KY-021 MINI MAGNETIC REED SWITCH MODULE SENSOR

- Presented by
R.kumar

Table of Contents

1. Project Summary

This project centers around an STM32 microcontroller-based embedded system designed for magnetic field detection. Utilizing an Analog-to-Digital Converter (ADC), the system continuously monitors sensor data to discern the presence or absence of a magnet.

Two-way UART communication is established with a Rugged Board and the Rightech Cloud, allowing seamless integration and data exchange. The system dynamically controls an LED based on sensor readings, providing visual feedback.

Commands and data are exchanged with the Rugged Board for configuration, and sensor data is transmitted to the Rightech Cloud for remote monitoring.The project's modular design facilitates the addition of sensors, while bidirectional communication handling ensures effective integration. This versatile system promises adaptability and extensibility for diverse applications.

The system will be deployed in a real-world environment to test its performance.The deployement results will be used to access the system's feasibility and identify any potential improvements.

2. Hardware Components

2.1 STM32 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 of applications, including

- 100 MHz CPU clock speed
- 128 KB of RAM
- 512 KB of Flash memory
- Floating point unit (FPU)
- 11 general-purpose timers
- 13 communication interfaces
- USB OTG
- RTC

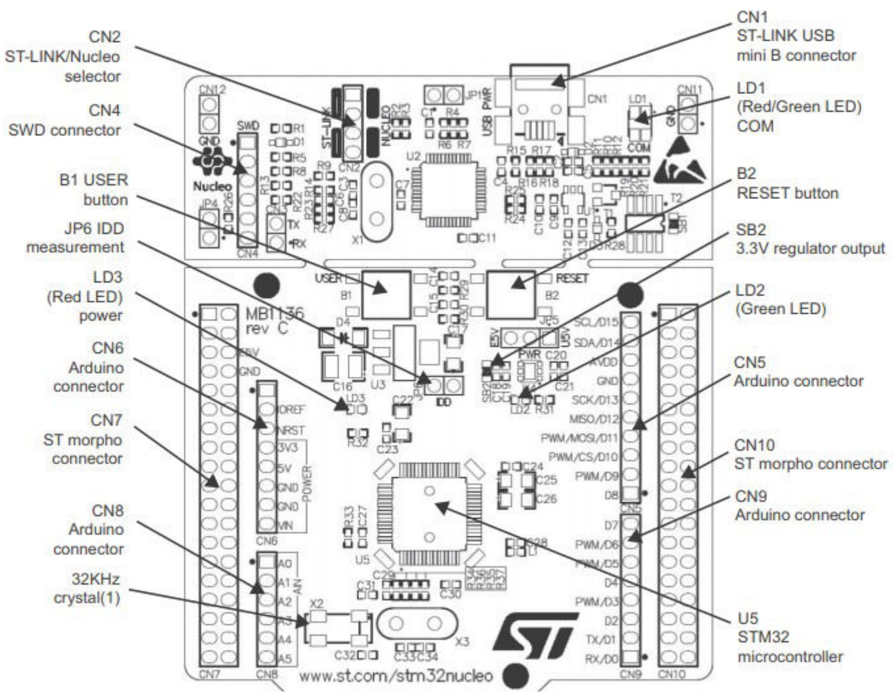


Fig: STM32 pin configuration

2.2 KY-021 Mini Reed Switch Sensor

The Mini Reed Sensor module is a compact and versatile component used to detect the presence or absence of a magnetic field.

It includes a 10K ohm resistor and a small switch reed switch actuated by a magnetic field. This Mini Reed Switch Module activates a switch closure when exposed to a magnetic field and remains open without a magnetic field.

- Reed capsules 20 mm long will typically resonate in the 1500 to 2000 Hz range.
- Operating voltage 3.3v – 5v.

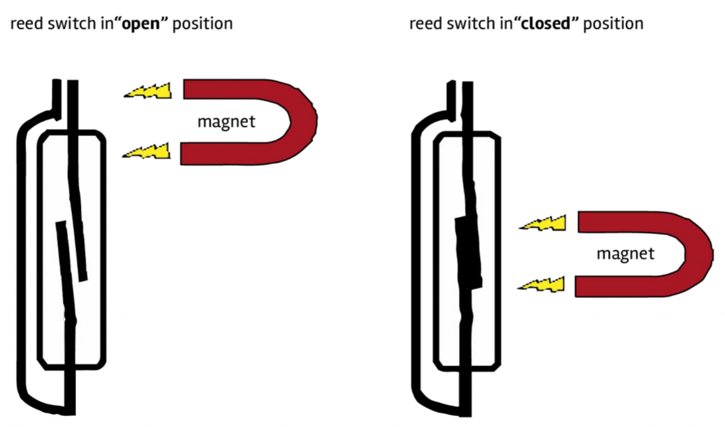
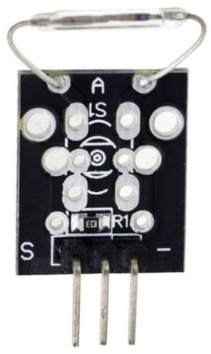

reed switch in **"open"** position
magnet
reed switch in **"closed"** position
magnet

Fig: Sensor magnet detection



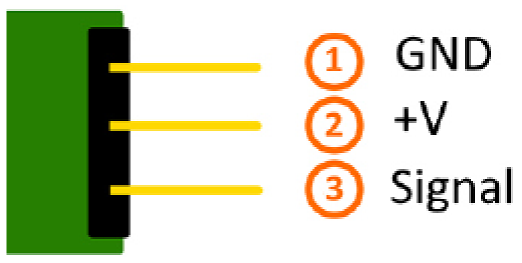


Fig: ky-021 sensor

Fig: Pinout

2.3 Rugged Board

RuggedBoard is an Open source Industrial single board computer powered by ARM Cortex-A5 SOC @500 MHz, implemented with the finest platform for rapid prototyping. The usage of System On Module over a System On Chip is the most rapid way to achieve time to market,curtail development risks for product quantities ranging from a few hundred to thousands.

RuggedBoard- A5D2x consists of Multiple Interfaces such as Ethernet,RS232, CAN, RS485, Digital Input and Digital Output with optically isolated, Standard MikroBus header for Add-On Sensors, Actuators and Multiple Wireless Modules such as ZigBee, LoRa,Bluetooth etc. mPCIeconnector with USB interface used for Cloud Connectivity modules 3G,4G, NB-IoT, WiFi. Expansion header with GPIO, UART, I2C, SPI,PWR etc.

