KY-038(MICROPHONE SMALL SOUND SENSOR MODULE)

-Presented by

S. Manindra

SBCS India pvt ltd.

TABLE OF CONTENTS:

1.PROJECT OVERVIEW:

The project configures a KY-038 Small Sound Sensor through serial communication across 4 stages initialization, device configuration, sensor data collection, and MQTT setup inSTM32 Microcontroller and Rugged board .

It establishes serial communication which reads analog values as input and provides digital output by blniking led and configures Wi-Fi and MQTT settings, reads sensor data, and transmits sound detection status via UART.

Additionally, it continuously reads serial data, transforms it into MQTT messages, and publishes it to a predefined topic. Potential improvements include enhancing error handling, optimizing code efficiency, modularizing functions, and implementing secure data handling practices for network communication. Ultimately, it orchestrates device setup, data collection, and remote monitoring through MQTT in a structured, iterative process.

2.HARDWARE COMPONENTS:

1.KY-038(SMALL SOUND SENSOR)

2.STM32F411RE

3.Rugged board –a5d2x

4.WE10 module

3.SOFTWARE COMPONENTS:

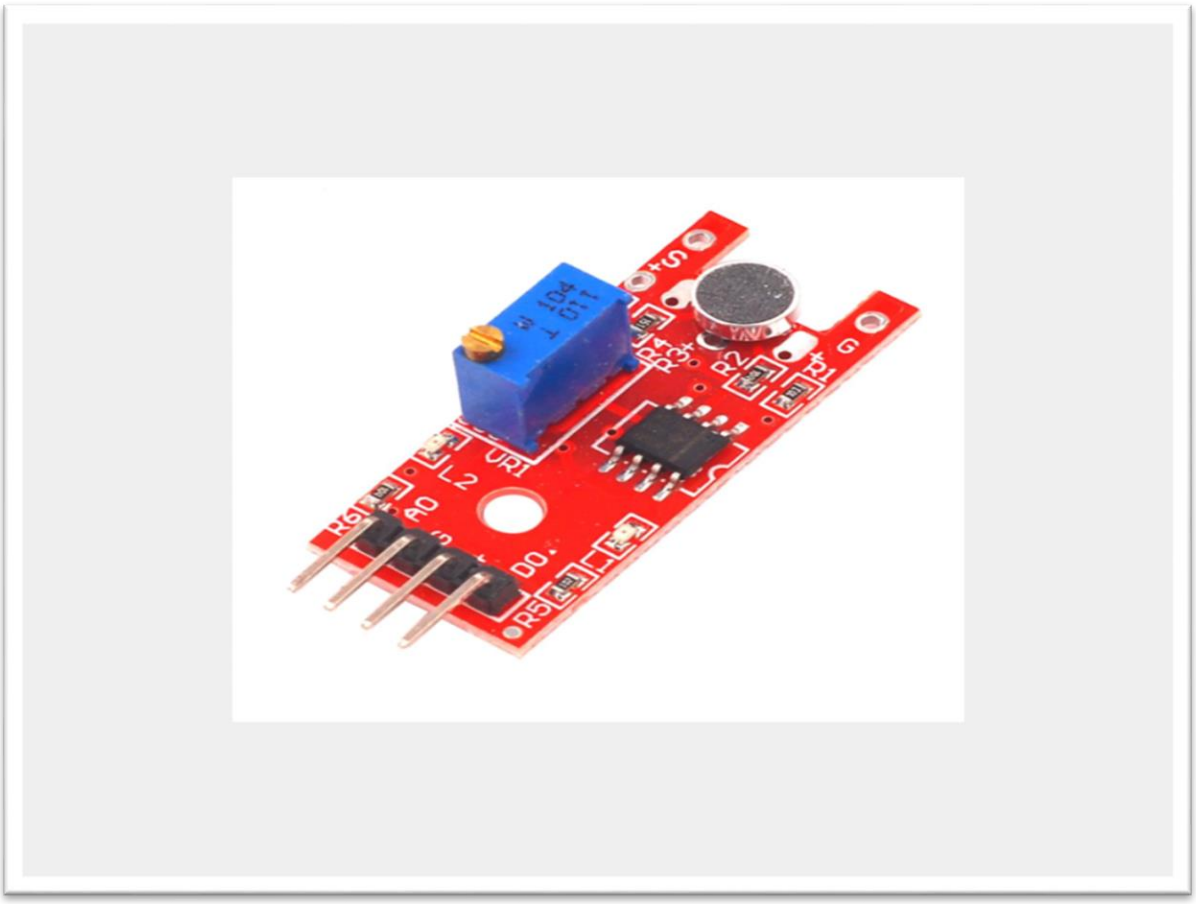
1.STM32 IDE tool

2.minicom

3.Right tech IOT cloud

HARDWARE COMPONENTS:

1.KY-038(SMALL SOUND SENSOR):



The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.

The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.

You can control the sensitivity by adjusting the potentiometer. (Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.)

This sensor doesn't show absolute values (like exact temperature in °C or magnetic field strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

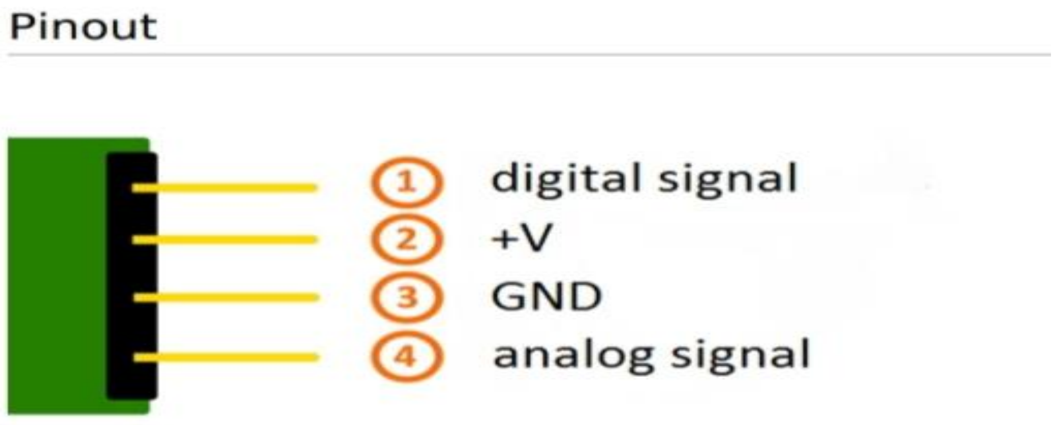
Module Features:

- **Sound Detection:** Sensitive to small sound variations and changes in the surrounding environment.

- **Analog Output:** Provides an analog output signal that varies based on the intensity of the detected sound.
- **AdjustableSensitivity:** Allows sensitivity adjustments through a potentiometer for customizing detection levels.
- **Integrated LM393 Comparator:** Equipped with an LM393 voltage comparator chip to enhance signal stability.
- **Versatile Interface:** Compatible with microcontrollers and Arduino boards through its analog output.
- **Low Power Consumption:** Operates on low power, making it suitable for energy-efficient applications.
- **Simple Integration:** Compact and easy to connect to various electronic circuits or projects.

