Microcontroller & Interfacing

CE205T

	CLO-2		CLO-3					Total
Part	B	C	A	D	E	F	G	
Marks	50	50	50	50	50	50	50	
Obt.								

Wireless Home Automation System

1: Muhammad Waleed Hussain (BSCE21030)

2: Ariba Mumtaz (BSCE22020)

3. Rameeza Rahim (BSCE22052

A. Overview [CLO-3, 50 Marks]

This project is about developing a home automation system that uses open-source software and unlike traditional systems, is not locked down to specific manufacturer or vendor specifications. To do this, we utilize MQTT protocol to establish device and server communication meanwhile use Home Assistant as our automation server. Here, the MQTT communication between Home Assistant and devices will be handled by EMQX server. In the usual scenario it is to be noted that each sensor or relay may be a device of its own but in the current scenario we aimed to create three devices. Device one is purely a sensor relay that is simulated to be present in a bedroom and measures various sensor values. The second device is simulated to be present next to a switchboard on part of the switch board to turn off or on any device connected to it. The third is a smart bell that is aimed at having the ability to unlock or lock the main gate. Since all these devices need to communicate wirelessly over the internet and we are bound to use STM32 platform; commonly available STM32 devices don't come with Wi-Fi, we used an ESP01 Wi-Fi module to provide Wi-Fi as well as server client communication.

GOALS

1. Interface ESP01 and STM32 microcontrollers to work together
2. Establish connection to server.
3. Receive commands from the server and send sensor values to the server over Wi-Fi and internet.
4. Use capabilities of the interfaces of the STM32 to control peripherals and read sensor values.

B. List of Components Used [CLO-2, 50 Marks]

Component	Quantity	Price per Unit (PKR)	Total Price (PKR)	Link	Working Principle
STM32f411CEU6 Black pill Board	3	1250	3750	https://digilog.pk/products/buy-stm32f411ceu6-blackpill-development-board-affordable-mcu-development-module?_pos=11&_sid=fe65a29db&_ss=r	STM32F411CEU6 microcontroller
ESP8266 based ESP01 Wi-Fi Module	3	290	870	https://digilog.pk/products/buy-stm32f411ceu6-blackpill-development-board-affordable-mcu-development-module?_pos=11&_sid=fe65a29db&_ss=r	ESP8266X based microcontroller. Uses UART to communicate
FT232 FTDI USB to TTL Serial Adapter	3	320	960	https://digilog.pk/products/ft232rl-ft232-serial-uart-3-ftdi-module?_pos=1&_sid=4f14b1c7c&_ss=r	USB programmer for ESP01
SHT30 I2C based temperature & humidity sensor	1	750	750	SHT30 I2C Humidity Sensor | Next-Gen SHT3x DIS Temperature Sensor Pakistan - Digilog.pk	!2C digital serial communication
KY-018 LDR Light Sensor module	2	150	300	LDR light sensor module In Pakistan - Digilog.pk	Analogue and digital output available
Magnetic Door Sensor Switch	2	250	500	Magnetic Door Sensor Magnetic Read Switch In Pakistan - Digilog.pk	Magnetic normally open switch
8 Channel Relay Module	1	700	700	5V 8 Channel Relay Module Relay Board Relay Arduino Relay Module - Digilog.pk	Active low-level trigger
Buzzer	1	80	80	Active Low Level Trigger Buzzer Alarm Module for DIY MCU SCM in Pakistan - Digilog.pk	Active low-level trigger
Push Button	5	15	60	12 x 12mm x 7.5mm Push Button - High Quality Momentary Switches in 5 Colors - Digilog.pk	Switch can be configured with pull up or pull down mode

Vero Board	3	150	450	BreadBoard Style Veroboard 100mm x 240mm Project Board Prototyping Board StripBoard In Pakistan - Digilog.pk	Prototyping Board
Jumper Wire	1	300	300	10Cm Hole To Hole Jumper Wire Dupont Line 40 Pin Female To Female Arduino Jumper Wires In Pakistan - Digilog.pk	Interconnections between components
Stlink v2 programmer	1	1300	1300	ST-LINK V2 ST Link V2 STLINK V2 Programmer - Digilog.pk	Programmer for STM devices

C. Peripherals of STM Microcontroller being used [CLO-2, 50 Marks]

Common Peripherals used for all devices

- UART: for communication between STM32 and ESP01

Sensor Array Board

- I2C: communication between SHT30 and STM32
- External Interrupt: for window sensor configured to detect rising and falling edge trigger
- ADC: read LDR values

Control Board

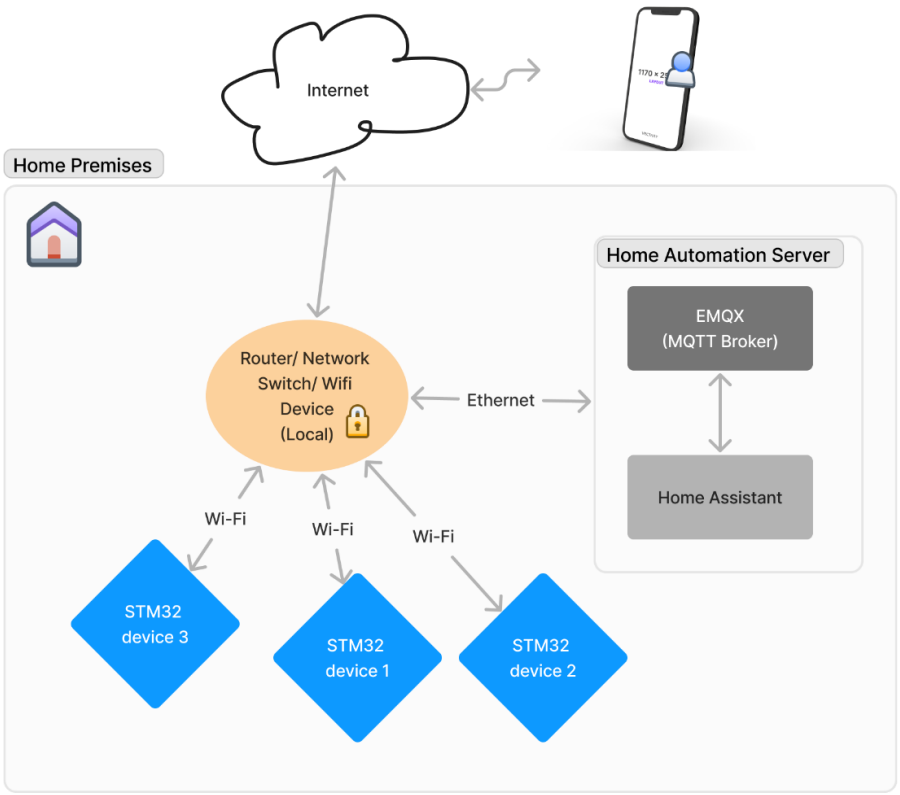
- ADC: read LDR values
- General GPIO output: control relays and buzzer

Smart Bell

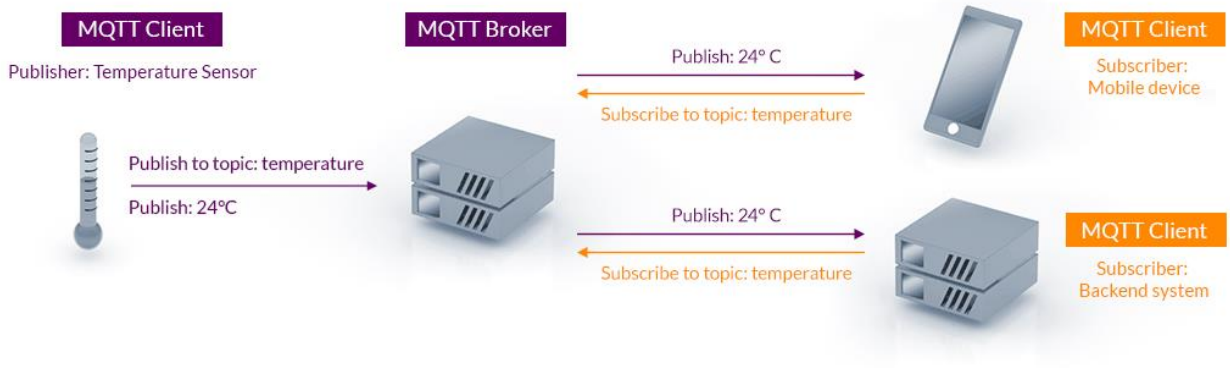
- External Interrupt: for window sensor configured to detect rising and falling edge trigger
- External Interrupt: for bell button configured to detect rising edge trigger
- General GPIO Output: control the door lock

