**Low-cost tracking solution using STM32L475E-IOTA1**

* LED-Button interface.

//Controlling LED using button on STM32 board.

STM32 IDE Code:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* 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 ---------------------------------------------------------\*/

DFSDM\_Channel\_HandleTypeDef hdfsdm1\_channel1;

I2C\_HandleTypeDef hi2c2;

QSPI\_HandleTypeDef hqspi;

SPI\_HandleTypeDef hspi3;

UART\_HandleTypeDef huart1;

UART\_HandleTypeDef huart3;

PCD\_HandleTypeDef hpcd\_USB\_OTG\_FS;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_DFSDM1\_Init**(**void**);

**static** **void** **MX\_I2C2\_Init**(**void**);

**static** **void** **MX\_QUADSPI\_Init**(**void**);

**static** **void** **MX\_SPI3\_Init**(**void**);

**static** **void** **MX\_USART1\_UART\_Init**(**void**);

**static** **void** **MX\_USART3\_UART\_Init**(**void**);

**static** **void** **MX\_USB\_OTG\_FS\_PCD\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* 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\_DFSDM1\_Init();

MX\_I2C2\_Init();

MX\_QUADSPI\_Init();

MX\_SPI3\_Init();

MX\_USART1\_UART\_Init();

MX\_USART3\_UART\_Init();

MX\_USB\_OTG\_FS\_PCD\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_GPIO\_WritePin(LED\_GPIO\_Port,LED\_Pin,*GPIO\_PIN\_RESET*);

**if**(HAL\_GPIO\_ReadPin(BUTTON\_GPIO\_Port,BUTTON\_Pin)==1)

{

HAL\_GPIO\_WritePin(LED\_GPIO\_Port,LED\_Pin,*GPIO\_PIN\_SET*);

}

**else**

{

HAL\_GPIO\_WritePin(LED\_GPIO\_Port,LED\_Pin,*GPIO\_PIN\_RESET*);

}

}

/\* 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

\*/

**if** (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure LSE Drive Capability

\*/

HAL\_PWR\_EnableBkUpAccess();

\_\_HAL\_RCC\_LSEDRIVE\_CONFIG(RCC\_LSEDRIVE\_LOW);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_LSE|RCC\_OSCILLATORTYPE\_MSI;

RCC\_OscInitStruct.LSEState = RCC\_LSE\_ON;

RCC\_OscInitStruct.MSIState = RCC\_MSI\_ON;

RCC\_OscInitStruct.MSICalibrationValue = 0;

RCC\_OscInitStruct.MSIClockRange = RCC\_MSIRANGE\_6;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_MSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 40;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV7;

RCC\_OscInitStruct.PLL.PLLQ = RCC\_PLLQ\_DIV2;

RCC\_OscInitStruct.PLL.PLLR = RCC\_PLLR\_DIV2;

**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\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_4) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Enable MSI Auto calibration

\*/

HAL\_RCCEx\_EnableMSIPLLMode();

}

/\*\*

\* @brief DFSDM1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_DFSDM1\_Init**(**void**)

{

/\* USER CODE BEGIN DFSDM1\_Init 0 \*/

/\* USER CODE END DFSDM1\_Init 0 \*/

/\* USER CODE BEGIN DFSDM1\_Init 1 \*/

/\* USER CODE END DFSDM1\_Init 1 \*/

hdfsdm1\_channel1.Instance = DFSDM1\_Channel1;

hdfsdm1\_channel1.Init.OutputClock.Activation = *ENABLE*;

hdfsdm1\_channel1.Init.OutputClock.Selection = DFSDM\_CHANNEL\_OUTPUT\_CLOCK\_SYSTEM;

hdfsdm1\_channel1.Init.OutputClock.Divider = 2;

hdfsdm1\_channel1.Init.Input.Multiplexer = DFSDM\_CHANNEL\_EXTERNAL\_INPUTS;

hdfsdm1\_channel1.Init.Input.DataPacking = DFSDM\_CHANNEL\_STANDARD\_MODE;

hdfsdm1\_channel1.Init.Input.Pins = DFSDM\_CHANNEL\_FOLLOWING\_CHANNEL\_PINS;

hdfsdm1\_channel1.Init.SerialInterface.Type = DFSDM\_CHANNEL\_SPI\_RISING;

hdfsdm1\_channel1.Init.SerialInterface.SpiClock = DFSDM\_CHANNEL\_SPI\_CLOCK\_INTERNAL;

hdfsdm1\_channel1.Init.Awd.FilterOrder = DFSDM\_CHANNEL\_FASTSINC\_ORDER;

hdfsdm1\_channel1.Init.Awd.Oversampling = 1;

hdfsdm1\_channel1.Init.Offset = 0;

hdfsdm1\_channel1.Init.RightBitShift = 0x00;

**if** (HAL\_DFSDM\_ChannelInit(&hdfsdm1\_channel1) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN DFSDM1\_Init 2 \*/

/\* USER CODE END DFSDM1\_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.Timing = 0x00000E14;

hi2c2.Init.OwnAddress1 = 0;

hi2c2.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c2.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c2.Init.OwnAddress2 = 0;

hi2c2.Init.OwnAddress2Masks = I2C\_OA2\_NOMASK;

hi2c2.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c2.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

**if** (HAL\_I2C\_Init(&hi2c2) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure Analogue filter

\*/

**if** (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c2, I2C\_ANALOGFILTER\_ENABLE) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Configure Digital filter

\*/

**if** (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c2, 0) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C2\_Init 2 \*/

/\* USER CODE END I2C2\_Init 2 \*/

}

/\*\*

\* @brief QUADSPI Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_QUADSPI\_Init**(**void**)