Module Pinout:

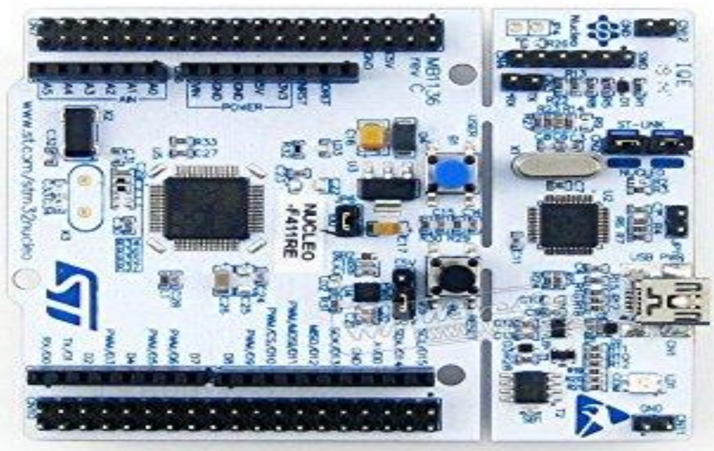
- digital Signal = [Pin 3]
- +V = [Pin 5V]
- GND = [Pin GND]
- analog Signal = [Pin 0]

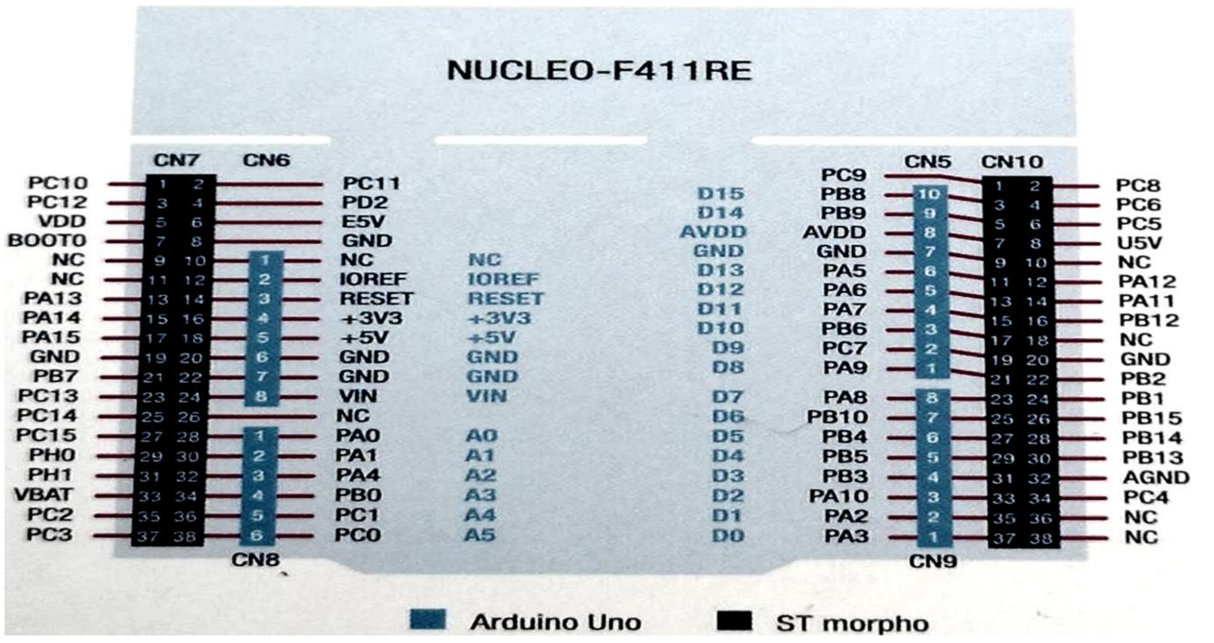


2.STM32F411RE:

- The STM32F446xC/E devices are based on the high-performance Arm Cortex -M4 32-bit RISC core operating at a frequency of up to 180 MHz.The Cortex-M4 core features a floating point unit (FPU) single precision supporting all Arm single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) that enhances application security.
- The STM32F446xC/E devices incorporate high-speed embedded memories (Flash memory up to 512 Kbytes, up to 128 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/O's and peripherals connected to two APB buses, two AHB buses and a32-bit multi-AHB bus matrix.
- All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers.
- They also feature standard and advanced communication interfaces.
- Integration of services from STM32CubeMX:STM32 microcontroller, microprocessor, development platform and example project selection Pinout, clock, peripheral, and middleware configuration Project creation and generation of the initialization code Software and middleware completed with enhanced STM32Cube Expansion Packages

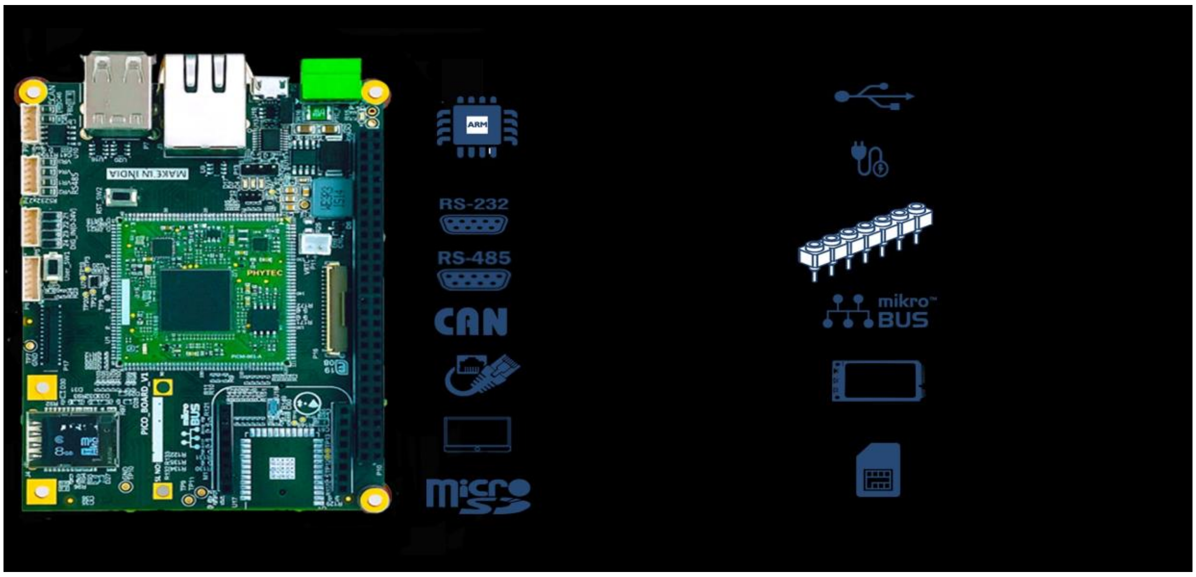
- Based on Eclipse ® /CDT™, with support for Eclipse ® add-ons, GNUC/C++ for Arm ® toolchain and GDB debugger
- STM32MP1 Series: Support for Open ST Linux projects: Linux Support for Linux
- Additional advanced debug features including: CPU core, peripheral register, and memory




NUCLEO-F411RE
CN7
CN6
CN5
CN10
CN8
CN9
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN
A0 A1 A2 A3 A4 A5
D15 D14 AVDD GND D13 D12 D11 D10 D9 D8
D7 D6 D5 D4 D3 D2 D1 D0
PC9 PB8 PB9 AVDD GND PA5 PA6 PA7 PB6 PC7 PA9
PA8 PB10 PB4 PB5 PB3 PA10 PA2 PA3
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 NC GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
Arduino Uno
ST morpho

3.Rugged board-a5d2x :

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



4.WE10 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 itcan 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

