Electrical and Computer Engineering Department

ECE 4510/5530 Microcontroller Applications

Project 2

Title: Laboratory Design Project

Masoud Panahi, Donovan J Colo

TEAM 8

April 15, 2022

Contents

Introduction....3
Procedure....4
Task a....4
Task b....8
Task c....9
Results....14
Conclusion....15
Appendix....16
Main.c....16
stm32f4xx_it.c....23
Main.lst....29
stm32f4xx_it.lst....39

Introduction

The theme of the project is to move an object that has been placed on a conveyor belt from a start position to a specific end position. Since this part of project is not a physical model of the plant would include the peripherals as follows: a Start Switch, a LED indicator, a Buzzer (a small speaker), an IR (Infra-Red) Emitter and Receiver Module, an H-Bridge Motor Driver Module, and a Motor Encoder Module, it is simulated by using function generator and oscilloscope to show the results.

Design Procedure

Task a

In this project, dip switches are used to simulate start and stop switches. Green and Red LEDs are used as indicators. To simulate buzzer, Timer 3, Channel 4 used as an output compared to provide different frequency. Oscilloscope channel 1 is used to show the buzzer performance. Function generator is used to provide input capture signal for timer 5 channel 4. To show different duty cycles at frequency 30kHz, PWM signal on timer 4 channel 4 is used. Ports and setting are listed below. Details of each part are shown in screenshots and pictures with caption and red squares for important part. C code is provided in appendix part, calculation for different type of timers are shown in comments.

List of ports and settings:

Start Switch: bounce-free switch, input, PE7 (EXTI7)

Stop Switch: active low, Input, IR Signal, PE8(EXTI8)

IR (Infra-Red) Emitter: Red LED, Output, PG14

LED indicator: Green LED, output (Six times,1 blink/s) and (10 times,2 blink/s), PG13

Buzzer (a small speaker): Output (5.5 kHz) TIM3_CH4_OC_Buzzer_PB1

Receiver Module:

H-Bridge Motor: PWM Signal (duty cycle is 50%), Output (verified using a Logic Analyzer), TIM4_CH4_PWM_PD15

Driver Module:

Motor Encoder Module: Function Generator (f= 5.5KHz), Input capture signal, TIM5_CH4_IC_PA3

