MCI lab # 05 Report

Ali Hussain Zaidi (sz10384)

Haider Zaidi (mz10270)

Task # 1

Code

```c
// PWM Parameters
#define PWM_MAX_VALUE    7999    // Maximum PWM value (ARR value)
#define PWM_MIN_VALUE    0       // Minimum PWM value
#define FADE_STEP        50      // Step size for fading (adjust for smooth fade)
#define FADE_DELAY_MS    5       // Delay between steps in milliseconds

// LED Configuration
#define LED_CHANNEL_RED    TIM_CHANNEL_1  // PA6
#define LED_CHANNEL_GREEN  TIM_CHANNEL_2  // PA7
#define LED_CHANNEL_BLUE   TIM_CHANNEL_3  // PB0
```

```c
void LED_PWM_Init(void)
{
    // Start PWM on Channel 1 (Red LED on PA6)
    HAL_TIM_PWM_Start(&htim3, LED_CHANNEL_RED);

    // Optional: Start additional LED channels for RGB control
    HAL_TIM_PWM_Start(&htim3, LED_CHANNEL_GREEN);
    HAL_TIM_PWM_Start(&htim3, LED_CHANNEL_BLUE);

    // Initialize with minimum brightness
    LED_SetBrightness(PWM_MIN_VALUE);
}
```

```c
void LED_SetBrightness(uint16_t value)
{
    // Clamp value to valid range
    if (value > PWM_MAX_VALUE)
        value = PWM_MAX_VALUE;

    // Update the PWM compare register for Channel 1
    __HAL_TIM_SET_COMPARE(&htim3, LED_CHANNEL_RED, value);
}
```

```c
void LED_Fade_Polling(void)
{
    // Fade IN: Increase brightness from 0% to 100%
    for (uint16_t pwm = PWM_MIN_VALUE; pwm <= PWM_MAX_VALUE; pwm += FADE_STEP)
    {
        LED_SetBrightness(pwm);
        HAL_Delay(FADE_DELAY_MS);
    }

    // Ensure we reach maximum
    LED_SetBrightness(PWM_MAX_VALUE);
    HAL_Delay(100);  // Hold at max brightness for 100ms

    // Fade OUT: Decrease brightness from 100% to 0%
    for (uint16_t pwm = PWM_MAX_VALUE; pwm >= PWM_MIN_VALUE; pwm -= FADE_STEP)
    {
        LED_SetBrightness(pwm);
        HAL_Delay(FADE_DELAY_MS);
    }

    // Ensure we reach minimum
    LED_SetBrightness(PWM_MIN_VALUE);
    HAL_Delay(100);  // Hold at min brightness for 100ms
}
```

Output:

Task # 02

Code:

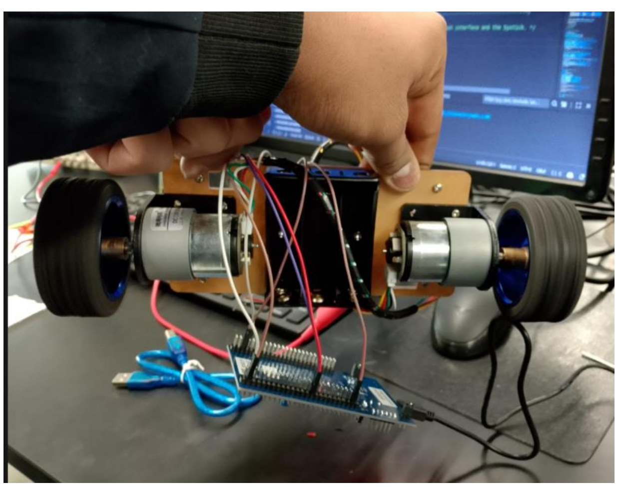
```c
uint8_t motor_speed = 0;
int8_t speed_direction = 1;
/* USER CODE END PV */

/* Private function prototypes ---------------------------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM3_Init(void);
static void MX_USB_PCD_Init(void);
/* USER CODE BEGIN PFP */
void set_motor_speed(uint8_t left_speed, uint8_t right_speed);
void set_motor_direction(uint8_t direction);
/* USER CODE END PFP */
```

```c
// Start PWM for both motors
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);  // Left motor PWM (D9)
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);  // Right motor PWM (D10)
```

```c
while (1)
{
  // Ramp speed up and down continuously
  motor_speed += speed_direction;

  // Reverse direction at limits
  if (motor_speed >= 250) {
    speed_direction = -1;
  } else if (motor_speed <= 5) {
    speed_direction = 1;

    // Every time we hit bottom, change direction
    static uint8_t dir_counter = 0;
    dir_counter++;

    if (dir_counter % 2 == 0) {
      // FORWARD
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET);
    } else {
      // BACKWARD
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET);
    }
  }

  // Apply the current speed to both motors
  set_motor_speed(motor_speed, motor_speed);

  // Small delay for smooth ramp (adjust for desired ramp speed)
  HAL_Delay(20);

  /* USER CODE END WHILE */

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

```c
void set_motor_speed(uint8_t left_speed, uint8_t right_speed) {
    // Clamp values to 0-255
    left_speed = (left_speed > 255) ? 255 : left_speed;
    right_speed = (right_speed > 255) ? 255 : right_speed;

    // Set PWM duty cycles
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, left_speed);   // Left motor
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, right_speed);  // Right motor
}

/**
  * @brief Set motor direction
  * @param direction: 0 = Stop, 1 = Forward, 2 = Backward
  */
void set_motor_direction(uint8_t direction) {
    switch(direction) {
        case 0: // STOP
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET);
            break;
        case 1: // FORWARD
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);   // D12 = HIGH
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // D8 = LOW
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);   // D7 = HIGH
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_RESET); // D6 = LOW
            break;
        case 2: // BACKWARD
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // D12 = LOW
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);   // D8 = HIGH
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET); // D7 = LOW
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, GPIO_PIN_SET);   // D6 = HIGH
            break;
    }
}
```

Output:



How does increasing or decreasing the duty cycle affect the speed of a DC motor?

Increasing the duty cycle of a DC motor increases the average voltage applied, making the motor spin faster, while decreasing the duty cycle lowers the average voltage, slowing it down. Essentially, higher duty cycle goes higher speed, lower duty cycle goes lower speed.

Given a timer clock of 72 MHz, how would you configure PSC and ARR to generate a 2 kHz PWM signal?

Let PSC as 35 and ARR as 999 so we will have (35+1)(999+1) = 36Khz, 72Mhz/36Khz is 2000hz i.e. 2kHz