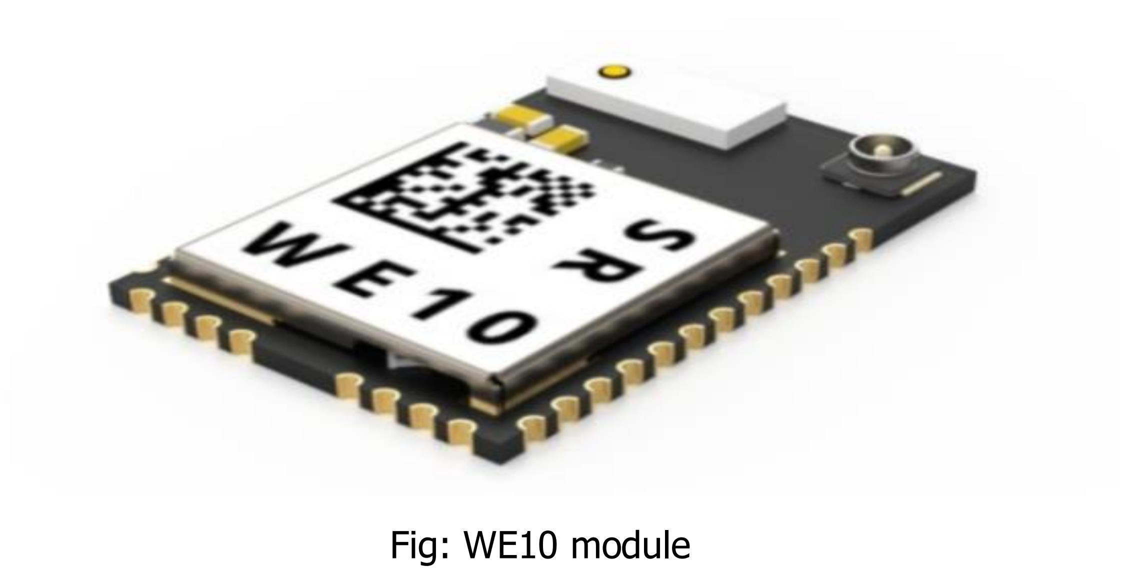


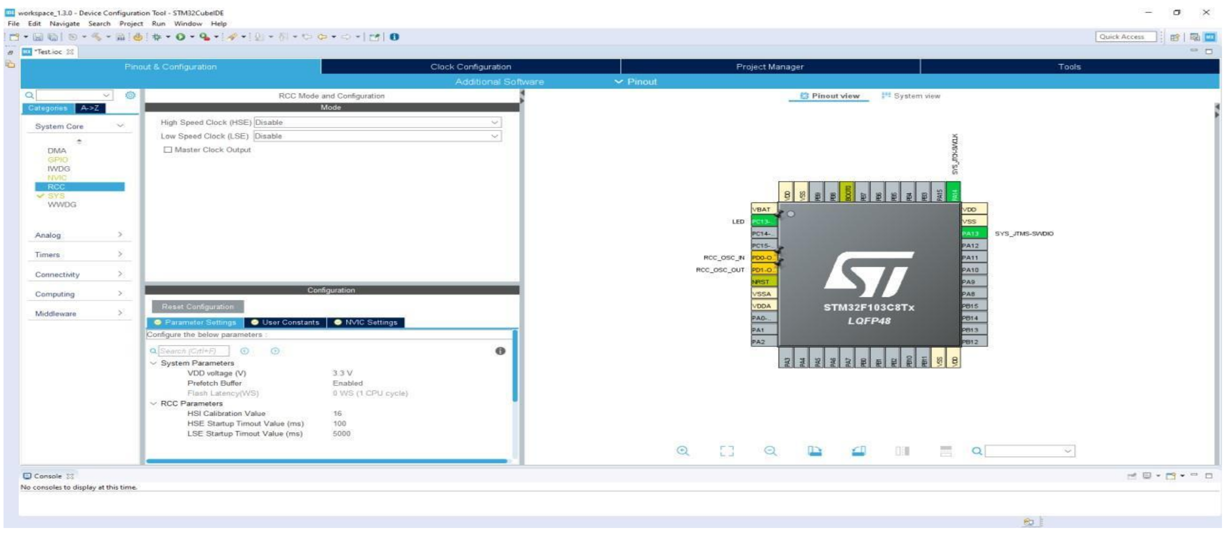
Fig: WE10 module

SOFTWARE COMPONENTS:

1.STM32 IDE tool:

The STM32 Integrated Development Environment (IDE) stands as a robust platform tailored for STM32 microcontrollers, serving as a one-stop solution for software development. It operates seamlessly across various operating systems, including Windows, macOS, and Linux, providing a versatile environment for developers. Its user-friendly interface streamlines the coding experience, featuring essential tools such as syntax highlighting, code completion, and intelligent suggestions.

This IDE simplifies project management by efficiently handling projects, libraries, and build configurations, while also offering debugging tools for real-time code analysis and error resolution. Additionally, it facilitates peripheral configuration and integration through graphical interfaces, complemented by extensive support for STM32Cube libraries, easing hardware abstraction and peripheral usage. Its compatibility spans across a wide range of STM32 microcontrollers and development boards. With an active community and reliable technical support, the STM32 IDE ensures a comprehensive and efficient development journey for software engineers and STM32 enthusiasts.



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.

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 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. For example, on a Debian-based system, you can use:
- Bash
- sudo apt-get install minicom

2. Connecting to a Serial Port:

- Minicom is primarily used for serial communication, so you need to 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:

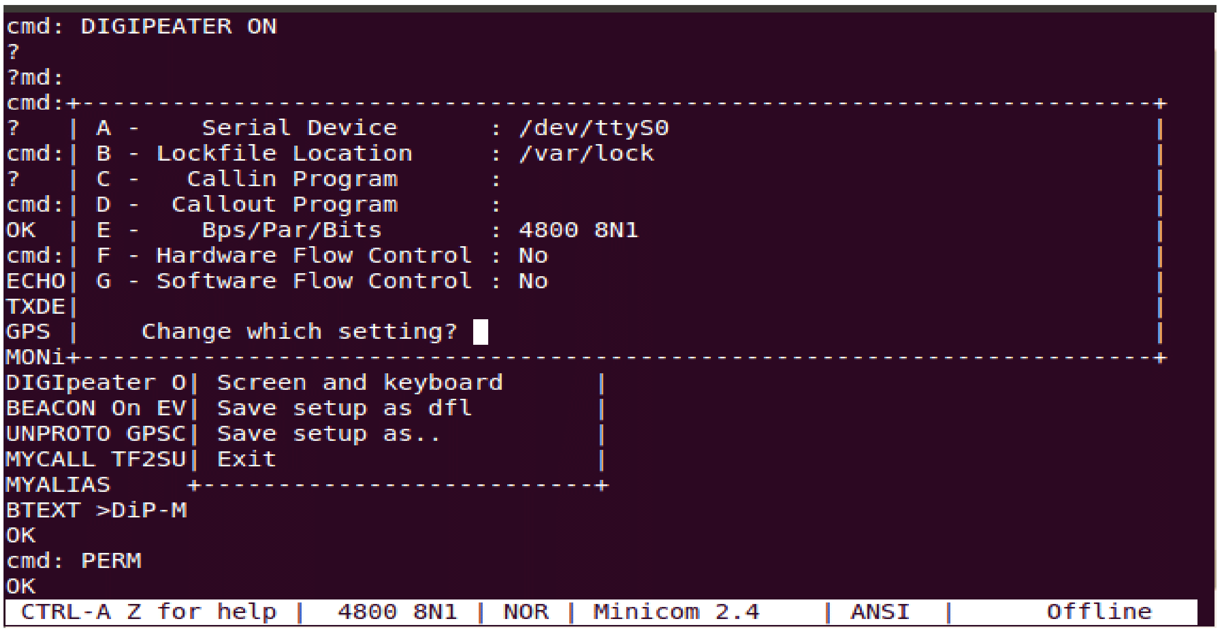
- After configuring the serial port, you can interact with the device. Minicom allows you to send commands and receive responses. This is particularly useful for debugging purposes and for configuring devices that have a serial console.

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.



```
cmd: DIGIPEATER ON
?
?md:
cmd:+-----------------------------------------------------------------------+
?    | A -    Serial Device      : /dev/ttyS0                              |
cmd:| B - Lockfile Location     : /var/lock                               |
?    | C -   Callin Program      :                                         |
cmd:| D -  Callout Program      :                                         |
OK   | E -    Bps/Par/Bits       : 4800 8N1                                |
cmd:| F - Hardware Flow Control : No                                      |
ECHO| G - Software Flow Control : No                                      |
TXDE|                                                                    |
GPS |    Change which setting?                                          |
MONi+-----------------------------------------------------------------------+
DIGIpeater O| Screen and keyboard      |
BEACON On EV| Save setup as dfl        |
UNPROTO GPSC| Save setup as..          |
MYCALL TF2SU| Exit                     |
MYALIAS     +--------------------------+
BTEXT >DiP-M
OK
cmd: PERM
OK
 CTRL-A Z for help |  4800 8N1 | NOR | Minicom 2.4      | ANSI   |       Offline
```

Fig: minicom

3.RIGHTTECH IOT CLOUD :