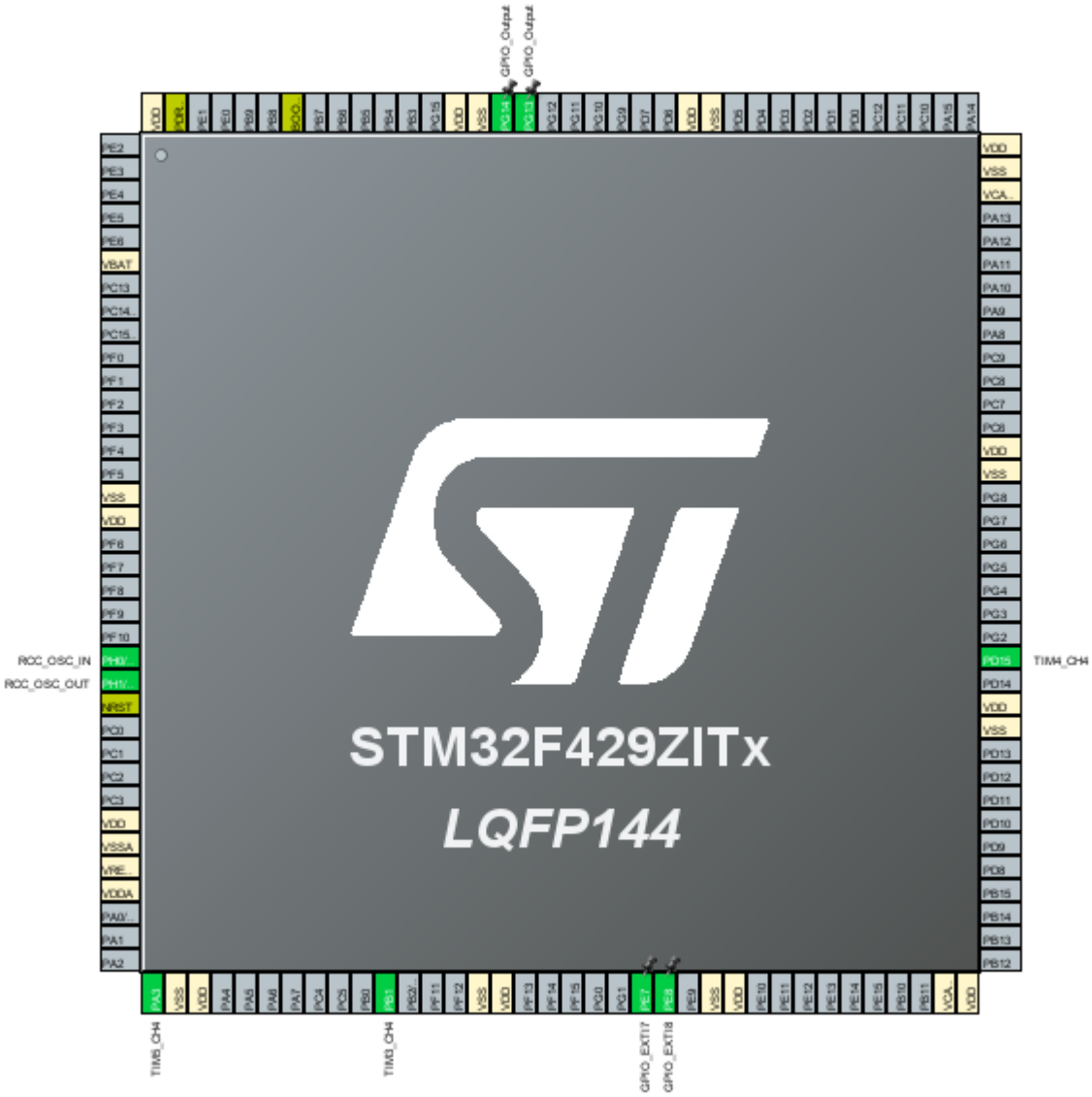


Figure 1. port configuration

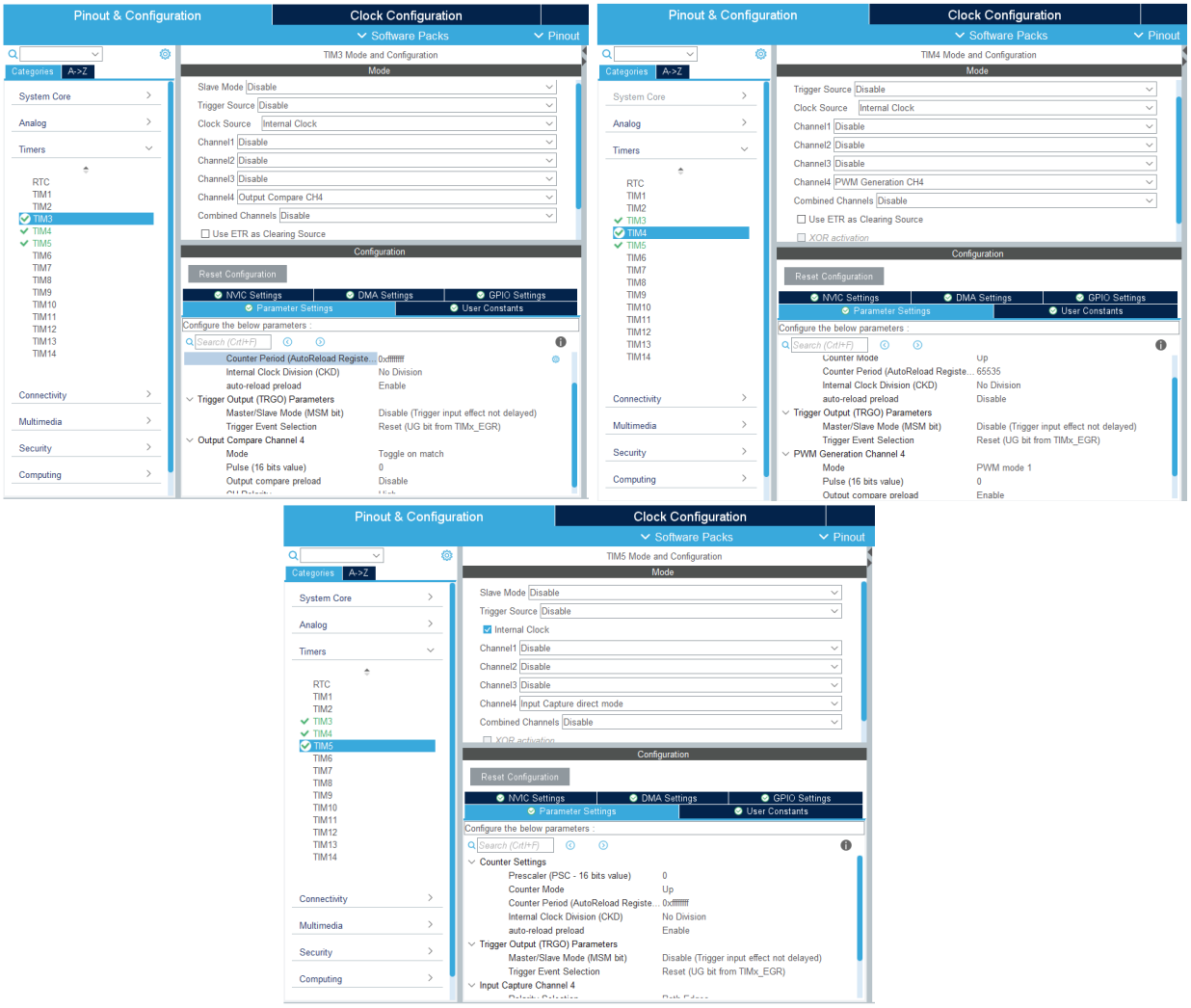


Figure 2. Timers' configuration

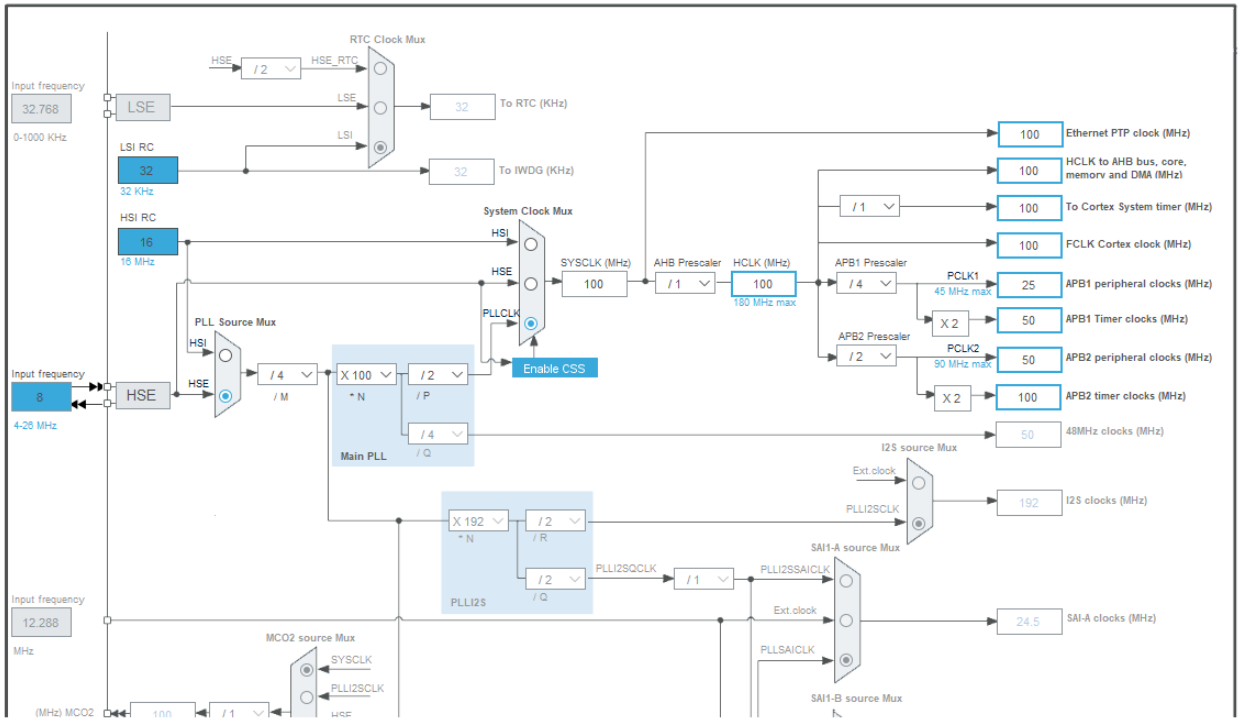