{

/\* USER CODE BEGIN QUADSPI\_Init 0 \*/

/\* USER CODE END QUADSPI\_Init 0 \*/

/\* USER CODE BEGIN QUADSPI\_Init 1 \*/

/\* USER CODE END QUADSPI\_Init 1 \*/

/\* QUADSPI parameter configuration\*/

hqspi.Instance = QUADSPI;

hqspi.Init.ClockPrescaler = 2;

hqspi.Init.FifoThreshold = 4;

hqspi.Init.SampleShifting = QSPI\_SAMPLE\_SHIFTING\_HALFCYCLE;

hqspi.Init.FlashSize = 23;

hqspi.Init.ChipSelectHighTime = QSPI\_CS\_HIGH\_TIME\_1\_CYCLE;

hqspi.Init.ClockMode = QSPI\_CLOCK\_MODE\_0;

**if** (HAL\_QSPI\_Init(&hqspi) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN QUADSPI\_Init 2 \*/

/\* USER CODE END QUADSPI\_Init 2 \*/

}

/\*\*

\* @brief SPI3 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_SPI3\_Init**(**void**)

{

/\* USER CODE BEGIN SPI3\_Init 0 \*/

/\* USER CODE END SPI3\_Init 0 \*/

/\* USER CODE BEGIN SPI3\_Init 1 \*/

/\* USER CODE END SPI3\_Init 1 \*/

/\* SPI3 parameter configuration\*/

hspi3.Instance = SPI3;

hspi3.Init.Mode = SPI\_MODE\_MASTER;

hspi3.Init.Direction = SPI\_DIRECTION\_2LINES;

hspi3.Init.DataSize = SPI\_DATASIZE\_4BIT;

hspi3.Init.CLKPolarity = SPI\_POLARITY\_LOW;

hspi3.Init.CLKPhase = SPI\_PHASE\_1EDGE;

hspi3.Init.NSS = SPI\_NSS\_SOFT;

hspi3.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_2;

hspi3.Init.FirstBit = SPI\_FIRSTBIT\_MSB;

hspi3.Init.TIMode = SPI\_TIMODE\_DISABLE;

hspi3.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;

hspi3.Init.CRCPolynomial = 7;

hspi3.Init.CRCLength = SPI\_CRC\_LENGTH\_DATASIZE;

hspi3.Init.NSSPMode = SPI\_NSS\_PULSE\_ENABLE;

**if** (HAL\_SPI\_Init(&hspi3) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN SPI3\_Init 2 \*/

/\* USER CODE END SPI3\_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;

huart1.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart1.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

**if** (HAL\_UART\_Init(&huart1) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\*\*

\* @brief USART3 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USART3\_UART\_Init**(**void**)

{

/\* USER CODE BEGIN USART3\_Init 0 \*/

/\* USER CODE END USART3\_Init 0 \*/

/\* USER CODE BEGIN USART3\_Init 1 \*/

/\* USER CODE END USART3\_Init 1 \*/

huart3.Instance = USART3;

huart3.Init.BaudRate = 115200;

huart3.Init.WordLength = UART\_WORDLENGTH\_8B;

huart3.Init.StopBits = UART\_STOPBITS\_1;

huart3.Init.Parity = UART\_PARITY\_NONE;

huart3.Init.Mode = UART\_MODE\_TX\_RX;

huart3.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart3.Init.OverSampling = UART\_OVERSAMPLING\_16;

huart3.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart3.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

**if** (HAL\_UART\_Init(&huart3) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART3\_Init 2 \*/

/\* USER CODE END USART3\_Init 2 \*/

}

/\*\*

\* @brief USB\_OTG\_FS Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_USB\_OTG\_FS\_PCD\_Init**(**void**)

{

/\* USER CODE BEGIN USB\_OTG\_FS\_Init 0 \*/

/\* USER CODE END USB\_OTG\_FS\_Init 0 \*/

/\* USER CODE BEGIN USB\_OTG\_FS\_Init 1 \*/

/\* USER CODE END USB\_OTG\_FS\_Init 1 \*/

hpcd\_USB\_OTG\_FS.Instance = USB\_OTG\_FS;

hpcd\_USB\_OTG\_FS.Init.dev\_endpoints = 6;

hpcd\_USB\_OTG\_FS.Init.speed = PCD\_SPEED\_FULL;

hpcd\_USB\_OTG\_FS.Init.phy\_itface = PCD\_PHY\_EMBEDDED;

hpcd\_USB\_OTG\_FS.Init.Sof\_enable = *DISABLE*;

hpcd\_USB\_OTG\_FS.Init.low\_power\_enable = *DISABLE*;

hpcd\_USB\_OTG\_FS.Init.lpm\_enable = *DISABLE*;

hpcd\_USB\_OTG\_FS.Init.battery\_charging\_enable = *DISABLE*;

hpcd\_USB\_OTG\_FS.Init.use\_dedicated\_ep1 = *DISABLE*;

hpcd\_USB\_OTG\_FS.Init.vbus\_sensing\_enable = *DISABLE*;

**if** (HAL\_PCD\_Init(&hpcd\_USB\_OTG\_FS) != *HAL\_OK*)

{

Error\_Handler();

}

/\* USER CODE BEGIN USB\_OTG\_FS\_Init 2 \*/

/\* USER CODE END USB\_OTG\_FS\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOE, M24SR64\_Y\_RF\_DISABLE\_Pin|M24SR64\_Y\_GPO\_Pin|ISM43362\_RST\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, ARD\_D10\_Pin|SPBTLE\_RF\_RST\_Pin|ARD\_D9\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB, ARD\_D8\_Pin|ISM43362\_BOOT0\_Pin|ISM43362\_WAKEUP\_Pin|LED\_Pin

|SPSGRF\_915\_SDN\_Pin|ARD\_D5\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, USB\_OTG\_FS\_PWR\_EN\_Pin|PMOD\_RESET\_Pin|STSAFE\_A100\_RESET\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(SPBTLE\_RF\_SPI3\_CSN\_GPIO\_Port, SPBTLE\_RF\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, VL53L0X\_XSHUT\_Pin|LED3\_WIFI\_\_LED4\_BLE\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(SPSGRF\_915\_SPI3\_CSN\_GPIO\_Port, SPSGRF\_915\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(ISM43362\_SPI3\_CSN\_GPIO\_Port, ISM43362\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pins : M24SR64\_Y\_RF\_DISABLE\_Pin M24SR64\_Y\_GPO\_Pin ISM43362\_RST\_Pin ISM43362\_SPI3\_CSN\_Pin \*/

