Supplementary Document for Experiment No: 04

- **Note: Green Color Highlighted Portion in the Given Codes is New Addition part**

Part 4A: DC Motor
Complete code for part 4A starts

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file         : main.c
 * @brief        : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"


/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */


/* USER CODE END Includes */


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

```
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim3;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM3_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();
 /* USER CODE BEGIN 2 */
 HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
```

```
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
          TIM3->CCR1=14000;
          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7, 0);
          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5, 1);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** 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.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
```

```
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  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_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != 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_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 = 65535;
htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_PWM_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_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
  Error_Handler();
```

```
  }
  /* USER CODE BEGIN TIM3_Init 2 */

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

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pins : PA5 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

```
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* 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 */
```

Complete Code for Part 4A Ends

Part 4B: Stepper Motor
Complete code for part 4B starts

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

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

```
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

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

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define IN1_PIN GPIO_PIN_6
#define IN1_PORT GPIOA
#define IN2_PIN GPIO_PIN_5
#define IN2_PORT GPIOA
#define IN3_PIN GPIO_PIN_4
```

```
#define IN3_PORT GPIOA
#define IN4_PIN GPIO_PIN_3
#define IN4_PORT GPIOA

void stepCCV (int steps, uint16_t delay) // CCV - Counter Clockwise
{
   for(int x=0; x<steps; x=x+1)
   {
            // Only Coil 4 is ON
                  HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
                  HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
                  HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);   // IN3
                  HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
                  HAL_Delay(delay);
                  // Coil 3 & Coil 4 is ON
                  HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
                  HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
                  HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET);   // IN3
                  HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET); // IN4
                  HAL_Delay(delay);
                  // Only Coil 3 is ON
                  HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
                  HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
                  HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET); // IN3
                  HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
                  HAL_Delay(delay);
                  // Coil 2 & Coil 3 is ON
                  HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
                  HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
                  HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET);   // IN3
                  HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
                  HAL_Delay(delay);
```

```
                    // Only Coil 2 is ON
                    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
                    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
                    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);   // IN3
                    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET); // IN4
                    HAL_Delay(delay);
                    // Coil 1 & Coil 2 is ON
                    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET);   // IN1
                    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
                    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
                    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
                    HAL_Delay(delay);
                    // Only Coil 1 is ON
                    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET);   // IN1
                    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
                    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
                    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
                    HAL_Delay(delay);
                    // Coil 1 & Coil 4 is ON
                    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET); // IN1
                    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
                    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
                    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
                    HAL_Delay(delay);
    }
}


void stepCV (int steps, uint16_t delay) // CV - Clockwise
{
   for(int x=0; x<steps; x=x+1)
   {
       HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET); // IN1
```

```
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);   // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
        HAL_Delay(delay);
        // Coil 1 & Coil 2 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET); // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);   // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET); // IN4
        // Only Coil 2 is ON
        HAL_Delay(delay);
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
        HAL_Delay(delay);
        // Coil 2 & Coil 3 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET);   // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);   // IN4
        HAL_Delay(delay);
        // Only Coil 3 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET); // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET);   // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET); // IN4
        HAL_Delay(delay);
        // Coil 2 & Coil 3 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);   // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET); // IN3
```

```
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
        HAL_Delay(delay);
        // Only Coil 4 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);   // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
        HAL_Delay(delay);
        // Coil 1 & Coil 4 is ON
        HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET); // IN1
        HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
        HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
        HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
        HAL_Delay(delay);

    }
}

/* 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();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */

// initialization of rotor angle to zero
 HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET);   // IN1
  HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET); // IN2
  HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET); // IN3
  HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);   // IN4
  HAL_Delay(500);

  while (1)
  {
```

```
            stepCCV(280, 2);
            HAL_Delay(100);
            stepCV(280, 2);
            HAL_Delay(100);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** 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_DIV2;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  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_DIV1;

 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
 {
  Error_Handler();
 }
}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

 /* GPIO Ports Clock Enable */
 __HAL_RCC_GPIOD_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6,
GPIO_PIN_RESET);

  /*Configure GPIO pins : PA3 PA4 PA5 PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* 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 */
```

Complete Code for Part 4B Ends

Part 4C: Servo Motor
Complete code for part 4C starts

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file         : main.c
 * @brief        : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

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

```
static void MX_TIM2_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_TIM2_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
            TIM2->CCR1=1000;
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** 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.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  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_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
```

```
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 40;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 36000;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &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(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
```

```
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* 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 */
```

Complete Code for Part 4C Ends