KY-027 Magic light cup MODULE

PRESENTED BY

ARAVIND POSINA

NOVEMBER-2023

Table of Contents

1.Project Summary

The "Magic Cup Light" project is a cutting-edge demonstration of smart lighting technology, seamlessly integrating the STM32F446 microcontroller, the WE-10 wireless communication module, and the KY-027 Magic Light Cup module. This summary encapsulates the core achievements, functionalities, and significance of the project.

Intelligent Lighting Control: The STM32F446 microcontroller empowers precise and responsive control over the KY-027 Magic Light Cup module, providing a dynamic lighting solution.

Wireless Connectivity: Leveraging the WE-10 module, the project achieves wireless communication, enabling remote monitoring and control, reducing the dependency on physical connections.

Cloud Integration: Data from the STM32F446 and the Rugged Board is seamlessly transmitted to the Rightech Cloud using the MQTT protocol, enabling real-time data visualization and analysis.

Environmental Adaptability: The project capitalizes on the environmental sensing capabilities of the Magic Light Cup module, creating a lighting system that intelligently adapts to changes in its surroundings.

Precise control over the Magic Cup Light module, allowing for dynamic adjustments based on sensor input.
Wireless Communication:

Seamless communication facilitated by the WE-10 module, eliminating the need for physical connections and enabling flexible deployment.

Utilization of the MQTT protocol for transmitting data to the Rightech Cloud platform, enabling centralized monitoring and control.

Integration of environmental sensing capabilities, allowing the lighting system to respond intelligently to changes in its environment.
Significance

The "Magic Cup Light" project signifies a convergence of embedded systems, wireless communication, and cloud technology. Its significance lies in showcasing the potential of Internet of Things (IoT) technology in creating intelligent and adaptive systems.
While the project successfully achieves its primary objectives, future enhancements could include the integration of additional sensors for more comprehensive environmental data, expanding the system's capabilities and applications.

In conclusion, the "Magic Cup Light" project provides a comprehensive and scalable solution for smart lighting control, exemplifying the capabilities of modern embedded systems and IoT technologies.

2.Key Features of the Project

1. Smart Lighting Control

Description: The project harnesses the power of the STM32F446 microcontroller to enable precise and responsive control over the KY-027 Magic Light Cup module.
Benefit: Intelligent lighting control allows for dynamic adjustments to lighting conditions based on real-time sensor input.

2. Wireless Connectivity with WE-10 Module

Description: The integration of the WE-10 module facilitates wireless communication, eliminating the need for physical connections and enabling flexible deployment in various environments.
Benefit: Seamless wireless connectivity enhances accessibility, enabling remote monitoring and control of the lighting system.

3. Cloud Integration through MQTT

Description: Data from both the STM32F446 and the Rugged Board is transmitted to the Rightech Cloud platform using the MQTT protocol.
Benefit: Real-time data visualization and analysis on the cloud platform provide centralized monitoring, enhancing overall system management.

4. Environmental Sensing with KY-027 Module

Description: The KY-027 Magic Light Cup module is equipped with environmental sensors, allowing the lighting system to adapt intelligently to changes in its surroundings.

Benefit: Environmental awareness enhances the system's adaptability, creating a more responsive and energy-efficient lighting solution.

5. STM32F446 Microcontroller Control
Description: The project leverages the capabilities of the STM32F446 microcontroller for precise control and coordination of the entire system.
Benefit: The STM32F446 serves as the central control unit, ensuring efficient and reliable operation of the smart lighting system.

6. Real-time Data Visualization and Analysis
Description: Through cloud integration, the project provides real-time data visualization and analysis on the Rightech Cloud platform.
Benefit: Real-time insights empower users to make informed decisions, optimize energy usage, and monitor the system's performance remotely.

7. Adaptive Lighting System
Description: The environmental sensing capabilities enable the lighting system to adapt and respond intelligently to changes in its environment.
Benefit: The adaptive nature of the lighting system enhances user comfort, energy efficiency, and overall system effectiveness.

8. Scalability and Future Enhancements
Description: The project is designed with scalability in mind, allowing for future enhancements such as the integration of additional sensors for more comprehensive environmental data.
Benefit: Scalability ensures that the system can evolve to meet changing requirements and incorporate new features in the future.
These key features collectively position the "Magic Cup Light" project as a sophisticated and adaptive smart lighting solution, showcasing the capabilities of modern embedded systems, wireless communication, and cloud integration in the context of the Internet of Things (IoT).

3.Hardware Components and Specifications

1. STM32F446 Microcontroller Board
Specifications:
Model: [Specify the model]
Key Specifications:
[List key specifications such as microcontroller type, clock speed, memory capacity, etc.]

2. KY-027 Magic Light Cup Module

Specifications:
Model: KY-027
Key Specifications:
RGB LED
PWM control
Environmental sensors (if applicable)
[Include any other relevant specifications]

3. WE-10 Wireless Communication Module
Specifications:
Model: WE-10
Key Specifications:
WiFi connectivity
MQTT support
[Include any other relevant specifications]

4. Rugged Board
Specifications:
Model: [Specify the model]
Key Specifications:
[List key specifications such as microcontroller type, connectivity options, etc.]

5. Additional Components
5.1 User Button (B1) and LED (LD2)
Specifications:
Button Type: [Specify the type of button]
LED Type: [Specify the type of LED]
5.2 Other GPIO Components
Specifications:
[Include specifications for any additional GPIO components used in the project]

4. INTERFACING WITH KY-027 Magic cup light

Hardware Connection:

Power Supply:
Connect the VCC pin of the KY-027 module to a 5V output on your STM32 board.
Connect the GND pin to the ground (GND) on your board.