GPIO\_InitStruct.Pin = M24SR64\_Y\_RF\_DISABLE\_Pin|M24SR64\_Y\_GPO\_Pin|ISM43362\_RST\_Pin|ISM43362\_SPI3\_CSN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : USB\_OTG\_FS\_OVRCR\_EXTI3\_Pin SPSGRF\_915\_GPIO3\_EXTI5\_Pin SPBTLE\_RF\_IRQ\_EXTI6\_Pin ISM43362\_DRDY\_EXTI1\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_OVRCR\_EXTI3\_Pin|SPSGRF\_915\_GPIO3\_EXTI5\_Pin|SPBTLE\_RF\_IRQ\_EXTI6\_Pin|ISM43362\_DRDY\_EXTI1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : BUTTON\_Pin VL53L0X\_GPIO1\_EXTI7\_Pin LSM3MDL\_DRDY\_EXTI8\_Pin \*/

GPIO\_InitStruct.Pin = BUTTON\_Pin|VL53L0X\_GPIO1\_EXTI7\_Pin|LSM3MDL\_DRDY\_EXTI8\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_A5\_Pin ARD\_A4\_Pin ARD\_A3\_Pin ARD\_A2\_Pin

ARD\_A1\_Pin ARD\_A0\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_A5\_Pin|ARD\_A4\_Pin|ARD\_A3\_Pin|ARD\_A2\_Pin

|ARD\_A1\_Pin|ARD\_A0\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D1\_Pin ARD\_D0\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D1\_Pin|ARD\_D0\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF8\_UART4;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D10\_Pin SPBTLE\_RF\_RST\_Pin ARD\_D9\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D10\_Pin|SPBTLE\_RF\_RST\_Pin|ARD\_D9\_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 : ARD\_D4\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D4\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF1\_TIM2;

HAL\_GPIO\_Init(ARD\_D4\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D7\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D7\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D7\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D13\_Pin ARD\_D12\_Pin ARD\_D11\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D13\_Pin|ARD\_D12\_Pin|ARD\_D11\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI1;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D3\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D3\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D3\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D6\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D6\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D6\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D8\_Pin ISM43362\_BOOT0\_Pin ISM43362\_WAKEUP\_Pin LED\_Pin

SPSGRF\_915\_SDN\_Pin ARD\_D5\_Pin SPSGRF\_915\_SPI3\_CSN\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D8\_Pin|ISM43362\_BOOT0\_Pin|ISM43362\_WAKEUP\_Pin|LED\_Pin

|SPSGRF\_915\_SDN\_Pin|ARD\_D5\_Pin|SPSGRF\_915\_SPI3\_CSN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : LPS22HB\_INT\_DRDY\_EXTI0\_Pin LSM6DSL\_INT1\_EXTI11\_Pin ARD\_D2\_Pin HTS221\_DRDY\_EXTI15\_Pin

PMOD\_IRQ\_EXTI12\_Pin \*/

GPIO\_InitStruct.Pin = LPS22HB\_INT\_DRDY\_EXTI0\_Pin|LSM6DSL\_INT1\_EXTI11\_Pin|ARD\_D2\_Pin|HTS221\_DRDY\_EXTI15\_Pin

|PMOD\_IRQ\_EXTI12\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : USB\_OTG\_FS\_PWR\_EN\_Pin SPBTLE\_RF\_SPI3\_CSN\_Pin PMOD\_RESET\_Pin STSAFE\_A100\_RESET\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_PWR\_EN\_Pin|SPBTLE\_RF\_SPI3\_CSN\_Pin|PMOD\_RESET\_Pin|STSAFE\_A100\_RESET\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : VL53L0X\_XSHUT\_Pin LED3\_WIFI\_\_LED4\_BLE\_Pin \*/

GPIO\_InitStruct.Pin = VL53L0X\_XSHUT\_Pin|LED3\_WIFI\_\_LED4\_BLE\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : PMOD\_SPI2\_SCK\_Pin \*/

GPIO\_InitStruct.Pin = PMOD\_SPI2\_SCK\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(PMOD\_SPI2\_SCK\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : PMOD\_UART2\_CTS\_Pin PMOD\_UART2\_RTS\_Pin PMOD\_UART2\_TX\_Pin PMOD\_UART2\_RX\_Pin \*/

GPIO\_InitStruct.Pin = PMOD\_UART2\_CTS\_Pin|PMOD\_UART2\_RTS\_Pin|PMOD\_UART2\_TX\_Pin|PMOD\_UART2\_RX\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF7\_USART2;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D15\_Pin ARD\_D14\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D15\_Pin|ARD\_D14\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_OD;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF4\_I2C1;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(*EXTI9\_5\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*EXTI9\_5\_IRQn*);

HAL\_NVIC\_SetPriority(*EXTI15\_10\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*EXTI15\_10\_IRQn*);

}

/\* USER CODE BEGIN 4 \*/

/\* 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 \*/

* Print data into SWV ITM data console.

Code:

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @author : Auto-generated by STM32CubeIDE

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#include** <stdint.h>

**#include** <stdio.h>

/\*#if !defined(\_\_SOFT\_FP\_\_) && defined(\_\_ARM\_FP)

#warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."

#endif\*/

**int** **main**(**void**)

{

**int** i=5;

**while**(i<10)

{

**printf**("Hello \n");

i++;

}

}

* Getting temperature and pressure data on mobile phone using STM32IDE.

