/\* 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"

**#include** "sht3x.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**);

**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();

}

/\*\* 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 \*/

}

/\*\*

\* @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*);

/\* 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\":\"close\"}\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 \*/