D. Block Diagram/Schematic [CLO-3, 50 Marks]

Client Server Architecture and overall setup

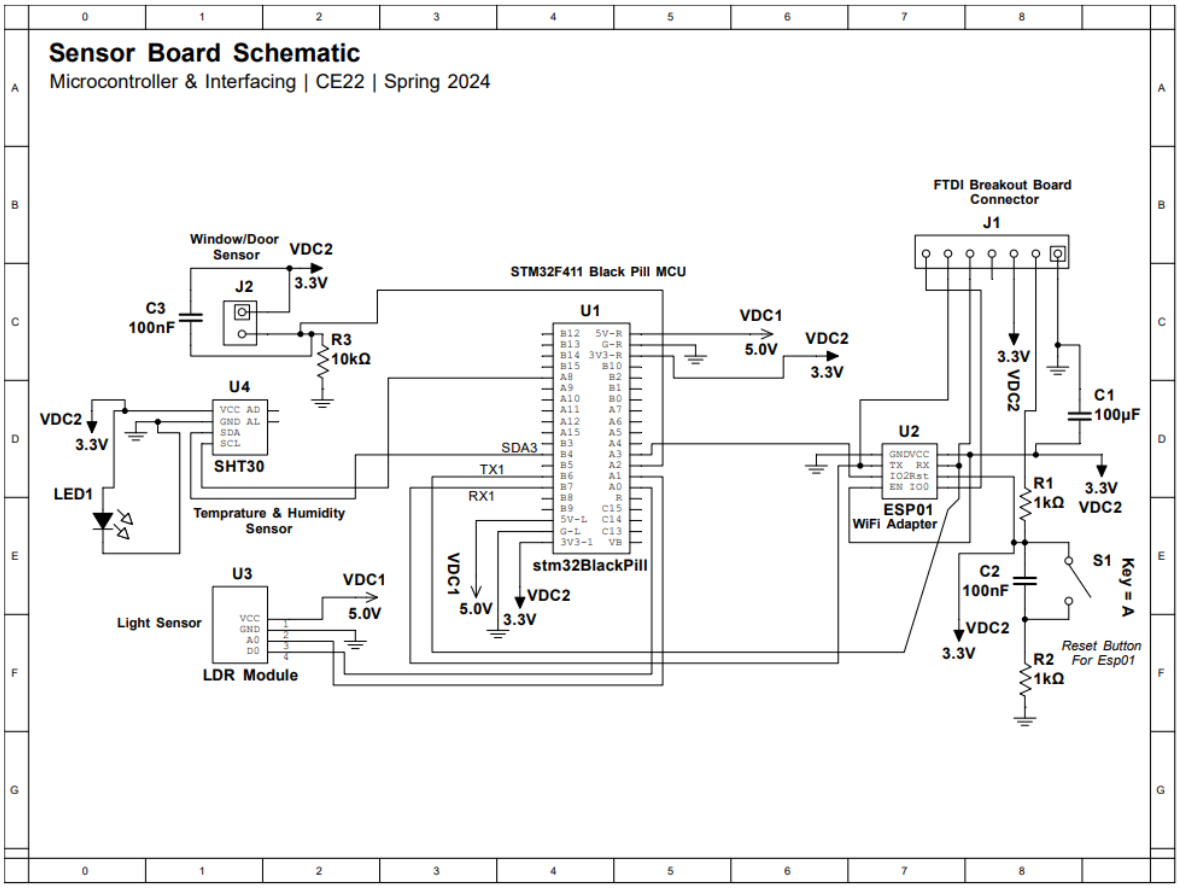


MQTT client and server communication

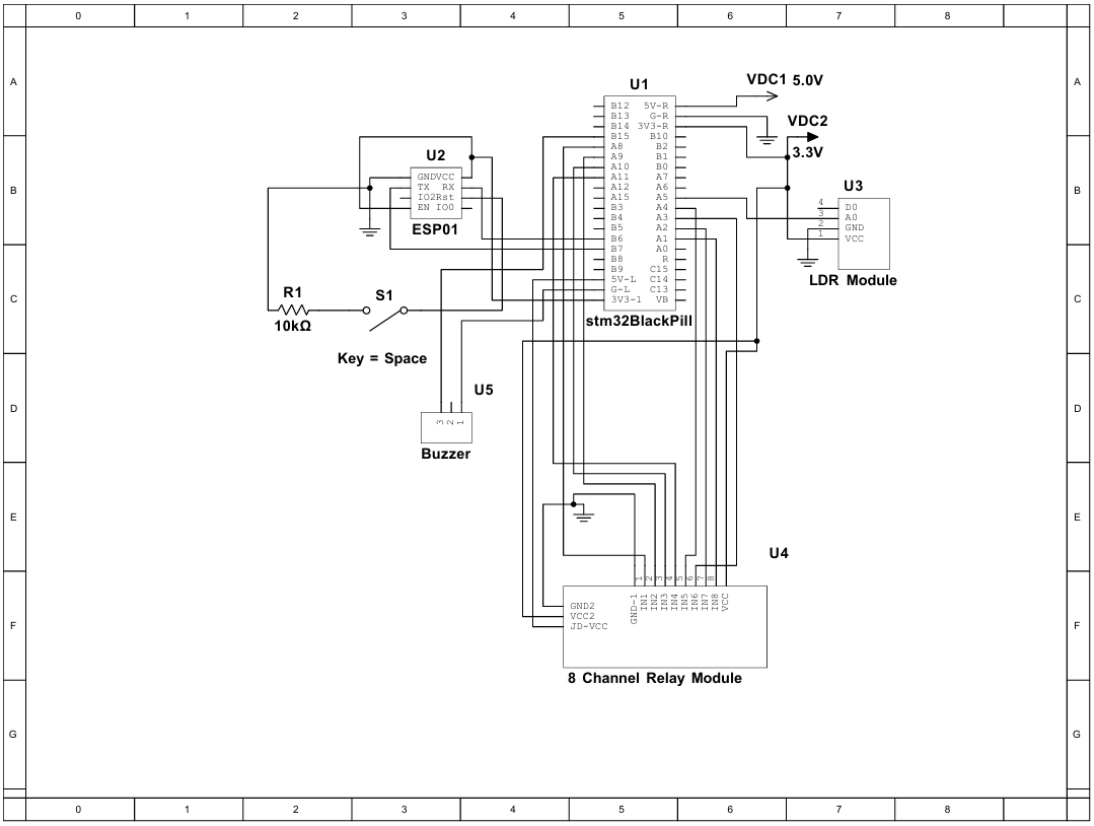


Source: mqtt.org

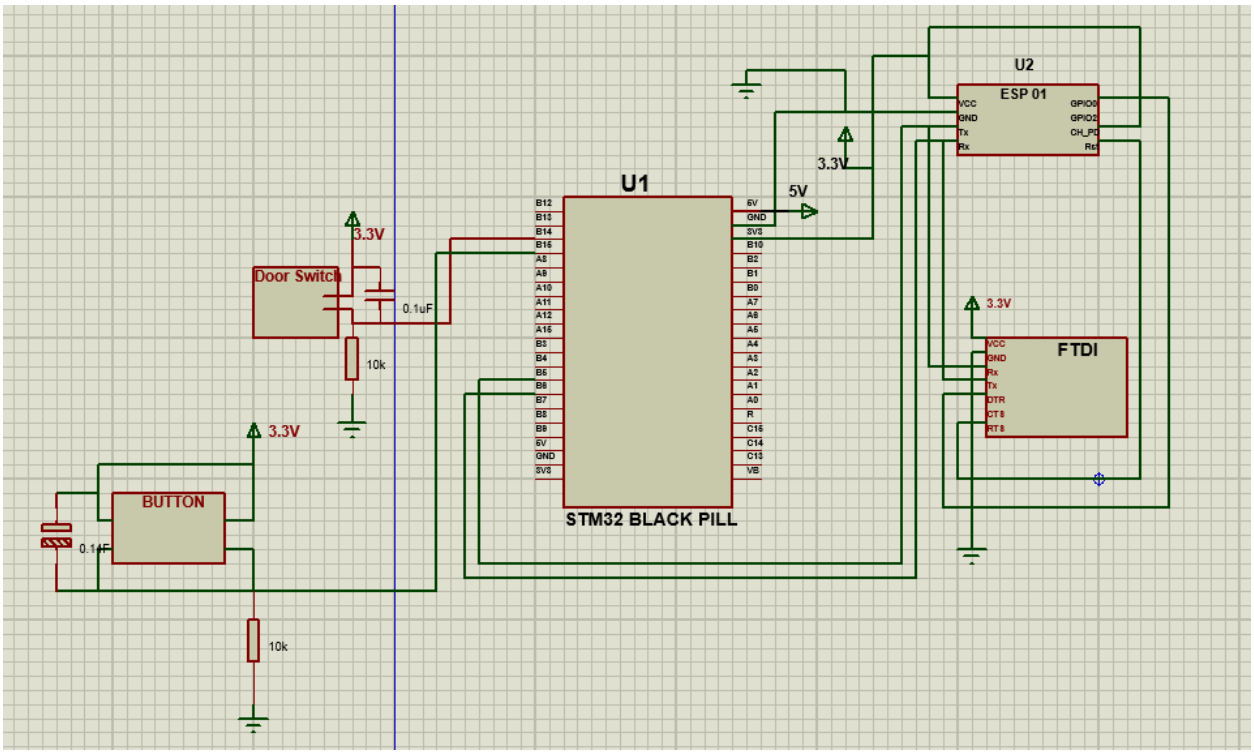
Sensor Array Board



Control Board

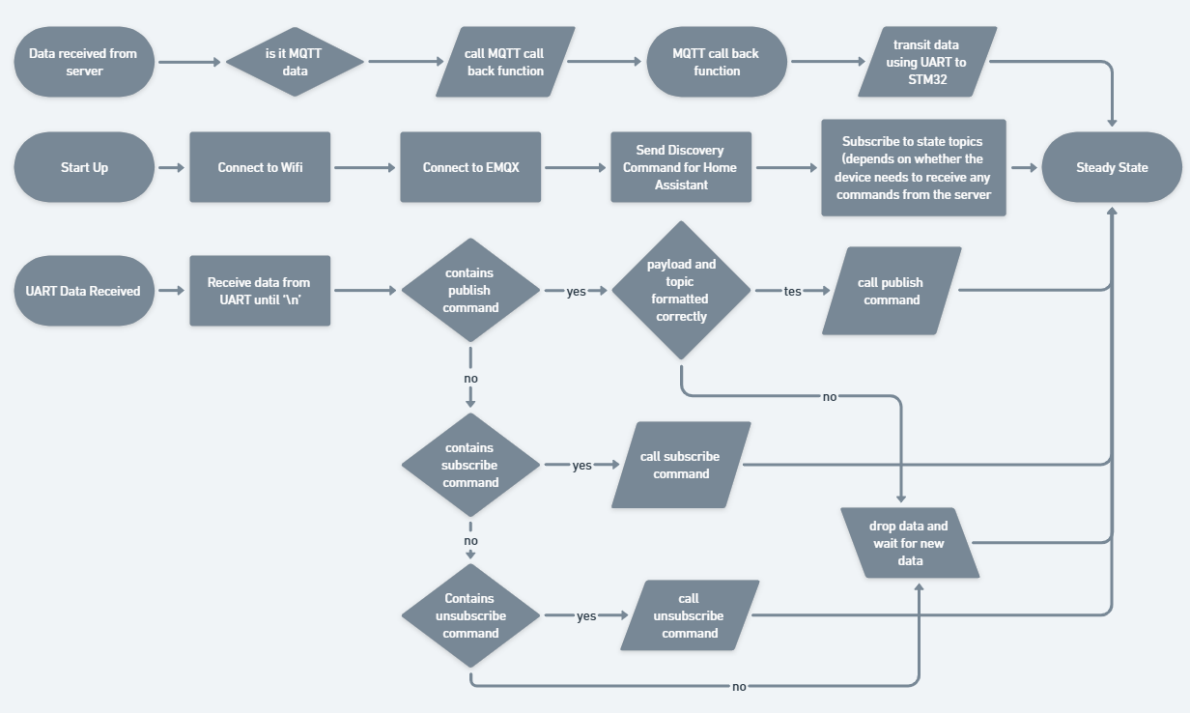


Smart Bell

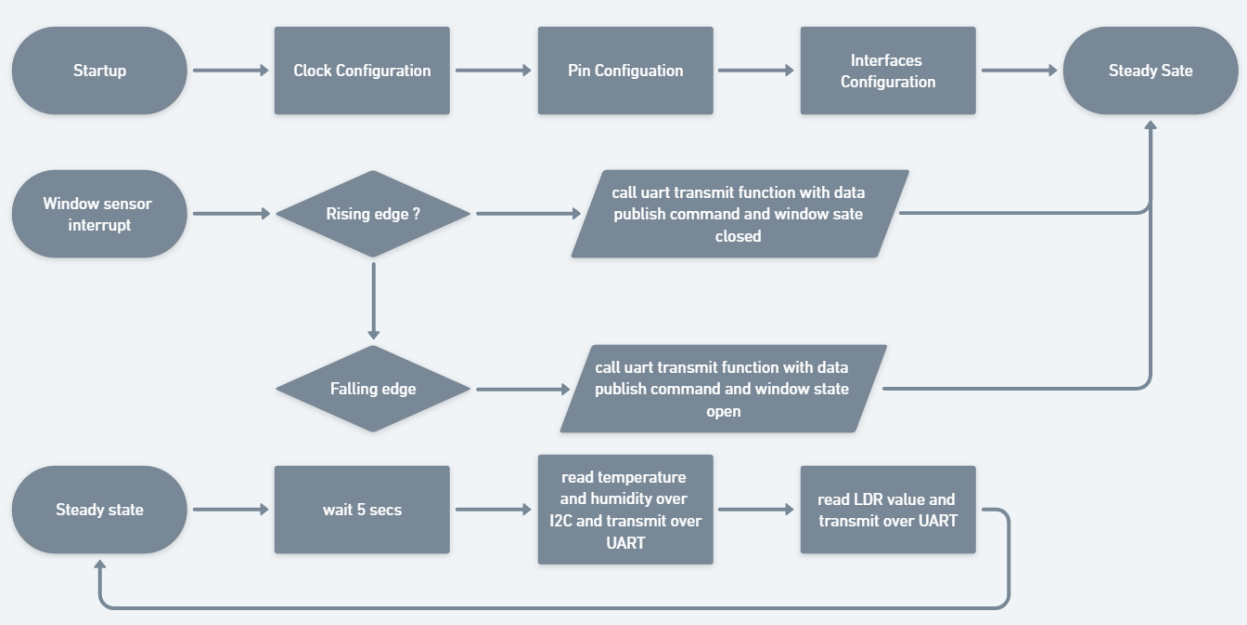


E. Flow Chart (Required at the time of final submission) [CLO-3, 50 Marks]

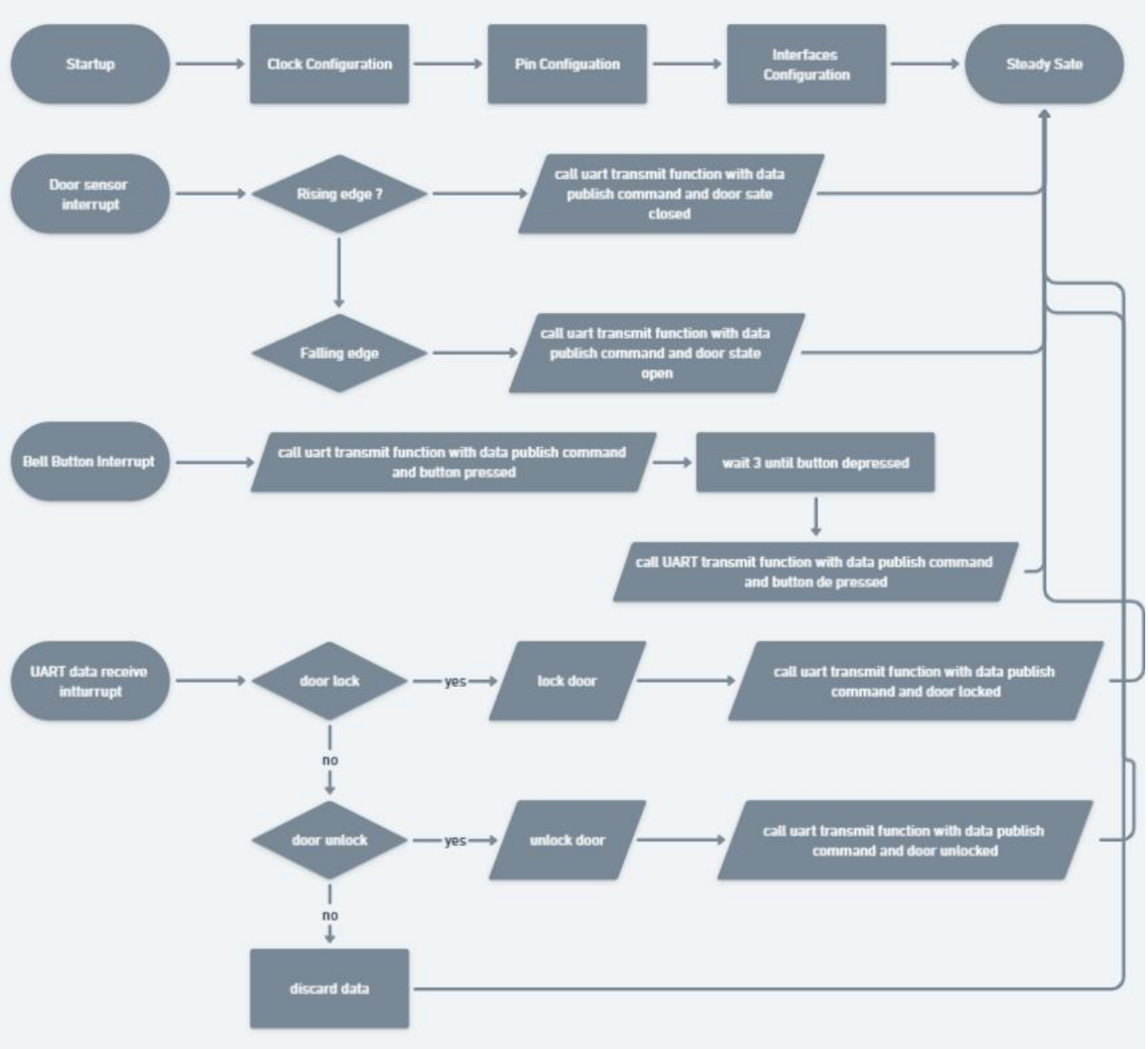