Analog Output (AO):
KY-027 may have an analog output pin (AO) representing the light intensity. Connect it to an analog input pin on your STM32 (e.g., ADC channel).

Digital Output (DO):
KY-027 may also have a digital output pin (DO) that gives a high or low signal based on a certain light threshold. Connect it to a digital input pin on your STM32.
LED Indicator (optional):

If the KY-027 module has an onboard LED, you can connect it to a digital output pin on your STM32 to visually indicate the light intensity.

Code Implementation:

Initialize GPIO Pins:
Initialize the GPIO pins for the AO, DO, and LED (if present) in your STM32 code.

Read Analog Input (AO):

Use the ADC peripheral to read the analog output from the KY-027 module. Convert the ADC readings to a meaningful light intensity value.
Read Digital Input (DO):
Read the digital output from the KY-027 module. This can be used to detect if the light intensity has crossed a certain threshold.

Control LED (optional):
If the KY-027 module has an onboard LED, use a digital output pin on your STM32 to control it based on the light intensity readings.

4.1 STAGES

STAGE:1

The STM32F446RE with KY-027 Magic cuo light is used to communicate with the module and board. It is used to read the values of the sensor..

STAGE:2

The STM32F446RE with KY-027,W10 is used to transmit the data into the MQTT server to Publish the data in the Right-tech.

STAGE:3

The Ruggedboard with STM32F446RE is used to transmit the data and get the KY-027 Magic Cup Light value in the ruggedboard minicom.

STAGE:4

The Ruggedboard with STM32F446RE is used to transmit the data and get the KY-027 Magic Cup Light value in the ruggedboard minicom and pass the value into the Right-tech by using W10 module is connected with the Rugged board.

5.SENDING THIS STATUS TO THE CLOUD

1.Wi-Fi Module Initialization (WE-10):

The WE10_Init function initializes the Wi-Fi module by sending a series of commands to configure it. These commands include resetting the module, setting the Wi-Fi mode, and connecting to a specified access point.

2.MQTT Initialization:
The MQTT_Init function initializes MQTT communication. It configures the MQTT network parameters, specifies the MQTT connection configuration (such as the broker address), and starts the MQTT communication.

3.Main Loop:
The main loop continuously reads the state of a sensor connected to pin GPIO_PIN_6 in GPIOA.
If the sensor state is GPIO_PIN_RESET (indicating the sensor is active):
A message is transmitted over UART to indicate that the LED in the sensor is on.
The LED connected to pin GPIO_PIN_7 is toggled.
The mqtt_data_send function is called with the argument 1 to send the MQTT message indicating that the LED is on.
If the sensor state is not GPIO_PIN_RESET:
A message is transmitted over UART to indicate that the LED in the sensor is off.
The LED connected to pin GPIO_PIN_7 is toggled.
The mqtt_data_send function is called with the argument 0 to send the MQTT message indicating that the LED is off.

4.MQTT Data Sending:
The mqtt_data_send function constructs a command in the form of "CMD+MQTTPUB=base/light/led1,1" or "CMD+MQTTPUB=base/light/led1,0" based on the input argument (1 or 0).
The constructed command is then transmitted over UART to the Wi-Fi module, which is expected to handle the MQTT communication with the cloud service.

5.UART Communication:

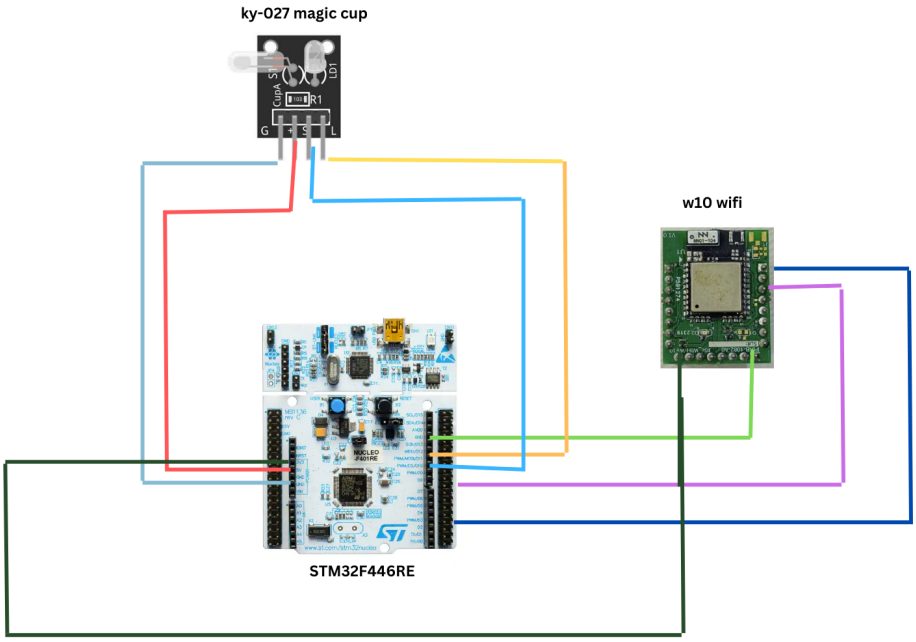
UART communication is used to send commands and receive responses between the microcontroller and the Wi-Fi module. This is done using the HAL_UART_Transmit and HAL_UART_Receive functions.

6.Delay:
The HAL_Delay function is used to introduce delays in the code. This is common in embedded systems to allow for proper timing and synchronization between different operations.

