**STM32 Code**

/\* USER CODE BEGIN Header \*/

/\*\*

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

\* @file : main.c

\* @brief : Main program body

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

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under BSD 3-Clause license,

\* the "License"; You may not use this file except in compliance with the

\* License. You may obtain a copy of the License at:

\* opensource.org/licenses/BSD-3-Clause

\*

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

\*/

/\* USER CODE END Header \*/

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

#include "main.h"

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

/\* USER CODE BEGIN Includes \*/

#include "string.h"

/\* USER CODE END Includes \*/

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

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

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

/\* USER CODE BEGIN PD \*/

#define MESSAGE 100

/\* USER CODE END PD \*/

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

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

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

UART\_HandleTypeDef huart1;

/\* USER CODE BEGIN PV \*/

uint8\_t RX\_MESSAGE[MESSAGE] = { 0 };

char voice[MESSAGE] = "";

int indx = 0;

/\* 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 \*/

HAL\_UART\_Receive\_IT(&huart1, RX\_MESSAGE, MESSAGE);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_SET);

strcpy(voice, (char\*) RX\_MESSAGE);

volatile int i;

for (i=indx; voice[i] != '\0'; i++)

{

if ((voice[i] == 'A') || (voice[i] == 'a'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'B') || (voice[i] == 'b'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'C') || (voice[i] == 'c'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'D') || (voice[i] == 'd'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'E') || (voice[i] == 'e'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'F') || (voice[i] == 'f'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'G') || (voice[i] == 'g'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'H') || (voice[i] == 'h'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'I') || (voice[i] == 'i'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'J') || (voice[i] == 'j'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'K') || (voice[i] == 'k'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'L') || (voice[i] == 'l'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'M') || (voice[i] == 'm'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'N') || (voice[i] == 'n'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'O') || (voice[i] == 'o'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'P') || (voice[i] == 'p'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'Q') || (voice[i] == 'q'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'R') || (voice[i] == 'r'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'S') || (voice[i] == 's'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'T') || (voice[i] == 't'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'U') || (voice[i] == 'u'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'V') || (voice[i] == 'v'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'W') || (voice[i] == 'w'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'X') || (voice[i] == 'x'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'Y') || (voice[i] == 'y'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if ((voice[i] == 'Z') || (voice[i] == 'z'))

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if(voice[i]=='1'||voice[i]=='2'||voice[i]=='3'||voice[i]=='4'||voice[i]=='5'||voice[i]=='6'||voice[i]=='7'||voice[i]=='8'||voice[i]=='9'||voice[i]=='0')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(500);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(400);

if (voice[i] == '1')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

}

if (voice[i] == '2')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

}

if (voice[i] == '3')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if (voice[i] == '4')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '5')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '6')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if (voice[i] == '7')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '8')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '9')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

}

if (voice[i] == '0')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

}

if (voice[i] == '.')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i]== ' ')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_SET);

}

if (voice[i] == ',')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

}

if (voice[i] == '#')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i] == ';')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

}

if (voice[i] == ':')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '?')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i] == '!')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

if (voice[i] == '-')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i] == '\_')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(500);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(400);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i] == '('||voice[i] == ')')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

HAL\_Delay(500);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_Delay(400);

if (voice[i] == '(')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

if (voice[i] == ')')

{

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

}

indx += 1;

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_3, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_4, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(1000);

}

if (strlen(voice) == MESSAGE)

{

memset(RX\_MESSAGE, 0, 100);

strcpy(voice, (char\*) RX\_MESSAGE);

indx = 0;

}

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* 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 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 = 9600;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

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

{

Error\_Handler();

}

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

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

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

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

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

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3

|GPIO\_PIN\_4|GPIO\_PIN\_5, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : PB0 PB1 PB2 PB3

PB4 PB5 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3

|GPIO\_PIN\_4|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(GPIOB, &GPIO\_InitStruct);

}

/\* 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 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/