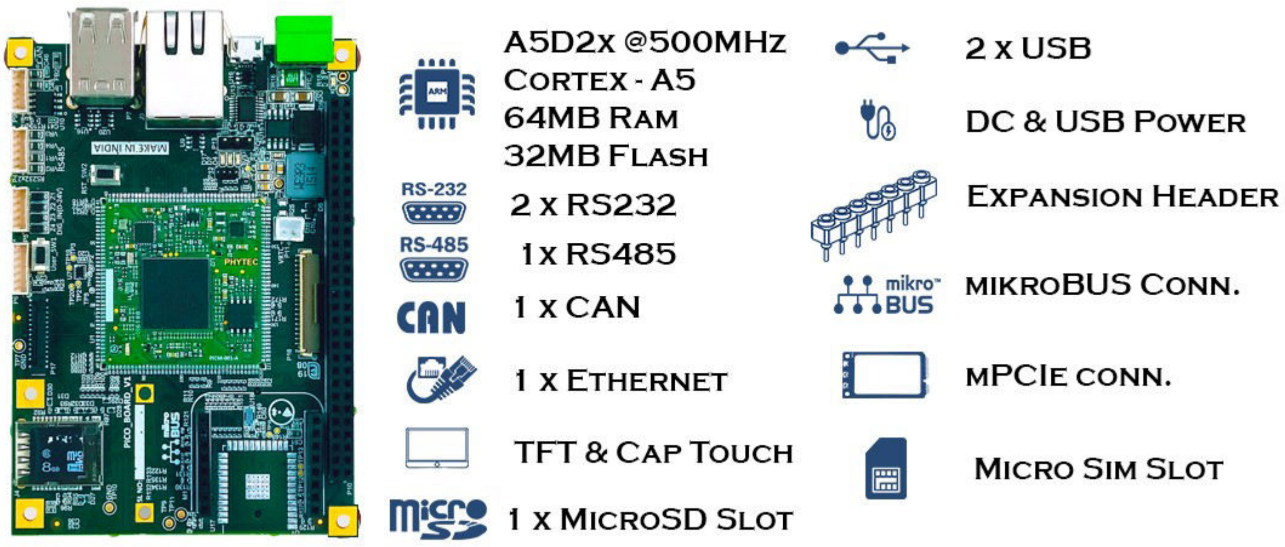


Fig: RuggedBoard-A5D2x

2.4 WE10 Module

The WE10 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 WE10 module is a compact wireless communication solution tailored for IoT applications. With Wi-Fi connectivity, it seamlessly integrates with microcontrollers, supporting diverse communication protocols. Known for its compact form factor, it suits space-constrained applications, coupled with low power consumption for extended device life. Delivering reliable performance, it ensures stable connectivity in various IoT environments. Security features are embedded for confidential data transmission. Its design allows for easy integration into existing hardware setups, simplifying development. The WE10 module finds versatile applications in smart homes, industrial automation, and remote monitoring systems. Its advanced features make it a reliable choice for developers seeking efficient and secure wireless connectivity in their IoT projects.

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

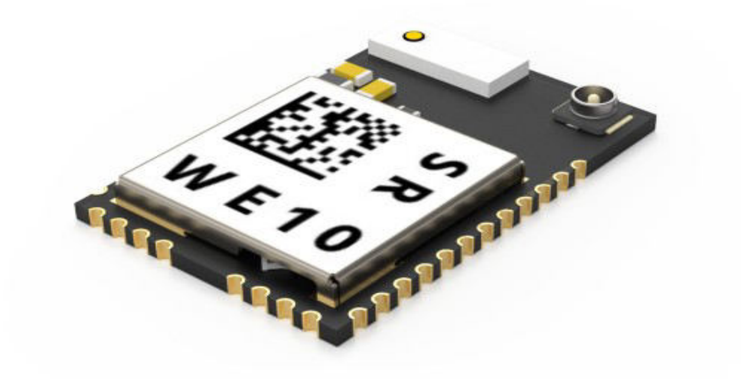


Fig: Wifi Module

3. Software Components

3.1 STM32 IDE Tool

STM32CubeIDE is an integrated development environment (IDE) developed by STMicroelectronics for STM32 microcontroller applications. It combines a user-friendly code editor, compiler, debugger, and peripheral configuration tools into a single platform.

This IDE streamlines development by integrating with STM32CubeMX, a graphical configuration tool for STM32 microcontrollers. Key features include a Hardware Abstraction Layer (HAL) for simplified peripheral interactions, advanced debugging capabilities compatible with popular probes like ST-Link and J-Link, and cross-platform support for Windows, Linux, and macOS.

STM32CubeIDE supports efficient source code management with built-in Git support and facilitates collaborative development. Its real-time operating system (RTOS) integration, particularly with FreeRTOS, enhances multitasking capabilities. The IDE's plugin system allows developers to extend functionality, adapting the environment to specific project needs. Overall, STM32CubeIDE accelerates STM32 microcontroller application development by providing an integrated and feature-rich platform for configuration, code generation, debugging, and collaborative coding.

3.1.1 Overview

1. Integrated Development Environment (IDE):

• The IDE is the primary interface for software development. It include code editing, debugging, and project management tools.

2. STM32CubeMX:

• STM32CubeMX is a graphical configuration tool that allows you to for your STM32 microcontroller.

3. HAL (Hardware Abstraction Layer) Library:

• STM32CubeIDE integrates the HAL library, which provides a set of high-level functions to interact with the microcontroller peripherals.

4. CMSIS (Cortex Microcontroller Software Interface Standard):

• CMSIS provides a standardized interface to the core functions of a Cortex-M microcontroller, including the NVIC (Nested Vector Interrupt Controller) and SysTick.

5. Debugger:

• The debugger allows you to set breakpoints, inspect variables, and step through code during the debugging process.

6. Project Configuration:

• You can configure various project settings, including compiler options, linker scripts, and build configurations.

7. Console Output:

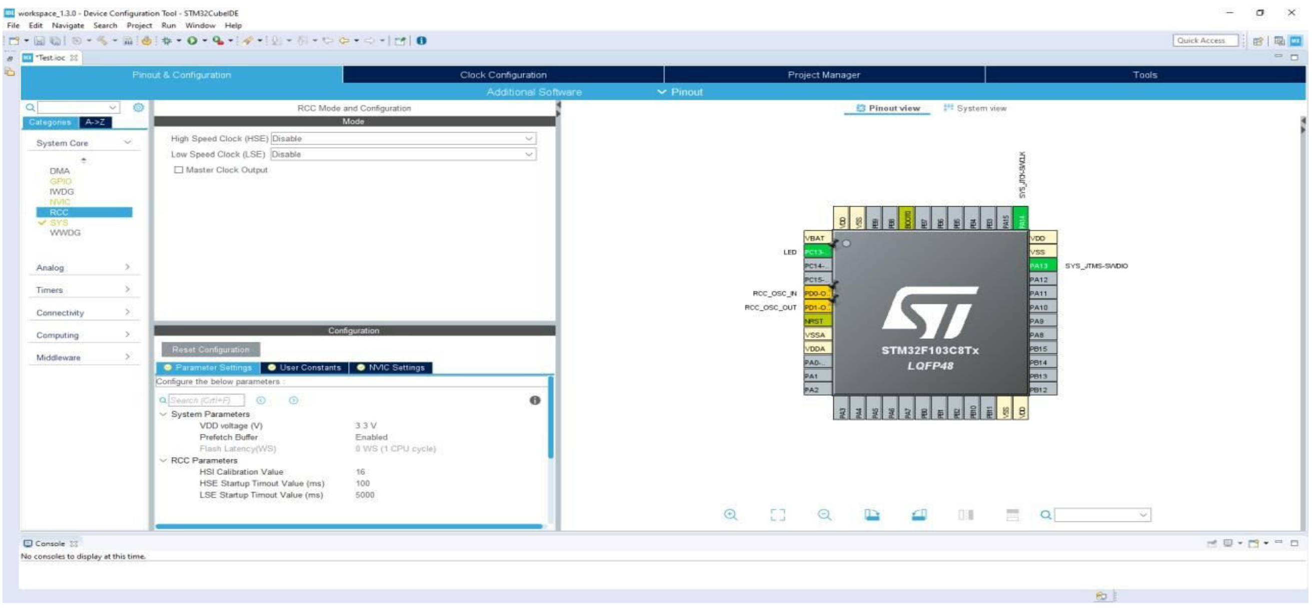
• The console provides feedback from the compiler and debugger, displaying messages, warnings, and errors.

8. Peripheral Configuration:

• STM32CubeMX generates initialization code based on your peripheral configurations, helping to set up the microcontroller's features.

9. Project Explorer:

• The Project Explorer organizes your project files and allows you to navigate through the source code and configuration files.



RCC Mode and Configuration

Mode

High Speed Clock (HSE) Disable

Low Speed Clock (LSE) Disable

☐ Master Clock Output

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings

Configure the below parameters :

```
System Parameters
    VDD voltage (V)                   3.3 V
    Prefetch Buffer                   Enabled
    Flash Latency(WS)                 0 WS (1 CPU cycle)
RCC Parameters
    HSI Calibration Value             16
    HSE Startup Timout Value (ms)     100
    LSE Startup Timout Value (ms)     5000
```

No consoles to display at this time.

3.1.2 Workflow

1. Project Initialization:

• Create a new project in STM32CubeIDE.

• Configure the microcontroller and peripherals using STM32CubeMX.

2. Code Development:

• Write and edit your C code in the IDE.

3. Build and Compilation:

• Compile your code to generate the binary file.

4. Debugging:

• Use the debugger to find and fix issues in your code.

5. Flash and Run:

• Flash the compiled code onto the STM32 microcontroller and run the application.

3.2 MINICOM

Minicom is a text-based serial communication program that is commonly used to connect to and communicate with devices over a serial port. It is often used 15 for debugging and configuring devices, especially in embedded systems and projects involving microcontrollers or other hardware components. Below is an explanation of how minicom can be used in a project.