7.Error Handling:
The Error_Handler function is a placeholder for error handling. In a real-world application, you would want to implement more robust error handling based on the specific needs of your project

6.Connection Diagram

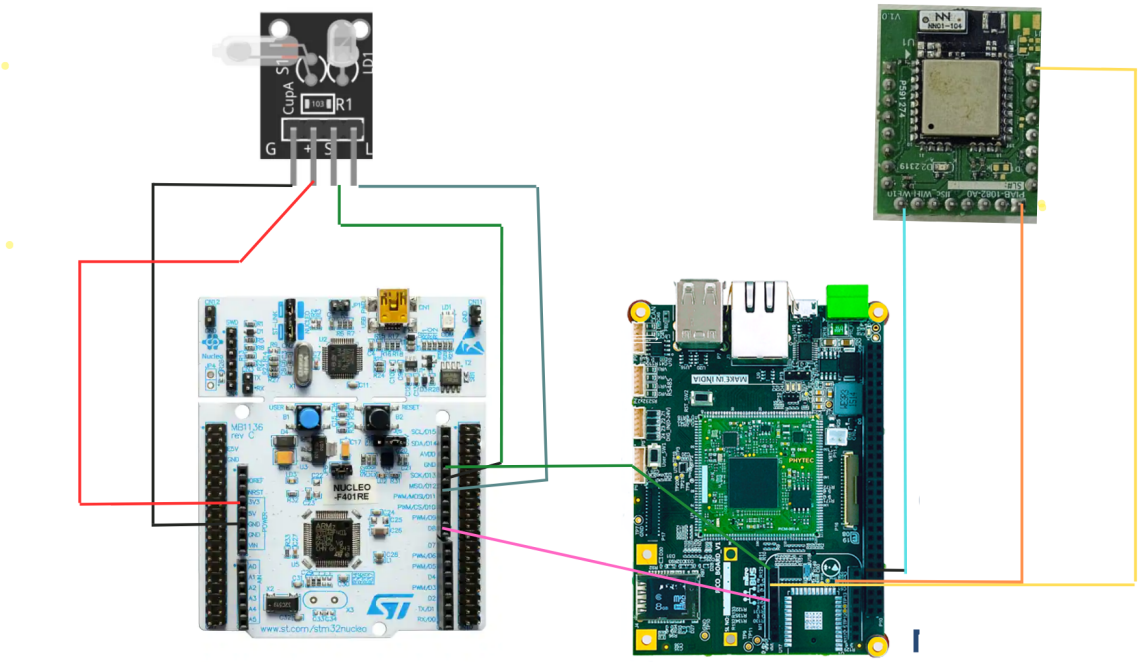
STAGE2:-



PIN	PIN NUMBER	COMPONENTS
VCC 5V	VCC	Ky-027 and w10
GROUND	GND	Ky-027 and w10
Signal	D10	ky-027
light	D9	stm32f446re

tx	PD8	w10
rx	PD2	w10

STAGE4:-



In stm32f446re with KY-027 CUP LIGHT:

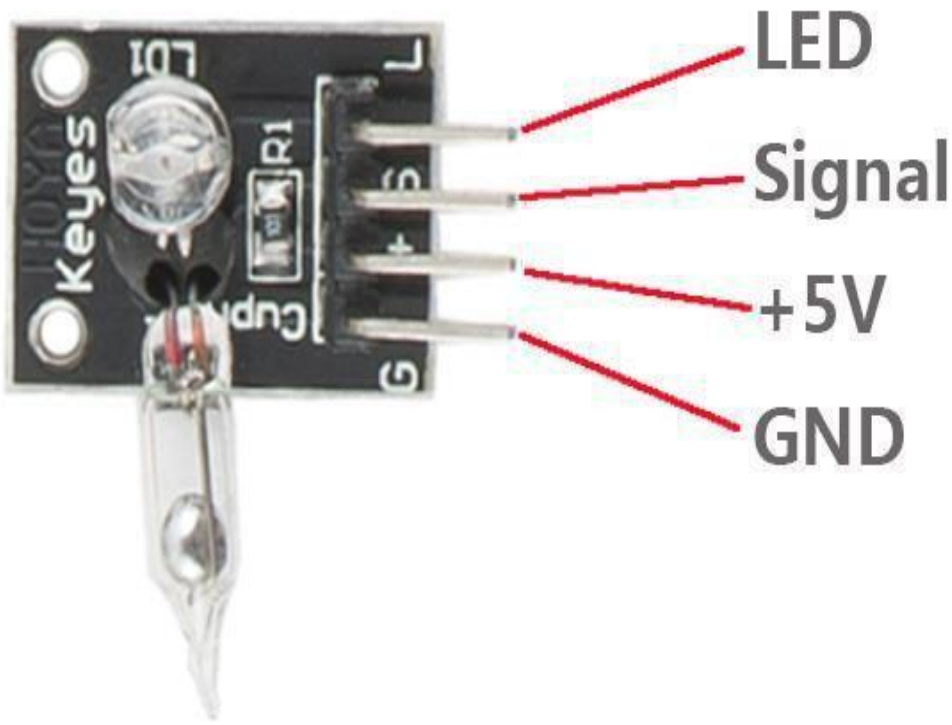
PIN	PIN NUMBER	COMPONENTS
VCC 5V	VCC	ky-027
GROUND	GND	ky-027
Signal	D10	ky-027
light	D9	stm32f446re

