EMBEDDED C CODE FOR INTERFACING MICROWAVE BASED ON STM32

/\* 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 \*/

#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 \*/

#define SEG\_THSD(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_1, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define SEG\_HNDR(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define SEG\_TENS(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define SEG\_UNIT(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_A(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_9, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_B(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_C(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_D(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_E(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_9, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_F(x) HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_5, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED\_G(x) HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_1, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define LED8(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_10, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define COL\_1(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define COL\_2(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_10, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define COL\_3(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_8, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define COL\_4(x) HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_9, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define COL\_5(x) HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_5, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define COL\_6(x) HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_1, (x)? GPIO\_PIN\_SET:GPIO\_PIN\_RESET)

#define DOOR1(x) HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_10, x ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET)

#define DOOR HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_9)

#define BUZZ(x) x ? HAL\_TIM\_PWM\_Start(&htim22, TIM\_CHANNEL\_1) : \

HAL\_TIM\_PWM\_Stop(&htim22, TIM\_CHANNEL\_1)

GPIO\_InitTypeDef GPIO\_InitStruct = { 0 };

/\* USER CODE END PM \*/

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

TIM\_HandleTypeDef htim6;

TIM\_HandleTypeDef htim22;

/\* USER CODE BEGIN PV \*/

volatile int displayValue;

volatile uint8\_t key;

volatile int dr;

/\* USER CODE END PV \*/

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

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM6\_Init(void);

static void MX\_TIM22\_Init(void);

/\* USER CODE BEGIN PFP \*/

void seven\_seg(int Z);

void display(uint8\_t x);

void updateSSD(int value);

void output(uint8\_t x);

uint8\_t keypad();

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) {

keypad();

GPIO\_InitStruct.Pin = GPIO\_PIN\_9;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

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

if (DOOR == GPIO\_PIN\_SET)

DOOR1(1);

else

DOOR1(0);

dr = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_9);

GPIO\_InitStruct.Pin = GPIO\_PIN\_9;

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);

if (htim == &htim6) {

static uint8\_t segmentCount = 0;

segmentCount++;

if (segmentCount > 3) {

segmentCount = 0;

}

seven\_seg(segmentCount);

}

}

/\* 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\_TIM6\_Init();

MX\_TIM22\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_Base\_Start\_IT(&htim6);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

updateSSD(0);

updateSSD(1);

updateSSD(1234);

TIM22->CCR1 = 500;

TIM22->CCR2 = 500;

HAL\_TIM\_PWM\_Start(&htim22, TIM\_CHANNEL\_1);

HAL\_Delay(1000);

HAL\_TIM\_PWM\_Stop(&htim22, TIM\_CHANNEL\_1);

HAL\_TIM\_PWM\_Start(&htim22, TIM\_CHANNEL\_2);

for (int i = 0; i < 10000; i += 1111) {

updateSSD(i);

HAL\_Delay(1000);

}

HAL\_TIM\_PWM\_Stop(&htim22, TIM\_CHANNEL\_2);

updateSSD(1234);

while (1) {

if (key != 0) {

updateSSD(key);

HAL\_Delay(10);

} else {

updateSSD(1234);

}

}

/\* 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\_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.PLLMUL = RCC\_PLLMUL\_4;

RCC\_OscInitStruct.PLL.PLLDIV = RCC\_PLLDIV\_2;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

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

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief TIM6 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM6\_Init(void)

{

/\* USER CODE BEGIN TIM6\_Init 0 \*/

/\* USER CODE END TIM6\_Init 0 \*/

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM6\_Init 1 \*/

/\* USER CODE END TIM6\_Init 1 \*/

htim6.Instance = TIM6;

htim6.Init.Prescaler = 0;

htim6.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim6.Init.Period = 64000;

htim6.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim6) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim6, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM6\_Init 2 \*/

/\* USER CODE END TIM6\_Init 2 \*/

}

/\*\*

\* @brief TIM22 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM22\_Init(void)

{

/\* USER CODE BEGIN TIM22\_Init 0 \*/

/\* USER CODE END TIM22\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

/\* USER CODE BEGIN TIM22\_Init 1 \*/

/\* USER CODE END TIM22\_Init 1 \*/

htim22.Instance = TIM22;

htim22.Init.Prescaler = 32-1;

htim22.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim22.Init.Period = 1000-1;

htim22.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim22.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim22) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim22, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_Init(&htim22) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim22, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 0;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim22, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_TIM\_PWM\_ConfigChannel(&htim22, &sConfigOC, TIM\_CHANNEL\_2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM22\_Init 2 \*/

/\* USER CODE END TIM22\_Init 2 \*/

HAL\_TIM\_MspPostInit(&htim22);

}

/\*\*

\* @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\_GPIOC\_CLK\_ENABLE();

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_1|GPIO\_PIN\_5, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_4|GPIO\_PIN\_5

|GPIO\_PIN\_8|GPIO\_PIN\_9, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_10|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_9, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PC1 PC5 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_1|GPIO\_PIN\_5;

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 : PA0 PA1 PA4 PA5

PA8 PA9 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_4|GPIO\_PIN\_5

|GPIO\_PIN\_8|GPIO\_PIN\_9;

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 pins : PA6 PA7 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_6|GPIO\_PIN\_7;

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

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

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

/\*Configure GPIO pins : PB0 PB6 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_6;

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

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

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

/\*Configure GPIO pins : PB10 PB4 PB5 PB9 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_10|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_9;

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);