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

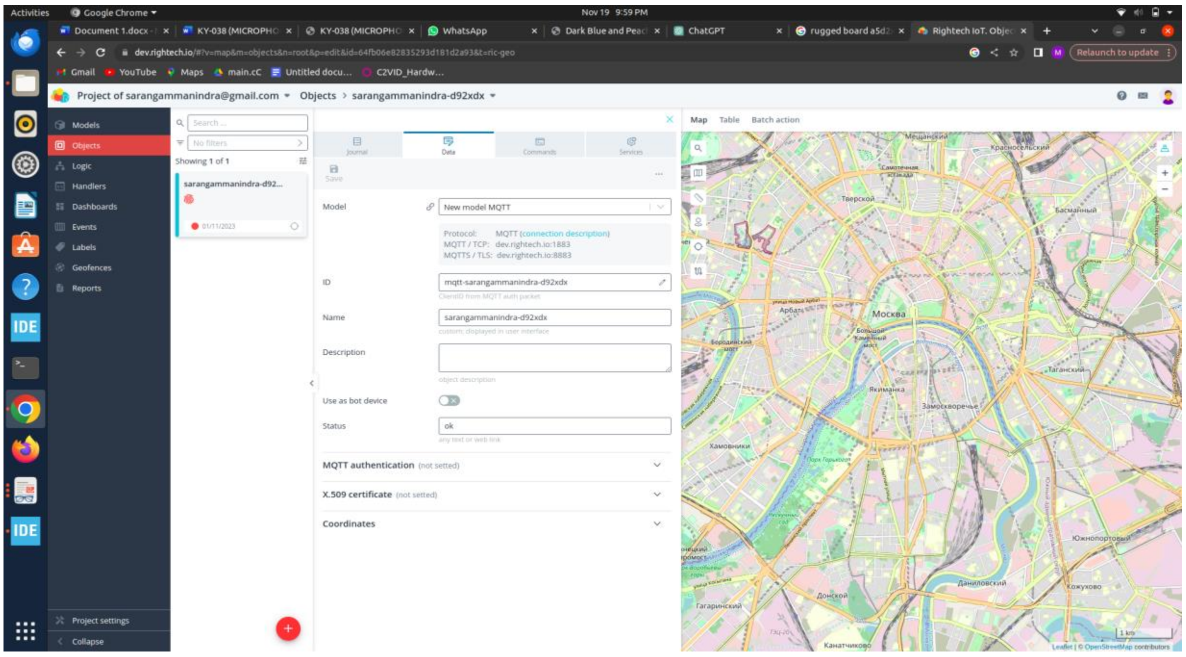
Registration: Look for a "Register" or "Sign Up" option on their website. Click on it to start the registration process.

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.

- 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. Follow the platform's instructions to create the parameters you need.


Project of sarangammanindra@gmail.com › Objects › sarangammanindra-d92xdx

Model: New model MQTT
Protocol: MQTT (connection description)
MQTT / TCP: dev.rightech.io:1883
MQTTS / TLS: dev.rightech.io:8883
ID: mqtt-sarangammanindra-d92xdx
Name: sarangammanindra-d92xdx
Description:
Use as bot device
Status: ok
MQTT authentication (not setted)
X.509 certificate (not setted)
Coordinates

fig: Rightech IOT cloud ID identification

4.STAGES:

We have different stages to do this module:

Stages-1:STM32 to minicom.

Stages-2: STM32 to Righttech IOT cloud.

Stages-3: STM32 to rugged board a5d2x.

Stages-4: rugged board a5d2x to Righttech IOT cloud.

4.1 STM32 to MINICOM:

4.1.1 Module connections:

The connections between the STM32F411RE and the KY-038 small sound sensor module are established as follows:

- **Analog Input(Sensor to STM32):** Connect the analog output pin of the sound sensor module to pin PA0 (Analog Input) on GPIO Port A of the STM32.
- **Power and Ground(Sensor to STM32)**: Link the VCC pin of the sound sensor to the appropriate power source (e.g., 5V) and connect the GND (ground) pin to the ground (GND) on the STM32.
- **Signal Processing(Sensor to STM32):** Utilize the sensor's analog signal output connected to pin PA0, enabling the microcontroller to process this signal through its built-in Analog-to-Digital Converter (ADC).
- **LED Control (STM32):** Establish a connection from pin PA5 (Digital Output) on GPIO Port A of the STM32F to control an LED, potentially indicating sound detection based on the programmed threshold value. Configure this pin as an output to manage the LED state.

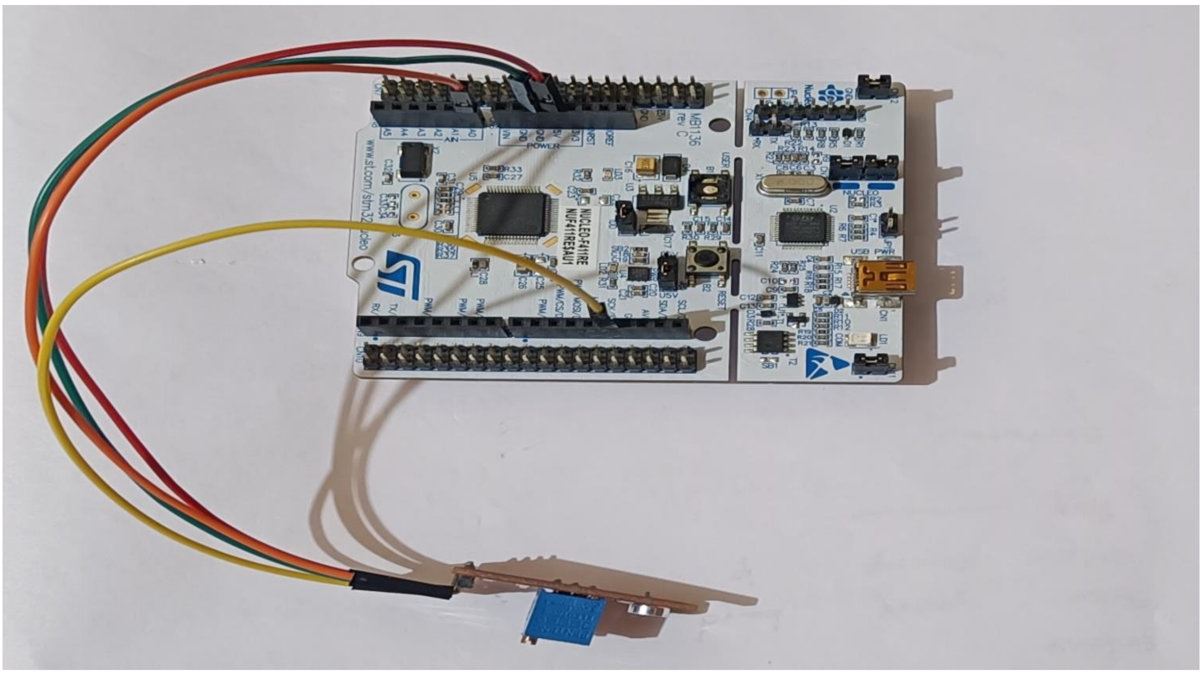


Fig: Stage-1 Pin Configuration

4.1.2 Simple HAL code for STM32:

```
while (1)
   {
     HAL_ADC_Start(&hadc1);
     if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
     {
       uint16_t analogValue = HAL_ADC_GetValue(&hadc1);
        if (analogValue > threshold)
       {
         HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_SET);
         sensor_value=1;
         sprintf(message, "sound is detected = %d\r\n", sensor_value);
         HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
         HAL_Delay(1000);
```