ESP Code



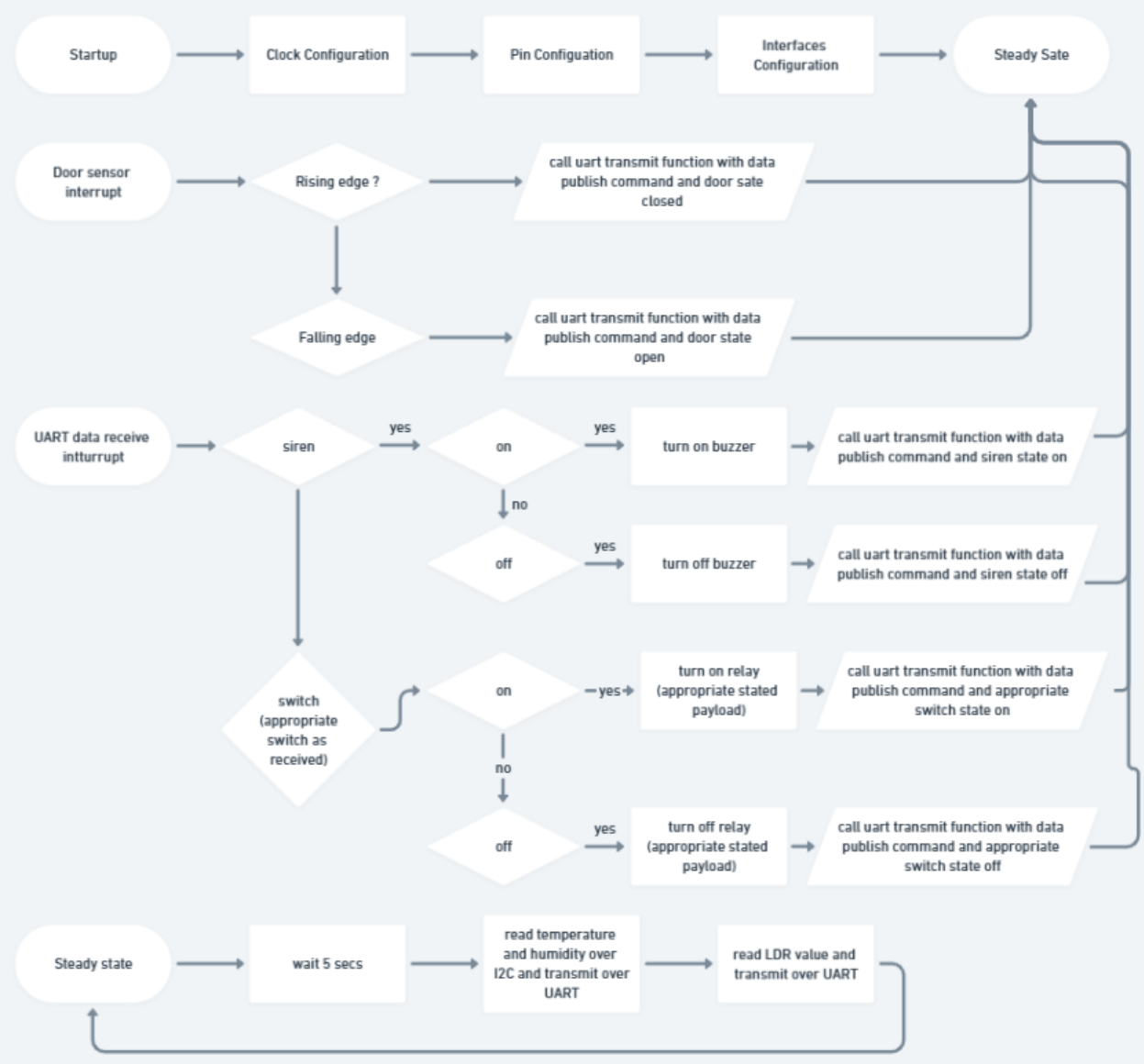
STM32 – Waleed : Sensor Array Board



STM32 – Rameeza : Smart Bell



STM32 – Ariba : Control Board



F. CEP (Project Complexity) Attributes - Describe Briefly [CLO-3, 50 Marks]

Attribute	Description	Complexity Level in your project
WP1: Depth of knowledge	The project shall involve in-depth engineering knowledge related to the	Interfacing two microcontrollers of different companies and architecture and also adding Wi-Fi capabilities