In Ruggedboard with w10:-

PIN	PIN NUMBER	COMPONENTS

VCC 5V	VCC	W10
GROUND	GND	W10
Tx	Tx	W10
Rx	Rx	stm32f446re

7.KY-027 Magic Cup Light Module

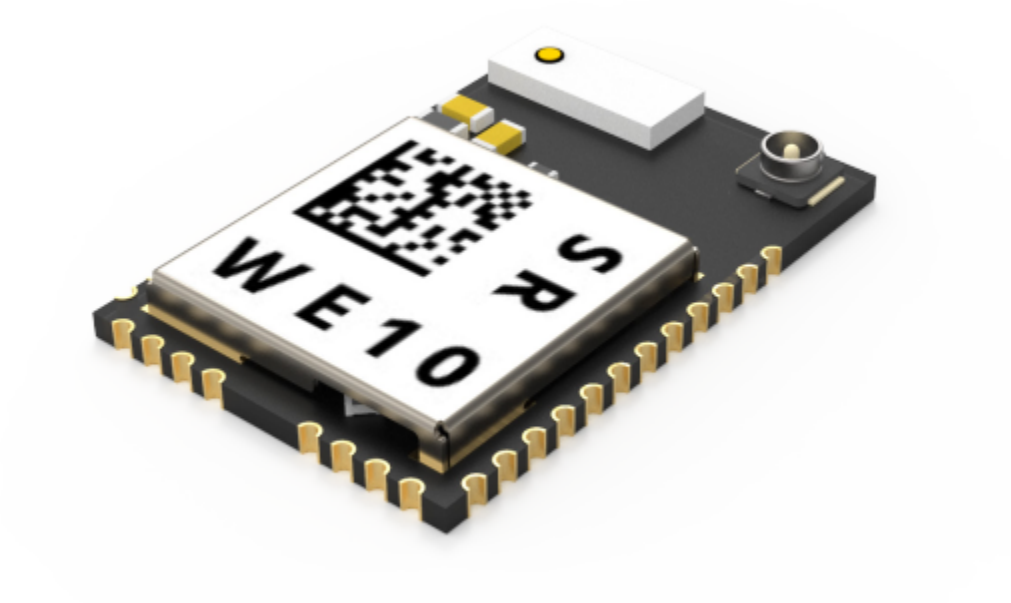


Components
- 1 * Arduino Uno board
- 1 * USB cable
- 2 * Magic light cup module
- DuPont wires(Female to Male)

PIN	Stm32f446re	
Vcc 5v	VCC	
GROUND	GND	

signal	D9	
light	D10	

8.W10 wifi 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

9.code snippet to wifi module initialization and connection with interrupt-based uart commands

```
void WE10_Init ()
{
	char buffer[128];
	/********* CMD+RESET ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	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);


	/*********  CMD+WIFIMODE=1  ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	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);
```

```
        /********* CMD+CONTOAP=SSID,PASSWD ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        sprintf (&buffer[0],"CMD+CONTOAP=iQOO Neo6,123456788\r\n");
        HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
        //memset(&buffer[0],0x00,strlen(buffer));
        HAL_Delay(2000);
        HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_Delay(500);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

        /********* CMD?WIFI**********/
        //memset(&buffer[0],0x00,strlen(buffer));
        sprintf (&buffer[0], "CMD?WIFI\r\n");
        HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
//      memset(&buffer[0],0x00,strlen(buffer));
//      HAL_Delay(500);
        HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_Delay(500);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);


}
```

.The code first declares a buffer of 128 characters. The buffer will be used to store the commands that are sent to the WE10 module.

.The next few lines of code send the CMD+RESET command to the WE10 module. This command resets the module to its default state.

.The next line of code sends the CMD+WIFIMODE=1 command to the WE10 module. This command sets the module to operate in WiFi Mode.

.The next line of code sends the CMD+CONTOAP=SSID, PASSWD command to the WE10 module. This command configures the module to connect to the WiFi network with the specified SSID and Password.

.The next line of code sends the CMD.WIFI command to the WE10

module. This command queries the module for its WiFi status. The last line of code waits for 2000 milliseconds and then receives a response from the WE10 module. The response is stored in the Buffer.

.The WE10_Init() function is a simple example of how to initialize a WE10 module and connect it to a WiFi network. The function takes no arguments and it returns void.

10 .code snippet to MQTT initialization and connection with interrupt-based uart commands

```
void MQTT_Init()
{

	char buffer[128];

	/*********CMD+MQTTNETCFG ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	sprintf (&buffer[0], "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n");
	HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
	//memset(&buffer[0],0x00,strlen(buffer));
	//HAL_Delay(500);
	HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_Delay(500);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);


	/*********CMD+MQTTCONCFG---->LED ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	sprintf (&buffer[0], "CMD+MQTTCONCFG=3,posinaaravind0205-2tq0rr,,,,,,,,,\r\n");
	HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
	//memset(&buffer[0],0x00,strlen(buffer));
	//HAL_Delay(500);
	HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_Delay(500);
```

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


        /*********CMD+MQTTSTART ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        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);
//      memset(&buffer[0],0x00,strlen(buffer));
        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);

        /*********CMD+MQTTSUB *******/
        //memset(&buffer[0],0x00,strlen(buffer));
        sprintf (&buffer[0], "CMD+MQTTSUB=base/light/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);

}
```

The code first declares a buffer of 128 characters. The buffer will be used to store the commands that are sent to the WE10 module.

The next few lines of code send the CMD+RESET command to the WE10 module. This command resets the module to its default state.

The next line of code sends the CMD+WIFIMODE=1 command to the WE10 module. This command sets the module to operate in WiFi Mode.

The next line of code sends the CMD+CONTOAP=SSID, PASSWD command to the WE10 module. This command configures the module to connect to the WiFi network with the specified SSID and Password.

The next line of code sends the CMD.WIFI command to the WE10 module. This command queries the module for its WiFi status.

The last line of code waits for 2000 milliseconds and then receives a response from the WE10 module. The response is stored in the Buffer.

The WE10_Init() function is a simple example of how to initialize a WE10 module and connect it to a WiFi network. The function takes no arguments and it returns void.

11. Send_task Function