Figure 3. Clock configuration

Task b

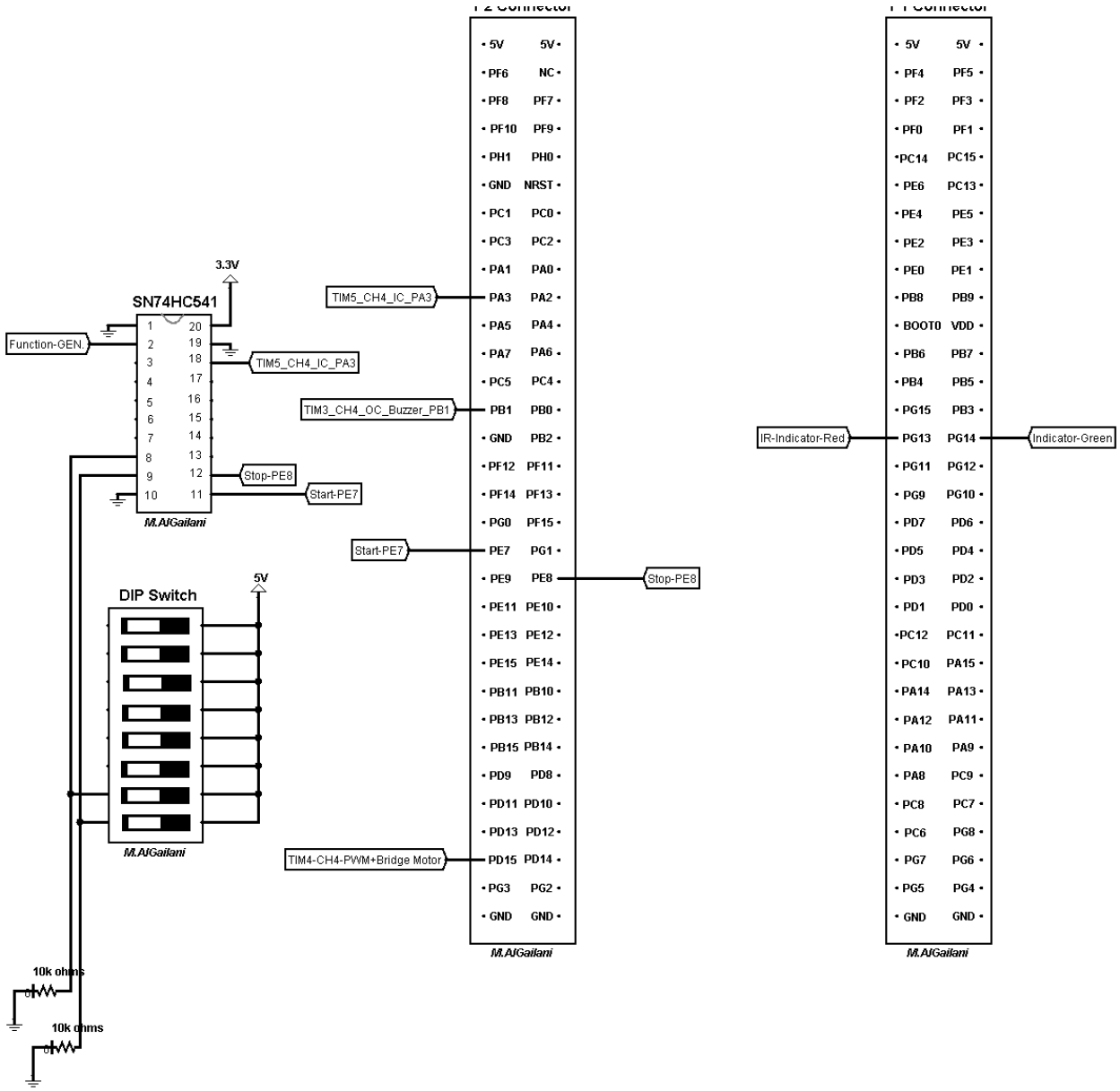


Figure 4. Schematic

Task c

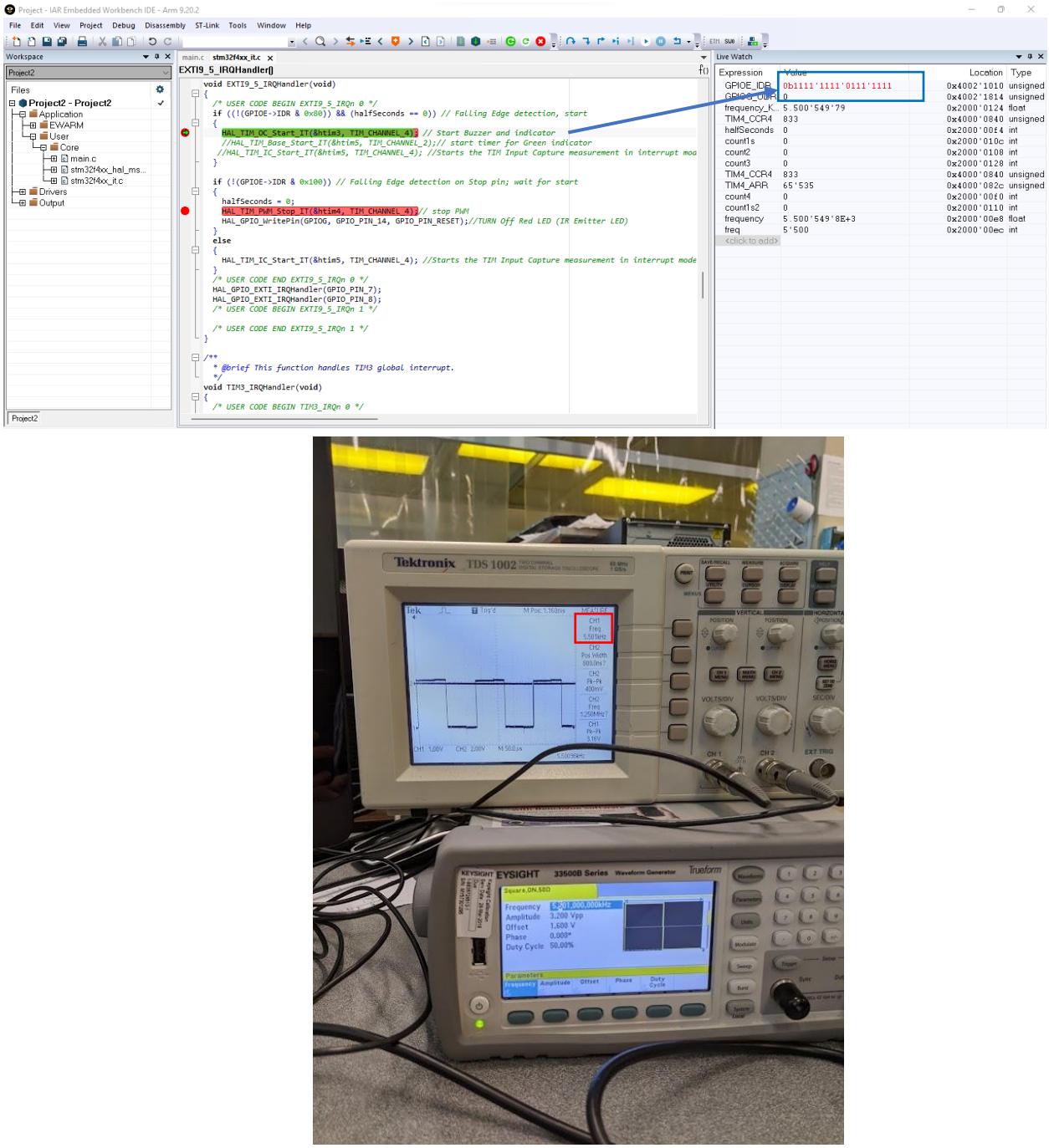
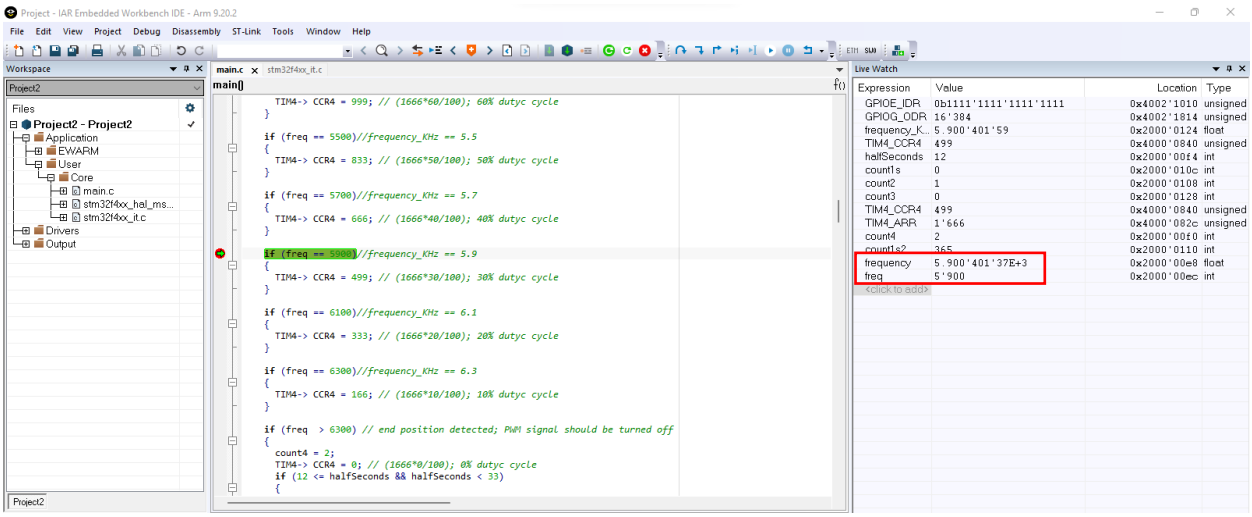