	area of Microprocessors, Microcontrollers & Interfacing [WK-4, Engineering Specialization].	
WP2: Range of conflicting requirements	The project has multiple conflicting requirements in terms of optimal usage of peripheral resources available on a Microcontroller.	Only specific pins on specific ports can be used for ADC. UART 1 couldn't be used in on the commonly used pins. In the control board managing a lot of pins is difficult. UART receive interrupt and call back is a challenge of its own as it is not always clear what the size of data will be. Also often in the transmitted data or received data, some garbage data would be present due to buffers used for each device and each buffer for TX and RX. On STM32 we had to use DMA to solve this issue and on ESP we had to do some to fix that any publish command would have its data in JSON format
WP5 Extent of applicable codes	The projects expose the students to broadly defined problems which require the development of codes that may be partially outside those encompassed by well-documented standards.	Handling multiple inturrupts is a challenge. The biggest one in code on STM was string maniulation as the library functions for this that are commonly available in c++ were not available in C. In ESP due to very little RAM, the code had to be written in a manner that would not saturate, this issue was faced in making discovery commands that why control board and bell button ESP have multiple functions for each discovery command. ADC values would be sometimes random so three conversion had to be done to get the average value that could be transmitted.
WP7 Interdependence	The projects shall have multiple components at the hardware and software level.	SHT30 is rather a less commonly used sensor, and its datasheet is not very well written. There were no exact drivers/libraries for this online that would work with STM32F4 series. Fortunately, a library written for STM32L series was found which was modified a bit to be compatible with STM32F4 series. The second challenge was controlling the relays required a certain amount of current an voltage levels and we were constantly having trouble with it. The relay board was orignally made to work on 5v system and to work with Arduino which 5v system and an onboard 7805 5v regulator capable of providing the necessary voltage and current meanwhile STM works on 3.3v system. Fortunately, separating relay voltage and trigger voltage by fully utilizing the opto couplers on the relay board solved this issue. There was also an issue with the buzzer as the buzzer we originally got required a sine wave

		but for simplicity we wanted to stick to digital input and output, so we swapped out the buzzer for a simple low level trigger buzzer.

G. Code [CLO-3, 50 Marks]

Sensor Array Board

ESP01-Waleed

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>

#define readyPin 2
// #define WDT_TIMEOUT 20000

// WiFi credentials
const char *ssid = "WIFI SSD";
const char *password = "WIFI PASSWORD";

// MQTT Broker
const char *mqtt_server = "emqx.home.mwaleedh.com.pk";
const int mqtt_port = 8883; // MQTT port
const char *mqtt_user = "USERNAME";
const char *mqtt_password = "PASSWORD";
const char *mqtt_client_id = "esp01_waleed";
bool switchState = false; // Initial state
float temperature = 0.0;  // Initial temperature value

WiFiClientSecure espClient;
PubSubClient client(espClient);

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
```

```
      Serial.println("MQTT connected");
      digitalWrite(readyPin, HIGH);
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
}

void callback(char *topic, byte *payload, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.println(topic);

  // Parse JSON payload
  JsonDocument doc;
  DeserializationError error = deserializeJson(doc, payload, length);
  if (error) {
    Serial.print("deserializeJson() failed: ");
    Serial.println(error.c_str());
    return;
  }
  // digitalWrite(LED, LOW);

  // Extract values from JSON and update variables
  // switchState = doc["switch"];
  // temperature = doc["temperature"];

  // Serial.print("Switch state: ");
  // Serial.println(switchState);
  // Serial.print("Temperature: ");
  // Serial.println(temperature);
  delay(1000);
}
unsigned long previousMillis = 0;

void nonBlockingDelay(unsigned long interval) {
  static unsigned long previousMillis = 0;
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    // Perform action here...
  }
```

```
}

void deviceDiscoveryHA() {
  char topic[128];
  char buffer1[512];
  char buffer2[512];
  char buffer3[512];
  char buffer4[512];
  char uid[128];
  JsonDocument doc;
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/binary_sensor/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_BS/config");

  // creating payload for Window Sensor
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_BS");
  doc["name"] = "Window Sensor";
  doc["obj_id"] = "mqtt_window_sensor";
  doc["uniq_id"] = uid;
  doc["stat_t"] = "esp01_waleed/sensors/window_sensor";
  doc["value_template"] = "{{value_json.state}}";
  doc["payload_on"] = "open";
  doc["payload_off"] = "close";
  doc["payload_available"] = "available";
  doc["not_payload_available"] = "not_available";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Sensing Device";
  device["mf"] = "Waleed";
  device["mdl"] = "ESP01";
  device["sw"] = "0.2";
  device["hw"] = "1.0";
  // device["cu"] = "http://192.168.1.226/config";  //web interface for device,
  // with discovery toggle
  serializeJson(doc, buffer1);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer1, true);

  // Creating topic for light sensor
  doc.clear();
  // creating topic here
```

```
strcpy(topic, "homeassistant/sensor/");
strcat(topic, mqtt_client_id);
strcat(topic, "_LS/config");

// creating payload for Light Sensor
strcpy(uid, mqtt_client_id);
strcat(uid, "_LS");
doc["name"] = "Light Sensor";
doc["obj_id"] = "mqtt_light_sensor";
doc["dev_cla"] = "illuminance";
doc["uniq_id"] = uid;
doc["stat_t"] = "esp01_waleed/sensors/lightlevel";
doc["unit_of_meas"] = "lx";
doc["value_template"] = "{{value_json.lux}}";
doc["not_payload_available"] = "not_available";

JsonObject deviceL = doc.createNestedObject("device");
deviceL["ids"] = mqtt_client_id;
deviceL["name"] = "Sensing Device";
serializeJson(doc, buffer2);
// Publish discovery topic and payload (with retained flag)
client.publish(topic, buffer2, true);

// creating topic for humidity sensor
doc.clear();
// creating topic here
strcpy(topic, "homeassistant/sensor/");
strcat(topic, mqtt_client_id);
strcat(topic, "_HM/config");

// creating payload for humidity Sensor
strcpy(uid, mqtt_client_id);
strcat(uid, "_HM");
doc["name"] = "Humidity";
doc["obj_id"] = "mqtt_RH_sensor";
doc["dev_cla"] = "humidity";
doc["uniq_id"] = uid;
doc["stat_t"] = "esp01_waleed/sensors/TH_sensor";
doc["unit_of_meas"] = "%";
doc["value_template"] = "{{value_json.humidity}}";

JsonObject deviceH = doc.createNestedObject("device");
deviceH["ids"] = mqtt_client_id;
deviceH["name"] = "Sensing Device";
serializeJson(doc, buffer3);
```

```
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer3, true);

  // creating topic for temprature sensor
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/sensor/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_TS/config");

  // creating payload for temprature Sensor
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_TS");
  doc["name"] = "Temprature";
  doc["obj_id"] = "mqtt_temprature_sensor";
  doc["dev_cla"] = "temperature";
  doc["uniq_id"] = uid;
  doc["stat_t"] = "esp01_waleed/sensors/TH_sensor";
  doc["unit_of_meas"] = "°C";
  doc["value_template"] = "{{value_json.temprature}}";

  JsonObject deviceT = doc.createNestedObject("device");
  deviceT["ids"] = mqtt_client_id;
  deviceT["name"] = "Sensing Device";
  serializeJson(doc, buffer4);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer4, true);
}

void setup() {
  Serial.begin(115200);
  Serial.println("Turning On...");
  delay(5000);

  WiFi.begin(ssid, password);
  Serial.println("Wifi Function called");
  while (WiFi.status() != WL_CONNECTED) {
    nonBlockingDelay(50000);
    Serial.println("Connecting to WiFi...");
    // delay(5000);
  }
  Serial.println("WiFi connected");

  // if (root_ca != NULL) {
  //   espClient.setCACert(root_ca);
```

```
  // } else {
  espClient.setInsecure();
  //}
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  client.setBufferSize(512); // increasing buffer size

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("MQTT connected");
      deviceDiscoveryHA();
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
  // send initial config message here, intial subscribe here
  // client.subscribe("topic_name");
  // Serial.println("Subscribed to topic");
  // client.publish(topic.c_str(), message.c_str());

  // action config here
  // pinMode(LED, OUTPUT);
}

void loop() {
  // ESP.wdtFeed();
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // digitalWrite(LED, HIGH);
  //  Read switch state and temperature from serial
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    if (input.startsWith("publish:")) {
      int separatorIndex = input.indexOf('|');
      if (separatorIndex != -1) {
        String topic = input.substring(8, separatorIndex);
        String payload = input.substring(separatorIndex + 1);
        JsonDocument doc;
        DeserializationError error = deserializeJson(doc, payload);
```