1. Installation:

• Before using minicom, you need to install it on your system. You can typically install it using your system's package manager.

• bash

• sudo apt-get install minicom

2. Connecting to a Serial Port:

• Minicom is primarily used for serial communication, so you need to be connect it to the serial port of the device you want to communicate with.Use the following command to open minicom:

• bash

• minicom -D /dev/ttyUSB0

• Here, /dev/ttyUSB0 is the path to the serial port. The actual port may vary depending on your system and the connected device.

3. Configuration:

• Once minicom is open, you may need to configure the serial port settings such as baud rate, data bits, stop bits, and parity. This is often necessary to match the settings of the device you are communicating with. You can access the configuration menu by pressing Ctrl-A followed by Z.

4. Interacting with the Device:

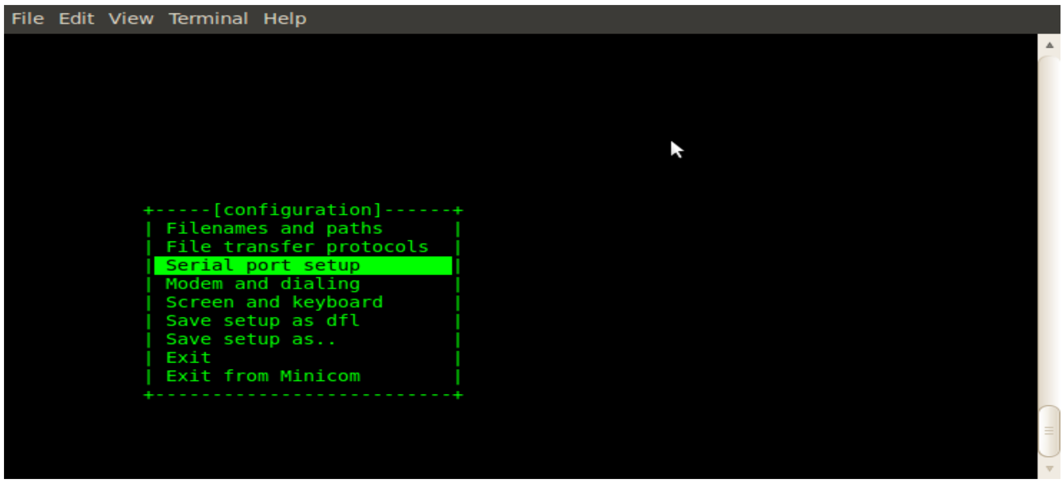
• After configuring the serial port, you can interact with the device. Minicom allows you to send commands and receive responses. This is particularly useful for debugging purposes and for configuring devices that have a serial console.

5. Exiting Minicom:

• To exit minicom, you can use the Ctrl-A followed by X shortcut.