```
void mqtt_data_send()

{

char buffer[50];

sprintf (&buffer[0], "CMD+MQTTPUB=base/state/distance,%.2f\r\n",dis1);

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

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

HAL_Delay(100);

}
```

This code is a function written in C that sends MQTT (Message Queuing Telemetry Transport) data using UART (Universal Asynchronous Receiver-Transmitter) communication.This declares a function named mqtt_data_send with no input parameters and no return value (void).

A character array (buffer) named buffer is declared with a size of 50 characters. This buffer will be used to store the formatted MQTT message.The sprintf function is used to format a string with the MQTT message. The message includes a command ("CMD+MQTTPUB"), a topic ("base/state/distance"), and a floating-point value (dis1) with two decimal places.

The formatted message stored in the buffer is transmitted via UART. The HAL_UART_Transmit function is used for this purpose. The message is sent to two UART interfaces (huart1 and huart2), and the third argument specifies the timeout duration (1000 milliseconds in this case).

A delay of 100 milliseconds is introduced using HAL_Delay. This delay allows time for the UART transmissions to complete before the function exits.In summary, this code snippet is part of a larger program, and its purpose is to format and transmit an MQTT message containing distance information (dis1) over two UART interfaces (huart1 and huart2). The delay at the end ensures that there is sufficient time for the transmissions to complete before moving on.

The code you provided defines the Send_Task function. The Send_Task is a task that will be executed by the code.

The Send_Task function first declares a variable of type data called DatatoSend.

The temp member of the data structure is used to store the temperature reading, and the humidity member of the data structure is used to store the humidity reading.

The Send_Task function then enters an infinite loop. In each iteration of the loop, the Send_Task function reads the temperature and humidity readings from the sensors, stores the readings in the DatatoSend structure, and then puts the DatatoSend structure on the myQueueTemp message queue. The osMessageQueuePut() function is used to put a message on a message queue. The first parameter is the handle of the message queue, the second parameter is a pointer to the message, the third parameter is the priority of the message, and the fourth parameter is the timeout value.

SendTask:

The SendTask is a task that will be created by the code.The osThreadId_t SendTaskHandle variable is used to store the handle of the SendTask. The osThreadAttr_t SendTask_attributes structure defines the attributes of the SendTask.

The SendTask_attributes structure has three members: name: The name of the task.stack_size The size of the stack that will be allocated to the task. priority: The priority of the task.

In this case, the name of the task is "SendTask", the stack_size is 128 * 4 bytes, and the priority is osPriorityNormal.

The osThreadAttr_t structure is used to configure the attributes of a task. The name member is used to set the name of the task. The stack_size member is used to set the size of the stack that will be allocated to the task. The priority member is used to set the priority of the task.

RecieveTask:

The RecieveTask is a task that will be created by the code.

The osThreadId_t RecieveTaskHandle variable is used to store the handle of the RecieveTask. The osThreadAttr_t RecieveTask_attributes structure defines the attributes of the RecieveTask.

The RecieveTask_attributes structure has three members: name: The name of the task.stack_size The size of the stack that will be allocated to the task. priority: The priority of the task.

In this case, the name of the task is "RecieveTask", the stack_size is 128* 4 bytes, and the priority is osPriorityLow.

The osThreadAttr_t structure is used to configure the attributes of a task. The name member is used to set the name of the task. The stack_size member is used to set the size of the stack that will be allocated to the task. The priority member is used to set the priority of the task.

12.Project Code

Stage 2:-(stm32f446re connected with w10)

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/*base/light/led1*/
```

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include"stdio.h"
#include"string.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
void WE10_Init ();
void MQTT_Init();
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
```

```
/* USER CODE BEGIN 0 */
uint32_t pinstate;
//uint32_t brightness=0;
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
void mqtt_data_send(uint8_t n)
{
char buffer[50];

sprintf(&buffer[0], "CMD+MQTTPUB=base/light/led1,%d\r\n", n);
HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), 1000);
HAL_Delay(100);
}
/*printf(&buffer[0], "CMD+MQTTPUB=reading/sensorval\r\n", n);
HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), 1000);
}*/
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
```

```
  MX_USART2_UART_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  WE10_Init();
  MQTT_Init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
          while (1)
            {
                  pinstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);

                        if (pinstate == GPIO_PIN_RESET)
                        {
                           HAL_UART_Transmit(&huart1, (uint8_t*)"LED IN SENSOR IS ON\r\n",
21, HAL_MAX_DELAY);
                           HAL_UART_Transmit(&huart2, (uint8_t*)"LED IN SENSOR IS ON\r\n",
21, HAL_MAX_DELAY); // Print to Minicom
                           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, !pinstate);
                           HAL_UART_Transmit(&huart1, (uint8_t*)"LED IN SENSOR IS ON\r\n",
21, HAL_MAX_DELAY);
                           mqtt_data_send(1);
                           HAL_Delay(1000);
                        }
                        else
                        {
                           HAL_UART_Transmit(&huart1, (uint8_t*)"LED IN SENSOR IS OFF\r\n",
22, HAL_MAX_DELAY);
                           HAL_UART_Transmit(&huart2, (uint8_t*)"LED IN SENSOR IS OFF\r\n",
22, HAL_MAX_DELAY); // Print to Minicom
                           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, !pinstate);
                           HAL_UART_Transmit(&huart1, (uint8_t*)"LED IN SENSOR IS OFF\r\n",
22, HAL_MAX_DELAY);
                           mqtt_data_send(0);
                           HAL_Delay(1000);
                        }