Figure 5. Debugging view and screenshot of start and buzzer signal at 5.5 kHz



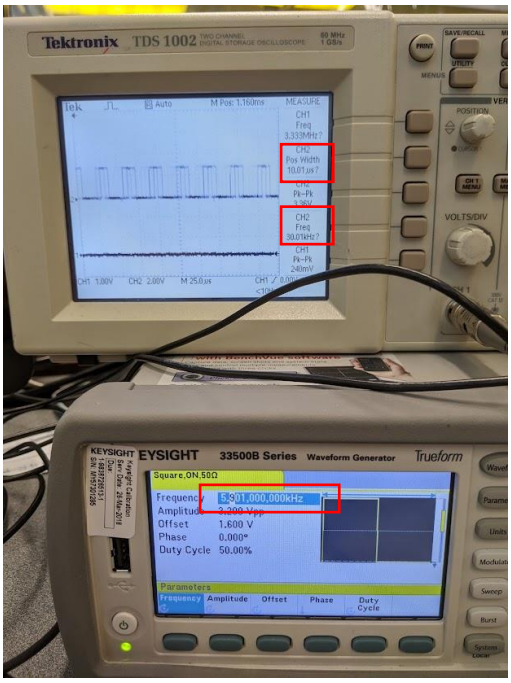


Figure 6. Show 30% duty cycle (10us*30kHz= 30%) on the debugging view and oscilloscope

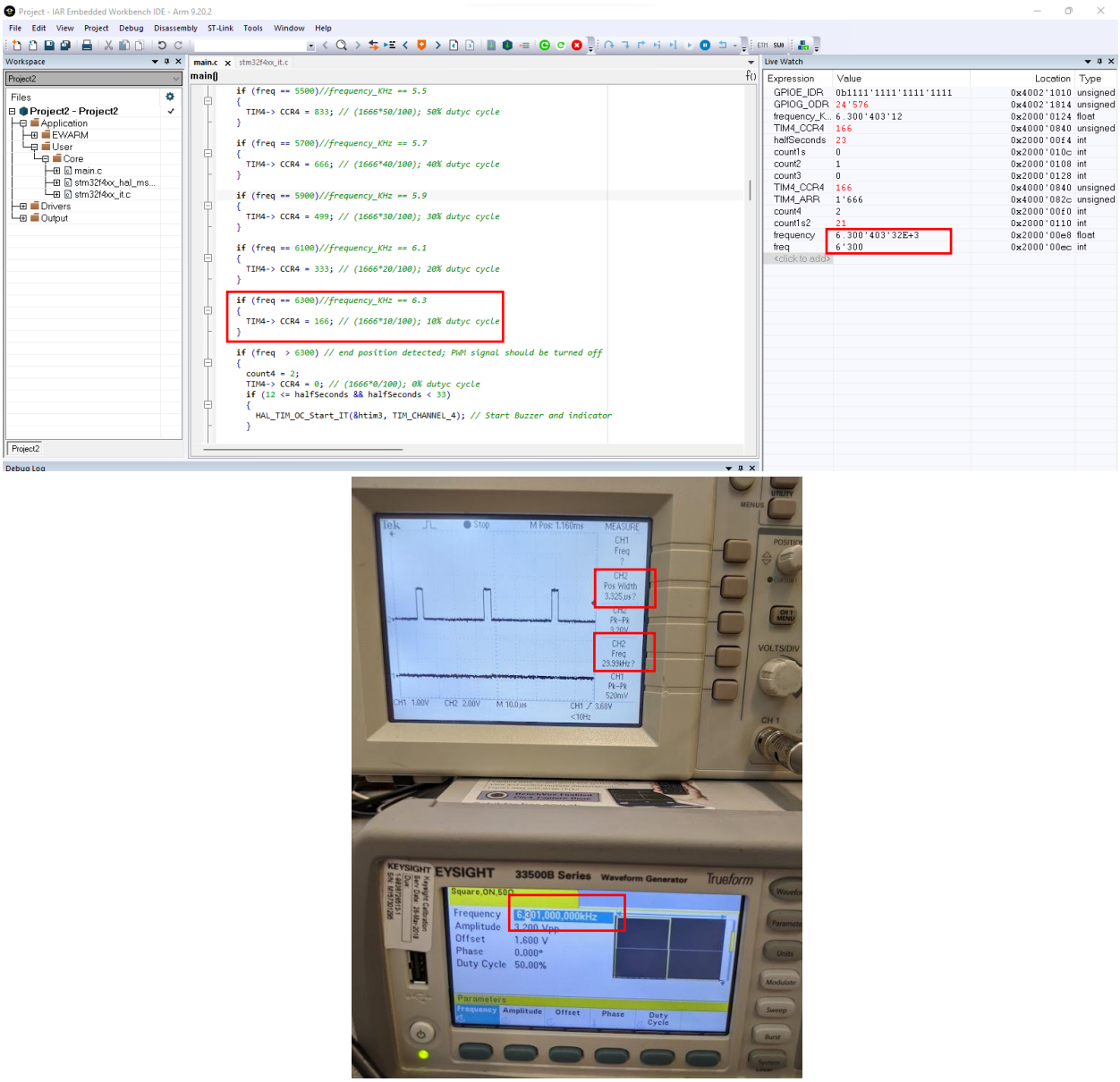


Figure 7. Show 10% duty cycle (3.25us*30kHz= 9.75%) on the debugging view and oscilloscope

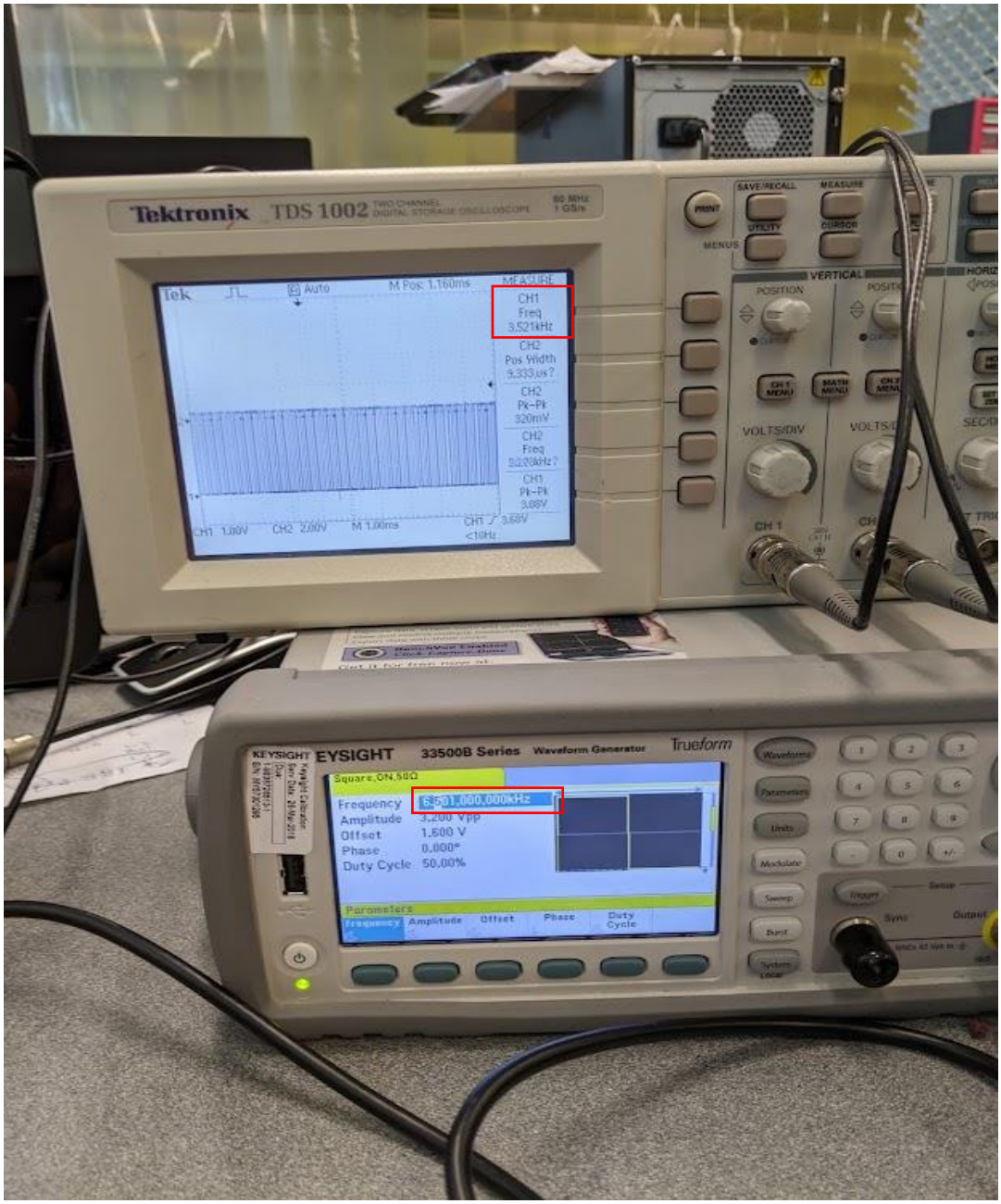


Figure 8. Picture of buzzer signal at 3.5 kHz when the duty cycle is 0% and input signal frequency is more than 6.3 kHz