6. File Transfer:

• Minicom also supports file transfer using protocols like Xmodem or Ymodem. This can be useful for updating firmware or transferring files between your computer and the connected device.



```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

Fig: Minicom Window

3.3 RIGHTTECH IOT CLOUD

Rightech IoT Cloud is a tool for developers. RIC is independent of specific equipment and protocols, which makes it easier for developers to combine different devices under one solution. Platform tools allow developers to create IoT solutions without extra code and reuse 90% of that solution to launch similar cases.

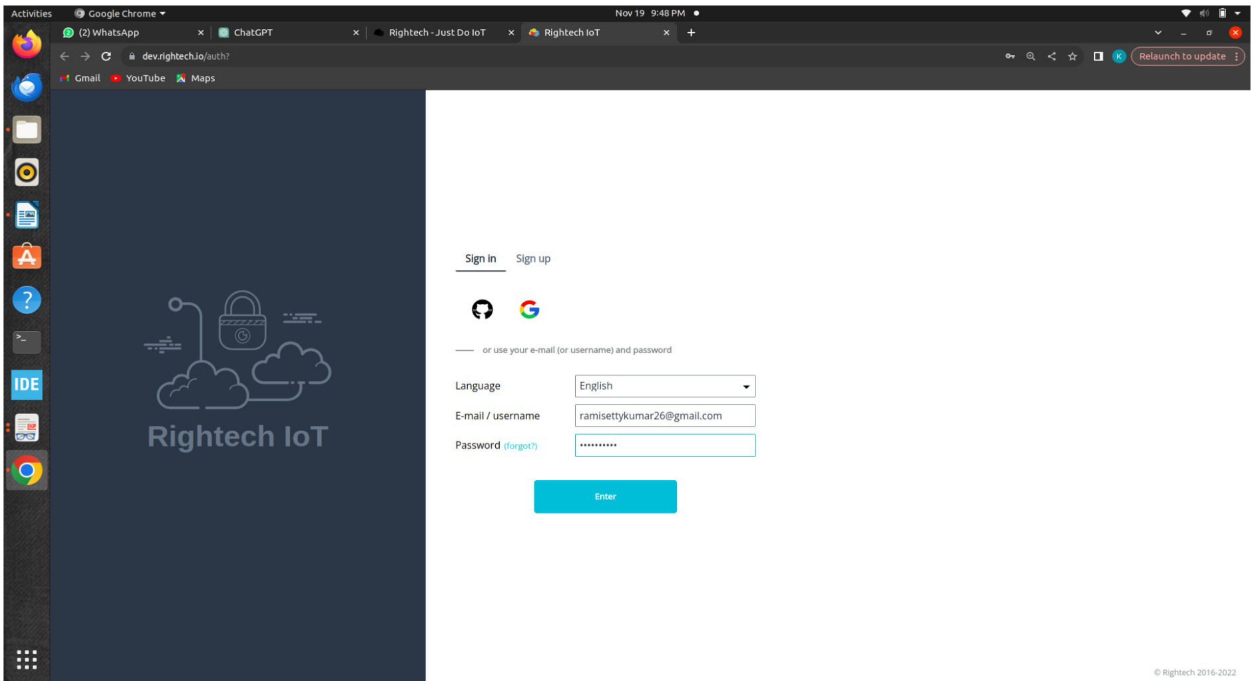
• Visit the "RightTech IoT" Website: Go to the official website or portal of "RightTech IoT."

• Registration: Look for a "Register" or "Sign Up" option on their website. Click on it to start the registration process.

• Fill Out Registration Form: Provide the required information, such as your name, email address, password, and any other details that the platform requests.

• Account Verification: Some platforms may require you to verify your email address by clicking on a verification link sent to your email. Complete the verification process if required.

• Log In: Once your registration is complete and your account is verified, log in to your "RightTech IoT" account using your credentials.

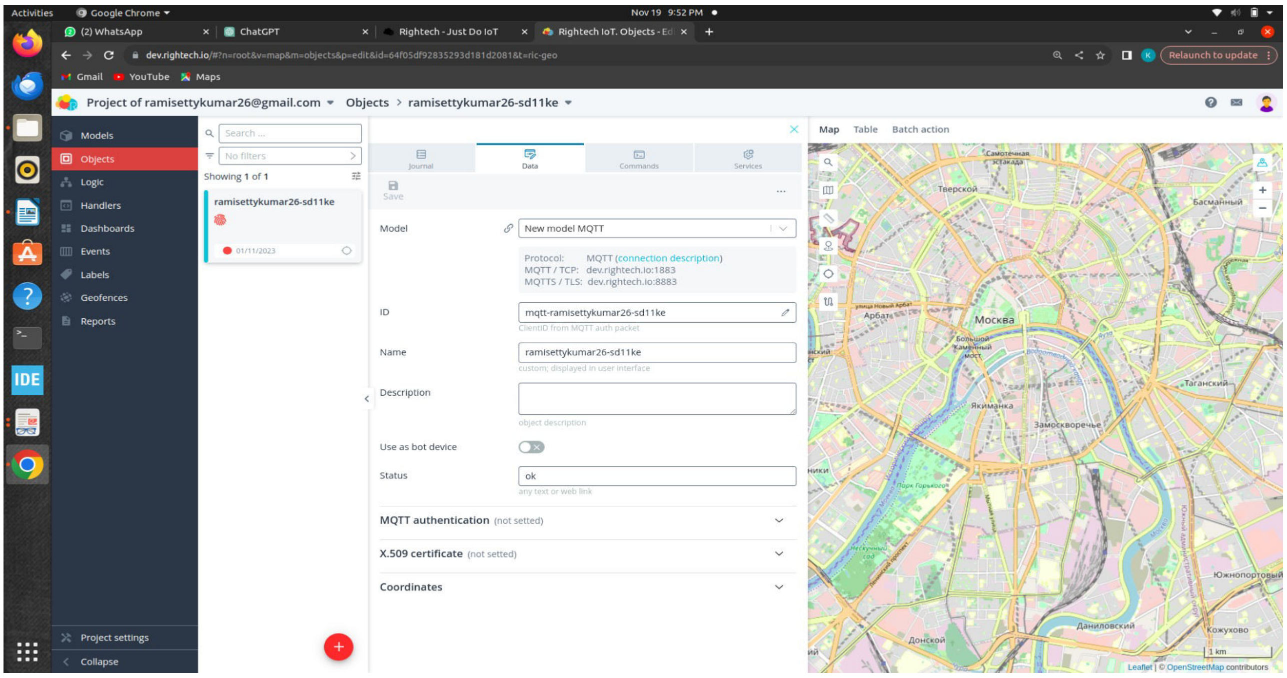


or use your e-mail (or username) and password

Language	English
E-mail / username	ramisettykumar26@gmail.com
Password (forgot?)	••••••••••

• Explore the Platform: Navigate through the platform to understand its features, dashboard, and settings. You should look for an option related to creating or managing parameters, which are typically settings or values used to configure and control IoT devices or data.

• Create Parameters: Depending on the platform's interface and options, you may find a section where you can create parameters or configure settings for your IoT devices or applications.



Model: New model MQTT

Protocol: MQTT (connection description)
MQTT / TCP: dev.rightech.io:1883
MQTTS / TLS: dev.rightech.io:8883

ID: mqtt-ramisettykumar26-sd11ke
ClientID from MQTT auth packet

Name: ramisettykumar26-sd11ke
custom; displayed in user interface

Description:
object description

Use as bot device

Status: ok
any text or web link

MQTT authentication (not setted)

X.509 certificate (not setted)

Coordinates

4. Stages

We have different stages to do this module:

- STM32 to Minicom
- STM32 to Minicom & Rightech IOT Cloud
- STM32 to Rugged board a5d2x
- Rugged board a5d2x to Rightech IOT Cloud

4.1 STM32 to Minicom

4.1.1 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.

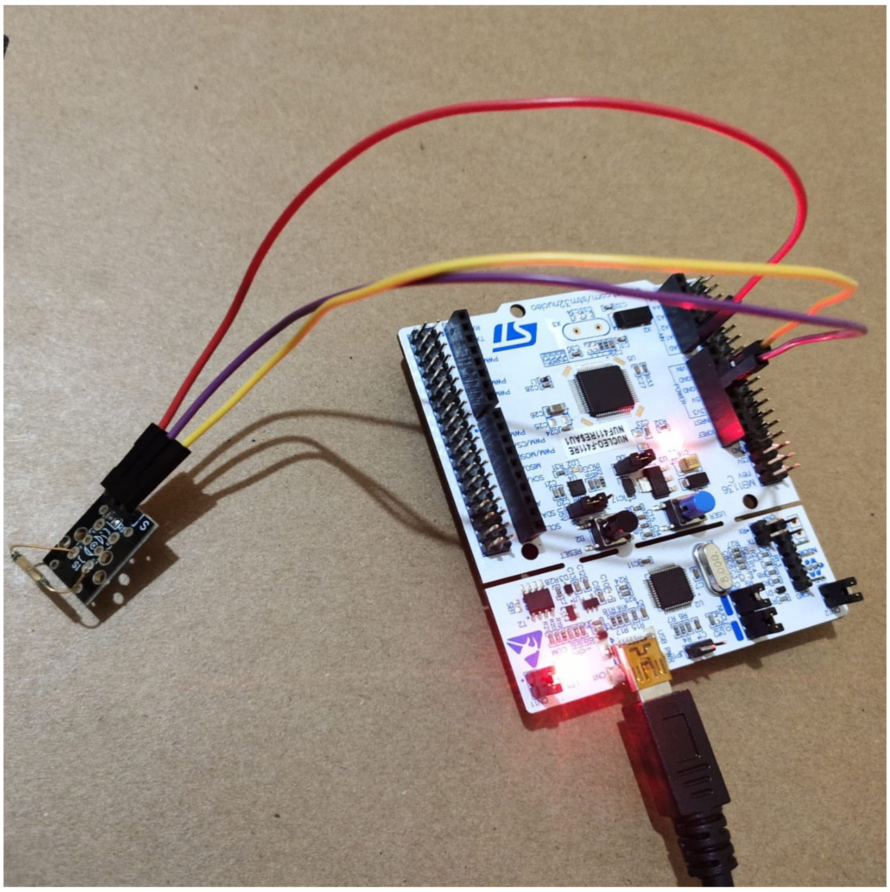


Fig: stage1 pin connections

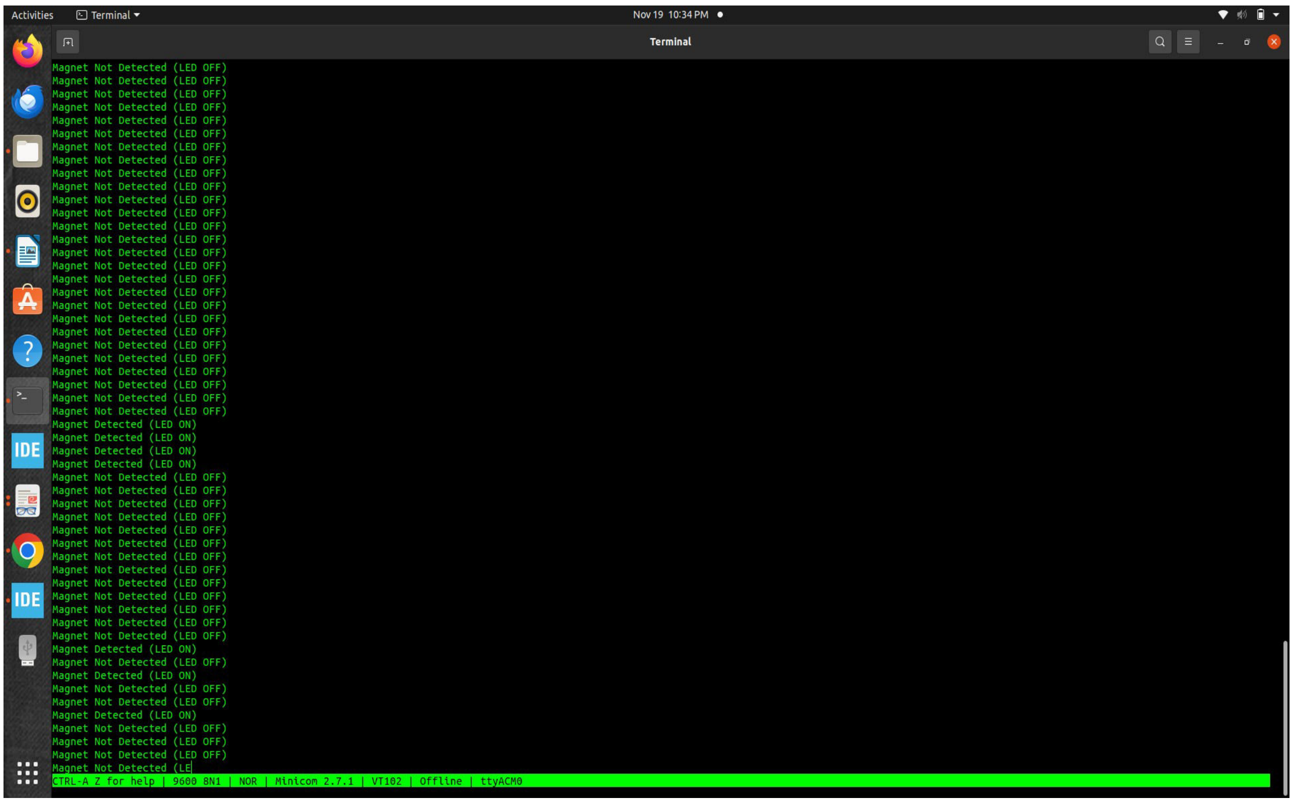
4.1.2 Code