    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

```
  }

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  RCC_OscInitStruct.PLL.PLLR = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
```

```
  {
    Error_Handler();
  }
}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 38400;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
```

```
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, LD2_Pin|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LD2_Pin PA7 */
  GPIO_InitStruct.Pin = LD2_Pin|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
void WE10_Init ()
{
        char buffer[128];
        /********* CMD+RESET ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        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);


        /*********  CMD+WIFIMODE=1  ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        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);


	/********* CMD+CONTOAP=SSID,PASSWD ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	sprintf (&buffer[0],"CMD+CONTOAP=iQOO Neo6,123456788\r\n");
	HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
	//memset(&buffer[0],0x00,strlen(buffer));
	HAL_Delay(2000);
	HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_Delay(500);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);

	/********* CMD?WIFI**********/
	//memset(&buffer[0],0x00,strlen(buffer));
	sprintf (&buffer[0], "CMD?WIFI\r\n");
	HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
//	memset(&buffer[0],0x00,strlen(buffer));
//	HAL_Delay(500);
	HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_Delay(500);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);


}

void MQTT_Init()
{

	char buffer[128];

	/*********CMD+MQTTNETCFG ********/
	//memset(&buffer[0],0x00,strlen(buffer));
	sprintf (&buffer[0], "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n");
	HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
	HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
	//memset(&buffer[0],0x00,strlen(buffer));
	//HAL_Delay(500);
	HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
```

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


        /*********CMD+MQTTCONCFG---->LED ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        sprintf (&buffer[0], "CMD+MQTTCONCFG=3,posinaaravind0205-2tq0rr,,,,,,,,,\r\n");
        HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);
        //memset(&buffer[0],0x00,strlen(buffer));
        //HAL_Delay(500);
        HAL_UART_Receive(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
        HAL_Delay(500);
        HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 1000);


        /*********CMD+MQTTSTART ********/
        //memset(&buffer[0],0x00,strlen(buffer));
        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);
//      memset(&buffer[0],0x00,strlen(buffer));
        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);

        /*********CMD+MQTTSUB *******/
        //memset(&buffer[0],0x00,strlen(buffer));
        sprintf (&buffer[0], "CMD+MQTTSUB=base/light/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);

}

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
```

```
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Stage 3:-

```
#include<stdio.h>

#include<string.h>

#include<errno.h>

#include<stdlib.h>
```

```
void error(const char *msg) {
perror(msg);
exit(1);
}
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, B115200); // Set the baud rate
cfsetispeed(&tty, B115200);
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");
}
while(1)
{
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;
```

Stage 4:- Ruggedboard code with w10

```
include <errno.h>
```

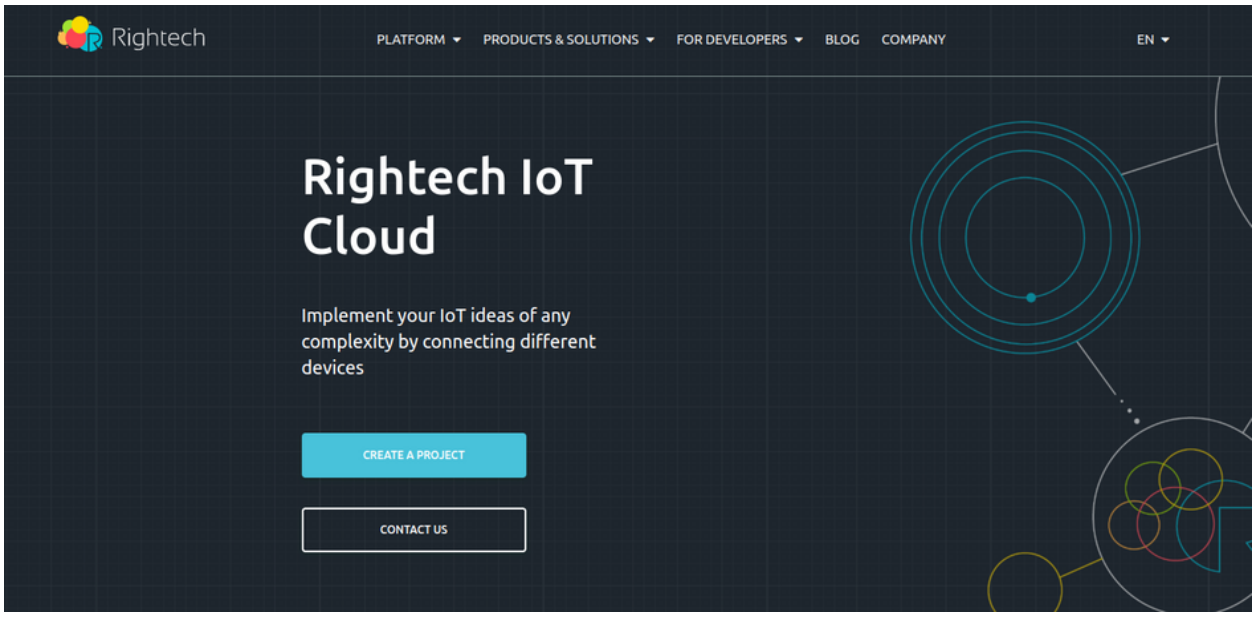
```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
int set_interface_attribs(int fd, int speed)
{
        struct termios tty;

        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=\"realme X7 Max\",\"0987654321\"\r\n";
    char arr3[] = "CMD+MQTTNETCFG=dev.rightech.io,1883\r\n";
    char arr4[] = "CMD+MQTTCONCFG=3,mqtt-posinaaravind0205-ow01tb,,,,,,,,\r\n";
    char arr5[] = "CMD+MQTTSTART=1\r\n";
    char arr6[] = "CMD+MQTTSUB=base/relay/led1\r\n";
//    char arr7[] = "CMD+MQTTPUB=sensor/voltage\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);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n",buf);

    // Send CMD+WIFIMODE=1
    printf("%s", arr2);
    wlen = write(fd, arr2, sizeof(arr2) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n",buf);

    // Send CMD+CONTOAP
    printf("%s", arr);
    wlen = write(fd, arr, sizeof(arr) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
```