```
        if (!error) {
          switchState = doc["switch"];
          temperature = doc["temperature"];
          client.publish(topic.c_str(), payload.c_str());
          Serial.println("Published message:");
          Serial.println(payload);
        }
        //   // char buffer[256];
        //   // size_t n = serializeJson(doc, buffer);
      }
    } else if (input.startsWith("subscribe:")) {
      String topic = input.substring(10);
      client.subscribe(topic.c_str());
      Serial.print("Subscribed to topic: ");
      Serial.println(topic);
    } else if (input.startsWith("unsubscribe:")) {
      String topic = input.substring(12);
      client.unsubscribe(topic.c_str());
      Serial.print("Unsubscribed from topic: ");
      Serial.println(topic);
    }
  }
}
```

STM32-WALEED

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2024 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.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
```

```c
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "sht3x.h" // library by https://github.com/henriheimann/stm32-hal-sht3x
#include "string.h"
#include <stdio.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 */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;

I2C_HandleTypeDef hi2c2;

UART_HandleTypeDef huart1;
DMA_HandleTypeDef hdma_usart1_rx;
DMA_HandleTypeDef hdma_usart1_tx;

/* USER CODE BEGIN PV */
int isDataSent = 1;
uint32_t adcValue = 0;
char data[100];
int windowStatus;
float humidity, temperature;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
```

```c
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_I2C2_Init(void);
/* USER CODE BEGIN PFP */
void transmistDataOverUART1(char *data);
void sendLightSenorData(void);
void setWindowSensorPinAsInput(void);
void sendTempAndHumidityData(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

// Data Transmit CALL BACK
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
    isDataSent = 1;
}

// SHT30 code
sht3x_handle_t handle = { .i2c_handle = &hi2c2, .device_address =
SHT3X_I2C_DEVICE_ADDRESS_ADDR_PIN_LOW };

int checkSHT30(void) {
    if (!sht3x_init(&handle)) {
        return 0;

    } else {
        return 1;
    }
}

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
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_DMA_Init();
    MX_ADC1_Init();
    MX_USART1_UART_Init();
    MX_I2C2_Init();
    /* USER CODE BEGIN 2 */
    void setWindowSensorPinAsInput(void);
    if (HAL_GPIO_ReadPin(WINDOW_SENSOR_GPIO_Port, WINDOW_SENSOR_Pin)
            == GPIO_PIN_SET) {
        windowStatus = 1;
        char buff[] =
                "publish:esp01_waleed/sensors/window_sensor|{\"state\":\"close\"}
\n\0";
        HAL_UART_Transmit(&huart1, (uint8_t*) buff, strlen(buff),
        HAL_MAX_DELAY);

    } else {
        windowStatus = 0;
        char buff[] =
                "publish:esp01_waleed/sensors/window_sensor|{\"state\":\"open\"}\
n\0";
        HAL_UART_Transmit(&huart1, (uint8_t*) buff, strlen(buff),
        HAL_MAX_DELAY);
    }
    MX_GPIO_Init();
    checkSHT30();

    /* USER CODE END 2 */
```

```
    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1) {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
        sendLightSenorData();
        sht3x_read_temperature_and_humidity(&handle, &temperature, &humidity);
        sendTempAndHumidityData();

        HAL_Delay(5000);
    }
    /* 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_SCALE2);

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    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_HSE;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
```

```
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK) {
        Error_Handler();
    }
}

/**
 * @brief ADC1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_ADC1_Init(void) {

    /* USER CODE BEGIN ADC1_Init 0 */

    /* USER CODE END ADC1_Init 0 */

    ADC_ChannelConfTypeDef sConfig = { 0 };

    /* USER CODE BEGIN ADC1_Init 1 */

    /* USER CODE END ADC1_Init 1 */

    /** Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)
     */
    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    hadc1.Init.DMAContinuousRequests = DISABLE;
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        Error_Handler();
    }
```

```c
    /** Configure for the selected ADC regular channel its corresponding rank in
the sequencer and its sample time.
     */
    sConfig.Channel = ADC_CHANNEL_1;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        Error_Handler();
    }
    /* USER CODE BEGIN ADC1_Init 2 */

    /* USER CODE END ADC1_Init 2 */

}

/**
 * @brief I2C2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_I2C2_Init(void) {

    /* USER CODE BEGIN I2C2_Init 0 */

    /* USER CODE END I2C2_Init 0 */

    /* USER CODE BEGIN I2C2_Init 1 */

    /* USER CODE END I2C2_Init 1 */
    hi2c2.Instance = I2C2;
    hi2c2.Init.ClockSpeed = 1000;
    hi2c2.Init.DutyCycle = I2C_DUTYCYCLE_2;
    hi2c2.Init.OwnAddress1 = 0;
    hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
    hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
    hi2c2.Init.OwnAddress2 = 0;
    hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
    hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
    if (HAL_I2C_Init(&hi2c2) != HAL_OK) {
        Error_Handler();
    }
    /* USER CODE BEGIN I2C2_Init 2 */

    /* USER CODE END I2C2_Init 2 */
```

```c
}

/**
 * @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 = 115200;
    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 */

}

/**
 * Enable DMA controller clock
 */
static void MX_DMA_Init(void) {

    /* DMA controller clock enable */
    __HAL_RCC_DMA2_CLK_ENABLE();

    /* DMA interrupt init */
    /* DMA2_Stream2_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);
```

```c
    /* DMA2_Stream7_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA2_Stream7_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream7_IRQn);

}

/**
 * @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_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

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

    /*Configure GPIO pin : WINDOW_SENSOR_Pin */
    GPIO_InitStruct.Pin = WINDOW_SENSOR_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(WINDOW_SENSOR_GPIO_Port, &GPIO_InitStruct);

    /* EXTI interrupt init*/
    HAL_NVIC_SetPriority(EXTI2_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(EXTI2_IRQn);

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

/* USER CODE BEGIN 4 */

// Interrupt handler
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
```

```
    if (GPIO_Pin == WINDOW_SENSOR_Pin) {

        if (HAL_GPIO_ReadPin(WINDOW_SENSOR_GPIO_Port, WINDOW_SENSOR_Pin)
                == GPIO_PIN_SET) {
            strcpy(data,
                    "publish:esp01_waleed/sensors/window_sensor|{\"state\":\"clos
e\"}\n\0");
//          HAL_UART_Transmit(&huart1, (uint8_t*) data, strlen(data),
//          HAL_MAX_DELAY);

            transmistDataOverUART1(data);
            windowStatus = 1;
            return;

        } else {

            strcpy(data,
                    "publish:esp01_waleed/sensors/window_sensor|{\"state\":\"open
\"}\n\0");

            transmistDataOverUART1(data);
            windowStatus = 0;
            return;

        }

    }
}

void transmistDataOverUART1(char *dataToTransmit) {
    while (isDataSent != 1) {
        // wait untill previous transmission is ongoing
    }
    HAL_UART_Transmit_DMA(&huart1, (uint8_t*) dataToTransmit,
            strlen(dataToTransmit));
    isDataSent = 0;

}

void sendLightSenorData(void) {
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    unsigned int adc_value = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    HAL_Delay(10);
```

```
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adc_value += HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    HAL_Delay(10);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adc_value += HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);

    adc_value = (adc_value / 3);
    adcValue = adc_value;

    char buffer[70];
    sprintf(buffer,
            "publish:esp01_waleed/sensors/lightlevel|{\"lux\":\"%d\"}\n",
            adc_value);
//  HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer),
//          HAL_MAX_DELAY);
    transmistDataOverUART1(buffer);
    HAL_Delay(10);
}

void setWindowSensorPinAsInput(void) {
    GPIO_InitTypeDef GPIO_InitStruct = { 0 };
    /*Configure GPIO pin : WINDOW_SENSOR_Pin */
    GPIO_InitStruct.Pin = WINDOW_SENSOR_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(WINDOW_SENSOR_GPIO_Port, &GPIO_InitStruct);
}

void sendTempAndHumidityData(void) {
    char buffer[70];
    sprintf(buffer,
            "publish:esp01_waleed/sensors/TH_sensor|{\"temprature\":\"%d\"}\n",
            (int) temperature);
    transmistDataOverUART1(buffer);
    HAL_Delay(10);
    sprintf(buffer,
            "publish:esp01_waleed/sensors/TH_sensor|{\"humidity\":\"%d\"}\n",
            (int) humidity);
    transmistDataOverUART1(buffer);
    HAL_Delay(10);
```

```
}
/* 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 */
```

Control Board

ESP01-Ariba

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>

#define readyPin 2
// #define WDT_TIMEOUT 20000
```

```
// WiFi credentials
const char *ssid = "";    // !! Enter WiFi name here !!
const char *password = ""; // !! Enter WiFi password here !!

// MQTT Broker settings
const char *mqtt_server = "emqx.home.mwaleedh.com.pk";
const int mqtt_port = 8883;
const char *mqtt_user = ""; // !! Provided server username here !!
const char *mqtt_password = ""; //!! Provided server password here !!
const char *mqtt_client_id = "esp01_ariba";

// function prototypes here
void deviceDiscoveryHA_LightSensor();
void deviceDiscoveryHA_Switch1();
void deviceDiscoveryHA_Switch2();
void deviceDiscoveryHA_Switch3();
void deviceDiscoveryHA_Switch4();
void deviceDiscoveryHA_Switch5();
void deviceDiscoveryHA_Switch6();
void deviceDiscoveryHA_Switch7();
void deviceDiscoveryHA_Switch8();
void deviceDiscoveryHA_Siren();

void reconnect();
void nonBlockingDelay(unsigned long interval);

// pre defined objects here

WiFiClientSecure espClient;
PubSubClient client(espClient);

void callback(char *topic, byte *payload, unsigned int length) {
  Serial.flush();
  Serial.print("topic:");
  Serial.print(topic);
  // String recv_payload = String((char *)payload);
  Serial.print("|payload:");
  char *recv_payload = (char *)malloc(length);
  for (int i = 0; i < length; i++) {
    recv_payload[i] = (char)payload[i];
    Serial.print((char)payload[i]);
  }
  Serial.print("\n");
  // strcat(recv_payload,"@");
  // Serial.print(recv_payload);
```

```
  // Serial.print("\n")
  // Serial.println(length);

  // Parse JSON payload
  // JsonDocument doc;
  // DeserializationError error = deserializeJson(doc, payload, length);
  // if (error) {
  //   String recv_payload = String((char *)payload);
  //   Serial.print("|payload:");
  //   Serial.println(recv_payload);
  // }
  // digitalWrite(LED, LOW);

  // Extract values from JSON and update variables
  // switchState = doc["switch"];
  // temperature = doc["temperature"];

  // Serial.print("Switch state: ");
  // Serial.println(switchState);
  // Serial.print("Temperature: ");
  // Serial.println(temperature);
  delay(1000);
}

unsigned long previousMillis = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("Turning On...");
  delay(5000);

  WiFi.begin(ssid, password);
  Serial.println("Wifi Function called");
  while (WiFi.status() != WL_CONNECTED) {
    nonBlockingDelay(50000);
    Serial.println("Connecting to WiFi...");
    // delay(5000);
  }
  Serial.println("WiFi connected");

  // if (root_ca != NULL) {
  //   espClient.setCACert(root_ca);
  // } else {
  espClient.setInsecure();
  //}
```

```
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  client.setBufferSize(512); // increasing buffer size

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("MQTT connected");
      deviceDiscoveryHA_LightSensor(); // home assistant config sent here
      deviceDiscoveryHA_Switch1();
      deviceDiscoveryHA_Switch2();
      deviceDiscoveryHA_Switch3();
      deviceDiscoveryHA_Switch4();
      deviceDiscoveryHA_Switch5();
      deviceDiscoveryHA_Switch6();
      deviceDiscoveryHA_Switch7();
      deviceDiscoveryHA_Switch8();
      deviceDiscoveryHA_Siren();

      digitalWrite(
          readyPin,
          HIGH); // GPIO2 goes high when server connection is established