Code:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

**#include** "app\_bluenrg\_ms.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* 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 ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* 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\_BlueNRG\_MS\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

MX\_BlueNRG\_MS\_Process();

/\* 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

\*/

**if** (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_MSI;

RCC\_OscInitStruct.MSIState = RCC\_MSI\_ON;

RCC\_OscInitStruct.MSICalibrationValue = 0;

RCC\_OscInitStruct.MSIClockRange = RCC\_MSIRANGE\_6;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_MSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 40;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV7;

RCC\_OscInitStruct.PLL.PLLQ = RCC\_PLLQ\_DIV2;

RCC\_OscInitStruct.PLL.PLLR = RCC\_PLLR\_DIV2;

**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\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_4) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOE, M24SR64\_Y\_RF\_DISABLE\_Pin|M24SR64\_Y\_GPO\_Pin|ISM43362\_RST\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, ARD\_D10\_Pin|SPBTLE\_RF\_RST\_Pin|ARD\_D9\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB, ARD\_D8\_Pin|ISM43362\_BOOT0\_Pin|ISM43362\_WAKEUP\_Pin|SPSGRF\_915\_SDN\_Pin

|ARD\_D5\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, USB\_OTG\_FS\_PWR\_EN\_Pin|PMOD\_RESET\_Pin|STSAFE\_A100\_RESET\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(SPBTLE\_RF\_SPI3\_CSN\_GPIO\_Port, SPBTLE\_RF\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, VL53L0X\_XSHUT\_Pin|LED3\_WIFI\_\_LED4\_BLE\_Pin, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(SPSGRF\_915\_SPI3\_CSN\_GPIO\_Port, SPSGRF\_915\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(ISM43362\_SPI3\_CSN\_GPIO\_Port, ISM43362\_SPI3\_CSN\_Pin, *GPIO\_PIN\_SET*);

/\*Configure GPIO pins : M24SR64\_Y\_RF\_DISABLE\_Pin M24SR64\_Y\_GPO\_Pin ISM43362\_RST\_Pin ISM43362\_SPI3\_CSN\_Pin \*/

GPIO\_InitStruct.Pin = M24SR64\_Y\_RF\_DISABLE\_Pin|M24SR64\_Y\_GPO\_Pin|ISM43362\_RST\_Pin|ISM43362\_SPI3\_CSN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : USB\_OTG\_FS\_OVRCR\_EXTI3\_Pin SPSGRF\_915\_GPIO3\_EXTI5\_Pin SPBTLE\_RF\_IRQ\_EXTI6\_Pin ISM43362\_DRDY\_EXTI1\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_OVRCR\_EXTI3\_Pin|SPSGRF\_915\_GPIO3\_EXTI5\_Pin|SPBTLE\_RF\_IRQ\_EXTI6\_Pin|ISM43362\_DRDY\_EXTI1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_A5\_Pin ARD\_A4\_Pin ARD\_A3\_Pin ARD\_A2\_Pin

ARD\_A1\_Pin ARD\_A0\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_A5\_Pin|ARD\_A4\_Pin|ARD\_A3\_Pin|ARD\_A2\_Pin

|ARD\_A1\_Pin|ARD\_A0\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D1\_Pin ARD\_D0\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D1\_Pin|ARD\_D0\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF8\_UART4;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D10\_Pin SPBTLE\_RF\_RST\_Pin ARD\_D9\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D10\_Pin|SPBTLE\_RF\_RST\_Pin|ARD\_D9\_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 : ARD\_D4\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D4\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF1\_TIM2;

HAL\_GPIO\_Init(ARD\_D4\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D7\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D7\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D7\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D13\_Pin ARD\_D12\_Pin ARD\_D11\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D13\_Pin|ARD\_D12\_Pin|ARD\_D11\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI1;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D3\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D3\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D3\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : ARD\_D6\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D6\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG\_ADC\_CONTROL;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(ARD\_D6\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D8\_Pin ISM43362\_BOOT0\_Pin ISM43362\_WAKEUP\_Pin SPSGRF\_915\_SDN\_Pin

ARD\_D5\_Pin SPSGRF\_915\_SPI3\_CSN\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D8\_Pin|ISM43362\_BOOT0\_Pin|ISM43362\_WAKEUP\_Pin|SPSGRF\_915\_SDN\_Pin

|ARD\_D5\_Pin|SPSGRF\_915\_SPI3\_CSN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : DFSDM1\_DATIN2\_Pin DFSDM1\_CKOUT\_Pin \*/

GPIO\_InitStruct.Pin = DFSDM1\_DATIN2\_Pin|DFSDM1\_CKOUT\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

GPIO\_InitStruct.Alternate = GPIO\_AF6\_DFSDM1;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : QUADSPI\_CLK\_Pin QUADSPI\_NCS\_Pin OQUADSPI\_BK1\_IO0\_Pin QUADSPI\_BK1\_IO1\_Pin

QUAD\_SPI\_BK1\_IO2\_Pin QUAD\_SPI\_BK1\_IO3\_Pin \*/

GPIO\_InitStruct.Pin = QUADSPI\_CLK\_Pin|QUADSPI\_NCS\_Pin|OQUADSPI\_BK1\_IO0\_Pin|QUADSPI\_BK1\_IO1\_Pin

|QUAD\_SPI\_BK1\_IO2\_Pin|QUAD\_SPI\_BK1\_IO3\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF10\_QUADSPI;

HAL\_GPIO\_Init(GPIOE, &GPIO\_InitStruct);

/\*Configure GPIO pins : INTERNAL\_I2C2\_SCL\_Pin INTERNAL\_I2C2\_SDA\_Pin \*/

GPIO\_InitStruct.Pin = INTERNAL\_I2C2\_SCL\_Pin|INTERNAL\_I2C2\_SDA\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_OD;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF4\_I2C2;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : INTERNAL\_UART3\_TX\_Pin INTERNAL\_UART3\_RX\_Pin \*/

GPIO\_InitStruct.Pin = INTERNAL\_UART3\_TX\_Pin|INTERNAL\_UART3\_RX\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF7\_USART3;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : LPS22HB\_INT\_DRDY\_EXTI0\_Pin LSM6DSL\_INT1\_EXTI11\_Pin ARD\_D2\_Pin HTS221\_DRDY\_EXTI15\_Pin

