/\* USER CODE BEGIN Header \*/

/\*\*

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

\* @file : main.c

\* @brief : Main program body

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

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2020 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 \*/

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

TIM\_HandleTypeDef htim1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

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

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM1\_Init(void);

/\* USER CODE BEGIN PFP \*/

void delay(uint32\_t time);

/\* USER CODE END PFP \*/

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

/\* USER CODE BEGIN 0 \*/

//Finding speed of sound in cm/uSec

const float speedofsound=0.0343/2; //Divided by 2 because the signal travels to and from the obstacle

float distance;

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

uint32\_t echoval;

/\* 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\_TIM1\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

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

delay(3);

//Send out the signal

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

delay(10);

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

//Polling till echo pin sets

while(HAL\_GPIO\_ReadPin(Echo\_GPIO\_Port,Echo\_Pin)==GPIO\_PIN\_RESET);

//read the echo pulse width

echoval=0;

while(HAL\_GPIO\_ReadPin(Echo\_GPIO\_Port,Echo\_Pin)==GPIO\_PIN\_SET)

{

echoval++;

delay(2);

}

//To find distance in cm

distance=(echoval)\*2\*speedofsound;

if(distance<10)

{

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_12,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_13,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_14,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_15,GPIO\_PIN\_RESET);

HAL\_Delay(10);

}

else if(distance<20)

{

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_13,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_12,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_14,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_15,GPIO\_PIN\_RESET);

HAL\_Delay(10);

}

else if(distance<40)

{

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_14,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_13,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_12,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_15,GPIO\_PIN\_RESET);

HAL\_Delay(10);

}

else

{

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_15,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_13,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_14,GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(GPIOD,GPIO\_PIN\_12,GPIO\_PIN\_RESET);

HAL\_Delay(10);

}

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

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

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

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

\* in the RCC\_OscInitTypeDef structure.

\*/

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

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

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

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

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 84;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

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

{

Error\_Handler();

}

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

\*/

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

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

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

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

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief TIM1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM1\_Init(void)

{

/\* USER CODE BEGIN TIM1\_Init 0 \*/

/\* USER CODE END TIM1\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM1\_Init 1 \*/

/\* USER CODE END TIM1\_Init 1 \*/

htim1.Instance = TIM1;

htim1.Init.Prescaler = 84-1;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 0;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM1\_Init 2 \*/

/\* USER CODE END TIM1\_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\_GPIOH\_CLK\_ENABLE();

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

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, Trigger\_Pin|GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14

|GPIO\_PIN\_15, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : Echo\_Pin \*/

GPIO\_InitStruct.Pin = Echo\_Pin;

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

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

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

/\*Configure GPIO pins : Trigger\_Pin PD12 PD13 PD14

PD15 \*/

GPIO\_InitStruct.Pin = Trigger\_Pin|GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14

|GPIO\_PIN\_15;

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);

}

/\* USER CODE BEGIN 4 \*/

//Function for microsecond delay

void delay(uint32\_t time)

{

TIM1->ARR=time-1;

TIM1->EGR=1;

TIM1->SR &=~1;

TIM1->CR1 |=1;

while((TIM1->SR&0x0001)!=1);

TIM1->SR &=~(0x0001);

}

/\* 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 \*/

/\* 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,

tex: 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\*\*\*\*/