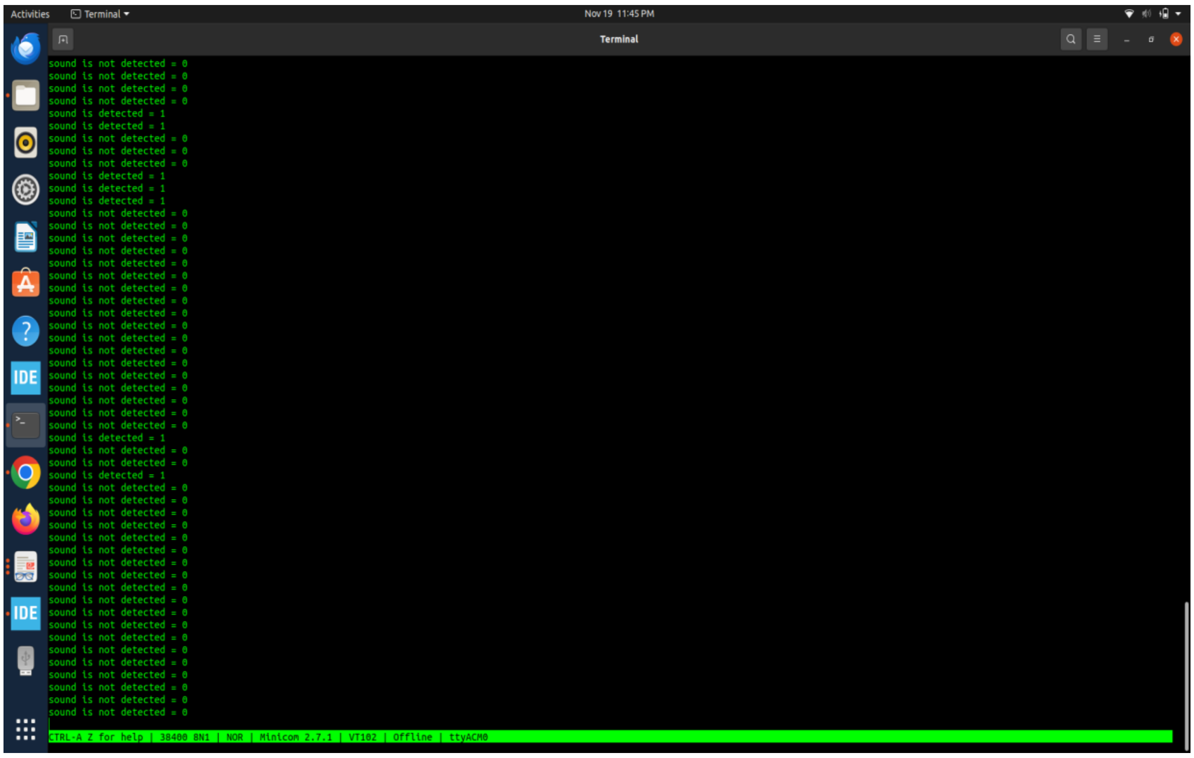
```
      }
      else
      {
        HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_RESET);
        sensor_value=0;
        sprintf(message, "sound is not detected = %d\r\n", sensor_value);
        HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
        HAL_Delay(1000);
      }
    }
    HAL_ADC_Stop(&hadc1);
  }
 }
```

4.1.3 CODE EXPLANATION:

- Loop Initialization: Begins an infinite loop for continuous operation.
- Start ADC: Initiates analog sensor reading via the STM32's ADC (Analog-to-Digital Converter).
- Read Analog Value: Obtains the converted analog value from the sensor.
- Threshold Check: Compares the obtained value against a preset threshold.
- LED Control & Messaging:If sound surpasses the threshold, Lights up an LED.Transmits a message via UART signaling sound detection. If sound doesn't exceed the threshold,Turns off the LED. Sends a message via UART indicating no sound detection.

- Stop ADC: Concludes the ADC reading process.Loop Continuation: Returns to the start of the loop for continuous monitoring.

4.1.4 OUTPUT:



```
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is detected = 1
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0

CTRL-A Z for help | 38400 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0
```

4.2. STM32 to Righttech IOT cloud:

4.2.1 Module connections:

The connections between the STM32F411RE and the KY-038 small sound sensor module and WE10 module are established as follows:

- **Analog Input(Sensor to STM32):** Connect the analog output pin of the sound sensor module to pin PA0 (Analog Input) on GPIO Port A of the STM32.

- **Power and Ground(Sensor to STM32)**: Link the VCC pin of the sound sensor to the appropriate power source (e.g., 5V) and connect the GND (ground) pin to the ground (GND) on the STM32.
- **Signal Processing(Sensor to STM32):** Utilize the sensor's analog signal output connected to pin PA0, enabling the microcontroller to process this signal through its built-in Analog-to-Digital Converter (ADC).
- **LED Control (STM32):** Establish a connection from pin PA5 (Digital Output) on GPIO Port A of the STM32 to control an LED, potentially indicating sound detection based on the programmed threshold value. Configure this pin as an output to manage the LED state.
- Connect WE10 module RX to STM32 UART1 TX and TX to STM32 UART RX.
- Connect WE10 GND to STM32 GND.
- Connect WE10 VCC to STM32 VCC(3.3v).

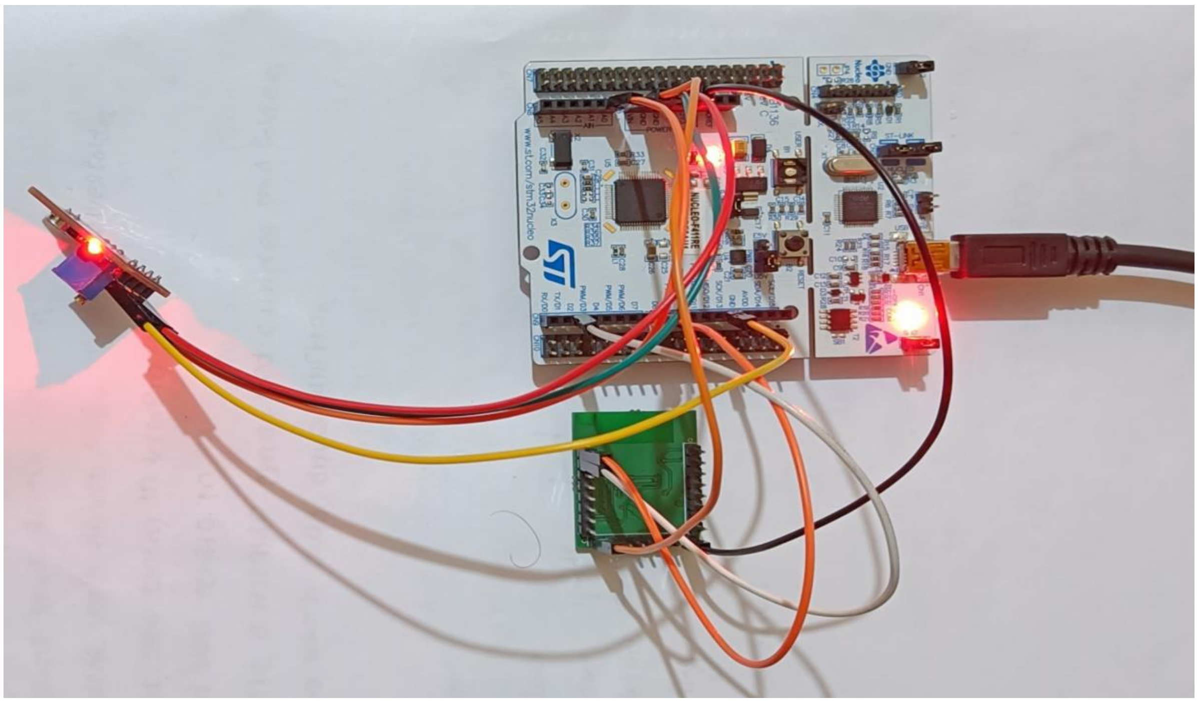


Fig: Stage-2 Pin Configuration

4.2.2 Simple HAL code for STM32 to Righttech IOT Cloud:

```
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_Delay(500);
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-sarangammanindra-
d92xdx,,,,,,,,,\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);
```