    } else {
      digitalWrite(readyPin, LOW); // GPIO2 goes low when server is disconnected
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
}

void loop() {
  // ESP.wdtFeed();
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  //  Read data from serial
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    if (input.startsWith("publish:")) {
      int separatorIndex = input.indexOf('|');
```

```
      if (separatorIndex != -1) {
        String topic = input.substring(8, separatorIndex);
        String payload = input.substring(separatorIndex + 1);
        JsonDocument doc;
        DeserializationError error = deserializeJson(doc, payload);
        if (!error) { // error checking json format
          client.publish(topic.c_str(), payload.c_str());

          // uncomment this for debugging

          // Serial.println("Published message:");
          // Serial.println(payload);
        }
      }
    } else if (input.startsWith("subscribe:")) {
      String topic = input.substring(10);
      client.subscribe(topic.c_str());
      Serial.print("Subscribed to topic: ");
      Serial.println(topic);
    } else if (input.startsWith("unsubscribe:")) {
      String topic = input.substring(12);
      client.unsubscribe(topic.c_str());
      Serial.print("Unsubscribed from topic: ");
      Serial.println(topic);
    }
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("MQTT connected");
      digitalWrite(
          readyPin,
          HIGH); // GPIO2 goes high when server connection is established
    } else {
      digitalWrite(readyPin, LOW); // GPIO2 goes low when server is disconnected
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
}
```

```
void nonBlockingDelay(unsigned long interval) {
  static unsigned long previousMillis = 0;
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    // Perform action here...
  }
}

void deviceDiscoveryHA_LightSensor() {
  char topic[128];
  char buffer[512];
  char uid[128];
  JsonDocument doc;
  // Creating topic for light sensor
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/sensor/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_LS/config");

  // creating payload for Light Sensor
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_LS");
  doc["name"] = "Light Sensor";
  doc["obj_id"] = "mqtt_light_sensor";
  doc["dev_cla"] = "illuminance";
  doc["uniq_id"] = uid;
  doc["stat_t"] = "esp01_ariba/sensors/lightlevel";
  doc["unit_of_meas"] = "lx";
  doc["value_template"] = "{{value_json.lux}}";
  doc["not_payload_available"] = "not_available";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  device["mf"] = "Ariba";
  device["mdl"] = "ESP01";
  device["sw"] = "1.0";
  device["hw"] = "1.0";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
```

```
}
void deviceDiscoveryHA_Switch1() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  // SWITCH 1 HERE
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw1/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW1");          // TODO: change this for further switches
  doc["name"] = "switch1";      // TODO: change this for further switches
  doc["obj_id"] = "switch_one"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch1"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch1/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch1/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}
void deviceDiscoveryHA_Switch2() {
```

```
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  // SWITCH 2 HERE
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw2/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW2");          // TODO: change this for further switches
  doc["name"] = "switch2";      // TODO: change this for further switches
  doc["obj_id"] = "switch_two"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch2"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch2/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch2/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Switch3() {
  char topic[128];
```

```
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw3/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW3");            // TODO: change this for further switches
  doc["name"] = "switch3";        // TODO: change this for further switches
  doc["obj_id"] = "switch_three"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch3"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch3/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch3/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Switch4() {
  char topic[128];
  char buffer[512];
  char uid[128];
```

```
  String subTopic;
  JsonDocument doc;
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw4/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW4");           // TODO: change this for further switches
  doc["name"] = "switch4";       // TODO: change this for further switches
  doc["obj_id"] = "switch_four"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch4"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch4/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch4/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Switch5() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;
  doc.clear();
```

```
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw5/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW5");           // TODO: change this for further switches
  doc["name"] = "switch5";       // TODO: change this for further switches
  doc["obj_id"] = "switch_five"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch5"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch5/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch5/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Switch6() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
```

```
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw6/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW6");          // TODO: change this for further switches
  doc["name"] = "switch6";      // TODO: change this for further switches
  doc["obj_id"] = "switch_six"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch6"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch6/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch6/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}
void deviceDiscoveryHA_Switch7() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw7/config"); // TODO: change this for further switches
```

```
  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW7");            // TODO: change this for further switches
  doc["name"] = "switch7";        // TODO: change this for further switches
  doc["obj_id"] = "switch_seven"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch7"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch7/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch7/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";

  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Switch8() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw8/config"); // TODO: change this for further switches

  // creating payload for switch1
```

```
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_SW8");            // TODO: change this for further switches
  doc["name"] = "switch8";        // TODO: change this for further switches
  doc["obj_id"] = "switch_eight"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/switch8"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/switch8/set"; // TODO: change this for further switches
  subTopic =
      "esp01_ariba/switch8/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";

  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_Siren() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/siren/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sr/config");

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
```

```
  strcat(uid, "_SR");      // TODO: change this for further switches
  doc["name"] = "siren";   // TODO: change this for further switches
  doc["obj_id"] = "siren"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_ariba/siren"; // TODO: change this for further switches
  doc["command_topic"] =
      "esp01_ariba/siren/set";        // TODO: change this for further switches
  subTopic = "esp01_ariba/siren/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  // doc["state_template"] = "{{value_json.state}}";
  // doc["command_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Control Device";