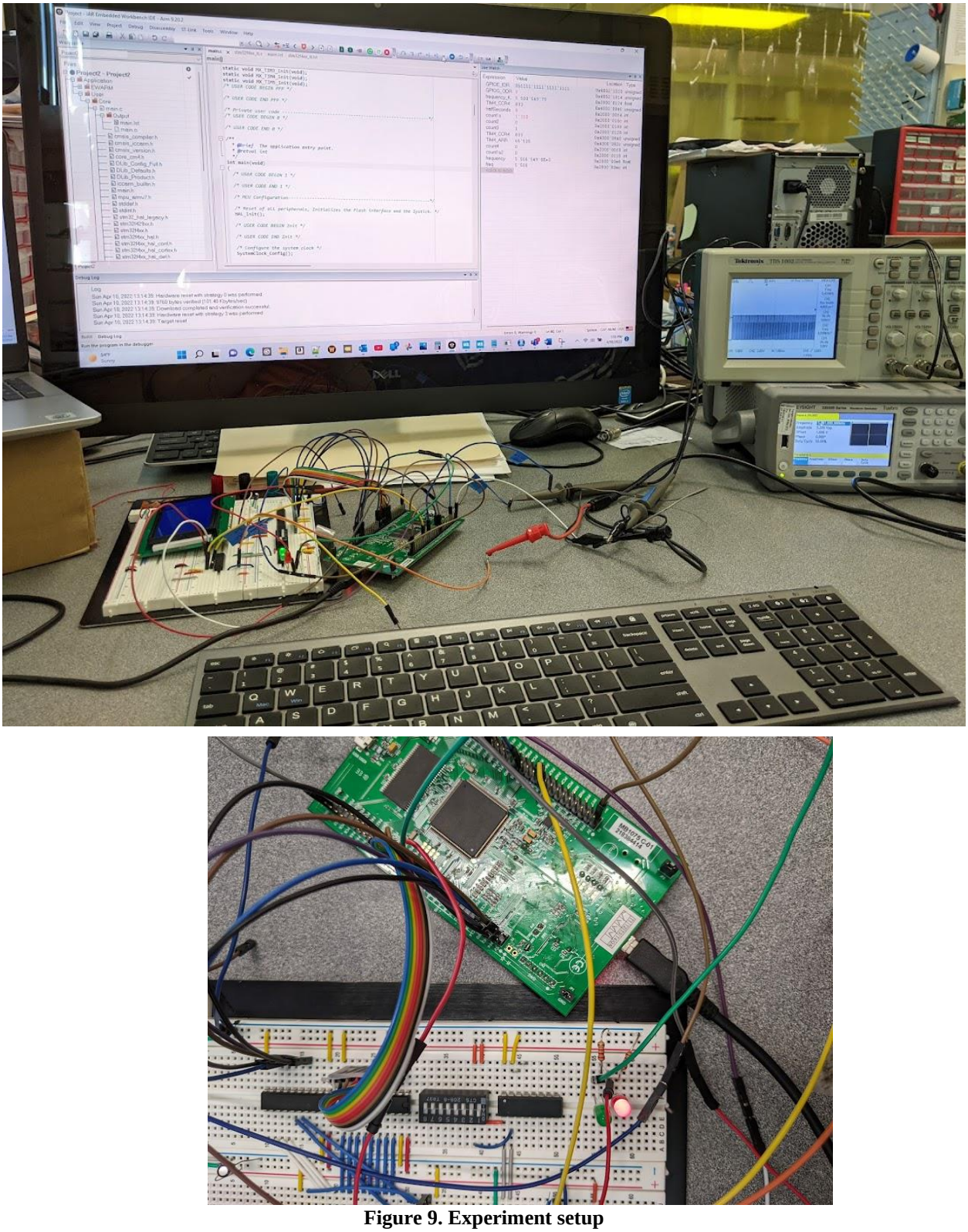


Figure 9. Experiment setup

Results

N/A

Conclusion

In this project an object that has been placed on a conveyor belt to move from a start position to a specific end position. Since in this part of project we didn't implement our code and circuit in physical model, we used oscilloscope and function generator to simulate the real system. Finally, we run the code successfully and show the results in each step by using screenshots of debugging view and pictures of oscilloscope.

Appendix

Main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 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 ---------------------------------------------------------*/
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim5;

/* USER CODE BEGIN PV */
float Hi_pulsewidth = 0x0000;
float Hi_pulsewidth11 = 0x0000; // for Ultrasonic sensor to measure distance
float Low_pulsewidth= 0x0000;
float frequency = 0x0000;
float dutyCycle = 0x0000;
float timePeriod = 0x0000;
int freq;

float frequency_KHz = 0x0000;
int count3 = 0;
int count4 = 0;
int halfSeconds = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
```

```
static void MX_GPIO_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void MX_TIM5_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_TIM3_Init();
  MX_TIM4_Init();
  MX_TIM5_Init();
  /* USER CODE BEGIN 2 */
  GPIOE->IDR  = 0x80; //start is off when the power is On at the first time
  //TIM3->ARR = 0XFFFF; // set the ARR register to count up to maximum value
  //HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4);//Start Buzzer and indicator,start the output capture Timer
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    freq = (int) frequency;

    if (freq == 4700) //frequency_KHz == 4.7
    {
      TIM4-> CCR4 = 1499; // (1666*90/100); 90% dutyc cycle
    }

    if (freq == 4900)//frequency_KHz == 4.9
    {
      TIM4-> CCR4 = 1332; // (1666*80/100); 80% dutyc cycle
    }
```

```
    if (freq == 5100)//frequency_KHz == 5.1
    {
      TIM4-> CCR4 = 1166; // (1666*70/100); 70% dutyc cycle
    }

    if (freq == 5300)//frequency_KHz == 5.3
    {
      TIM4-> CCR4 = 999; // (1666*60/100); 60% dutyc cycle
    }

    if (freq == 5500)//frequency_KHz == 5.5
    {
      TIM4-> CCR4 = 833; // (1666*50/100); 50% dutyc cycle
    }

    if (freq == 5700)//frequency_KHz == 5.7
    {
      TIM4-> CCR4 = 666; // (1666*40/100); 40% dutyc cycle
    }

    if (freq == 5900)//frequency_KHz == 5.9
    {
      TIM4-> CCR4 = 499; // (1666*30/100); 30% dutyc cycle
    }

    if (freq == 6100)//frequency_KHz == 6.1
    {
      TIM4-> CCR4 = 333; // (1666*20/100); 20% dutyc cycle
    }

    if (freq == 6300)//frequency_KHz == 6.3
    {
      TIM4-> CCR4 = 166; // (1666*10/100); 10% dutyc cycle
    }

    if (freq  > 6300) // end position detected; PWM signal should be turned off
    {
      count4 = 2;
      TIM4-> CCR4 = 0; // (1666*0/100); 0% dutyc cycle
      if (12 <= halfSeconds && halfSeconds < 33)
      {
        HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer and indicator
      }

      //HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4); // Stop PWM
      //HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer

    }

  /* 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_SCALE3);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
```

```
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 100;
  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_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 0xffffffff;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_OC_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
```

```
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_TOGGLE;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 0;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 0xffffffff;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &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(&htim4, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */
  HAL_TIM_MspPostInit(&htim4);

}

/**
  * @brief TIM5 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM5_Init(void)
{

  /* USER CODE BEGIN TIM5_Init 0 */

  /* USER CODE END TIM5_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};

  /* USER CODE BEGIN TIM5_Init 1 */

  /* USER CODE END TIM5_Init 1 */
  htim5.Instance = TIM5;
  htim5.Init.Prescaler = 0;
  htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim5.Init.Period = 0xffffffff;
  htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_IC_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim5, &sConfigIC, TIM_CHANNEL_4) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM5_Init 2 */

  /* USER CODE END TIM5_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_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13|GPIO_PIN_14, GPIO_PIN_RESET);

  /*Configure GPIO pins : PE7 PE8 */
  GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pins : PG13 PG14 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI9_5_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 */
```

stm32f4xx_it.c

```
  /* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 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 "stm32f4xx_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 */

//EXTI
extern int halfSeconds;

//TIM3
int COUNT = 0;
int count2 = 0;
extern uint8_t count3;
extern uint8_t count4;

int count = 0;
uint32_t risingedge1 = 0x0000;
```

```
uint32_t fallingedge = 0x0000;
uint32_t risingedge2 = 0x0000;
extern float Hi_pulsewidth ;
extern float Low_pulsewidth ;
extern float frequency;
extern float dutyCycle;
extern float timePeriod;
extern float frequency_KHz;

//TIM5
int count1s = 0;
int count1s2 = 0;
//uint8_t count2 = 0x00;
/* 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 --------------------------------------------------------*/
extern TIM_HandleTypeDef htim3;
extern TIM_HandleTypeDef htim4;
extern TIM_HandleTypeDef htim5;
/* 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 Pre-fetch 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 */
}

/******************************************************************************/
/* STM32F4xx 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_stm32f4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles EXTI line[9:5] interrupts.
  */
void EXTI9_5_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI9_5_IRQn 0 */
  if ((!(GPIOE->IDR & 0x80)) && (halfSeconds == 0)) // Falling Edge detection, start
  {
    111HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer and indicator
    //HAL_TIM_Base_Start_IT(&htim5, TIM_CHANNEL_2);// start timer for Green indicator
   //HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_4); //Starts the TIM Input Capture measurement in interrupt mode.
  }

  if (!(GPIOE->IDR & 0x100)) // Falling Edge detection on Stop pin; wait for start
  {
    halfSeconds = 0;
    HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4);// stop PWM
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);//TURN Off Red LED (IR Emitter LED)
  }
  else
  {
    HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_4); //Starts the TIM Input Capture measurement in interrupt mode.
  }
  /* USER CODE END EXTI9_5_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_7);
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
  /* USER CODE BEGIN EXTI9_5_IRQn 1 */

  /* USER CODE END EXTI9_5_IRQn 1 */
}

/**
  * @brief This function handles TIM3 global interrupt.
```

```
  */
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */
  if (halfSeconds < 12)
  {
    if (count2 == 0)
    {
      TIM3->CCR4 += 4545;// Tcnt for one periode = (1/f) * fapb1_clock = (1/5.5kHz)* 50MHz = 9090
      count2 = 1;
    }
    else if (count2 == 1)
    {
    TIM3->CCR4 += (9090-4545);// 50% duty cycle:(50/100)*9090=4545
    count2 = 0;
    count1s += 1;
    }
    count3= 1;
  }
  if ((count1s == 2750) && (count3 != 2)) // To mak 1 halfSeconds, 1/5.5KHz =0.1818ms; 0.1818*X/2 = 1000ms; x = 5500/2=2750
  //else if ((count1s == 2750) && (halfSeconds <= 12)) // To mak 1 halfSeconds, 1/5.5KHz =0.1818ms; 0.1818*X/2 = 1000ms; x =
5500/2=2750
  {
    count1s = 0;
    HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13); //Toggle GREEN LED
    halfSeconds +=1;

  }

  ////////////////////////////////////////////////////
  //if (((GPIOE->IDR & 0x80)) && (halfSeconds == 12))// Falling Edge is not detected on start and after 12 halfSeconds
  if (((count3== 1)) && (halfSeconds == 12))// Falling Edge is not detected on start and after 12 halfSeconds
  //if (count3 == 1)
  {
    HAL_TIM_OC_Stop_IT(&htim3, TIM_CHANNEL_4); // Turn off Buzzer
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);//TURN ON Red LED (IR Emitter LED)

    //PWM signal driving the motor on
    HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_4);// to make a pulse with f= 30KHz and duty cycle=50%  0.016666 ms width and
0.03333 ms period.
    TIM4-> ARR = 1666; // (1/30KHz)*50MHz=1666
    TIM4-> CCR4 = 833; // (1/2*30KHz)*50MHz=833; 50% dutyc cycle
    count3= 2;
  }
//////////////////////////////////////////////////////////////

//  if ((TIM4->CCR4 == 0) && ( 12 <= halfSeconds && halfSeconds <= 32))
  //if ((TIM4->CCR4 == 0) && (halfSeconds <= 32))
  //count1s2 = 0;
  if (TIM4->CCR4 == 0)
  {
    if (count2 == 0)
    {
      TIM3->CCR4 += 7143;// Tcnt for one periode = (1/f) * fapb1_clock = (1/3.5kHz)* 50MHz = 14286
      count2 = 1;
    }
    else if (count2 == 1)
    {
    TIM3->CCR4 += (14286-7143);// 50% duty cycle:(50/100)*14286=7143
    count3 = 0;
    count1s2 += 1;
    }
  }
   if((count1s2 == 1375) && (halfSeconds <= 32)) // To make 1 halfSeconds, 1/3.5KHz =0.2857ms; 0.2857*2X = 1000ms; x = 3500/4 for 2
blinks/sec
     {
       count1s2 = 0;
       HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13); //Toggle GREEN LED
```

```
      halfSeconds +=1;
    }

  else if ( halfSeconds > 32)
  {
   HAL_TIM_OC_Stop_IT(&htim3, TIM_CHANNEL_4); // Stop Buzzer Start
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);//TURN Off Green LED (Indicator LED)
   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);//TURN Off Red LED (IR Emitter LED)
   halfSeconds = 0;
   count2 = 0;
   count3 = 0;
  }
  /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */


  /* USER CODE END TIM3_IRQn 1 */
}

/**
  * @brief This function handles TIM4 global interrupt.
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */

  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/**
  * @brief This function handles TIM5 global interrupt.
  */
void TIM5_IRQHandler(void)
{
  /* USER CODE BEGIN TIM5_IRQn 0 */
if ((GPIOA->IDR & 0x0008) && (count == 0)) // Rising-1 edge detection
  {
    risingedge1 = TIM5->CCR4;
    count = 1;
  }
  else if ((!(GPIOA->IDR & 0x0008)) && (count == 1))// Falling Edge detection
  {
    fallingedge = TIM5->CCR4;
    count = 2;
  }
else if ((GPIOA->IDR & 0x0008) && (count == 2)) //Rising-2 edge detection
  {
    risingedge2 = TIM5->CCR4;
    timePeriod = ((risingedge2 - risingedge1)*0.02) ; // HLCK=100MHz  and APB1=50MHz. Period (us).1/50MHz = 0.02us
    frequency = (1000000/timePeriod); // frequency (HZ)=1/timePeriod(us)= 10*6/timePeriod
    Hi_pulsewidth = ((fallingedge - risingedge1)* 0.02); // Hi_Pulsewidth (ms).