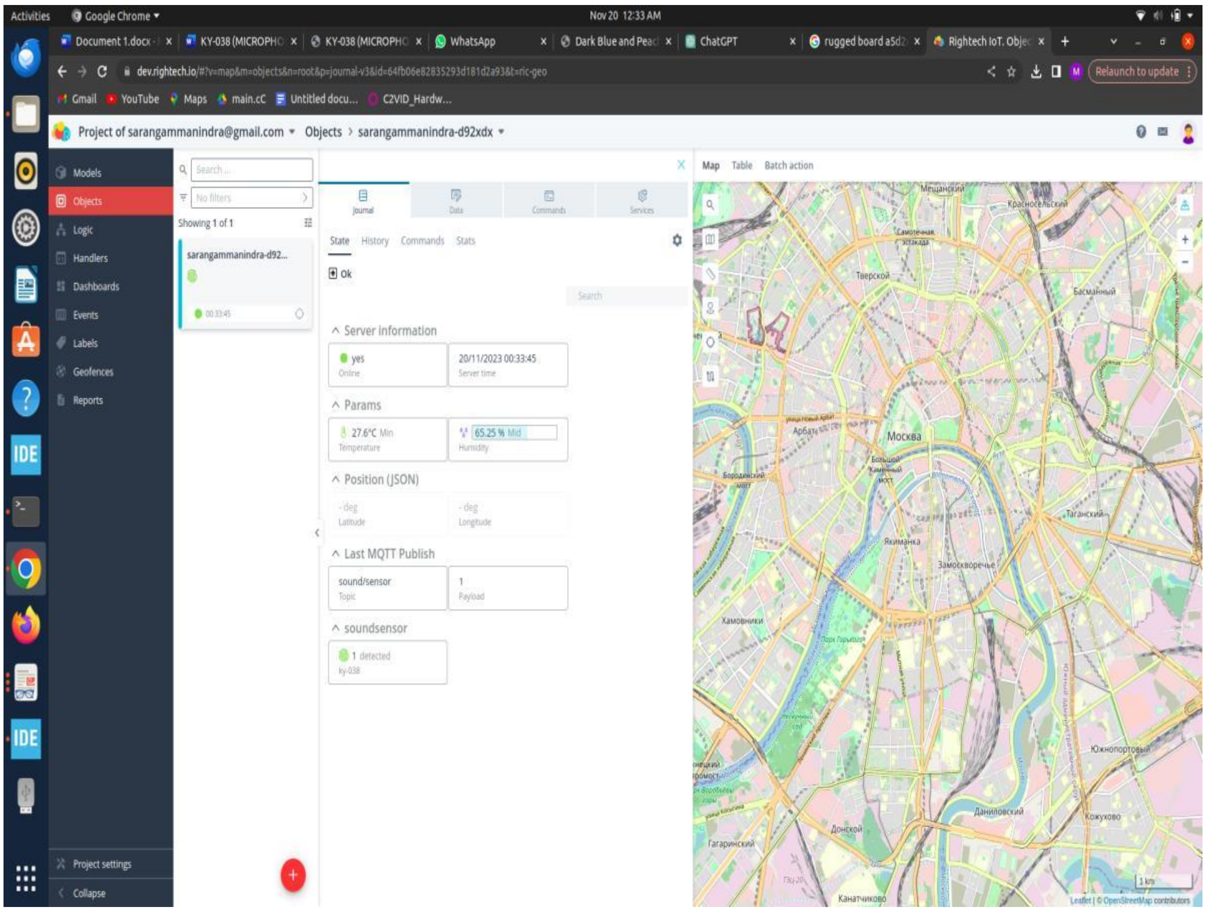
```
            HAL_Delay(5000);
            HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
            HAL_Delay(500);
            HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

            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);
      if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
      {
        uint16_t analogValue = HAL_ADC_GetValue(&hadc1);
        if (analogValue > threshold)
        {
             HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_SET);
             sensor_value=1;
             sprintf(message, "sound is detected = %d\r\n", sensor_value);
             HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
             mqtt_data_send(1);
             HAL_Delay(1000);
        }
        else
        {
          HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_RESET);
          sensor_value=0;
          sprintf(message, "sound is not detected = %d\r\n", sensor_value);
         HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
         mqtt_data_send(0);
          HAL_Delay(1000);
        }
      }
      HAL_ADC_Stop(&hadc1);
    }
  }
```

4.2.3 CODE EXPLANATION:

- The code continuously monitors an analog input (PA0) for sound detection using an ADC. It starts the ADC conversion and checks if the analog value surpasses a defined threshold.
- If the analog value exceeds the threshold, it activates an LED on pin PA5 and sends a message indicating sound detection via UART and MQTT.
- When no sound is detected (analog value below the threshold), it turns off the LED, updates sensor_value, sends a corresponding message via UART and MQTT.
- The program employs a delay of 1 second after each detection/no-detection event using HAL_Delay(1000). It utilizes HAL functions from the STM32 HAL library for ADC, GPIO, UART, and MQTT operations.

- The sprintf function formats messages to display the status of sound detection along with the sensor_value (1 if sound detected, 0 if not).
- The code continuously loops through this process, consistently checking for sound and updating LED and messaging accordingly.
- It effectively controls the LED output on PA5 based on the sound threshold and sends status updates via UART and MQTT. The loop iterates indefinitely, maintaining the monitoring and response mechanism for sound detection.

4.2.4 OUTPUT:


Project of sarangammanindra@gmail.com > Objects > sarangammanindra-d92xdx

Server information
Online: yes
Server time: 20/11/2023 00:33:45

Params
Temperature: 27.6°C Min
Humidity: 65.25 % Mid

Position (JSON)
Latitude: - deg
Longitude: - deg

Last MQTT Publish
Topic: sound/sensor
Payload: 1

soundsensor
ky-038: 1 detected

4.3 STM32 to Rugged Board-a5d2x:

4.3.1 Module connections:

The connections between the STM32F411RE and the KY-038 small sound sensor module and WE10 module are established as follows:

- **Analog Input(Sensor to STM32):** Connect the analog output pin of the sound sensor module to pin PA0 (Analog Input) on GPIO Port A of the STM32.
- **Power and Ground(Sensor to STM32)**: Link the VCC pin of the sound sensor to the appropriate power source (e.g., 5V) and connect the GND (ground) pin to the ground (GND) on the STM32.
- **Signal Processing(Sensor to STM32):** Utilize the sensor's analog signal output connected to pin PA0, enabling the microcontroller to process this signal through its built-in Analog-to-Digital Converter (ADC).
- **LED Control (STM32):** Establish a connection from pin PA5 (Digital Output) on GPIO Port A of the STM32 to control an LED, potentially indicating sound detection based on the programmed threshold value. Configure this pin as an output to manage the LED state.
- Connect the STM32 UART TX keeping for the rugged board- a5d2x(UART1) to RX (UART3) of rugged board-a5d2x.
- Connect STM32 GND TO rugged board –a5d2x GND.

