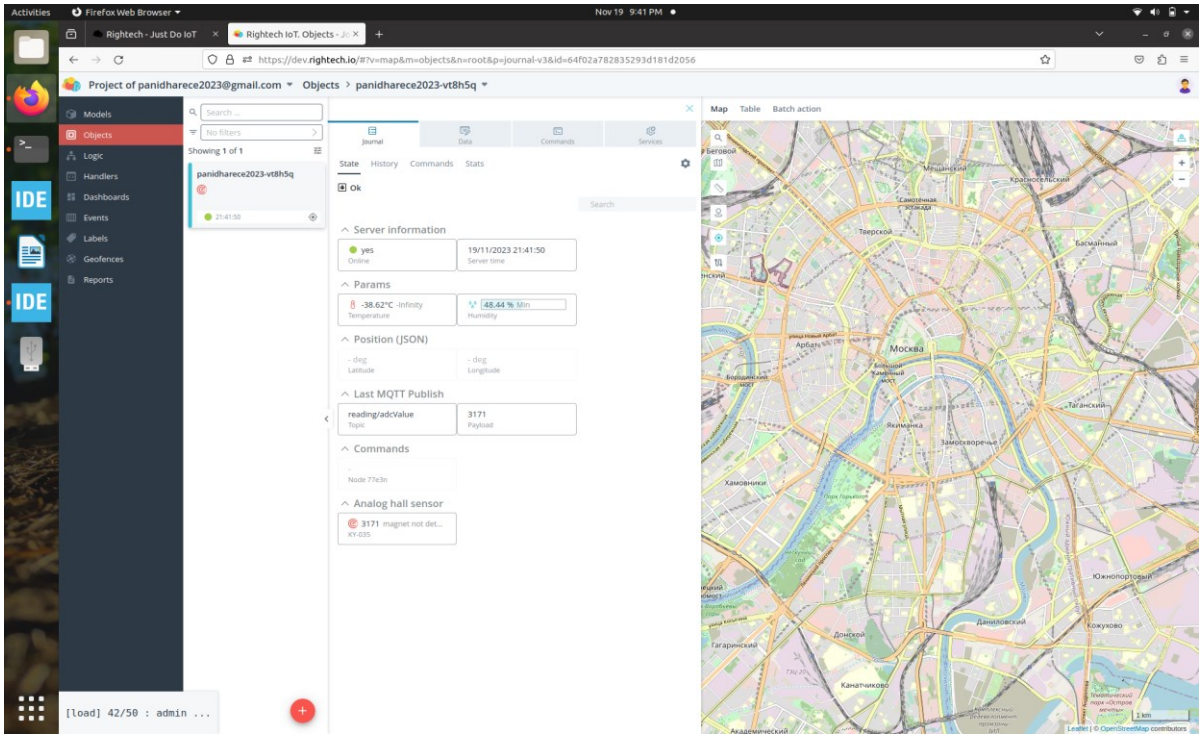
4.4.3 Explanation

The program initializes a serial port, sends a sequence of commands to an external device, receives data, processes it, and sends formatted messages back over the serial port. The communication seems to involve commands related to resetting, configuring Wi-Fi, MQTT setup, and continuous communication with another device. The program uses sleep delays to control the timing of commands and responses.

4.4.4 Connections



4.4.5 Output



5.Advantages

- Contactless Operation
- High Sensitivity
- Wide Operating Range
- Low Power Consumption
- Durability
- Cost-Effective
- Fast Response Time

6.Dis-advantages

- Temperature Sensitivity
- Dependency on Magnetic Fields
- Limited Sensing Range

7.Real-Time applications

- Door and Window Position Detection in Home Automation
- Anti-lock Braking Systems (ABS) in Vehicles
- Current Sensing
- Home Automation

8.References

- https://datasheetspdf.com/pdf/1402045/Joy-IT/KY-035/
- https://sensorkit.joy-it.net/en/sensors/ky-035/
- https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html/

9.Conclusion

The proposed system provides an efficient and scalable solution for real-time data monitoring and transmission. By combining the capabilities of the STM32F446RE, KY-035 analog hall sensor, WE10 Wi-Fi module, and Rightech Cloud, the project addresses a broad spectrum of applications, emphasizing reliability and versatility.