```
while (1)
{
   if(HAL_GPIO_ReadPin(GPIO_PORT, REED_SWITCH_PIN)==GPIO_PIN_RESET)
     {
        HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_SET
        sprintf(message, "Magnet Detected (LED ON)\r\n");
        HAL_UART_Transmit(&uart_handle, (uint8_t*)message,strlen(message),HAL_MAX_DELAY);
     }
   else
      {
        HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_RESET);
        sprintf(message, "Magnet Not Detected (LED OFF)\r\n");
        HAL_UART_Transmit(&uart_handle, (uint8_t*)message, strlen(message),HAL_MAX_DELAY);
      }
      HAL_Delay(1000);
}
```

4.1.3 Explaination

- This code, running in an embedded system, continuously monitors a reed switch using an STM32 microcontroller.
- If the reed switch is closed (indicating the presence of a magnet), it turns on an LED and sends a message over UART stating "Magnet Detected (LED ON)".
- If the reed switch is open (no magnet detected), it turns off the LED and sends a message saying "Magnet Not Detected (LED OFF)".
- The process repeats with a 1-second delay between iterations. This setup could be part of a larger system for magnet sensing and reporting.

4.1.4 Output



```
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LE
CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0
```

4.2 STM32 to Minicom & Rightech IOT Cloud

4.2.1 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 R_X to STM UART1 T_X(D8) and WE10 T_X to STM UART1 R_X(D2).

• Connect WE10 GND to STM32 GND.

• Connect WE10 VCC to STM VCC(3.3V).

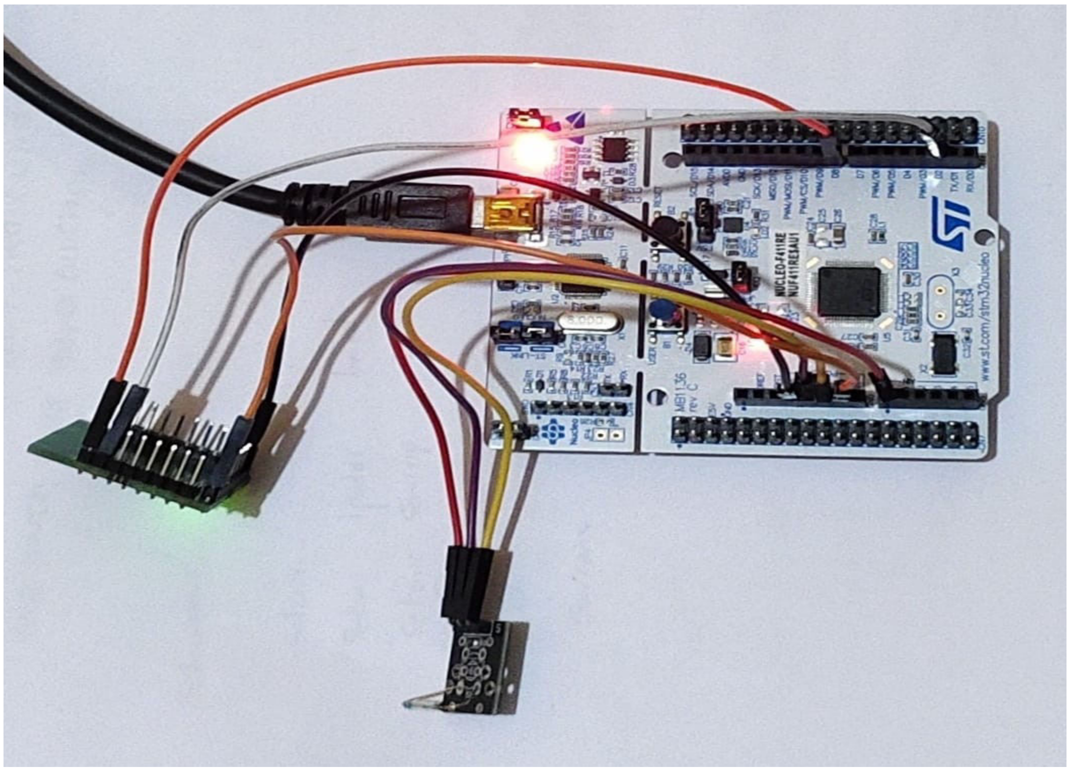


Fig: Stage2 Pin Connections

4.2.2 Code

```
void WE10_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=vivo 1917,11111111\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);
}
void MQTT_Init()
{
  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-ramisettykumar26-sd11ke,,,,,,,,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");
```

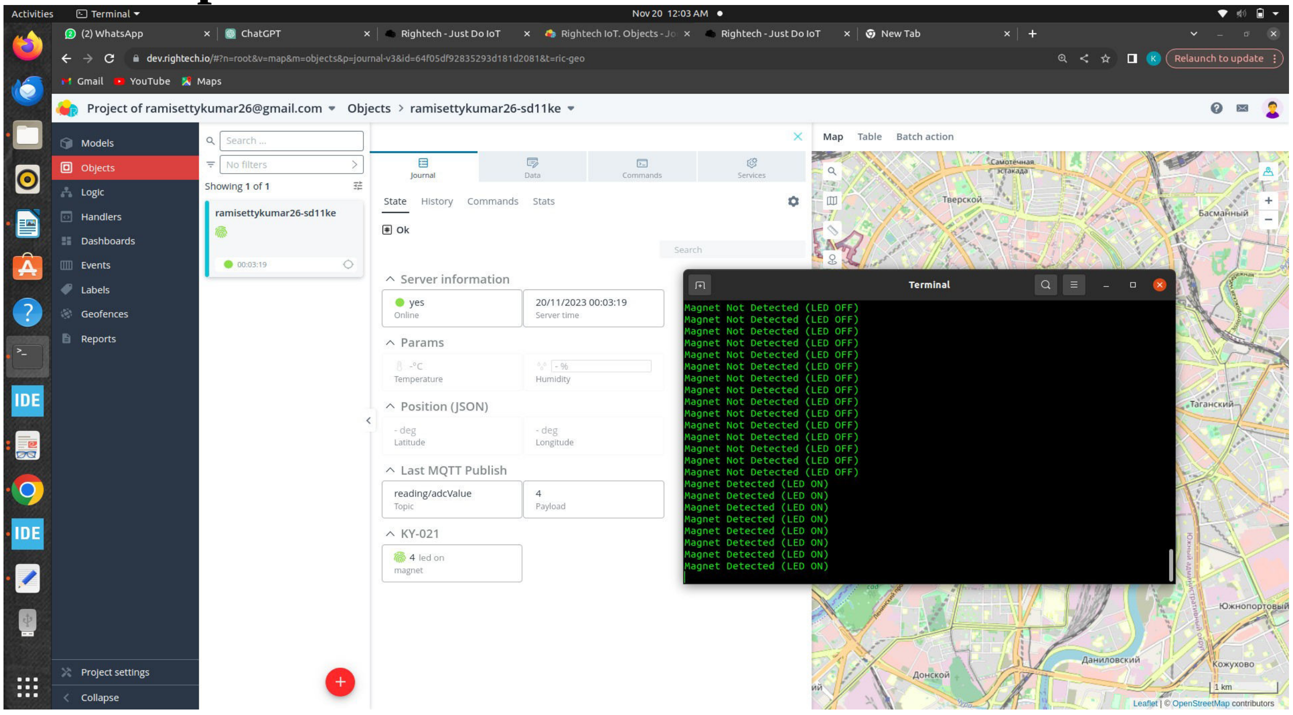
```
        HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_Delay(5000);
        HAL_UART_Receive(&huart1, (uint8_t*)buffer,  strlen(buffer),  1000);
        HAL_Delay(500);
        HAL_UART_Transmit(&huart2, (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)
  {
    readSensor();
      if(value < 15)
       {
          HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_SET);
          sprintf(message, "Magnet Detected (LED ON)\r\n");
          HAL_UART_Transmit(&huart2,(uint8_t*)message,strlen(message),HAL_MAX_DELAY);
          mqtt_data_send(value);
       }
      else
       {
          HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_RESET);
          sprintf(message, "Magnet Not Detected (LED OFF)\r\n");
          HAL_UART_Transmit(&huart2,(uint8_t*)message, trlen(message),
          HAL_MAX_DELAY);
```