  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}
```

STM32-Ariba

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2024 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.
 *
```

```
 ******************************************************************************
 */
/* 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 */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;

UART_HandleTypeDef huart1;
DMA_HandleTypeDef hdma_usart1_tx;
DMA_HandleTypeDef hdma_usart1_rx;

/* USER CODE BEGIN PV */
int isDataSent = 1;
uint32_t adcValue = 0;
char data[100];
uint8_t rxByte;
char rxBuffer[100]; // Buffer to store received data
uint16_t rxIndex = 0;  // Index for the received data buffer
char topic[30];
char payload[10];
int counter = 0;
/* USER CODE END PV */
```

```c
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void sendLightSenorData(void);
void turnOnRelay(char *topic, char *payload);
void transmistDataOverUART1(char *dataToTransmit);
// Data Transmit CALL BACK
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
    isDataSent = 1;

}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    counter++;
//
//  rxBuffer[rxIndex++] = (char) rxByte;
//
//  // Prevent buffer overflow
//  if (rxIndex >= sizeof(rxBuffer)) {
//      rxIndex = 0; // Reset index if buffer overflows
//  }
//  if (rxIndex == 10) {
//      transmistDataOverUART1(rxBuffer);
//
//  }
    HAL_UART_Receive_DMA(&huart1, &rxByte, 1);
}
//void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
//  if ((char) rxByte == '\n') { //  if end of transmission then start processing
//      rxBuffer[rxIndex] = '\0';
//      rxIndex = 0;
//      transmistDataOverUART1(rxBuffer);
//      if (strcmp(rxBuffer,
//              "topic:esp01_ariba/siren/set|{\"state\"}:{\"on\"}")) {
//          HAL_GPIO_WritePin(BUZZER_GPIO_Port, BUZZER_Pin, GPIO_PIN_SET);
//          char data[] = "publish:esp01_ariba/siren|{\"state\":\"on\"}";
```

```
//          transmistDataOverUART1(data);
//      }
//      if (strcmp(rxBuffer,
//              "topic:esp01_ariba/siren/set|{\"state\"}:{\"off\"}")) {
//          HAL_GPIO_WritePin(BUZZER_GPIO_Port, BUZZER_Pin, GPIO_PIN_RESET);
//          char data[] = "publish:esp01_ariba/siren|{\"state\":\"off\"}";
//          transmistDataOverUART1(data);
//      }
////          char switch1[] = "switch";
////          char siren[] = "siren";
////          char payloadOn[] = "on";
////          char payloadOff[] = "off";
////
////          //transmistDataOverUART1((char*) rxBuffer);
////          // processing incoming data here
////          // Separate the topic and payload
////          const char delim[] = ":|";
////
////          // Tokenize the string to get the first token
////          char *token = strtok(rxBuffer, delim);
////
////          // Check if the first token is "topic"
////          if (token != NULL && strcmp(token, "topic") == 0) {
////              // Get the topic
////              char *topic = strtok(NULL, delim);
////              // Get the payload
////              char *payload = strtok(NULL, delim);
////
////              // Print the results
////              if (topic != NULL && payload != NULL) {
////                  if (strstr(topic, switch1)) {
////                      turnOnRelay(topic, payload);
////                  }
////                  if (strstr(topic, switch1)) {
////                      if (strstr(payload, payloadOn)) {
////                          HAL_GPIO_WritePin(BUZZER_GPIO_Port, BUZZER_Pin,
////                                  GPIO_PIN_SET);
////                          char data[] =
////                                  "publish:esp01_ariba/siren|{\"state\":\"on\"}
";
////                          transmistDataOverUART1(data);
////                      }
////                      if (strstr(payload, payloadOff)) {
////                          HAL_GPIO_WritePin(BUZZER_GPIO_Port, BUZZER_Pin,
////                                  GPIO_PIN_RESET);
```

```
////                        char data[] =
////                                "publish:esp01_ariba/siren|{\"state\":\"off\"
}";
////                        transmistDataOverUART1(data);
////                    }
////
////                }
////
////            } else {
////                printf("Failed to parse the string.\n");
////            }
////        } else {
////            printf("Invalid format.\n");
////        }
//
//  } else {
//      rxBuffer[rxIndex++] = (char) rxByte;
//
//      // Prevent buffer overflow
//      if (rxIndex >= sizeof(rxBuffer)) {
//          rxIndex = 0; // Reset index if buffer overflows
//      }
//  }
//
//  HAL_UART_Receive_DMA(&huart1, &rxByte, 1);
//
//}

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
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();
```

```c
  /* 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_DMA_Init();
  MX_ADC1_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_GPIO_WritePin(RELAY_1_GPIO_Port, RELAY_1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_2_GPIO_Port, RELAY_2_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_3_GPIO_Port, RELAY_3_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_4_GPIO_Port, RELAY_4_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_5_GPIO_Port, RELAY_5_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_6_GPIO_Port, RELAY_6_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_7_GPIO_Port, RELAY_7_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RELAY_8_GPIO_Port, RELAY_8_Pin, GPIO_PIN_SET);
  HAL_UART_Receive_DMA(&huart1, &rxByte, 1);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1) {
      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */
      //HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
      //HAL_GPIO_TogglePin(ONBOARD_LED_GPIO_Port, ONBOARD_LED_Pin);
      //HAL_GPIO_TogglePin(RELAY_1_GPIO_Port, RELAY_1_Pin);
      HAL_GPIO_TogglePin(BUZZER_GPIO_Port, BUZZER_Pin);
      //HAL_GPIO_WritePin(RELAY_3_GPIO_Port, RELAY_3_Pin, GPIO_PIN_RESET);
      sendLightSenorData();
      HAL_Delay(3000);
      //HAL_GPIO_WritePin(RELAY_3_GPIO_Port, RELAY_3_Pin, GPIO_PIN_SET);
      //HAL_Delay(3000);
```

```c
    }
    /* 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_SCALE1);

    /** 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 = 8;
    RCC_OscInitStruct.PLL.PLLN = 100;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    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_3) != HAL_OK) {
        Error_Handler();
    }
```

```c
}

/**
 * @brief ADC1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_ADC1_Init(void) {

    /* USER CODE BEGIN ADC1_Init 0 */

    /* USER CODE END ADC1_Init 0 */

    ADC_ChannelConfTypeDef sConfig = { 0 };

    /* USER CODE BEGIN ADC1_Init 1 */

    /* USER CODE END ADC1_Init 1 */

    /** Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)
     */
    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    hadc1.Init.DMAContinuousRequests = DISABLE;
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        Error_Handler();
    }

    /** Configure for the selected ADC regular channel its corresponding rank in
the sequencer and its sample time.
     */
    sConfig.Channel = ADC_CHANNEL_5;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
```

```
        Error_Handler();
    }
    /* USER CODE BEGIN ADC1_Init 2 */

    /* USER CODE END ADC1_Init 2 */

}

/**
 * @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 = 115200;
    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 */

}

/**
 * Enable DMA controller clock
 */
static void MX_DMA_Init(void) {

    /* DMA controller clock enable */
```

```c
    __HAL_RCC_DMA2_CLK_ENABLE();

    /* DMA interrupt init */
    /* DMA2_Stream2_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);
    /* DMA2_Stream7_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA2_Stream7_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream7_IRQn);

}

/**
 * @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_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(ONBOARD_LED_GPIO_Port, ONBOARD_LED_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA,
            RELAY_7_Pin | RELAY_8_Pin | RELAY_3_Pin | RELAY_4_Pin | RELAY_5_Pin
                    | RELAY_6_Pin | RELAY_1_Pin | RELAY_2_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(BUZZER_GPIO_Port, BUZZER_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin : ONBOARD_LED_Pin */
    GPIO_InitStruct.Pin = ONBOARD_LED_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(ONBOARD_LED_GPIO_Port, &GPIO_InitStruct);
```

```
    /*Configure GPIO pins : RELAY_7_Pin RELAY_8_Pin RELAY_3_Pin RELAY_4_Pin
     RELAY_5_Pin RELAY_6_Pin RELAY_1_Pin RELAY_2_Pin */
    GPIO_InitStruct.Pin = RELAY_7_Pin | RELAY_8_Pin | RELAY_3_Pin | RELAY_4_Pin
            | RELAY_5_Pin | RELAY_6_Pin | RELAY_1_Pin | RELAY_2_Pin;
    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 : BUZZER_Pin */
    GPIO_InitStruct.Pin = BUZZER_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(BUZZER_GPIO_Port, &GPIO_InitStruct);

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

/* USER CODE BEGIN 4 */
void transmistDataOverUART1(char *dataToTransmit) {
    while (isDataSent != 1) {
        // wait untill previous transmission is ongoing
    }
    HAL_UART_Transmit_DMA(&huart1, (uint8_t*) dataToTransmit,
            strlen(dataToTransmit));
    isDataSent = 0;

}
void sendLightSenorData(void) {
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    unsigned int adc_value = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    HAL_Delay(10);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adc_value += HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    HAL_Delay(10);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adc_value += HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
```

```
    adc_value = (adc_value / 3);
    adcValue = adc_value;
    if (adcValue > 2000) {
        HAL_GPIO_WritePin(RELAY_3_GPIO_Port, RELAY_3_Pin, GPIO_PIN_SET);
    } else {
        HAL_GPIO_WritePin(RELAY_3_GPIO_Port, RELAY_3_Pin, GPIO_PIN_RESET);
    }

    char buffer[70];
    sprintf(buffer, "publish:esp01_ariba/sensors/lightlevel|{\"lux\":\"%d\"}\n",
            adc_value);
//  HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer),
//          HAL_MAX_DELAY);
    transmistDataOverUART1(buffer);
    HAL_Delay(10);
}
void turnOnRelay(char *topic, char *payload) {
    if (strstr(topic, "switch1") && strcmp(payload, "on")) {
        HAL_GPIO_WritePin(RELAY_1_GPIO_Port, RELAY_1_Pin, GPIO_PIN_RESET);
        char buffer[70];
        strcpy(buffer, "publish:esp01_ariba/switch1|{\"state\":\"on\"}\n");
        transmistDataOverUART1(buffer);
        HAL_Delay(10);
    } else if (strstr(topic, "switch1"), strcmp(payload, "off")) {
        HAL_GPIO_WritePin(RELAY_1_GPIO_Port, RELAY_1_Pin, GPIO_PIN_SET);
        char buffer[70];
        strcpy(buffer, "publish:esp01_ariba/switch1|{\"state\":\"off\"}\n");
        transmistDataOverUART1(buffer);
        HAL_Delay(10);
    }
}
/* 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 */
```

Smart Bell

ESP01

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>

#define readyPin 2
// #define WDT_TIMEOUT 20000

// WiFi credentials
const char *ssid = "";    // !! Enter WiFi name here !!
const char *password = ""; // !! Enter WiFi password here !!

// MQTT Broker settings
const char *mqtt_server = "emqx.home.mwaleedh.com.pk";
const int mqtt_port = 8883;
const char *mqtt_user = ""; // !! Provided server username here !!
const char *mqtt_password = ""; //!! Provided server password here !!
const char *mqtt_client_id = "esp01_rameeza";

// function prototypes here
```

```
void deviceDiscoveryHA_BellButton();
void deviceDiscoveryHA_DoorSensor();
void deviceDiscoveryHA_DoorSwitch();

void reconnect();
void nonBlockingDelay(unsigned long interval);

// pre defined objects here

WiFiClientSecure espClient;
PubSubClient client(espClient);

void callback(char *topic, byte *payload, unsigned int length) {
  Serial.flush();
  Serial.print("topic:");
  Serial.print(topic);
  // String recv_payload = String((char *)payload);
  Serial.print("|payload:");
  char *recv_payload = (char *)malloc(length);
  for (int i = 0; i < length; i++) {
    recv_payload[i] = (char)payload[i];
    Serial.print((char)payload[i]);
  }
  Serial.print("\n");
  // strcat(recv_payload,"@");
  // Serial.print(recv_payload);
  // Serial.print("\n")
  // Serial.println(length);

  // Parse JSON payload
  // JsonDocument doc;
  // DeserializationError error = deserializeJson(doc, payload, length);
  // if (error) {
  //   String recv_payload = String((char *)payload);
  //   Serial.print("|payload:");
  //   Serial.println(recv_payload);
  // }
  // digitalWrite(LED, LOW);

  // Extract values from JSON and update variables
  // switchState = doc["switch"];
  // temperature = doc["temperature"];

  // Serial.print("Switch state: ");
  // Serial.println(switchState);
```

```
  // Serial.print("Temperature: ");
  // Serial.println(temperature);
  delay(1000);
}

unsigned long previousMillis = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("Turning On...");
  delay(5000);

  WiFi.begin(ssid, password);
  Serial.println("Wifi Function called");
  while (WiFi.status() != WL_CONNECTED) {
    nonBlockingDelay(50000);
    Serial.println("Connecting to WiFi...");
    // delay(5000);
  }
  Serial.println("WiFi connected");

  // if (root_ca != NULL) {
  //   espClient.setCACert(root_ca);
  // } else {
  espClient.setInsecure();
  //}
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  client.setBufferSize(512); // increasing buffer size

  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("MQTT connected");
      deviceDiscoveryHA_DoorSensor();
      deviceDiscoveryHA_DoorSwitch();
      deviceDiscoveryHA_BellButton();

      digitalWrite(
          readyPin,
          HIGH); // GPIO2 goes high when server connection is established

    } else {
      digitalWrite(readyPin, LOW); // GPIO2 goes low when server is disconnected
      Serial.print("Failed, rc=");
```

```
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
}

void loop() {
  // ESP.wdtFeed();
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  //  Read switch state and temperature from serial
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    if (input.startsWith("publish:")) {
      int separatorIndex = input.indexOf('|');
      if (separatorIndex != -1) {
        String topic = input.substring(8, separatorIndex);
        String payload = input.substring(separatorIndex + 1);
        JsonDocument doc;
        DeserializationError error = deserializeJson(doc, payload);
        if (!error) { // error checking json format
          client.publish(topic.c_str(), payload.c_str());

          // uncomment this for debugging

          // Serial.println("Published message:");
          // Serial.println(payload);
        }
      }
    } else if (input.startsWith("subscribe:")) {
      String topic = input.substring(10);
      client.subscribe(topic.c_str());
      Serial.print("Subscribed to topic: ");
      Serial.println(topic);
    } else if (input.startsWith("unsubscribe:")) {
      String topic = input.substring(12);
      client.unsubscribe(topic.c_str());
      Serial.print("Unsubscribed from topic: ");
      Serial.println(topic);
    }
  }
```

```
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");
    if (client.connect(mqtt_client_id, mqtt_user, mqtt_password)) {
      Serial.println("MQTT connected");
      digitalWrite(
          readyPin,
          HIGH); // GPIO2 goes high when server connection is established
    } else {
      digitalWrite(readyPin, LOW); // GPIO2 goes low when server is disconnected
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retry in 5 seconds");
      delay(5000);
    }
  }
}

void nonBlockingDelay(unsigned long interval) {
  static unsigned long previousMillis = 0;
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    // Perform action here...
  }
}

void deviceDiscoveryHA_DoorSensor() {
  char topic[128];
  char buffer[512];
  char uid[128];
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/binary_sensor/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_BS/config");

  // creating payload for Window Sensor
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_BS");
```