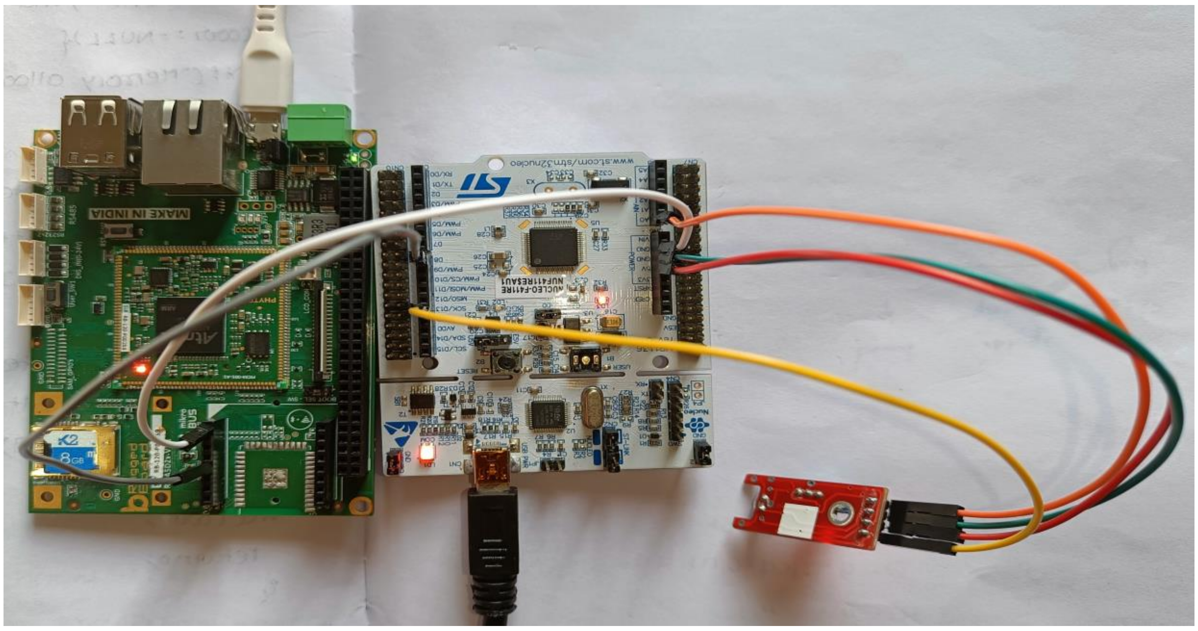


Fig: Stage-3 Pin Configuration

4.3.2 Simple HAL code for STM32 and Rugged Board a5d2x:

6.2.1 CODE FOR STM32:

```
while (1)
  {
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
    {
      uint16_t analogValue = HAL_ADC_GetValue(&hadc1);
       if (analogValue > threshold)
      {
        HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_SET);
        sensor_value=1;
        sprintf(message, "sound is detected = %d\r\n", sensor_value);
        HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
        HAL_Delay(1000);
      }

else
      {
        HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_RESET);
        sensor_value=0;
        sprintf(message, "sound is not detected = %d\r\n", sensor_value);
        HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
        HAL_Delay(1000);
      }
    }
    HAL_ADC_Stop(&hadc1);
  }
}
```

6.2.2 CODE FOR Rugged Board a5d2x:

```
#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);
```

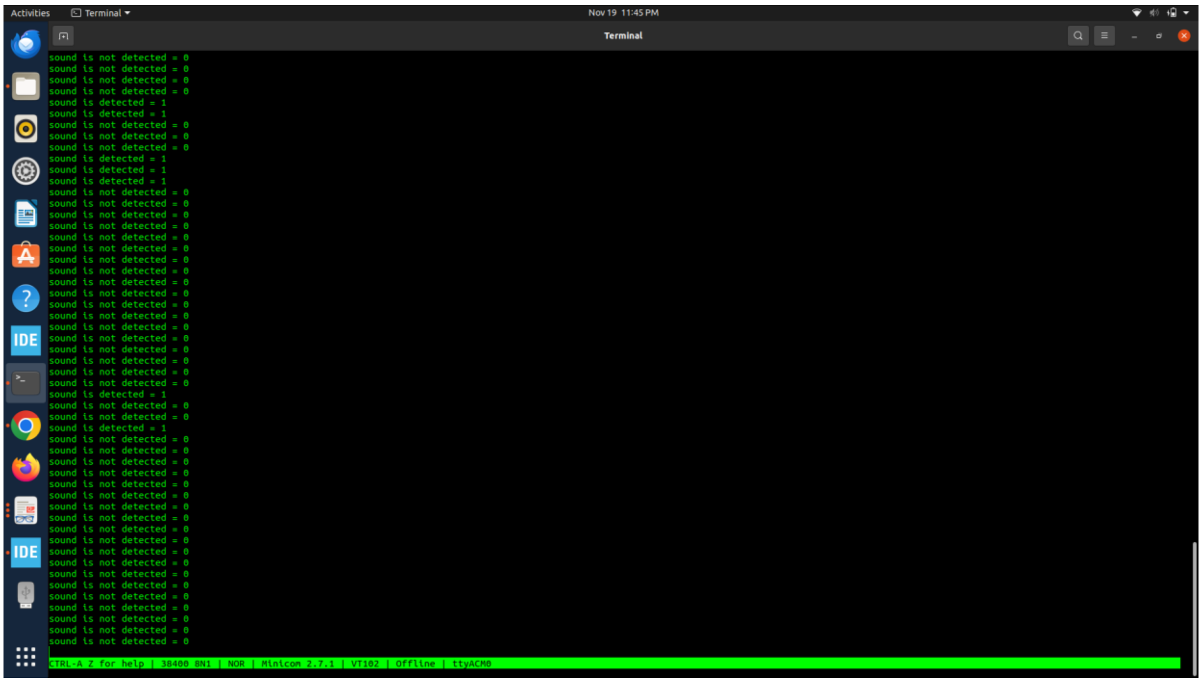
```
    tty.c_cflag |= (CLOCAL | CREAD); /* Ignore modem controls */
    tty.c_cflag &= ~CSIZE;
    tty.c_cflag |= CS8;           /* 8-bit characters */
    tty.c_cflag &= ~PARENB;       /* No parity bit */
    tty.c_cflag &= ~CSTOPB;       /* Only need 1 stop bit */
    tty.c_cflag &= ~CRTSCTS;      /* 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); // Read data into the buffer
        if (rdlen > 0) {
            buf[rdlen] = '\0'; // Null-terminate the received data
            printf("Received data: %s\n", buf);
        } else {
            printf("No data received.\n");
        }
    }
    close(fd);
    return 0;
}
```

4.3.3 RUGGED BOARD CODE EXPLANATION:

- The set_interface_attribs function configures serial port attributes like baud rate, character size, parity, and stop bits for communication.
- It opens the port in read-write mode without controlling terminal and synchronous mode.
- The set_interface_attribs function sets the serial port attributes to operate at 9600 baud rate and specific configurations for robust data transmission.
- Inside the infinite loop, it continuously reads data from the serial port into a buffer.
- If data is received (read length > 0), it null-terminates the buffer and prints the received data.
- If no data is received, it displays a message indicating the absence of incoming data.
- The code indefinitely loops, continuously monitoring and printing incoming serial data, crucial for rugged board operations.

- The buffer size for reading data can be adjusted by modifying the buf array size.
- It effectively handles serial communication setup and reception, catering to the robust demands of the rugged board environment.
- The program remains in an everlasting loop, ensuring consistent monitoring and display of incoming serial data on the rugged board.

4.3.4 OUTPUT:



```
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is detected = 1
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is detected = 1
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0
sound is not detected = 0

CTRL-A Z for help | 38400 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0
```

4.4 Rugged Board a5d2x to Righttech IOT Cloud.

4.4.1 Module connections:

- **Analog Input(Sensor to STM32):** Connect the analog output pin of the sound sensor module to pin PA0 (Analog Input) on GPIO Port A of the STM32.
- **Power and Ground(Sensor to STM32)**: Link the VCC pin of the sound sensor to the appropriate power source (e.g., 5V) and connect the GND (ground) pin to the ground (GND) on the STM32.
- **Signal Processing(Sensor to STM32):** Utilize the sensor's analog signal output connected to pin PA0, enabling the microcontroller to process this signal through its built-in Analog-to-Digital Converter (ADC).
- **LED Control (STM32):** Establish a connection from pin PA5 (Digital Output) on GPIO Port A of the STM32 to control an LED, potentially indicating sound detection based on the programmed threshold value. Configure this pin as an output to manage the LED state.
- Connect STM TX (UART1) to rugged board-a5d2x RX(UART3).
- Connect WE10 RX to rugged board-a5d2xRX(UART3).
- Connect WE10 GND to rugged board-a5d2x.
- Connect WE10 VCC to rugged board-a5d2x (3.3v).