PMOD\_IRQ\_EXTI12\_Pin \*/

GPIO\_InitStruct.Pin = LPS22HB\_INT\_DRDY\_EXTI0\_Pin|LSM6DSL\_INT1\_EXTI11\_Pin|ARD\_D2\_Pin|HTS221\_DRDY\_EXTI15\_Pin

|PMOD\_IRQ\_EXTI12\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : USB\_OTG\_FS\_PWR\_EN\_Pin SPBTLE\_RF\_SPI3\_CSN\_Pin PMOD\_RESET\_Pin STSAFE\_A100\_RESET\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_PWR\_EN\_Pin|SPBTLE\_RF\_SPI3\_CSN\_Pin|PMOD\_RESET\_Pin|STSAFE\_A100\_RESET\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : VL53L0X\_XSHUT\_Pin LED3\_WIFI\_\_LED4\_BLE\_Pin \*/

GPIO\_InitStruct.Pin = VL53L0X\_XSHUT\_Pin|LED3\_WIFI\_\_LED4\_BLE\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pins : VL53L0X\_GPIO1\_EXTI7\_Pin LSM3MDL\_DRDY\_EXTI8\_Pin \*/

GPIO\_InitStruct.Pin = VL53L0X\_GPIO1\_EXTI7\_Pin|LSM3MDL\_DRDY\_EXTI8\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : USB\_OTG\_FS\_VBUS\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_VBUS\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(USB\_OTG\_FS\_VBUS\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : USB\_OTG\_FS\_ID\_Pin USB\_OTG\_FS\_DM\_Pin USB\_OTG\_FS\_DP\_Pin \*/

GPIO\_InitStruct.Pin = USB\_OTG\_FS\_ID\_Pin|USB\_OTG\_FS\_DM\_Pin|USB\_OTG\_FS\_DP\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF10\_OTG\_FS;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : PMOD\_SPI2\_SCK\_Pin \*/

GPIO\_InitStruct.Pin = PMOD\_SPI2\_SCK\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF5\_SPI2;

HAL\_GPIO\_Init(PMOD\_SPI2\_SCK\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : PMOD\_UART2\_CTS\_Pin PMOD\_UART2\_RTS\_Pin PMOD\_UART2\_TX\_Pin PMOD\_UART2\_RX\_Pin \*/

GPIO\_InitStruct.Pin = PMOD\_UART2\_CTS\_Pin|PMOD\_UART2\_RTS\_Pin|PMOD\_UART2\_TX\_Pin|PMOD\_UART2\_RX\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF7\_USART2;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

/\*Configure GPIO pins : ARD\_D15\_Pin ARD\_D14\_Pin \*/

GPIO\_InitStruct.Pin = ARD\_D15\_Pin|ARD\_D14\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_OD;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF4\_I2C1;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(*EXTI9\_5\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*EXTI9\_5\_IRQn*);

HAL\_NVIC\_SetPriority(*EXTI15\_10\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*EXTI15\_10\_IRQn*);

}

/\* USER CODE BEGIN 4 \*/

/\* 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 \*/

* Printing data on SWV ITM Data Console using USART1.

**#include**<stdio.h>

**#include** <stdint.h>

**#include**"stm32l4\_gpio\_driver.h"

**#include**"stm32l475.h"

**#include** <string.h>

**char** text[] = "Car";

// Initializes the output pin A2

**void** **InitOutputPinA2**()

{

GPIO\_Handle\_t output\_pin;

output\_pin.pGPIOx = GPIOA;

output\_pin.GPIO\_PinConfig.GPIO\_PinNumber = GPIO\_PIN\_NO\_2;

output\_pin.GPIO\_PinConfig.GPIO\_PinMode = GPIO\_MODE\_OUT;

output\_pin.GPIO\_PinConfig.GPIO\_PinOPType = GPIO\_OP\_TYPE\_PP;

output\_pin.GPIO\_PinConfig.GPIO\_PinPuPdControl = GPIO\_PIN\_PD;

GPIO\_Init(&output\_pin);

}

// Initializes the input pin C5

**void** **InitInputPinC5**()

{

GPIO\_Handle\_t input\_pin;

input\_pin.pGPIOx = GPIOC;

input\_pin.GPIO\_PinConfig.GPIO\_PinNumber = GPIO\_PIN\_NO\_5;

input\_pin.GPIO\_PinConfig.GPIO\_PinMode = GPIO\_MODE\_IN;

input\_pin.GPIO\_PinConfig.GPIO\_PinSpeed = GPIO\_SPEED\_HIGH;

input\_pin.GPIO\_PinConfig.GPIO\_PinPuPdControl = GPIO\_PIN\_PD;

GPIO\_PeriClockControl(GPIOC,ENABLE);

GPIO\_Init(&input\_pin);

}

// Sends bits out to pin A2 and reads bits in from pin C5

**void** **BitBanging**(**int** binary[], **int** Binary[])

{

**for**(**int** i = 0 ; i < **strlen**(text)\*8; i ++)

{

**if** (binary[i] == 1){

GPIO\_WriteToOutputPin(GPIOA, GPIO\_PIN\_NO\_2, GPIO\_PIN\_SET);

}**else** **if** (binary[i]== 0){

GPIO\_WriteToOutputPin(GPIOA, GPIO\_PIN\_NO\_2, GPIO\_PIN\_RESET);

}

**if** (GPIO\_ReadFromInputPin(GPIOC, GPIO\_PIN\_NO\_5) == 1){

Binary[i] = 1;

}**else** **if** (GPIO\_ReadFromInputPin(GPIOC, GPIO\_PIN\_NO\_5) == 0){

Binary[i] = 0;

}

}

}

//function to convert text to binary

**void** **textToBinary**(**int** binary[]) {

**int** len = **strlen**(text);

**int** index = 0;

**for**(**int** i = 0; i < len; i++) {

**int** c = (**int**) text[i];

**for**(**int** j = 7; j >= 0; j--) {

binary[index] = (c >> j) & 1;

index++;