```
  doc["name"] = "Dooor Sensor";
  doc["obj_id"] = "mqtt_door_sensor";
  doc["uniq_id"] = uid;
  doc["stat_t"] = "esp01_rameeza/sensors/door_sensor";
  doc["value_template"] = "{{value_json.state}}";
  doc["payload_on"] = "open";
  doc["payload_off"] = "close";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Door Bell";
  device["mf"] = "Rameeza";
  device["mdl"] = "ESP01";
  device["sw"] = "1.0";
  device["hw"] = "1.0";
  // device["cu"] = "http://192.168.1.226/config";  //web interface for device,
  // with discovery toggle
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
}

void deviceDiscoveryHA_DoorSwitch() {
  char topic[128];
  char buffer[512];
  char uid[128];
  String subTopic;
  JsonDocument doc;

  // SWITCH 1 HERE
  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/switch/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_sw1/config"); // TODO: change this for further switches

  // creating payload for switch1
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_DS");            // TODO: change this for further switches
  doc["name"] = "Door Switch";   // TODO: change this for further switches
  doc["obj_id"] = "door_switch"; // TODO: change this for further switches
  doc["uniq_id"] = uid;
  doc["state_topic"] =
      "esp01_rameeza/door_switch"; // TODO: change this for further switches
  doc["command_topic"] =
```

```
      "esp01_rameeza/door_switch/set"; // TODO: change this for further switches
  subTopic =
      "esp01_rameeza/door_switch/set"; // TODO: change this for further switches

  doc["value_template"] = "{{value_json.state}}";
  doc["state_template"] = "{{value_json.state}}";
  doc["payload_on"] = "on";
  doc["payload_off"] = "off";
  doc["state_on"] = "on";
  doc["state_off"] = "off";
  doc["optimistic"] = "false";

  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Door Bell";
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
  client.subscribe(subTopic.c_str());
}

void deviceDiscoveryHA_BellButton() {

  char topic[128];
  char buffer[512];
  char uid[128];
  JsonDocument doc;

  doc.clear();
  // creating topic here
  strcpy(topic, "homeassistant/binary_sensor/");
  strcat(topic, mqtt_client_id);
  strcat(topic, "_BTN/config");

  // creating payload for Window Sensor
  strcpy(uid, mqtt_client_id);
  strcat(uid, "_BTN");
  doc["name"] = "Bell Button";
  doc["obj_id"] = "mqtt_bell_button";
  doc["uniq_id"] = uid;
  doc["stat_t"] = "esp01_rameeza/buttons/bell_button/state";
  doc["value_template"] = "{{value_json.state}}";
  doc["payload_on"] = "pressed";
  doc["payload_off"] = "de_pressed";
```

```
  JsonObject device = doc.createNestedObject("device");
  device["ids"] = mqtt_client_id;
  device["name"] = "Door Bell";
  // device["cu"] = "http://192.168.1.226/config";  //web interface for device,
  // with discovery toggle
  serializeJson(doc, buffer);
  // Publish discovery topic and payload (with retained flag)
  client.publish(topic, buffer, true);
}
```

STM32-Rameeza

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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.
  *
  ******************************************************************************
  */
/* 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 */
```

```c
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;
DMA_HandleTypeDef hdma_usart1_tx;
DMA_HandleTypeDef hdma_usart1_rx;

/* USER CODE BEGIN PV */
char data[100];
int doorStatus;
int isDataSent=1;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */
void transmistDataOverUART1(char *dataToTransmit);
void setDoorSensorPinAsInput(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
  isDataSent=1;
}
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
uint8_t rxByte;
char rxBuffer[100]; // Buffer to store received data
uint16_t rxIndex = 0;  // Index for the received data buffer
char topic[30];
char payload[10];
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
  if ((char) rxByte == '\n') { //  if end of transmission then start processing
    rxBuffer[rxIndex] = '\0';
    rxIndex = 0;
    char switch1[] = "switch";
```

```
    char payloadOn[] = "on";
    char payloadOff[] = "off";
    //transmistDataOverUART1((char*) rxBuffer);
    // processing incoming data here
    // Separate the topic and payload
    const char delim[] = ":|";

    // Tokenize the string to get the first token
    char *token = strtok(rxBuffer, delim);

    // Check if the first token is "topic"
    if (token != NULL && strcmp(token, "topic") == 0) {
      // Get the topic
      char *topic = strtok(NULL, delim);
      // Get the payload
      char *payload = strtok(NULL, delim);

      // Print the results
      if (topic != NULL && payload != NULL) {
        if (strstr(topic, switch1)) {
          turnOnRelay(topic, payload);
        }

      } else {
        printf("Failed to parse the string.\n");
      }
    } else {
      printf("Invalid format.\n");
    }

  } else {
    rxBuffer[rxIndex++] = (char) rxByte;

    // Prevent buffer overflow
    if (rxIndex >= sizeof(rxBuffer)) {
      rxIndex = 0; // Reset index if buffer overflows
    }
  }

  HAL_UART_Receive_DMA(&huart1, &rxByte, 1);

}
/* USER CODE END 0 */

/**
```

```c
  * @brief  The application entry point.
  * @retval int
  */
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_DMA_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  //void setDoorSensorPinAsInput(void);

  if (HAL_GPIO_ReadPin(DOOR_SENSOR_GPIO_Port, DOOR_SENSOR_Pin)== GPIO_PIN_SET) {
    doorStatus = 1;
      char buff[]
="publish:esp01_rameeza/sensors/door_sensor|{\"state\":\"close\"}\n\0";
      HAL_UART_Transmit(&huart1, (uint8_t*) buff, strlen(buff),
      HAL_MAX_DELAY);

  } else {
      doorStatus = 0;
      char buff[] =
"publish:esp01_rameeza/sensors/door_sensor|{\"state\":\"open\"}\n\0";
```

```c
      HAL_UART_Transmit(&huart1, (uint8_t*) buff, strlen(buff),
      HAL_MAX_DELAY);
  }

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* 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_SCALE1);

  /** 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 = 8;
  RCC_OscInitStruct.PLL.PLLN = 100;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  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_3) != 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 = 115200;
  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();
  }
```

```c
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA2_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA2_Stream2_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);
  /* DMA2_Stream7_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA2_Stream7_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream7_IRQn);

}

/**
  * @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_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(DOOR_LOCK_GPIO_Port, DOOR_LOCK_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : DOOR_LOCK_Pin */
  GPIO_InitStruct.Pin = DOOR_LOCK_Pin;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(DOOR_LOCK_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : DOOR_SENSOR_Pin */
  GPIO_InitStruct.Pin = DOOR_SENSOR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DOOR_SENSOR_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : BELL_BUTTON_Pin */
  GPIO_InitStruct.Pin = BELL_BUTTON_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BELL_BUTTON_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

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

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
  if (GPIO_Pin == DOOR_SENSOR_Pin) {
    if (HAL_GPIO_ReadPin(DOOR_SENSOR_GPIO_Port, DOOR_SENSOR_Pin) == GPIO_PIN_SET)
{
      strcpy(data,
"publish:esp01_rameeza/sensors/door_sensor|{\"state\":\"close\"}\n\0");
//      HAL_UART_Transmit(&huart1, (uint8_t*) data, strlen(data),
//      HAL_MAX_DELAY);

      transmistDataOverUART1(data);
      doorStatus = 1;
      return;

    } else {

      strcpy(data,
          "publish:esp01_rameeza/sensors/door_sensor|{\"state\":\"open\"}\n\0");

      transmistDataOverUART1(data);
```

```
      doorStatus = 0;
      return;

    }
  }
  else if (GPIO_Pin == BELL_BUTTON_Pin) {
        if (HAL_GPIO_ReadPin(BELL_BUTTON_GPIO_Port, BELL_BUTTON_Pin) ==
GPIO_PIN_SET) {
            strcpy(data,
"publish:esp01_rameeza/sensors/bell_button|{\"state\":\"pressed\"}\n\0");
            transmistDataOverUART1(data);
        }

        HAL_Delay(100);

        if (HAL_GPIO_ReadPin(BELL_BUTTON_GPIO_Port, BELL_BUTTON_Pin) ==
GPIO_PIN_RESET) {
            strcpy(data,
"publish:esp01_rameeza/sensors/bell_button|{\"state\":\"de_pressed\"}\n\0");
            transmistDataOverUART1(data);
        }
    }
}
void transmistDataOverUART1(char *dataToTransmit) {
  while (isDataSent != 1) {
    // wait until previous transmission is ongoing
  }
  HAL_UART_Transmit_DMA(&huart1, (uint8_t*)
dataToTransmit,strlen(dataToTransmit));
  isDataSent=0;

}

void setDoorSensorPinAsInput(void) {
  GPIO_InitTypeDef GPIO_InitStruct = { 0 };
  /*Configure GPIO pin : DOOR_SENSOR_Pin */
  GPIO_InitStruct.Pin = DOOR_SENSOR_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DOOR_SENSOR_GPIO_Port, &GPIO_InitStruct);
}

/* 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 */
```

H. References [Negative Marking of 20% if this section is skipped]

Resources and links used for guidance and documentation as references:

- https://github.com/henriheimann/stm32-hal-sht3x
- https://www.geeksforgeeks.org/strlen-function-in-c/
- https://controllerstech.com/stm32-uart-2-use-interrupt-dma-to-transmit-data/
- https://www.youtube.com/watch?v=JaMwNT0m3Sw
- https://stackoverflow.com/questions/20458489/include-double-quote-in-c-string
- https://github.com/davolesh/mqtt_pushbutton/blob/master/mqtt_button_basement.ino
- https://www.youtube.com/watch?v=--XLR3WlTqo

- https://forum.arduino.cc/t/serial-communications-end-char/42484/3
- https://www.geeksforgeeks.org/cpp-malloc/
- https://www.arduino.cc/reference/tr/language/functions/communication/serial/flush/
- https://www.home-assistant.io/integrations/switch.mqtt/
- https://arduino.stackexchange.com/questions/55024/how-to-convert-byte-payload-to-string
- https://www.w3schools.com/cpp/cpp_strings_concat.asp
- https://resinchemtech.blogspot.com/2023/12/mqtt-auto-discovery.html
- https://resinchemtech.blogspot.com/2023/12/mqtt-auto-discovery.html
- https://community.home-assistant.io/t/mqtt-button-using-esp8266/45/15
- https://www.home-assistant.io/docs/configuration/templating/#using-templates-with-the-mqtt-integration
- https://www.home-assistant.io
- https://www.emqx.com
- https://digilog.pk
- https://mqtt.org