```
            mqtt_data_send(value);
        }
            HAL_Delay(1000);
    }
```

4.2.3 Explaination

- Continuous loop monitors sensor for magnet presence.
- If sensor < 15, turns on LED, sends "Magnet Detected" via UART, and transmits sensor data via MQTT.
- If sensor ≥ 15, turns off LED, sends "Magnet Not Detected" via UART, and transmits sensor data via MQTT.
- Utilizes STM32 HAL GPIO functions for LED control.
- Real-time UART communication for status updates.
- LED and messages dynamically reflect magnet status changes.
- Overall, the code efficiently manages magnet detection with LED indication and communication through UART and MQTT.

4.2.4 Output



Ok

Server information
yes
Online
20/11/2023 00:03:19
Server time

Params
- °C
Temperature
- %
Humidity

Position (JSON)
- deg
Latitude
- deg
Longitude

Last MQTT Publish
reading/adcValue
Topic
4
Payload

KY-021
4 led on
magnet

Terminal

```
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
```

4.3 STM32 to Minicom & Rugged Board a5d2x

4.3.1 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 UART1 T_X(D8) to R_X (UART3) of rugged board-a5d2x.
- Connect STM GND to Rugged board –a5d2x GND.

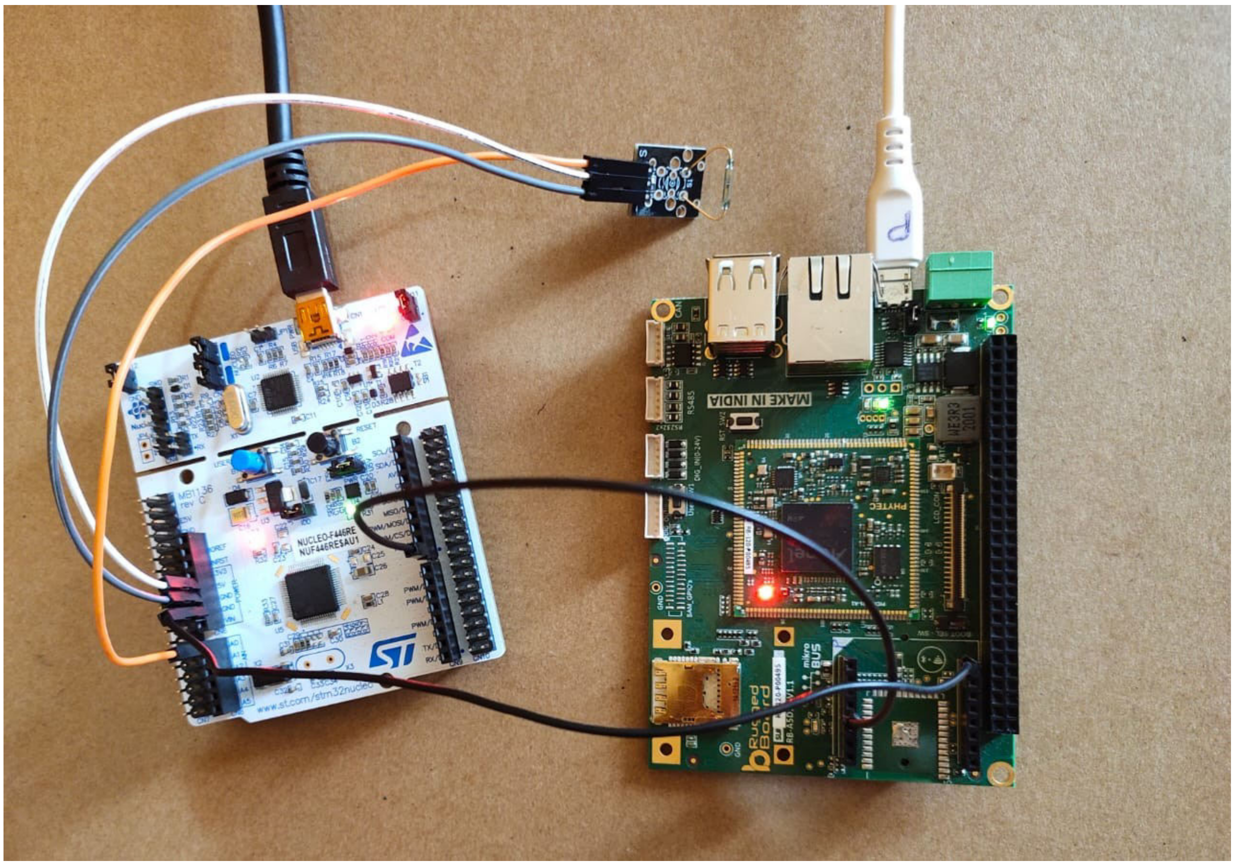


Fig: Stage3 Pin Connections

4.3.2 Code For STM32

```
while (1)
 {
   readSensor();
     if(value < 15)
       {
           HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_SET);
           sprintf(message, "Magnet Detected (LED ON) with adcvalue = %dr\n", value);
           HAL_UART_Transmit(&huart2,(uint8_t*)message,strlen(message), HAL_MAX_DELAY);
           HAL_UART_Transmit(&huart1,(uint8_t*)message,strlen(message), HAL_MAX_DELAY);
       }
       else
       {
         HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_RESET);
         sprintf(message, "Magnet Not Detected (LED OFF) with adcvalue = %d\r\n",value);
         HAL_UART_Transmit(&huart2, (uint8_t*)message, strlen(message), HAL_MAX_DELAY);
         HAL_UART_Transmit(&huart1,(uint8_t*)message, strlen(message), HAL_MAX_DELAY);
        }
          HAL_Delay(1000);
  }
```

4.3.2 Code For Rugged Board