    Low_pulsewidth = ((risingedge2 - fallingedge) * 0.02); // Low_pulsewidth (ms).
    //printf("Period = %lf msec\n", timePeriod);
    dutyCycle = (Hi_pulsewidth/timePeriod)*100;


    count = 0;
    TIM5->CNT = 0; // Reset the counter register

    frequency_KHz = frequency/1000;
  }
```

```
  /* USER CODE END TIM5_IRQn 0 */
  HAL_TIM_IRQHandler(&htim5);
  /* USER CODE BEGIN TIM5_IRQn 1 */

  /* USER CODE END TIM5_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

Main.lst

```
###############################################################################
#
# IAR ANSI C/C++ Compiler V9.20.2.320/W64 for ARM         10/Apr/2022  12:24:21
# Copyright 1999-2021 IAR Systems AB.
#
#    Cpu mode         =  thumb
#    Endian           =  little
#    Source file      =
#        C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\Core\Src\main.c
#    Command line     =
#        -f "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\main.o.rsp"
#        ("C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\Core\Src\main.c" -D
#        USE_HAL_DRIVER -D STM32F429xx -lcN "C:\Users\Student\OneDrive -
#        Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\List\Application\User\Core"
#        -o "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core"
#        --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
#        --dlib_config "C:\Program Files\IAR Systems\Embedded Workbench
#        9.0\arm\inc\c\DLib_Config_Full.h" -I "C:\Users\Student\OneDrive -
#        Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM/../Core/Inc\\" -I
#        "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc\\"
#        -I "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\\"
#        -I "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\\"
#        -I "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/CMSIS/Include\\"
#        -Ohz) --dependencies=n "C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\main.o.d"
#    Locale           =  C
#    List file        =
#        C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\List\Application\User\Core\main.lst
#    Object file      =
#        C:\Users\Student\OneDrive - Western Michigan
#        University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\main.o
#    Runtime model:
#      __CPP_Runtime   = 1
#      __SystemLibrary = DLib
#      __dlib_version  = 6
#      __size_limit    = 32768|ARM.EW.LINKER
```

```
#
##############################################################################

C:\Users\Student\OneDrive - Western Michigan University\Documents\MIcrocontroller\Project2\Core\Src\main.c
  1  /* USER CODE BEGIN Header */
  2  /**
  3    ******************************************************************************
  4    * @file           : main.c
  5    * @brief          : Main program body
  6    ******************************************************************************
  7    * @attention
  8    *
  9    * Copyright (c) 2022 STMicroelectronics.
 10    * All rights reserved.
 11    *
 12    * This software is licensed under terms that can be found in the LICENSE file
 13    * in the root directory of this software component.
 14    * If no LICENSE file comes with this software, it is provided AS-IS.
 15    *
 16    ******************************************************************************
 17    */
 18  /* USER CODE END Header */
 19  /* Includes ------------------------------------------------------------------*/
 20  #include "main.h"
 21
 22  /* Private includes ----------------------------------------------------------*/
 23  /* USER CODE BEGIN Includes */
 24
 25  /* USER CODE END Includes */
 26
 27  /* Private typedef -----------------------------------------------------------*/
 28  /* USER CODE BEGIN PTD */
 29
 30  /* USER CODE END PTD */
 31
 32  /* Private define ------------------------------------------------------------*/
 33  /* USER CODE BEGIN PD */
 34  /* USER CODE END PD */
 35
 36  /* Private macro -------------------------------------------------------------*/
 37  /* USER CODE BEGIN PM */
 38
 39  /* USER CODE END PM */
 40
 41  /* Private variables ---------------------------------------------------------*/
 42  TIM_HandleTypeDef htim3;
 43  TIM_HandleTypeDef htim4;
 44  TIM_HandleTypeDef htim5;
 45
 46  /* USER CODE BEGIN PV */
 47  float Hi_pulsewidth = 0x0000;
 48  float Hi_pulsewidth11 = 0x0000; // for Ultrasonic sensor to measure distance
 49  float Low_pulsewidth= 0x0000;
 50  float frequency = 0x0000;
 51  float dutyCycle = 0x0000;
 52  float timePeriod = 0x0000;
 53  int freq;
 54
 55  float frequency_KHz = 0x0000;
 56  int count3 = 0;
 57  int count4 = 0;
 58  int halfSeconds = 0;
 59  /* USER CODE END PV */
 60
 61  /* Private function prototypes -----------------------------------------------*/
 62  void SystemClock_Config(void);
 63  static void MX_GPIO_Init(void);
 64  static void MX_TIM3_Init(void);
```

```
65  static void MX_TIM4_Init(void);
66  static void MX_TIM5_Init(void);
67  /* USER CODE BEGIN PFP */
68
69  /* USER CODE END PFP */
70
71  /* Private user code ---------------------------------------------------------*/
72  /* USER CODE BEGIN 0 */
73
74  /* USER CODE END 0 */
75
76  /**
77    * @brief  The application entry point.
78    * @retval int
79    */
80  int main(void)
81  {
82    /* USER CODE BEGIN 1 */
83
84    /* USER CODE END 1 */
85
86    /* MCU Configuration--------------------------------------------------------*/
87
88    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
89    HAL_Init();
90
91    /* USER CODE BEGIN Init */
92
93    /* USER CODE END Init */
94
95    /* Configure the system clock */
96    SystemClock_Config();
97
98    /* USER CODE BEGIN SysInit */
99
100   /* USER CODE END SysInit */
101
102   /* Initialize all configured peripherals */
103   MX_GPIO_Init();
104   MX_TIM3_Init();
105   MX_TIM4_Init();
106   MX_TIM5_Init();
107   /* USER CODE BEGIN 2 */
108   GPIOE->IDR  = 0x80; //start is off when the power is On at the first time
109   //TIM3->ARR = 0XFFFF; // set the ARR register to count up to maximum value
110   //HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4);//Start Buzzer and indicator,start the output capture Timer
111   /* USER CODE END 2 */
112
113   /* Infinite loop */
114   /* USER CODE BEGIN WHILE */
115   while (1)
116   {
117     /* USER CODE END WHILE */
118
119     /* USER CODE BEGIN 3 */
120     freq = (int) frequency;
121
122     if (freq == 4700) //frequency_KHz == 4.7
123     {
124       TIM4-> CCR4 = 1499; // (1666*90/100); 90% dutyc cycle
125     }
126
127     if (freq == 4900)//frequency_KHz == 4.9
128     {
129       TIM4-> CCR4 = 1332; // (1666*80/100); 80% dutyc cycle
130     }
131
132     if (freq == 5100)//frequency_KHz == 5.1
```

```
133         {
134           TIM4-> CCR4 = 1166; // (1666*70/100); 70% dutyc cycle
135         }
136
137         if (freq == 5300)//frequency_KHz == 5.3
138         {
139           TIM4-> CCR4 = 999; // (1666*60/100); 60% dutyc cycle
140         }
141
142         if (freq == 5500)//frequency_KHz == 5.5
143         {
144           TIM4-> CCR4 = 833; // (1666*50/100); 50% dutyc cycle
145         }
146
147         if (freq == 5700)//frequency_KHz == 5.7
148         {
149           TIM4-> CCR4 = 666; // (1666*40/100); 40% dutyc cycle
150         }
151
152         if (freq == 5900)//frequency_KHz == 5.9
153         {
154           TIM4-> CCR4 = 499; // (1666*30/100); 30% dutyc cycle
155         }
156
157         if (freq == 6100)//frequency_KHz == 6.1
158         {
159           TIM4-> CCR4 = 333; // (1666*20/100); 20% dutyc cycle
160         }
161
162         if (freq == 6300)//frequency_KHz == 6.3
163         {
164           TIM4-> CCR4 = 166; // (1666*10/100); 10% dutyc cycle
165         }
166
167         if (freq  > 6300) // end position detected; PWM signal should be turned off
168         {
169           count4 = 2;
170           TIM4-> CCR4 = 0; // (1666*0/100); 0% dutyc cycle
171           if (12 <= halfSeconds && halfSeconds < 33)
172           {
173             HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer and indicator
174           }
175
176           //HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4); // Stop PWM
177           //HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer
178
179         }
180
181       //    if (4.69 < frequency_KHz && frequency_KHz < 4.71) //frequency_KHz == 4.7
182       //    {
183       //      TIM4-> CCR4 = 1499; // (1666*90/100); 90% dutyc cycle
184       //    }
185       //
186       //    if (4.89 < frequency_KHz && frequency_KHz < 4.91)//frequency_KHz == 4.9
187       //    {
188       //      TIM4-> CCR4 = 1332; // (1666*80/100); 80% dutyc cycle
189       //    }
190       //
191       //    if (5.09< frequency_KHz && frequency_KHz < 5.11)//frequency_KHz == 5.1
192       //    {
193       //      TIM4-> CCR4 = 1166; // (1666*70/100); 70% dutyc cycle
194       //    }
195       //
196       //    if (5.29 < frequency_KHz && frequency_KHz < 5.31)//frequency_KHz == 5.3
197       //    {
198       //      TIM4-> CCR4 = 999; // (1666*60/100); 60% dutyc cycle
199       //    }
200       //
```

```
201     //   if (5.49 < frequency_KHz && frequency_KHz < 5.51)//frequency_KHz == 5.5
202     //   {
203     //     TIM4-> CCR4 = 833; // (1666*50/100); 50% dutyc cycle
204     //   }
205     //
206     //   if (5.69 < frequency_KHz && frequency_KHz < 5.71)//frequency_KHz == 5.7
207     //   {
208     //     TIM4-> CCR4 = 666; // (1666*40/100); 40% dutyc cycle
209     //   }
210     //
211     //   if (5.89 < frequency_KHz && frequency_KHz < 5.91)//frequency_KHz == 5.9
212     //   {
213     //     TIM4-> CCR4 = 499; // (1666*30/100); 30% dutyc cycle
214     //   }
215     //
216     //   if (6.09 < frequency_KHz && frequency_KHz < 6.11)//frequency_KHz == 6.1
217     //   {
218     //     TIM4-> CCR4 = 333; // (1666*20/100); 20% dutyc cycle
219     //   }
220     //
221     //   if (6.29 < frequency_KHz && frequency_KHz < 6.31)//frequency_KHz == 6.3
222     //   {
223     //     TIM4-> CCR4 = 166; // (1666*10/100); 10% dutyc cycle
224     //   }
225     //
226     //   if (frequency_KHz > 6.31) // end position detected; PWM signal should be turned off
227     //   {
228     //     count4 = 2;
229     //     TIM4-> CCR4 = 0; // (1666*0/100); 0% dutyc cycle
230     //
231     //     //HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4); // Stop PWM
232     //     //HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer
233     //
234     //   }
235       }
236     //   if (4.6999 < frequency_KHz && frequency_KHz < 4.7001) //frequency_KHz == 4.7
237     //   {
238     //     TIM4-> CCR4 = 1499; // (1666*90/100); 90% dutyc cycle
239     //   }
240     //
241     //   if (4.8999 < frequency_KHz && frequency_KHz < 4.9001)//frequency_KHz == 4.9
242     //   {
243     //     TIM4-> CCR4 = 1332; // (1666*80/100); 80% dutyc cycle
244     //   }
245     //
246     //   if (5.0999< frequency_KHz && frequency_KHz < 5.1001)//frequency_KHz == 5.1
247     //   {
248     //     TIM4-> CCR4 = 1166; // (1666*70/100); 70% dutyc cycle
249     //   }
250     //
251     //   if (5.2999 < frequency_KHz && frequency_KHz < 5.3001)//frequency_KHz == 5.3
252     //   {
253     //     TIM4-> CCR4 = 999; // (1666*60/100); 60% dutyc cycle
254     //   }
255     //
256     //   if (5.4999 < frequency_KHz && frequency_KHz < 5.5001)//frequency_KHz == 5.5
257     //   {
258     //     TIM4-> CCR4 = 833; // (1666*50/100); 50% dutyc cycle
259     //   }
260     //
261     //   if (5.6999 < frequency_KHz && frequency_KHz < 5.7001)//frequency_KHz == 5.7
262     //   {
263     //     TIM4-> CCR4 = 666; // (1666*40/100); 40% dutyc cycle
264     //   }
265     //
266     //   if (5.8999 < frequency_KHz && frequency_KHz < 5.9001)//frequency_KHz == 5.9
267     //   {
268     //     TIM4-> CCR4 = 499; // (1666*30/100); 30% dutyc cycle
```

```
269         //   }
270         //
271         //   if (6.0999 < frequency_KHz && frequency_KHz < 6.1001)//frequency_KHz == 6.1
272         //   {
273         //     TIM4-> CCR4 = 333; // (1666*20/100); 20% dutyc cycle
274         //   }
275         //
276         //   if (6.2999 < frequency_KHz && frequency_KHz < 6.3001)//frequency_KHz == 6.3
277         //   {
278         //     TIM4-> CCR4 = 166; // (1666*10/100); 10% dutyc cycle
279         //   }
280         //
281         //   if (frequency_KHz > 6.3001) // end position detected; PWM signal should be turned off
282         //   {
283         //     HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4); // Stop PWM
284         //     HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer
285         //     count3 = 2;
286         //   }
287         // }
288           /* USER CODE END 3 */
289         }
290
291         /**
292           * @brief System Clock Configuration