}

}

}

//convert binary to text

**void** **BinaryToText**(**int** binary[], **int** size) {

**printf**("form Binary to text\n");

**char** text2[size/8 + 1];

**int** i, j, k, decimal;

**for** (i = 0, j = 0; i < size; i += 8, j++) {

decimal = 0;

**for** (k = 0; k < 8; k++) {

decimal += binary[i + k] \* (1 << (7 - k));

}

text2[j] = decimal;

}

text2[j] = '\0';

**printf**("%s\n", text2);

}

**int** **main**(**void**)

{

//defining array to store binary value

**int** y = 8 \* **strlen**(text);

**int** binary[y];

**int** Binary[y];

//Init and configure GPIO

InitOutputPinA2();

InitInputPinC5();

textToBinary(binary);

**for** (**int** i = 0; i < y; i++ ) {

**printf**("binary sending : %d\n",binary[i]);

}

BitBanging(binary, Binary);

**for** (**int** i = 0; i < y; i++ ) {

**printf**("binary received : %d\n",Binary[i]);

}

BinaryToText(Binary,y);

}

///\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* @file : main.c

// \* @author : Auto-generated by STM32CubeIDE

// \* @brief : Main program body

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* @attention

// \*

// \* Copyright (c) 2023 STMicroelectronics.

// \* All rights reserved.

// \*

// \* This software is licensed under terms that can be found in the LICENSE file

// \* in the root directory of this software component.

// \* If no LICENSE file comes with this software, it is provided AS-IS.

// \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*/

//

//#include <stdio.h>

//#include <math.h>

//#include <stdint.h>

//#include"stm32l4\_gpio\_driver.h"

//#include"stm32l475.h"

//#include <string.h>

//

//int arr[7];

////int res[7] = {0,0,0,0,0,0,0};

////

////// Initializes the output pin A2

////void InitOutputPinA2()

////{

//// GPIO\_Handle\_t output\_pin;

//// output\_pin.pGPIOx = GPIOA;

//// output\_pin.GPIO\_PinConfig.GPIO\_PinNumber = GPIO\_PIN\_NO\_2;

//// output\_pin.GPIO\_PinConfig.GPIO\_PinMode = GPIO\_MODE\_OUT;

//// output\_pin.GPIO\_PinConfig.GPIO\_PinOPType = GPIO\_OP\_TYPE\_PP;

//// output\_pin.GPIO\_PinConfig.GPIO\_PinPuPdControl = GPIO\_PIN\_PD;

//// GPIO\_Init(&output\_pin);

////}

////

////void InitInputPinC5()

////{

//// GPIO\_Handle\_t input\_pin;

//// input\_pin.pGPIOx = GPIOC;

//// input\_pin.GPIO\_PinConfig.GPIO\_PinNumber = GPIO\_PIN\_NO\_5;

//// input\_pin.GPIO\_PinConfig.GPIO\_PinMode = GPIO\_MODE\_IN;

//// input\_pin.GPIO\_PinConfig.GPIO\_PinSpeed = GPIO\_SPEED\_HIGH;

//// input\_pin.GPIO\_PinConfig.GPIO\_PinPuPdControl = GPIO\_PIN\_PD;

//// GPIO\_PeriClockControl(GPIOC,ENABLE);

//// GPIO\_Init(&input\_pin);

////}

////

////void BitBanging()

////{

//// for(int i = 0 ; i < 7; i ++)

//// {

//// if (arr[i] == 1){

//// GPIO\_WriteToOutputPin(GPIOA, GPIO\_PIN\_NO\_2, GPIO\_PIN\_SET);

//// }else if (arr[i]== 0){

//// GPIO\_WriteToOutputPin(GPIOA, GPIO\_PIN\_NO\_2, GPIO\_PIN\_RESET);

//// }

////

//// if (GPIO\_ReadFromInputPin(GPIOC, GPIO\_PIN\_NO\_5) == 1){

//// res[i] = 1;

//// }else if (GPIO\_ReadFromInputPin(GPIOC, GPIO\_PIN\_NO\_5) == 0){

//// res[i] = 0;

//// }

//// }

////}

//

////void CharToBinary(char c)

////{

//// for (int i = 6; i >= 0; i--) {

//// arr[i] = c & 1;

//// c >>= 1;

//// }

////}

////// Converts a binary array to a character

////char BinaryToChar()

////{

//// int a =0 , base = 1;

//// for (int i = 6; i >= 0; i--)

//// {

//// a += res[i] \* base;

//// base \*= 2;

//// }

//// return a;

////}

//void string\_to\_binary(char string[]) {

// int i;

// for(i=0;i<8\*strlen(string);i++)

// {

// arr[i]=string[i/8]&1;

// printf("%d",0 != (string[i/8] & 1 << (~i&7)));

// printf("%d",arr[i]);

// }

// printf("\n");

//

//}

//

//int main()

//{

// char \*string = "String to Binary In C";

// printf("Char send is %s \n", string);

// //InitOutputPinA2();

// //InitInputPinC5();

// string\_to\_binary(string);

// //BitBanging();