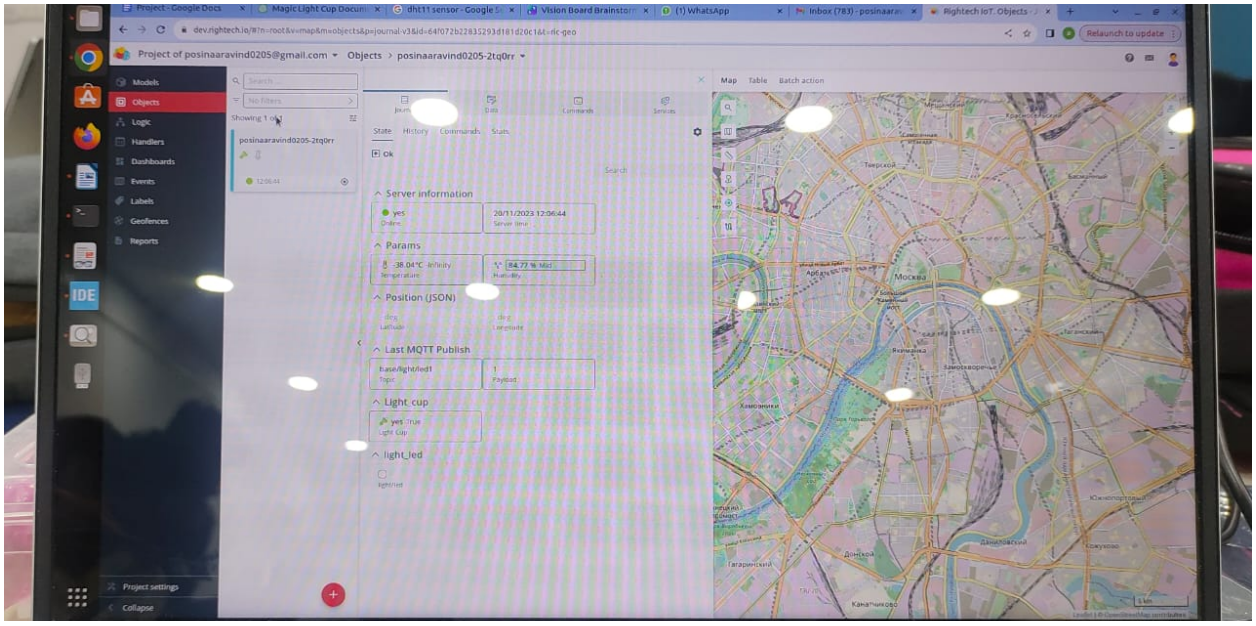
```
    //printf("%s\n", buf);

    printf("%s", arr3);
    wlen = write(fd, arr3, sizeof(arr3) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n", buf);

    printf("%s", arr4);
    wlen = write(fd, arr4, sizeof(arr4) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n", buf);

    printf("%s", arr5);
    wlen = write(fd, arr5, sizeof(arr5) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n", buf);
//while(1){
    printf("%s", arr6);
    wlen = write(fd, arr6, sizeof(arr6) - 1);
    sleep(3);
    //rdlen = read(fd, buf, sizeof(buf));
    //buf[rdlen]='\0';
    //printf("%s\n", buf);

//    printf("%s", arr7);
  //  wlen = write(fd, arr7, sizeof(arr7) - 1);
//    sleep(3);
//    char buff[10]="c";
//    rdlen = read(fd, buf, sizeof(buf) - 1); // Read data into the buffer
  //      if (rdlen > 0) {
    //        buf[rdlen-3] = '\0'; // Null-terminate the received data
      //      printf("Received data: %s\n", buf);    }

//    wlen = write(fd , buf, sizeof(buf));
//   wlen = write(fd ,"CMD+MQTTPUB=sensor/voltage,%s\r\n", buf);
char buffer[100]; // Create a buffer to hold the formatted message
//char buf[100]="42";    // Create a buffer to store the value read
```

```
//int fd; // Your file descriptor
//set_interface_attribs(fd, B9600);
// Read data into the 'buf' buffer
while(1){
 rdlen = read(fd, buf, sizeof(buf) - 1);
if (rdlen > 0) {
    buf[rdlen] = '\0'; // Null-terminate the received data
        printf("%s\n", buf);
    // Format the data from 'buf' into 'buffer'
    int ret = snprintf(buffer, sizeof(buffer), "CMD+MQTTPUB=sensor/voltage,%s\r\n", buf);

    if (ret < 0) {
        // Handle the error if snprintf fails
    } else {
        // Open the file descriptor 'fd' if not already opened

        // Write the formatted message to the file descriptor
        ssize_t wlen = write(fd, buffer, ret);
        sleep(3);
        if (wlen == -1) {
            // Handle the write error if needed
        }
    }
}
}



    close(fd);
    return 0;
}
```

13.Rightech 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.