293           * @retval None
294           */
295         void SystemClock_Config(void)
296         {
297           RCC_OscInitTypeDef RCC_OscInitStruct = {0};
298           RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
299
300           /** Configure the main internal regulator output voltage
301           */
302           __HAL_RCC_PWR_CLK_ENABLE();
303           __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);
304           /** Initializes the RCC Oscillators according to the specified parameters
305           * in the RCC_OscInitTypeDef structure.
306           */
307           RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_HSE;
308           RCC_OscInitStruct.HSEState = RCC_HSE_ON;
309           RCC_OscInitStruct.LSIState = RCC_LSI_ON;
310           RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
311           RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
312           RCC_OscInitStruct.PLL.PLLM = 4;
313           RCC_OscInitStruct.PLL.PLLN = 100;
314           RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
315           RCC_OscInitStruct.PLL.PLLQ = 4;
316           if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
317           {
318             Error_Handler();
319           }
320           /** Initializes the CPU, AHB and APB buses clocks
321           */
322           RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
323                                         |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
324           RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
325           RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
326           RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
327           RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
328
329           if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
330           {
331             Error_Handler();
332           }
333         }
334
335         /**
336           * @brief TIM3 Initialization Function
```

```
337         * @param None
338         * @retval None
339         */
340        static void MX_TIM3_Init(void)
341        {
342
343         /* USER CODE BEGIN TIM3_Init 0 */
344
345         /* USER CODE END TIM3_Init 0 */
346
347         TIM_ClockConfigTypeDef sClockSourceConfig = {0};
348         TIM_MasterConfigTypeDef sMasterConfig = {0};
349         TIM_OC_InitTypeDef sConfigOC = {0};
350
351         /* USER CODE BEGIN TIM3_Init 1 */
352
353         /* USER CODE END TIM3_Init 1 */
354         htim3.Instance = TIM3;
355         htim3.Init.Prescaler = 0;
356         htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
357         htim3.Init.Period = 0xffffffff;
358         htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
359         htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
360         if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
361         {
362           Error_Handler();
363         }
364         sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
365         if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
366         {
367           Error_Handler();
368         }
369         if (HAL_TIM_OC_Init(&htim3) != HAL_OK)
370         {
371           Error_Handler();
372         }
373         sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
374         sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
375         if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
376         {
377           Error_Handler();
378         }
379         sConfigOC.OCMode = TIM_OCMODE_TOGGLE;
380         sConfigOC.Pulse = 0;
381         sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
382         sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
383         if (HAL_TIM_OC_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
384         {
385           Error_Handler();
386         }
387         /* USER CODE BEGIN TIM3_Init 2 */
388
389         /* USER CODE END TIM3_Init 2 */
390         HAL_TIM_MspPostInit(&htim3);
391
392        }
393
394        /**
395         * @brief TIM4 Initialization Function
396         * @param None
397         * @retval None
398         */
399        static void MX_TIM4_Init(void)
400        {
401
402         /* USER CODE BEGIN TIM4_Init 0 */
403
404         /* USER CODE END TIM4_Init 0 */
```

```
405
406         TIM_ClockConfigTypeDef sClockSourceConfig = {0};
407         TIM_MasterConfigTypeDef sMasterConfig = {0};
408         TIM_OC_InitTypeDef sConfigOC = {0};
409
410         /* USER CODE BEGIN TIM4_Init 1 */
411
412         /* USER CODE END TIM4_Init 1 */
413         htim4.Instance = TIM4;
414         htim4.Init.Prescaler = 0;
415         htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
416         htim4.Init.Period = 0xffffffff;
417         htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
418         htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
419         if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
420         {
421           Error_Handler();
422         }
423         sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
424         if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
425         {
426           Error_Handler();
427         }
428         if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
429         {
430           Error_Handler();
431         }
432         sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
433         sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
434         if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
435         {
436           Error_Handler();
437         }
438         sConfigOC.OCMode = TIM_OCMODE_PWM1;
439         sConfigOC.Pulse = 0;
440         sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
441         sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
442         if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
443         {
444           Error_Handler();
445         }
446         /* USER CODE BEGIN TIM4_Init 2 */
447
448         /* USER CODE END TIM4_Init 2 */
449         HAL_TIM_MspPostInit(&htim4);
450
451       }
452
453       /**
454         * @brief TIM5 Initialization Function
455         * @param None
456         * @retval None
457         */
458       static void MX_TIM5_Init(void)
459       {
460
461         /* USER CODE BEGIN TIM5_Init 0 */
462
463         /* USER CODE END TIM5_Init 0 */
464
465         TIM_ClockConfigTypeDef sClockSourceConfig = {0};
466         TIM_MasterConfigTypeDef sMasterConfig = {0};
467         TIM_IC_InitTypeDef sConfigIC = {0};
468
469         /* USER CODE BEGIN TIM5_Init 1 */
470
471         /* USER CODE END TIM5_Init 1 */
472         htim5.Instance = TIM5;
```

```
473         htim5.Init.Prescaler = 0;
474         htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
475         htim5.Init.Period = 0xffffffff;
476         htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
477         htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
478         if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
479         {
480           Error_Handler();
481         }
482         sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
483         if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
484         {
485           Error_Handler();
486         }
487         if (HAL_TIM_IC_Init(&htim5) != HAL_OK)
488         {
489           Error_Handler();
490         }
491         sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
492         sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
493         if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) != HAL_OK)
494         {
495           Error_Handler();
496         }
497         sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
498         sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
499         sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
500         sConfigIC.ICFilter = 0;
501         if (HAL_TIM_IC_ConfigChannel(&htim5, &sConfigIC, TIM_CHANNEL_4) != HAL_OK)
502         {
503           Error_Handler();
504         }
505         /* USER CODE BEGIN TIM5_Init 2 */
506
507         /* USER CODE END TIM5_Init 2 */
508
509       }
510
511       /**
512         * @brief GPIO Initialization Function
513         * @param None
514         * @retval None
515         */
516       static void MX_GPIO_Init(void)
517       {
518         GPIO_InitTypeDef GPIO_InitStruct = {0};
519
520         /* GPIO Ports Clock Enable */
521         __HAL_RCC_GPIOH_CLK_ENABLE();
522         __HAL_RCC_GPIOA_CLK_ENABLE();
523         __HAL_RCC_GPIOB_CLK_ENABLE();
524         __HAL_RCC_GPIOE_CLK_ENABLE();
525         __HAL_RCC_GPIOD_CLK_ENABLE();
526         __HAL_RCC_GPIOG_CLK_ENABLE();
527
528         /*Configure GPIO pin Output Level */
529         HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13|GPIO_PIN_14, GPIO_PIN_RESET);
530
531         /*Configure GPIO pins : PE7 PE8 */
532         GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_8;
533         GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
534         GPIO_InitStruct.Pull = GPIO_NOPULL;
535         HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
536
537         /*Configure GPIO pins : PG13 PG14 */
538         GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14;
539         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
540         GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
  541         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  542         HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
  543
  544         /* EXTI interrupt init*/
  545         HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
  546         HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
  547
  548       }
  549
  550       /* USER CODE BEGIN 4 */
  551
  552       /* USER CODE END 4 */
  553
  554       /**
  555         * @brief  This function is executed in case of error occurrence.
  556         * @retval None
  557         */
  558       void Error_Handler(void)
  559       {
  560         /* USER CODE BEGIN Error_Handler_Debug */
  561         /* User can add his own implementation to report the HAL error return state */
  562         __disable_irq();
  563         while (1)
  564         {
  565         }
  566         /* USER CODE END Error_Handler_Debug */
  567       }
  568
  569       #ifdef  USE_FULL_ASSERT
  570       /**
  571         * @brief  Reports the name of the source file and the source line number
  572         *         where the assert_param error has occurred.
  573         * @param  file: pointer to the source file name
  574         * @param  line: assert_param error line source number
  575         * @retval None
  576         */
  577       void assert_failed(uint8_t *file, uint32_t line)
  578       {
  579         /* USER CODE BEGIN 6 */
  580         /* User can add his own implementation to report the file name and line number,
  581            ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  582         /* USER CODE END 6 */
  583       }
  584       #endif /* USE_FULL_ASSERT */
  585

Maximum stack usage in bytes:

.cstack Function
------- --------
    0   Error_Handler
   80   SystemClock_Config
     80   -> HAL_RCC_ClockConfig
     80   -> HAL_RCC_OscConfig
     80   -> memset
   80   main
     80   -> Error_Handler
     80   -> HAL_GPIO_Init
     80   -> HAL_GPIO_WritePin
     80   -> HAL_Init
     80   -> HAL_NVIC_EnableIRQ
     80   -> HAL_NVIC_SetPriority
     80   -> HAL_TIMEx_MasterConfigSynchronization
     80   -> HAL_TIM_Base_Init
     80   -> HAL_TIM_ConfigClockSource
     80   -> HAL_TIM_IC_ConfigChannel
     80   -> HAL_TIM_IC_Init
     80   -> HAL_TIM_MspPostInit
```

```
      80   -> HAL_TIM_OC_ConfigChannel
      80   -> HAL_TIM_OC_Init
      80   -> HAL_TIM_OC_Start_IT
      80   -> HAL_TIM_PWM_ConfigChannel
      80   -> HAL_TIM_PWM_Init
      80   -> SystemClock_Config
      80   -> memset


  Section sizes:

  Bytes  Function/Label
  -----  --------------
     4  ??DataTable1
     4  ??DataTable1_1
     4  ??DataTable1_2
     4  ??DataTable1_3
     4  ??DataTable1_4
     4  ??DataTable1_5
     4  ??DataTable1_6
     4  ??DataTable1_7
     4  ??DataTable1_8
    16  ?Subroutine0
    10  ?Subroutine1
    10  ?Subroutine2
    10  ?Subroutine3
    10  ?Subroutine4
    10  ?Subroutine5
     4  Error_Handler
     4  Hi_pulsewidth
     4  Hi_pulsewidth11
     4  Low_pulsewidth
   162  SystemClock_Config
     4  count3
     4  dutyCycle
     4  frequency_KHz
   232  htim3
        htim4
        htim5
        frequency
        freq
        count4
        halfSeconds
   794  main
     4  timePeriod


   260 bytes in section .bss
 1'062 bytes in section .text

 1'062 bytes of CODE memory
   260 bytes of DATA memory

Errors: none
Warnings: none
```