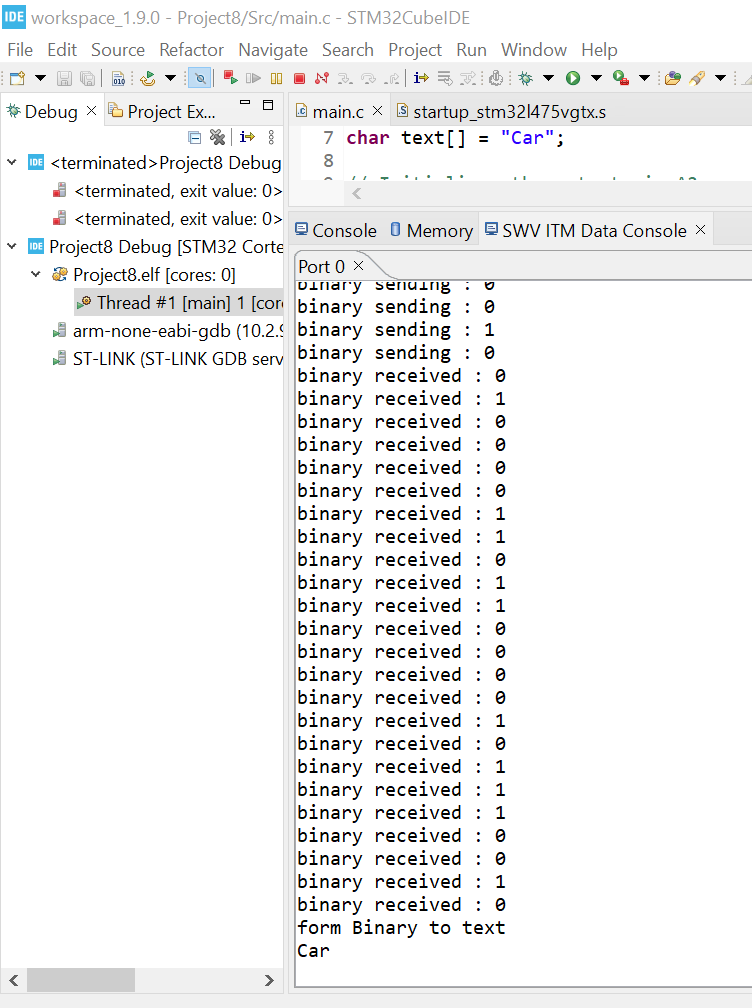
// //char c = BinaryToChar();

// //printf("Char received is %c \n", c);

// return 0;

//}

Output:



* Printing on tera term.

Code:

#include "main.h"  
#include <stdio.h>  
#include <string.h>  
/\* Private includes ----------------------------------------------------------\*/  
/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

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

struct bdata{  
    int a1 ;  
    int b;  
    char c ;

};

void saveCSV(){  
    FILE \*pt;  
    int data = 10;  
    pt = fopen("DATA.txt", "a+");  
    for (char ch = 'a'; ch < 'z'; ch++)  
        fputc(ch, pt);  
//    fprintf(pt, "a, b, c");  
//    fprintf(pt, "%d, %d, %d", data, data, data);  
    fclose(pt);  
}

/\* 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 ---------------------------------------------------------\*/  
UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/  
void SystemClock\_Config(void);  
static void MX\_GPIO\_Init(void);  
static void MX\_USART1\_UART\_Init(void);  
/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

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

/\* 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\_USART1\_UART\_Init();  
  /\* USER CODE BEGIN 2 \*/  
  struct bdata data;  
  data.a1 = 1;  
  data.b = 3;  
  data.c = 'a';  
  int a[] = {10,20};  
  int j;  
  j = 1;  
  int size  =  sizeof(a)/sizeof(a[0]);

  char buffer[100];

  FILE \*pt;

  //sprintf(buffer, "%d\r\n ", a[i]);

  /\* USER CODE END 2 \*/

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

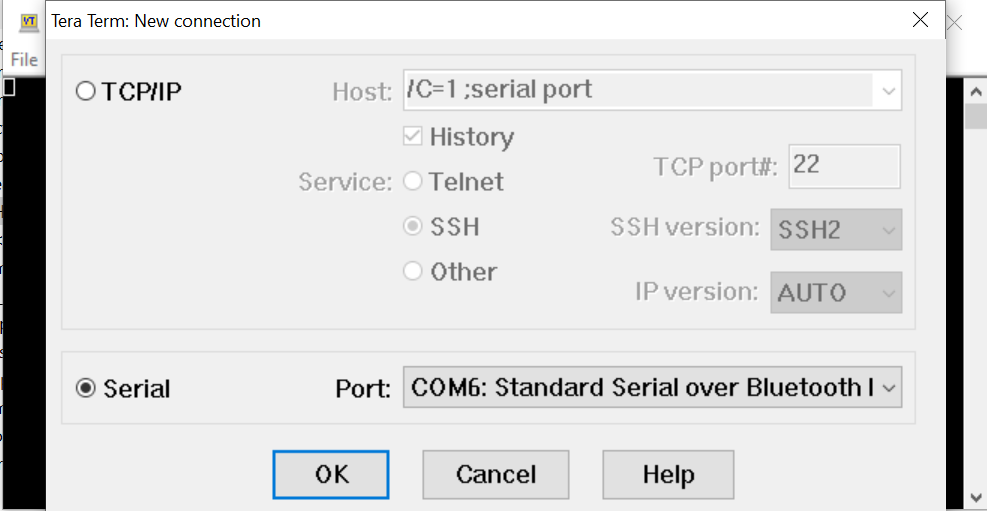
    /\* USER CODE END WHILE \*/

    /\* USER CODE BEGIN 3 \*/  
      for (int i = 0 ; i < size; i++){  
      sprintf(buffer, "%d) %d\r\n ",j, a[i]);  
     // saveCSV();  
      j++;  
//      saveCSV( data);  
      pt = fopen("DATA.csv", "a+");  
      fprintf(pt, "1 , 3 , 4");  
      fclose(pt);

      HAL\_UART\_Transmit(&huart1, buffer, strlen(buffer), 100);  
      HAL\_Delay(1000);  
      }

  }  
  /\* USER CODE END 3 \*/

Configuration of tera term:



A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Scanning for other Bluetooth beacons(half done).

Code:

/\* USER CODE BEGIN Header \*/

/\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @file    stm32l4xx\_it.c

  \* @brief   Interrupt Service Routines.

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \* @attention

  \*

  \* Copyright (c) 2023 STMicroelectronics.

  \* All rights reserved.

  \*

  \* This software is licensed under terms that can be found in the LICENSE file

  \* in the root directory of this software component.