{

"id": "centimeter",

"name": "centimeter",

"active": true,

"type": "argument",

"source": "state",

"dataType": "number",

"unit": "length-centimeter",

"reference": "base/state/centimeter",

"display": {},

"factor": 1,

"linear": true,

"children": [],

```
"levels": {
"type": "range",
"value": [
{
"color": "#f16059",
"name": "-Infinity",
"value": {
"a": "-Infinity",
"b": 10
}
},
{
"color": "#a2ce4b",
"name": "Min",
"value": {
"a": 10,
"b": 30
}
},
{
"color": "#ffae62",
"name": "Mid",
"value": {
"a": 30,
```

```
"b": 50

}

},

{

"color": "#f16059",

"name": "Max",

"value": {

"a": 50,

"b": "Infinity"

}

}

],

"svg": "<svg xmlns=\"http://www.w3.org/2000/svg\" x=\"0px\" y=\"0px\"\nviewBox=\"0 0 16 16\"\n><g id=\"surface1\"><path style=\" \" d=\"M 7.511719 1 C 6.140625 1 5.011719 2.128906 5.011719 3.5 L 5.011719 9.121094 C 4.414063 9.746094 4 10.5625 4 11.5 C 4 13.425781 5.574219 15 7.5 15 C 9.425781 15 11 13.425781 11 11.5 C 11 10.574219 10.597656 9.765625 10.011719 9.140625 L 10.011719 3.5 C 10.011719 2.128906 8.882813 1 7.511719 1 Z M 7.511719 2 C 8.339844 2 9.011719 2.671875 9.011719 3.5 L 9.011719 4 L 7 4 L 7 5 L 9.011719 5 L 9.011719 6 L 7 6 L 7 7 L 9.011719 7 L 9.011719 8 L 7 8 L 7 9 L 9.011719 9 L 9.011719 9.289063 C 9.011719 9.429688 9.070313 9.5625 9.175781 9.65625 C 9.683594 10.117188 10 10.765625 10 11.5 C 10 12.886719 8.886719 14 7.5 14 C 6.113281 14 5 12.886719 5 11.5 C 5 10.757813 5.328125 10.105469 5.84375 9.640625 C 5.949219 9.546875 6.011719 9.414063 6.011719 9.269531 L 6.011719 3.5 C 6.011719 2.671875 6.683594 2 7.511719 2 Z \"></path></g></svg>"

},

"usage": null

},
```

14.Real Time Applications

1.Industrial Automation:

Example: Robotic assembly lines, PLC (Programmable Logic Controller) systems for manufacturing.

2.Automotive Systems:

Example: Anti-lock Braking Systems (ABS), Engine Control Units (ECUs), Advanced Driver Assistance Systems (ADAS).

Telecommunications:

Example: Real-time communication systems, VoIP (Voice over Internet Protocol) applications.

3.Medical Devices:

Example: Patient monitoring systems, infusion pumps, medical imaging devices.

Aerospace and Defense:

Example: Flight control systems, radar systems, missile guidance systems.

4.Power Systems:

Example: Power grid control systems, smart grid applications.

5.Financial Trading Systems:

Example: High-frequency trading platforms, algorithmic trading systems.

6.Embedded Systems:

Example: Real-time operating systems (RTOS), control systems in embedded devices.

7.Gaming:

Example: Real-time multiplayer games, game physics engines.

8.Audio and Video Processing:

Example: Audio processing in live concerts, video streaming applications.

9.Robotics:

Example: Real-time control of robotic arms, autonomous vehicles.

10.Traffic Control Systems:

Example: Intelligent transportation systems, traffic light control.
12.Healthcare Systems:

Example: Remote patient monitoring, wearable health devices.
13.Environmental Monitoring:

Example: Real-time monitoring of pollution levels, weather forecasting.
14.Smart Grids:

Example: Monitoring and control of electricity distribution in real-time.
15.Networked Control Systems:

Example: Systems where control loops are closed over a network**.**
16.Simulation and Training Systems:

Example: Flight simulators, military training simulations.

15.FEATURE SCOPES:-

1.Project Objectives:

Clearly state the overall objectives and goals of the project. What problem or need is the project addressing?
2.Target Audience:

Identify the target audience or users for whom the project is being developed. Understand their needs and preferences.
3.Functional Requirements:

List the specific features and functionalities that the project must deliver to meet its objectives. These are the core capabilities of the system.
4.Non-functional Requirements:

Include non-functional requirements such as performance, scalability, security, and usability. These criteria are often as important as functional requirements.

5.Constraints:

Identify any constraints or limitations that might impact the project, such as budget constraints, time constraints, or technological constraints.

6.Prioritization:

Prioritize features based on their importance and impact on the project goals. This helps in case of resource constraints or phased development.

7.Dependencies:

Identify any dependencies between features. Some features may need to be implemented before others can be developed.

8.Exclusions:

Clearly state what is not included in the project. This helps manage expectations and prevents scope creep.

9.User Stories or Use Cases:

Describe user stories or use cases to illustrate how users will interact with the system and what features they will use.

10.Prototypes or Mockups:

Create prototypes or mockups to visualize the user interface and interactions. This can help stakeholders better understand the planned features.

11.Feedback Mechanism:

Define how feedback from stakeholders will be collected and incorporated into the project scope.

12.Change Control Process:

Establish a process for handling changes to the project scope. Changes should be carefully evaluated and approved.

13.Risk Analysis:

Conduct a risk analysis to identify potential risks that could impact the project scope. Develop contingency plans if needed.