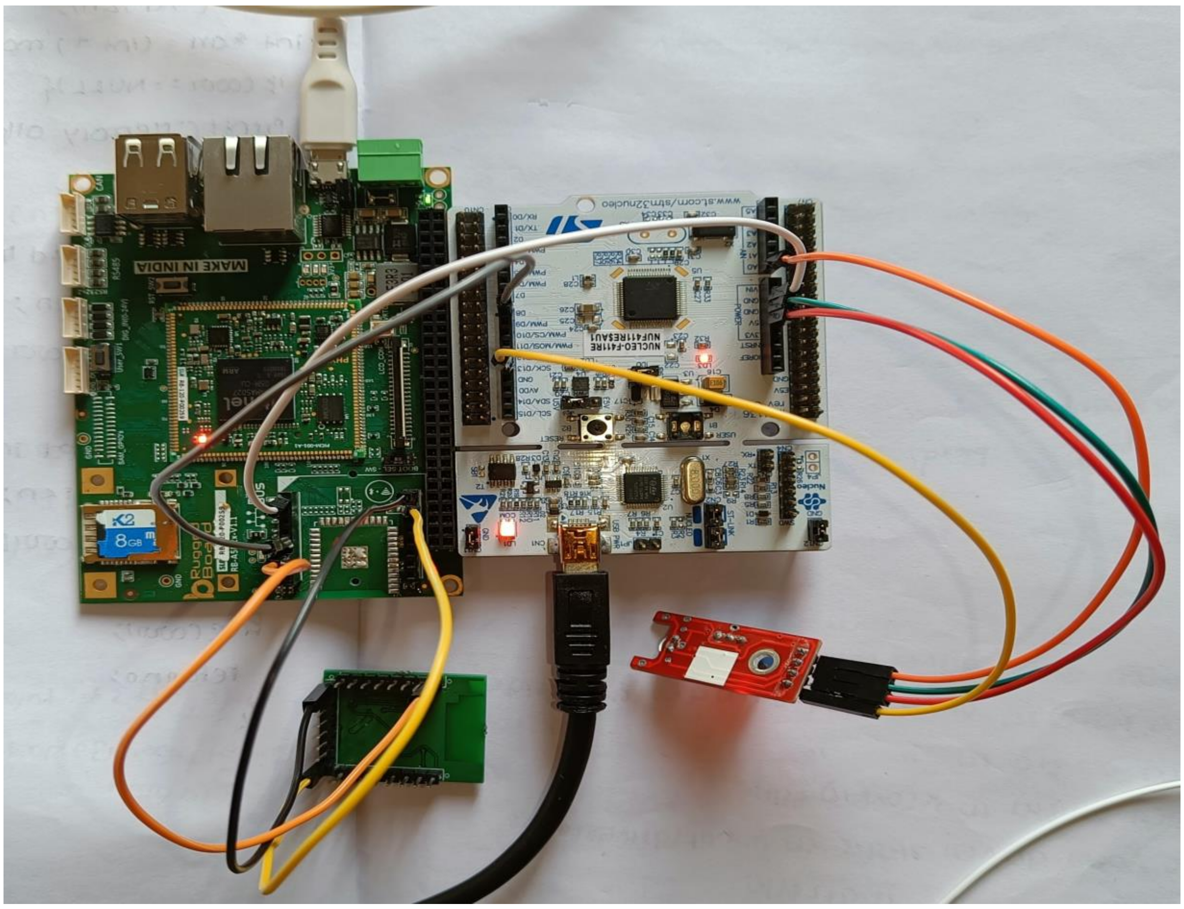


Fig: Stage-4 Pin Configuration

4.4.2 Simple HAL code for STM32 and Rugged Board a5d2x to Righttech IOT Cloud.:

4.4.2.1 CODE FOR STM32:

```
while (1)
  {
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
    {
      uint16_t analogValue = HAL_ADC_GetValue(&hadc1);
       if (analogValue > threshold)
```

```
        {
          HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_SET);
          sensor_value=1;
          sprintf(message, "sound is detected = %d\r\n", sensor_value);
          HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
          HAL_Delay(1000);
        }

else
        {
          HAL_GPIO_WritePin(GPIOA, Digital_Output, GPIO_PIN_RESET);
          sensor_value=0;
          sprintf(message, "sound is not detected = %d\r\n", sensor_value);
          HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message),
HAL_MAX_DELAY);
          HAL_Delay(1000);
        }
      }
      HAL_ADC_Stop(&hadc1);
    }
  }
```

4.4.2.2 CODE FOR Rugged Board a5d2x:

```
#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()
{
```

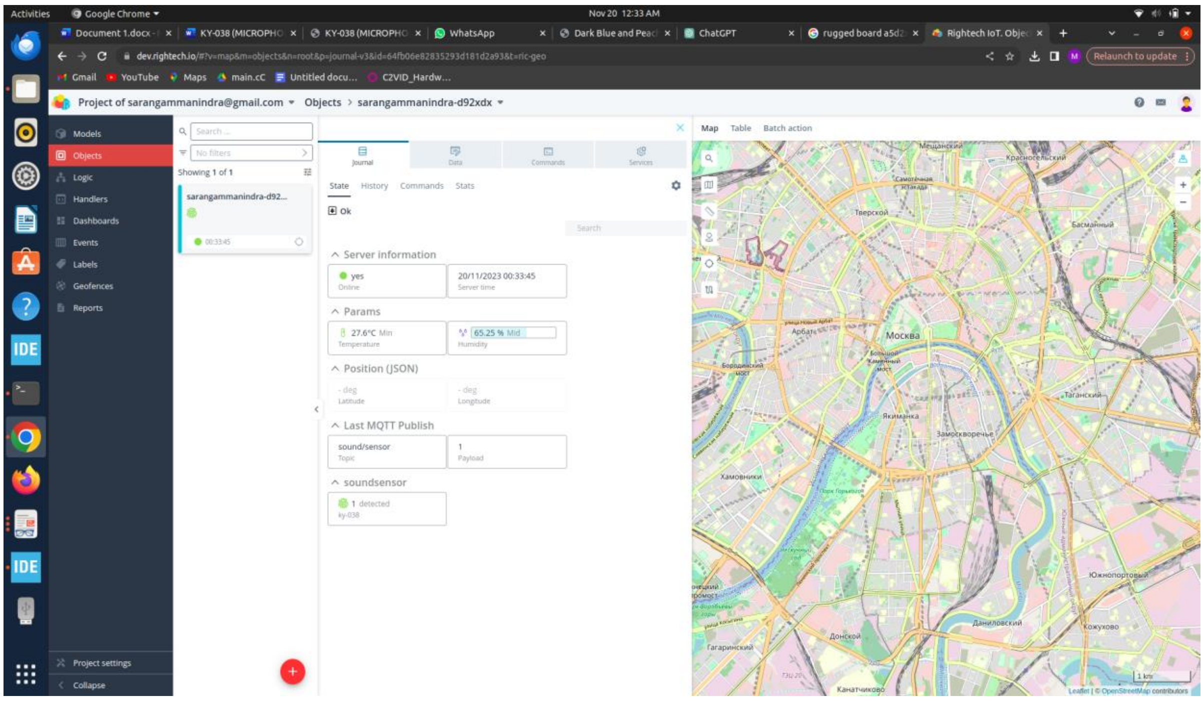
```
char *portname = "/dev/ttyS3";
int fd;
int wlen;
int rdlen;
int ret;

char res[5];
char arr1[] = "CMD+RESET\r\n";
char arr2[] = "CMD+WIFIMODE=1\r\n";
char arr[] = "CMD+CONTOAP=\"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=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 RUGGED BOARD CODE EXPLANATION:

- Rugged Board Code: This programme uses UART3 connectivity to configure a WE10 module. It sends commands to reset the device, change the Wi-Fi mode, connect to an access point, configure MQTT parameters, and activate MQTT.
- After configuring, it begins an unending loop, reading data from the serial port indefinitely.
- The data received is printed, and prepared MQTT messages are constructed and returned to the device.
- The loop continues indefinitely, with a 3-second delay between transmissions. After exiting the loop, the serial port is closed.

4.4.4 OUTPUT:



Project of sarangammanindra@gmail.com › Objects › sarangammanindra-d92xdx

Showing 1 of 1

sarangammanindra-d92...

Ok

Server information

yes
Online

20/11/2023 00:33:45
Server time

Params

27.6°C Min
Temperature

65.25 % Mid
Humidity

Position (JSON)

- deg
Latitude

- deg
Longitude

Last MQTT Publish

sound/sensor
Topic

1
Payload

soundsensor

1 detected
ky-038

5.Advantages:

- High Sensitivity
- Wide Applicability
- Ease of Use
- Compact Size
- Low Power Consumption
- Cost-Effective

6.Disadvantages:

- Limited Range
- Sensitivity to Interference
- Calibration Challenges
- Response Time

7. Real-Time Applications:

- Sound-Activated Switches
- Security Systems
- Noise Level Monitoring
- Voice-Activated Controls
- Sound-Based Data Transmission

8.References:

- **https://sensorkit.joy-it.net/en/sensors/ky-038**
- **https://arduinomodules.info/ky-037-high-sensitivity-sound-detection-module/**
- **https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html**

Thank
You