  \* If no LICENSE file comes with this software, it is provided AS-IS.

  \*

  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  \*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "stm32l4xx\_it.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN TD \*/

/\* USER CODE END TD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\* External variables --------------------------------------------------------\*/

/\* USER CODE BEGIN EV \*/

/\* USER CODE END EV \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*           Cortex-M4 Processor Interruption and Exception Handlers          \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

  \* @brief This function handles Non maskable interrupt.

  \*/

void NMI\_Handler(void)

{

  /\* USER CODE BEGIN NonMaskableInt\_IRQn 0 \*/

  /\* USER CODE END NonMaskableInt\_IRQn 0 \*/

  /\* USER CODE BEGIN NonMaskableInt\_IRQn 1 \*/

  while (1)

  {

  }

  /\* USER CODE END NonMaskableInt\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Hard fault interrupt.

  \*/

void HardFault\_Handler(void)

{

  /\* USER CODE BEGIN HardFault\_IRQn 0 \*/

  /\* USER CODE END HardFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_HardFault\_IRQn 0 \*/

    /\* USER CODE END W1\_HardFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Memory management fault.

  \*/

void MemManage\_Handler(void)

{

  /\* USER CODE BEGIN MemoryManagement\_IRQn 0 \*/

  /\* USER CODE END MemoryManagement\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_MemoryManagement\_IRQn 0 \*/

    /\* USER CODE END W1\_MemoryManagement\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Prefetch fault, memory access fault.

  \*/

void BusFault\_Handler(void)

{

  /\* USER CODE BEGIN BusFault\_IRQn 0 \*/

  /\* USER CODE END BusFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_BusFault\_IRQn 0 \*/

    /\* USER CODE END W1\_BusFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles Undefined instruction or illegal state.

  \*/

void UsageFault\_Handler(void)

{

  /\* USER CODE BEGIN UsageFault\_IRQn 0 \*/

  /\* USER CODE END UsageFault\_IRQn 0 \*/

  while (1)

  {

    /\* USER CODE BEGIN W1\_UsageFault\_IRQn 0 \*/

    /\* USER CODE END W1\_UsageFault\_IRQn 0 \*/

  }

}

/\*\*

  \* @brief This function handles System service call via SWI instruction.

  \*/

void SVC\_Handler(void)

{

  /\* USER CODE BEGIN SVCall\_IRQn 0 \*/

  /\* USER CODE END SVCall\_IRQn 0 \*/

  /\* USER CODE BEGIN SVCall\_IRQn 1 \*/

  /\* USER CODE END SVCall\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Debug monitor.

  \*/

void DebugMon\_Handler(void)

{

  /\* USER CODE BEGIN DebugMonitor\_IRQn 0 \*/

  /\* USER CODE END DebugMonitor\_IRQn 0 \*/

  /\* USER CODE BEGIN DebugMonitor\_IRQn 1 \*/

  /\* USER CODE END DebugMonitor\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles Pendable request for system service.

  \*/

void PendSV\_Handler(void)

{

  /\* USER CODE BEGIN PendSV\_IRQn 0 \*/

  /\* USER CODE END PendSV\_IRQn 0 \*/

  /\* USER CODE BEGIN PendSV\_IRQn 1 \*/

  /\* USER CODE END PendSV\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles System tick timer.

  \*/

void SysTick\_Handler(void)

{

  /\* USER CODE BEGIN SysTick\_IRQn 0 \*/

  /\* USER CODE END SysTick\_IRQn 0 \*/

  HAL\_IncTick();

  /\* USER CODE BEGIN SysTick\_IRQn 1 \*/

  /\* USER CODE END SysTick\_IRQn 1 \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* STM32L4xx Peripheral Interrupt Handlers                                    \*/

/\* Add here the Interrupt Handlers for the used peripherals.                  \*/

/\* For the available peripheral interrupt handler names,                      \*/

/\* please refer to the startup file (startup\_stm32l4xx.s).                    \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

  \* @brief This function handles EXTI line[9:5] interrupts.

  \*/

void EXTI9\_5\_IRQHandler(void)

{

  /\* USER CODE BEGIN EXTI9\_5\_IRQn 0 \*/

  /\* USER CODE END EXTI9\_5\_IRQn 0 \*/

  HAL\_GPIO\_EXTI\_IRQHandler(SPSGRF\_915\_GPIO3\_EXTI5\_Pin);

  HAL\_EXTI\_IRQHandler(&H\_EXTI\_6);

  HAL\_GPIO\_EXTI\_IRQHandler(VL53L0X\_GPIO1\_EXTI7\_Pin);

  HAL\_GPIO\_EXTI\_IRQHandler(LSM3MDL\_DRDY\_EXTI8\_Pin);

  /\* USER CODE BEGIN EXTI9\_5\_IRQn 1 \*/

  /\* USER CODE END EXTI9\_5\_IRQn 1 \*/

}

/\*\*

  \* @brief This function handles EXTI line[15:10] interrupts.

  \*/

void EXTI15\_10\_IRQHandler(void)

{

  /\* USER CODE BEGIN EXTI15\_10\_IRQn 0 \*/

  /\* USER CODE END EXTI15\_10\_IRQn 0 \*/

  HAL\_GPIO\_EXTI\_IRQHandler(LPS22HB\_INT\_DRDY\_EXTI0\_Pin);

  HAL\_GPIO\_EXTI\_IRQHandler(LSM6DSL\_INT1\_EXTI11\_Pin);

  HAL\_EXTI\_IRQHandler(&H\_EXTI\_13);

  HAL\_GPIO\_EXTI\_IRQHandler(ARD\_D2\_Pin);

  HAL\_GPIO\_EXTI\_IRQHandler(HTS221\_DRDY\_EXTI15\_Pin);

  /\* USER CODE BEGIN EXTI15\_10\_IRQn 1 \*/

  /\* USER CODE END EXTI15\_10\_IRQn 1 \*/

}

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

A screenshot of a computer

Description automatically generated with medium confidence