}

/\* USER CODE BEGIN 4 \*/

void seven\_seg(int Z) {

static uint8\_t singleDigit;

switch (Z) {

case 0: {

SEG\_UNIT(0);

SEG\_TENS(1);

SEG\_HNDR(1);

SEG\_THSD(1);

singleDigit = (displayValue) % 10;

break;

}

case 1: {

SEG\_UNIT(1);

SEG\_TENS(0);

SEG\_HNDR(1);

SEG\_THSD(1);

singleDigit = (displayValue / 10) % 10;

break;

}

case 2: {

SEG\_UNIT(1);

SEG\_TENS(1);

SEG\_HNDR(0);

SEG\_THSD(1);

singleDigit = (displayValue / 100) % 10;

break;

}

case 3: {

SEG\_UNIT(1);

SEG\_TENS(1);

SEG\_HNDR(1);

SEG\_THSD(0);

singleDigit = (displayValue / 1000) % 10;

break;

}

}

switch (singleDigit) {

case 0: {

LED\_G(0);

LED\_F(1);

LED\_E(1);

LED\_D(1);

LED\_C(1);

LED\_B(1);

LED\_A(1);

break;

}

case 1: {

LED\_G(0);

LED\_F(0);

LED\_E(0);

LED\_D(0);

LED\_C(1);

LED\_B(1);

LED\_A(0);

break;

}

case 2: {

LED\_G(1);

LED\_F(0);

LED\_E(1);

LED\_D(1);

LED\_C(0);

LED\_B(1);

LED\_A(1);

break;

}

case 3: {

LED\_G(1);

LED\_F(0);

LED\_E(0);

LED\_D(1);

LED\_C(1);

LED\_B(1);

LED\_A(1);

break;

}

case 4: {

LED\_G(1);

LED\_F(1);

LED\_E(0);

LED\_D(0);

LED\_C(1);

LED\_B(1);

LED\_A(0);

break;

}

case 5: {

LED\_G(1);

LED\_F(1);

LED\_E(0);

LED\_D(1);

LED\_C(1);

LED\_B(0);

LED\_A(1);

break;

}

case 6: {

LED\_G(1);

LED\_F(1);

LED\_E(1);

LED\_D(1);

LED\_C(1);

LED\_B(0);

LED\_A(1);

break;

}

case 7: {

LED\_G(0);

LED\_F(0);

LED\_E(0);

LED\_D(0);

LED\_C(1);

LED\_B(1);

LED\_A(1);

break;

}

case 8: {

LED\_G(1);

LED\_F(1);

LED\_E(1);

LED\_D(1);

LED\_C(1);

LED\_B(1);

LED\_A(1);

break;

}

case 9: {

LED\_G(1);

LED\_F(1);

LED\_E(0);

LED\_D(1);

LED\_C(1);

LED\_B(1);

LED\_A(1);

break;

}

}

}

void updateSSD(int value) {

displayValue = value;

}

uint8\_t keypad() {

volatile uint8\_t a, b, c, d;

key = 0;

COL\_1(1);

COL\_2(0);

COL\_3(0);

COL\_4(0);

COL\_5(0);

COL\_6(0);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 22;

} else if (b == GPIO\_PIN\_SET) {

key = 21;

} else if (c == GPIO\_PIN\_SET) {

key = 5;

} else if (d == GPIO\_PIN\_SET) {

key = 23;

}

COL\_1(0);

COL\_2(1);

COL\_3(0);

COL\_4(0);

COL\_5(0);

COL\_6(0);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 19;

} else if (b == GPIO\_PIN\_SET) {

key = 18;

} else if (c == GPIO\_PIN\_SET) {

key = 4;

} else if (d == GPIO\_PIN\_SET) {

key = 20;

}

COL\_1(0);

COL\_2(0);

COL\_3(1);

COL\_4(0);

COL\_5(0);

COL\_6(0);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 17;

} else if (b == GPIO\_PIN\_SET) {

key = 16;

} else if (c == GPIO\_PIN\_SET) {

key = 3;

} else if (d == GPIO\_PIN\_SET) {

key = 9;

}

COL\_1(0);

COL\_2(0);

COL\_3(0);

COL\_4(1);

COL\_5(0);

COL\_6(0);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 15;

} else if (b == GPIO\_PIN\_SET) {

key = 14;

} else if (c == GPIO\_PIN\_SET) {

key = 2;

} else if (d == GPIO\_PIN\_SET) {

key = 8;

}

COL\_1(0);

COL\_2(0);

COL\_3(0);

COL\_4(0);

COL\_5(1);

COL\_6(0);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 13;

} else if (b == GPIO\_PIN\_SET) {

key = 12;

} else if (c == GPIO\_PIN\_SET) {

key = 1;

} else if (d == GPIO\_PIN\_SET) {

key = 7;

}

COL\_1(0);

COL\_2(0);

COL\_3(0);

COL\_4(0);

COL\_5(0);

COL\_6(1);

a = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_7);

b = HAL\_GPIO\_ReadPin(GPIOA, GPIO\_PIN\_6);

c = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0);

d = HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_6);

if (a == GPIO\_PIN\_SET) {

key = 11;

} else if (b == GPIO\_PIN\_SET) {

key = 10;

} else if (c == GPIO\_PIN\_SET) {

key = 0;

} else if (d == GPIO\_PIN\_SET) {

key = 6;

}

if(key)

BUZZ(1);

else

BUZZ(0);

return 0;

}

/\* 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 \*/