```
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
```

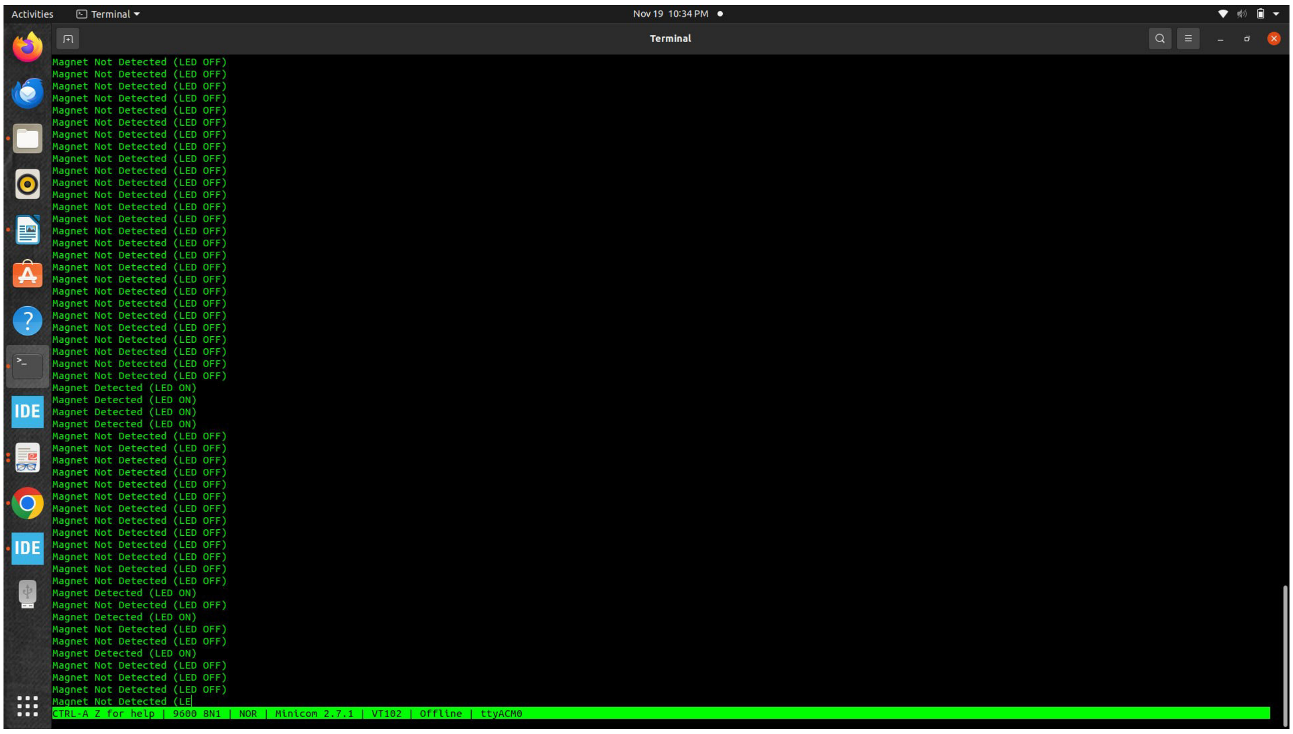
```
int set_interface_attribs(int fd, int speed)
 {
   struct termios tty;
   if (tcgetattr(fd, &tty) < 0)
     {
      printf("Error from tcgetattr: %s\n", strerror(errno));
      return -1;
     }
   cfsetispeed(&tty, (speed_t)speed);
   tty.c_cflag |= (CLOCAL | CREAD); /* Ignore modem controls */
   tty.c_cflag &= ~CSIZE;
   tty.c_cflag |= CS8; /* 8-bit characters */
   tty.c_cflag &= ~PARENB; /* No parity bit */
   tty.c_cflag &= ~CSTOPB; /* Only need 1 stop bit */
   tty.c_cflag &= ~CRTSCTS; /* No hardware flow control */
   tty.c_iflag = IGNPAR;
   tty.c_lflag = 0;
   tty.c_cc[VMIN] = 1;
   tty.c_cc[VTIME] = 1;
   if (tcsetattr(fd, TCSANOW, &tty) != 0)
     {
      printf("Error from tcsetattr: %s\n", strerror(errno));
      return -1;
     }
   return 0;
}
int main()
  {
   char *portname = "/dev/ttyS3";
   int fd;
   int rdlen;
   unsigned char buf[256]; // Adjust buffer size as needed
   fd = open(portname, O_RDWR | O_NOCTTY | O_SYNC);
   if (fd < 0)
     {
      printf("Error opening %s: %s\n", portname, strerror(errno));
      return -1;
     }
   if (set_interface_attribs(fd, B9600) != 0)
    {
```

```
    close(fd);
    return -1;
  }
 while (1)
   {
    rdlen = read(fd, buf, sizeof(buf) - 1);
    if (rdlen > 0)
     {
       buf[rdlen] = '\0';
       printf("Received data: %s\n", buf);
     }
   else
    {
       printf("No data received.\n");
    }
 }
  close(fd);
  return 0;
 }
```

4.3.3 Explaination

- The STM32 code continuously reads a sensor, controls an LED, and transmits sensor data via UART2 to minicom and UART1 to the rugged board.
- If sensor < 15, LED is turned on, and "Magnet Detected" message is sent with sensor value to both UART2 and UART1.
- The rugged board code initializes UART3, opens the serial port "/dev/ttyS3," and configures it for a baud rate of 9600.
- In an infinite loop, the rugged board code reads data from UART3, prints received data if available, and notifies if no data is received.
- Both devices use common baud rates and message formats for effective communication.
- UART configurations such as baud rates and buffer sizes are adjustable in both codes for flexibility.
- Both codes include error handling mechanisms for UART initialization and data reception.

4.3.4 Output



```
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LE
CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0
```

4.4 STM32 to Minicom & Rugged Board And Rugged Board to Rightech Cloud

4.4.1 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 UART1 T_X(D8) to R_X (UART3) of Rugged board.
- Connect WE10 R_X to Rugged board-a5d2x T_X(UART3).
- Connect WE10 GND to Rugged board-a5d2x GND.
- Connect WE10 VCC to Rugged board-a5d2x (3.3v).

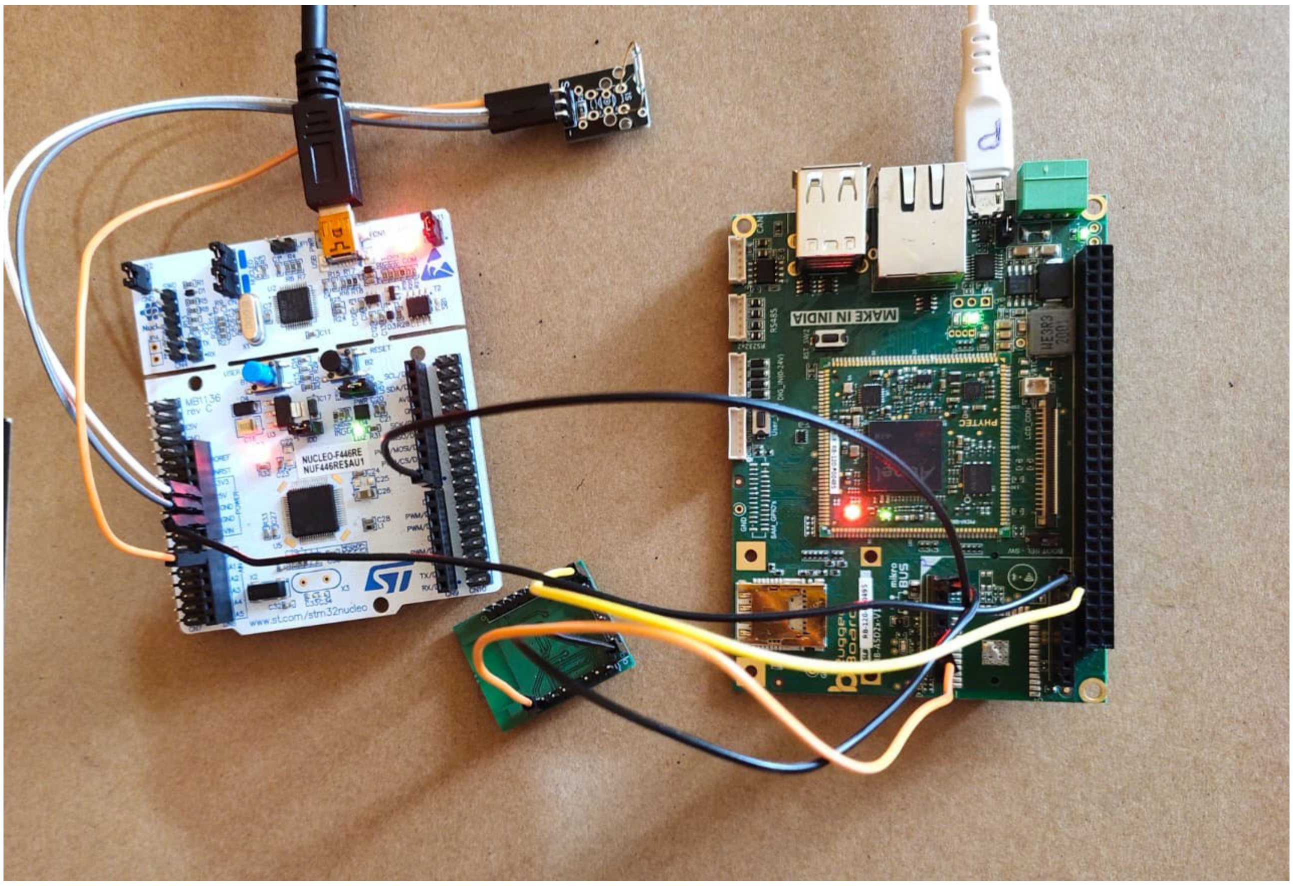


Fig: Stage4 Pin Configuration

4.4.2 Code For STM32