stm32f4xx_it.lst

```
#################################################################################
#
# IAR ANSI C/C++ Compiler V9.20.2.320/W64 for ARM         10/Apr/2022  12:55:41
# Copyright 1999-2021 IAR Systems AB.
#
#    Cpu mode         =  thumb
#    Endian           =  little
```

```
#   Source file      =
#      C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\Core\Src\stm32f4xx_it.c
#   Command line     =
#      -f "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\stm32f4xx_it.o.rsp"
#      ("C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\Core\Src\stm32f4xx_it.c"
#      -D USE_HAL_DRIVER -D STM32F429xx -lcN "C:\Users\Student\OneDrive -
#      Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\List\Application\User\Core"
#      -o "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core"
#      --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
#      --dlib_config "C:\Program Files\IAR Systems\Embedded Workbench
#      9.0\arm\inc\c\DLib_Config_Full.h" -I "C:\Users\Student\OneDrive -
#      Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM/../Core/Inc\\" -I
#      "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc\\"
#      -I "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\\"
#      -I "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\\"
#      -I "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM/../Drivers/CMSIS/Include\\"
#      -Ohz) --dependencies=n "C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\stm32f4xx_it.o.d"
#   Locale           = C
#   List file        =
#      C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\List\Application\User\Core\stm32f4xx_it.lst
#   Object file      =
#      C:\Users\Student\OneDrive - Western Michigan
#      University\Documents\MIcrocontroller\Project2\EWARM\Project2\Obj\Application\User\Core\stm32f4xx_it.o
#   Runtime model:
#     __CPP_Runtime   = 1
#     __SystemLibrary = DLib
#     __dlib_version  = 6
#     __size_limit    = 32768|ARM.EW.LINKER
#
###############################################################################

C:\Users\Student\OneDrive - Western Michigan University\Documents\MIcrocontroller\Project2\Core\Src\stm32f4xx_it.c
      1          /* USER CODE BEGIN Header */
      2         /**
      3          ******************************************************************************
      4          * @file    stm32f4xx_it.c
      5          * @brief   Interrupt Service Routines.
      6          ******************************************************************************
      7          * @attention
      8          *
      9          * Copyright (c) 2022 STMicroelectronics.
     10           * All rights reserved.
     11           *
     12           * This software is licensed under terms that can be found in the LICENSE file
     13           * in the root directory of this software component.
     14           * If no LICENSE file comes with this software, it is provided AS-IS.
     15           *
     16           ******************************************************************************
     17           */
     18         /* USER CODE END Header */
     19
     20         /* Includes ------------------------------------------------------------------*/
     21         #include "main.h"
     22         #include "stm32f4xx_it.h"
     23         /* Private includes ----------------------------------------------------------*/
     24         /* USER CODE BEGIN Includes */
```

```
25      /* USER CODE END Includes */
26
27      /* Private typedef -----------------------------------------------------------*/
28      /* USER CODE BEGIN TD */
29
30      /* USER CODE END TD */
31
32      /* Private define ------------------------------------------------------------*/
33      /* USER CODE BEGIN PD */
34
35      /* USER CODE END PD */
36
37      /* Private macro -------------------------------------------------------------*/
38      /* USER CODE BEGIN PM */
39
40      /* USER CODE END PM */
41
42      /* Private variables ---------------------------------------------------------*/
43      /* USER CODE BEGIN PV */
44
45      //EXTI
46      extern int halfSeconds;
47
48      //TIM3
49      int COUNT = 0;
50      int count2 = 0;
51      extern uint8_t count3;
52      extern uint8_t count4;
53
54      int count = 0;
55      uint32_t risingedge1 = 0x0000;
56      uint32_t fallingedge = 0x0000;
57      uint32_t risingedge2 = 0x0000;
58      extern float Hi_pulsewidth ;
59      extern float Low_pulsewidth ;
60      extern float frequency;
61      extern float dutyCycle;
62      extern float timePeriod;
63      extern float frequency_KHz;
64
65      //TIM5
66      int count1s = 0;
67      int count1s2 = 0;
68      //uint8_t count2 = 0x00;
69      /* USER CODE END PV */
70
71      /* Private function prototypes -----------------------------------------------*/
72      /* USER CODE BEGIN PFP */
73
74      /* USER CODE END PFP */
75
76      /* Private user code ---------------------------------------------------------*/
77      /* USER CODE BEGIN 0 */
78
79      /* USER CODE END 0 */
80
81      /* External variables --------------------------------------------------------*/
82      extern TIM_HandleTypeDef htim3;
83      extern TIM_HandleTypeDef htim4;
84      extern TIM_HandleTypeDef htim5;
85      /* USER CODE BEGIN EV */
86
87      /* USER CODE END EV */
88
89      /******************************************************************************/
90      /*           Cortex-M4 Processor Interruption and Exception Handlers          */
91      /******************************************************************************/
92      /**
```

```
93        * @brief This function handles Non maskable interrupt.
94        */
95       void NMI_Handler(void)
96       {
97        /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
98
99        /* USER CODE END NonMaskableInt_IRQn 0 */
100        /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
101        while (1)
102        {
103        }
104        /* USER CODE END NonMaskableInt_IRQn 1 */
105       }
106
107       /**
108        * @brief This function handles Hard fault interrupt.
109        */
110       void HardFault_Handler(void)
111       {
112        /* USER CODE BEGIN HardFault_IRQn 0 */
113
114        /* USER CODE END HardFault_IRQn 0 */
115        while (1)
116        {
117          /* USER CODE BEGIN W1_HardFault_IRQn 0 */
118          /* USER CODE END W1_HardFault_IRQn 0 */
119        }
120       }
121
122       /**
123        * @brief This function handles Memory management fault.
124        */
125       void MemManage_Handler(void)
126       {
127        /* USER CODE BEGIN MemoryManagement_IRQn 0 */
128
129        /* USER CODE END MemoryManagement_IRQn 0 */
130        while (1)
131        {
132          /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
133          /* USER CODE END W1_MemoryManagement_IRQn 0 */
134        }
135       }
136
137       /**
138        * @brief This function handles Pre-fetch fault, memory access fault.
139        */
140       void BusFault_Handler(void)
141       {
142        /* USER CODE BEGIN BusFault_IRQn 0 */
143
144        /* USER CODE END BusFault_IRQn 0 */
145        while (1)
146        {
147          /* USER CODE BEGIN W1_BusFault_IRQn 0 */
148          /* USER CODE END W1_BusFault_IRQn 0 */
149        }
150       }
151
152       /**
153        * @brief This function handles Undefined instruction or illegal state.
154        */
155       void UsageFault_Handler(void)
156       {
157        /* USER CODE BEGIN UsageFault_IRQn 0 */
158
159        /* USER CODE END UsageFault_IRQn 0 */
160        while (1)
```

```
161         {
162           /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
163           /* USER CODE END W1_UsageFault_IRQn 0 */
164         }
165       }
166
167       /**
168         * @brief This function handles System service call via SWI instruction.
169         */
170       void SVC_Handler(void)
171       {
172         /* USER CODE BEGIN SVCall_IRQn 0 */
173
174         /* USER CODE END SVCall_IRQn 0 */
175         /* USER CODE BEGIN SVCall_IRQn 1 */
176
177         /* USER CODE END SVCall_IRQn 1 */
178       }
179
180       /**
181         * @brief This function handles Debug monitor.
182         */
183       void DebugMon_Handler(void)
184       {
185         /* USER CODE BEGIN DebugMonitor_IRQn 0 */
186
187         /* USER CODE END DebugMonitor_IRQn 0 */
188         /* USER CODE BEGIN DebugMonitor_IRQn 1 */
189
190         /* USER CODE END DebugMonitor_IRQn 1 */
191       }
192
193       /**
194         * @brief This function handles Pendable request for system service.
195         */
196       void PendSV_Handler(void)
197       {
198         /* USER CODE BEGIN PendSV_IRQn 0 */
199
200         /* USER CODE END PendSV_IRQn 0 */
201         /* USER CODE BEGIN PendSV_IRQn 1 */
202
203         /* USER CODE END PendSV_IRQn 1 */
204       }
205
206       /**
207         * @brief This function handles System tick timer.
208         */
209       void SysTick_Handler(void)
210       {
211         /* USER CODE BEGIN SysTick_IRQn 0 */
212
213         /* USER CODE END SysTick_IRQn 0 */
214         HAL_IncTick();
215         /* USER CODE BEGIN SysTick_IRQn 1 */
216
217         /* USER CODE END SysTick_IRQn 1 */
218       }
219
220       /******************************************************************************/
221       /* STM32F4xx Peripheral Interrupt Handlers                                    */
222       /* Add here the Interrupt Handlers for the used peripherals.                  */
223       /* For the available peripheral interrupt handler names,                      */
224       /* please refer to the startup file (startup_stm32f4xx.s).                    */
225       /******************************************************************************/
226
227       /**
228         * @brief This function handles EXTI line[9:5] interrupts.
```

```
229          */
230         void EXTI9_5_IRQHandler(void)
231         {
232           /* USER CODE BEGIN EXTI9_5_IRQn 0 */
233           if ((!(GPIOE->IDR & 0x80)) && (halfSeconds == 0)) // Falling Edge detection, start
234           {
235             HAL_TIM_OC_Start_IT(&htim3, TIM_CHANNEL_4); // Start Buzzer and indicator
236             //HAL_TIM_Base_Start_IT(&htim5, TIM_CHANNEL_2);// start timer for Green indicator
237            //HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_4); //Starts the TIM Input Capture measurement in interrupt mode.
238           }
239
240           if (!(GPIOE->IDR & 0x100)) // Falling Edge detection on Stop pin; wait for start
241           {
242             halfSeconds = 0;
243             HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_4);// stop PWM
244             HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);//TURN Off Red LED (IR Emitter LED)
245           }
246           else
247           {
248             HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_4); //Starts the TIM Input Capture measurement in interrupt mode.
249           }
250           /* USER CODE END EXTI9_5_IRQn 0 */
251           HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_7);
252           HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
253           /* USER CODE BEGIN EXTI9_5_IRQn 1 */
254
255           /* USER CODE END EXTI9_5_IRQn 1 */
256         }
257
258         /**
259           * @brief This function handles TIM3 global interrupt.
260           */
261         void TIM3_IRQHandler(void)
262         {
263           /* USER CODE BEGIN TIM3_IRQn 0 */
264           if (halfSeconds < 12)
265           {
266             if (count2 == 0)
267             {
268               TIM3->CCR4 += 4545;// Tcnt for one periode = (1/f) * fapb1_clock = (1/5.5kHz)* 50MHz = 9090
269               count2 = 1;
270             }
271             else if (count2 == 1)
272             {
273             TIM3->CCR4 += (9090-4545);// 50% duty cycle:(50/100)*9090=4545
274             count2 = 0;
275             count1s += 1;
276             }
277             count3= 1;
278           }
279           if ((count1s == 2750) && (count3 != 2)) // To mak 1 halfSeconds, 1/5.5KHz =0.1818ms; 0.1818*X/2 = 1000ms; x =
5500/2=2750
280           //else if ((count1s == 2750) && (halfSeconds <= 12)) // To mak 1 halfSeconds, 1/5.5KHz =0.1818ms; 0.1818*X/2 = 1000ms; x
= 5500/2=2750
281           {
282             count1s = 0;
283             HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13); //Toggle GREEN LED
284             halfSeconds +=1;
285
286           }
287
288           //////////////////////////////////////////////////////
289           //if (((GPIOE->IDR & 0x80)) && (halfSeconds == 12))// Falling Edge is not detected on start and after 12 halfSeconds
290           if (((count3== 1)) && (halfSeconds == 12))// Falling Edge is not detected on start and after 12 halfSeconds
291           //if (count3 == 1)
292           {
293             HAL_TIM_OC_Stop_IT(&htim3, TIM_CHANNEL_4); // Turn off Buzzer
294             HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);//TURN ON Red LED (IR Emitter LED)
```

```
295
296             //PWM signal driving the motor on
297             HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_4);// to make a pulse with f= 30KHz and duty cycle=50%  0.016666
ms width and 0.03333 ms period.
298             TIM4-> ARR = 1666; // (1/30KHz)*50MHz=1666
299             TIM4-> CCR4 = 833; // (1/2*30KHz)*50MHz=833; 50% dutyc cycle
300             count3= 2;
301           }
302         ////////////////////////////////////////////////////////////
303
304         //  if ((TIM4->CCR4 == 0) && ( 12 <= halfSeconds && halfSeconds <= 32))
305           //if ((TIM4->CCR4 == 0) && (halfSeconds <= 32))
306           //count1s2 = 0;
307           if (TIM4->CCR4 == 0)
308           {
309             if (count2 == 0)
310             {
311               TIM3->CCR4 += 7143;// Tcnt for one periode = (1/f) * fapb1_clock = (1/3.5kHz)* 50MHz = 14286
312               count2 = 1;
313             }
314             else if (count2 == 1)
315             {
316             TIM3->CCR4 += (14286-7143);// 50% duty cycle:(50/100)*14286=7143
317             count3 = 0;
318             count1s2 += 1;
319             }
320           }
321           if((count1s2 == 1375) && (halfSeconds <= 32)) // To make 1 halfSeconds, 1/3.5KHz =0.2857ms; 0.2857*2X = 1000ms; x =
3500/4 for 2 blinks/sec
322             {
323               count1s2 = 0;
324               HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13); //Toggle GREEN LED
325               halfSeconds +=1;
326             }
327
328           else if ( halfSeconds > 32)
329           {
330            HAL_TIM_OC_Stop_IT(&htim3, TIM_CHANNEL_4); // Stop Buzzer Start
331            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);//TURN Off Green LED (Indicator LED)
332            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_RESET);//TURN Off Red LED (IR Emitter LED)
333            halfSeconds = 0;
334            count2 = 0;
335            count3 = 0;
336           }
337           /* USER CODE END TIM3_IRQn 0 */
338           HAL_TIM_IRQHandler(&htim3);
339           /* USER CODE BEGIN TIM3_IRQn 1 */
340
341
342           /* USER CODE END TIM3_IRQn 1 */
343         }
344
345         /**
346           * @brief This function handles TIM4 global interrupt.
347           */
348         void TIM4_IRQHandler(void)
349         {
350           /* USER CODE BEGIN TIM4_IRQn 0 */
351
352           /* USER CODE END TIM4_IRQn 0 */
353           HAL_TIM_IRQHandler(&htim4);
354           /* USER CODE BEGIN TIM4_IRQn 1 */
355
356           /* USER CODE END TIM4_IRQn 1 */
357         }
358
359         /**
360           * @brief This function handles TIM5 global interrupt.
```

```
361          */
362         void TIM5_IRQHandler(void)
363         {
364           /* USER CODE BEGIN TIM5_IRQn 0 */
365         if ((GPIOA->IDR & 0x0008) && (count == 0)) // Rising-1 edge detection
366           {
367             risingedge1 = TIM5->CCR4;
368             count = 1;
369           }
370           else if ((!(GPIOA->IDR & 0x0008)) && (count == 1))// Falling Edge detection
371           {
372             fallingedge = TIM5->CCR4;
373             count = 2;
374           }
375         else if ((GPIOA->IDR & 0x0008) && (count == 2)) //Rising-2 edge detection
376           {
377             risingedge2 = TIM5->CCR4;
378             timePeriod = ((risingedge2 - risingedge1)*0.02) ; // HLCK=100MHz  and APB1=50MHz. Period (us).1/50MHz = 0.02us
379             frequency = (1000000/timePeriod); // frequency (HZ)=1/timePeriod(us)= 10*6/timePeriod
380             Hi_pulsewidth = ((fallingedge - risingedge1)* 0.02); // Hi_Pulsewidth (ms).
381
382             Low_pulsewidth = ((risingedge2 - fallingedge) * 0.02); // Low_pulsewidth (ms).
383             //printf("Period = %lf msec\n", timePeriod);
384             dutyCycle = (Hi_pulsewidth/timePeriod)*100;
385
386
387             count = 0;
388             TIM5->CNT = 0; // Reset the counter register
389
390             frequency_KHz = frequency/1000;
391           }
392
393
394           /* USER CODE END TIM5_IRQn 0 */
395           HAL_TIM_IRQHandler(&htim5);
396           /* USER CODE BEGIN TIM5_IRQn 1 */
397
398           /* USER CODE END TIM5_IRQn 1 */
399         }
400
401         /* USER CODE BEGIN 1 */
402
403         /* USER CODE END 1 */
404
405
406
407

Maximum stack usage in bytes:

.cstack Function
------- --------
    0   BusFault_Handler
    0   DebugMon_Handler
   16   EXTI9_5_IRQHandler
      0   -> HAL_GPIO_EXTI_IRQHandler
     16   -> HAL_GPIO_EXTI_IRQHandler
     16   -> HAL_GPIO_WritePin
     16   -> HAL_TIM_IC_Start_IT
     16   -> HAL_TIM_OC_Start_IT
     16   -> HAL_TIM_PWM_Stop_IT
    0   HardFault_Handler
    0   MemManage_Handler
    0   NMI_Handler
    0   PendSV_Handler
    0   SVC_Handler
    0   SysTick_Handler
      0   -> HAL_IncTick
```

```
  32   TIM3_IRQHandler
    32   -> HAL_GPIO_TogglePin
    32   -> HAL_GPIO_WritePin
     0   -> HAL_TIM_IRQHandler
    32   -> HAL_TIM_OC_Stop_IT
    32   -> HAL_TIM_PWM_Start_IT
   0   TIM4_IRQHandler
     0   -> HAL_TIM_IRQHandler
  48   TIM5_IRQHandler
     0   -> HAL_TIM_IRQHandler
    48   -> __aeabi_d2f
    48   -> __aeabi_dmul
    48   -> __aeabi_ui2d
   0   UsageFault_Handler


Section sizes:

Bytes  Function/Label
-----  --------------
   4  ??DataTable4
   4  ??DataTable4_1
   4  ??DataTable4_10
   4  ??DataTable4_11
   8  ??DataTable4_12
   4  ??DataTable4_13
   4  ??DataTable4_14
   4  ??DataTable4_15
   4  ??DataTable4_16
   4  ??DataTable4_17
   4  ??DataTable4_18
   4  ??DataTable4_19
   4  ??DataTable4_2
   4  ??DataTable4_20
   4  ??DataTable4_21
   4  ??DataTable4_22
   4  ??DataTable4_3
   4  ??DataTable4_4
   4  ??DataTable4_5
   4  ??DataTable4_6
   4  ??DataTable4_7
   4  ??DataTable4_8
   4  ??DataTable4_9
  10  ?Subroutine0
  10  ?Subroutine1
  12  ?Subroutine2
   8  ?Subroutine3
   2  BusFault_Handler
   4  COUNT
   2  DebugMon_Handler
  82  EXTI9_5_IRQHandler
   2  HardFault_Handler
   2  MemManage_Handler
   2  NMI_Handler
   2  PendSV_Handler
   2  SVC_Handler
   4  SysTick_Handler
 270  TIM3_IRQHandler
   6  TIM4_IRQHandler
 214  TIM5_IRQHandler
   2  UsageFault_Handler
  16  count
      risingedge1
      fallingedge
      risingedge2
  12  count2
      count1s
      count1s2
```

```
 32 bytes in section .bss
728 bytes in section .text

728 bytes of CODE memory
 32 bytes of DATA memory

Errors: none
Warnings: none
```