```
while (1)
 {
   readSensor();
     if(value < 15)
       {
           HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_SET);
           sprintf(message, "Magnet Detected (LED ON) with adcvalue = %dr\n", value);
           HAL_UART_Transmit(&huart2,(uint8_t*)message,strlen(message), HAL_MAX_DELAY);
           HAL_UART_Transmit(&huart1,(uint8_t*)message,strlen(message), HAL_MAX_DELAY);
       }
else
    {
     HAL_GPIO_WritePin(GPIO_PORT, LED_PIN, GPIO_PIN_RESET);
     sprintf(message, "Magnet Not Detected (LED OFF) with adcvalue = %d\r\n",value);
      HAL_UART_Transmit(&huart2, (uint8_t*)message, strlen(message), HAL_MAX_DELAY);
     HAL_UART_Transmit(&huart1,(uint8_t*)message, strlen(message), HAL_MAX_DELAY);
    }
      HAL_Delay(1000);
 }
```

4.4.3 Code For Rugged Board

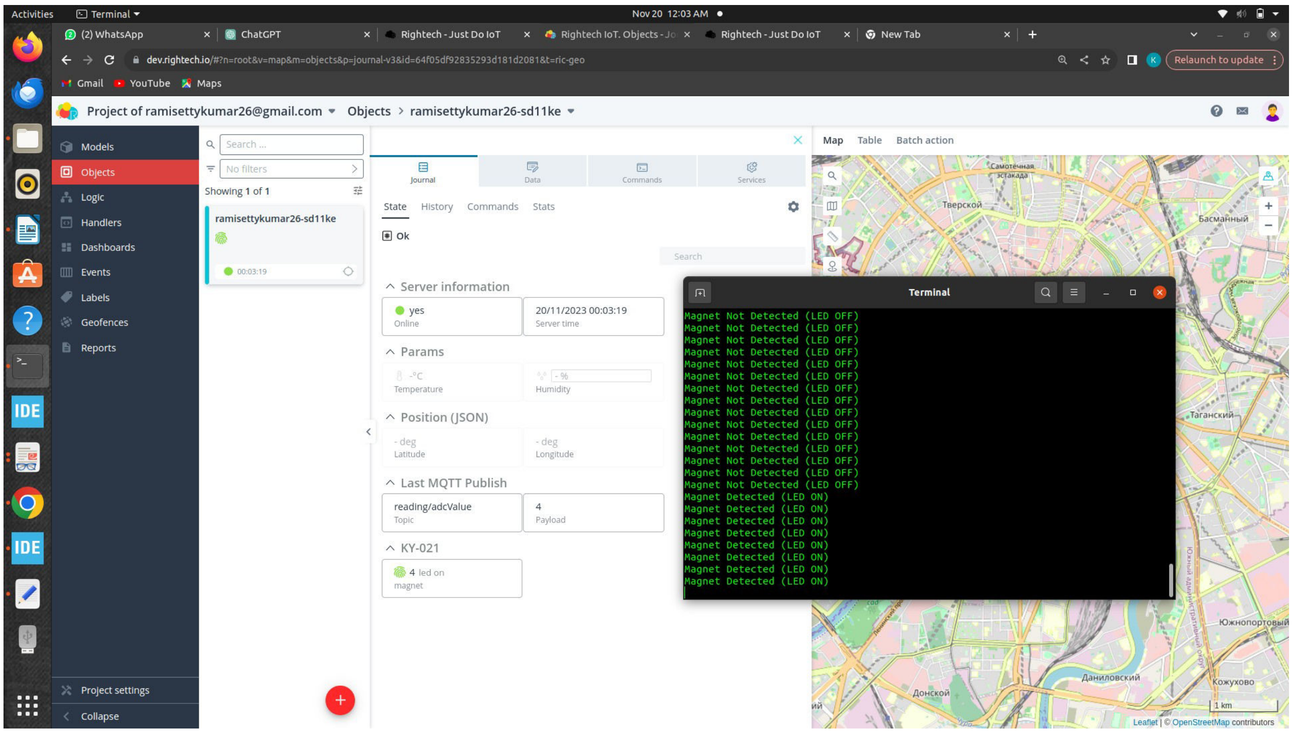
```
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
int set_interface_attribs(int fd, int speed)
 {
        struct termios tty;
        if (tcgetattr(fd, &tty) < 0)
        {
                printf("Error from tcgetattr: %s\n", strerror(errno));
                return -1;
        }
        cfsetispeed(&tty, (speed_t)speed);
        tty.c_cflag |= (CLOCAL | CREAD); /* ignore modem controls */
        tty.c_cflag &= ~CSIZE;
        tty.c_cflag |= CS8; /* 8-bit characters */
        tty.c_cflag &= ~PARENB; /* no parity bit */
        tty.c_cflag &= ~CSTOPB; /* only need 1 stop bit */
        tty.c_cflag &= ~CRTSCTS; /* no hardware flowcontrol */
        tty.c_iflag = IGNPAR;
        tty.c_lflag = 0;
        tty.c_cc[VMIN] = 1;
        tty.c_cc[VTIME] = 1;
        if (tcsetattr(fd, TCSANOW, &tty) != 0)
         {
            printf("Error from tcsetattr: %s\n", strerror(errno));
            return -1;
         }
        return 0;
 }
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=\"vivo 1917\",\"11111111\"\r\n";
    char arr3[] = "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n";
    char arr4[] = "CMD+MQTTCONCFG=3,mqtt-ramisettykumar26-sd11ke,,,,,,,,\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=1
    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;
}
```

4.4.3 Explaination

- The program configures a WE10 module through UART3 communication. It sends commands for resetting, setting Wi-Fi mode, connecting to an access point, configuring MQTT parameters, and starting MQTT.
- After configuring, it enters an infinite loop, continuously reading data from the serial port.
- The received data is printed, and formatted MQTT messages are created and sent back to the device.
- The loop runs indefinitely with a 3-second delay after each transmission. The serial port is closed after exiting the loop.

4.4.4 Output



Server information

yes Online	20/11/2023 00:03:19 Server time

Params

- °C Temperature	- % Humidity

Position (JSON)

- deg Latitude	- deg Longitude

Last MQTT Publish

reading/adcValue Topic	4 Payload

KY-021

4 led on magnet

```
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Not Detected (LED OFF)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
Magnet Detected (LED ON)
```

5. Real Time Applications

The KY-021 mini reed switch sensor can be employed in various real-time applications due to its sensitivity to magnetic fields. Some common real-time applications include

1. Door and Window Security Systems:
   - The reed switch can be used to detect the opening and closing of doors and windows.
   - In security systems, it can trigger an alarm or alert when unauthorized access is detected.
2. Proximity Sensing:
   - The sensor can be used for proximity sensing in industrial automation and robotics.
   - It can detect the presence or absence of magnetic objects in close proximity.
3. Automotive Applications:
   - Reed switches are used in automotive applications for sensing the position of vehicle doors and trunks.
   - They can also be employed in speed sensors for applications like bicycle speedometers.
4. Liquid Level Sensing:
   - Placing a magnet on a float and using the reed switch can create a liquid level sensor.
   - This can be applied in tanks or containers to monitor and control liquid levels.
5. Home Automation:
   - Reed switches can be integrated into smart home systems to monitor the status of doors, windows, or cabinets.
   - They can trigger specific actions, such as turning on lights or adjusting the thermostat, based on the magnetic field changes.

6. Medical Devices:
   - Reed switches can be used in certain medical devices to detect the position of movable parts.
   - For example, they might be employed in infusion pumps or other medical equipment.
7. Counting and Position Sensing:
   - In industrial applications, the reed switch can be used for counting revolutions or detecting the position of moving parts in machinery.
8. Magnetic Triggering in Electronics:
   - Reed switches are used in electronic devices like relays for magnetic triggering.
   - They can control the flow of electrical current based on the presence or absence of a magnetic field.
9. Wireless Switches:
   - Reed switches can be used in wireless switches that activate or deactivate devices remotely.
   - These can be applied in smart home setups or industrial control systems.
10. Burglar Alarms:
   - Reed switches are commonly used in burglar alarm systems to detect unauthorized entry through doors or windows.
   - The opening of a door or window breaks the magnetic field, triggering the alarm.

These real-time applications showcase the versatility of the KY-021 mini reed switch sensor in various industries and electronic systems.

6. References

1. https://arduinomodules.info/?s=ky-021+mini+reed+switch+module
2. https://oshwlab.com/adrirobot/KY-021-Mini-magnetic-reed-module
3. https://sensorkit.joy-it.net/en/sensors/ky-021
4.https://www.phippselectronics.com/using-the-mini-magnetic-reed-module-ky-021-with-arduino/

THANK YOU!