세종대학교
SEJONG UNIVERSITY

2024학년도 2학기 지능기전공학과

임베디드시스템 결과보고서

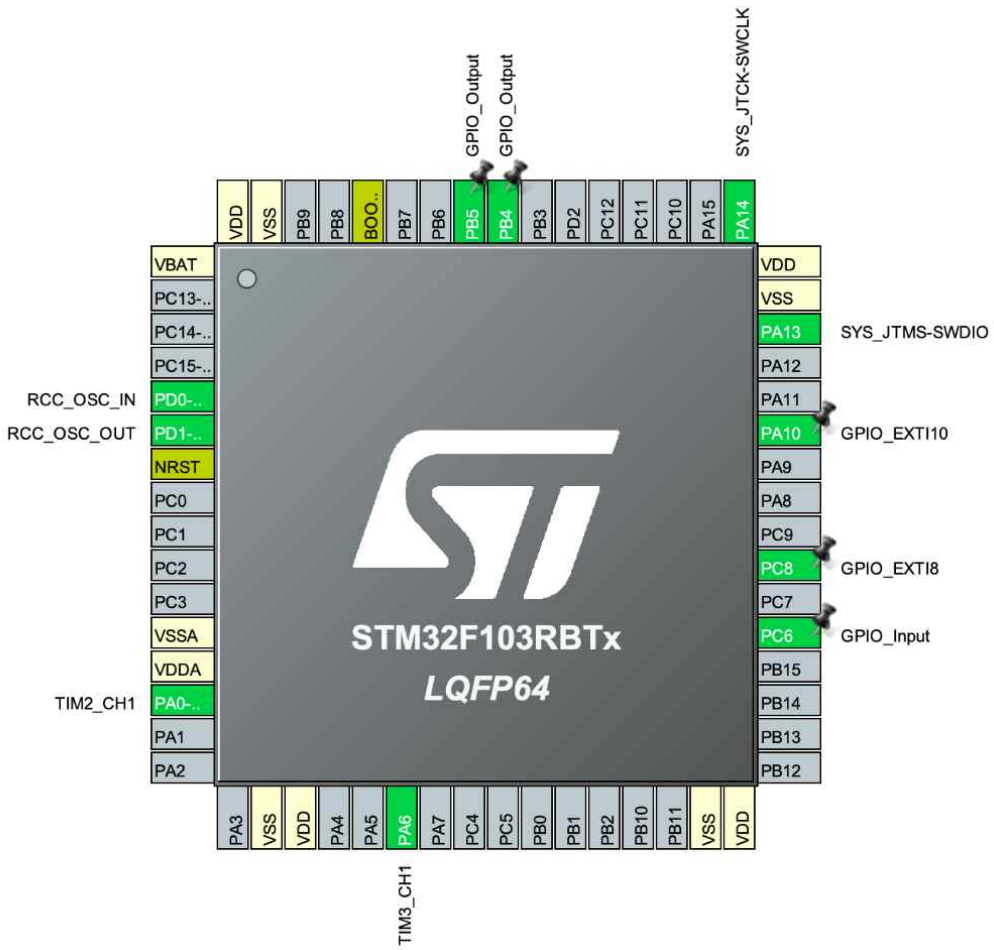
조: 4조
조원: 송종찬(조장), 김상우, 이종호, 정기남

1. 연구개발의 목표 및 범위

최종목표	정밀하고 안정적인 모터 위치 제어 시스템 구현
세부목표	1. 모터 제어 정확도 ±0.5% 이내 달성(오차 최소화) 2. 안정적인 회로 구성 및 목표치에 근사한 PID 설계를 통한 모터 제어 3. 성공적인 PBL(Problem-Based Learning) 프로젝트 수행

2. 연구과제의 수행 과정 및 수행 내용

- **<마이크로컨트롤러 설정에 대한 이미지 및 설명>**



**구성한 마이크로컨트롤러에 대한 설명:

GPIO_Input: PC6 -> 시작 신호를 받고 모터 구동을 시작함

GPIO_Output: PB4, PB5 -> 모터 방향 제어

GPIO_EXTI: PC8, PA10 -> 모터의 현재 각도를 구하기 위해 홀센서 값을 읽는 외부 인터럽트

TIM3_CH1: PA6 -> PID 제어로 도출한 값을 PWM 신호로 보내어, 모터의 Duty Cycle 을 제어함

- 프로그램 User Code

**코드 내 주석 참고

```
/* USER CODE BEGIN Header */
        /**
          ******************************************************************************
          * @file           : main.c
          * @brief          : Main program body
          ******************************************************************************
          * @attention
          *
          * Copyright (c) 2024 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"  // 수학 연산에 필요한 헤더 파일 포함

        /* Private includes ----------------------------------------------------------*/
        /* USER CODE BEGIN Includes */
        #include <math.h>
        /* 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;  // 타이머 2 핸들러 구조체
TIM_HandleTypeDef htim3; // 타이머 3 핸들러 구조체

/* USER CODE BEGIN PV */
const double PI = 3.1415926; // 원주율 상수 정의
const double dt = 0.001; // 제어 주기 설정
volatile double duty = 0; // PWM 듀티 값
uint32_t t1; // 타이머 주기 설정
uint32_t t2;
uint32_t flag_start; // 시작 플래그
uint32_t debug; // 디버깅 변수
volatile uint32_t time_1ns; // 마이크로 초 단위 시간
volatile uint32_t time_1ms; // 밀리초 단위 시간
volatile uint32_t time_1s; // 초 단위 시간
volatile uint32_t Tim;
volatile double property; // 목표 위치 설정 값

volatile double deg = 0; // 현재 각도
volatile uint32_t cnt_1 = 0; // 인코더 1 카운터
volatile uint32_t flag_1 = 0; // 인코더 1 플래그
volatile uint32_t cnt_2 = 0; // 인코더 2 카운터
volatile uint32_t flag_2 = 0; // 인코더 2 플래그

double int_error; // 적분 에러
double diff_error; // 미분 에러
double error_old; // 이전 에러 값
double error_current; // 현재 에러 값
int timer_entire; // 전체 타이머
int timer_start; // 시작 타이머
double targetval; // 목표 값

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void); // 시스템 클록 설정 함수
static void MX_GPIO_Init(void); // GPIO 초기화 함수
static void MX_TIM2_Init(void); // 타이머 2 초기화 함수
static void MX_TIM3_Init(void); // 타이머 3 초기화 함수
/* USER CODE BEGIN PFP */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim); // 타이머 인터럽트 콜백 함수
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin); // GPIO 외부 인터럽트 콜백 함수
double MotorPIDControl(); // 모터 PID 제어 함수
double MotorErrorFunction(double target, double output); // PID 에러 계산 함수
double MotorIntegralErrorFunction(); // PID 적분 에러 계산 함수
double MotorDiffErrorFunction(); // PID 미분 에러 계산 함수
```

```
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

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();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */

        HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_1);// 타이머 2에서 PWM 시작
        duty = 0; // 초기 듀티 값
        TIM2->CCR1 = duty; // 듀티 설정
        if ( HAL_TIM_Base_Start_IT(&htim3)!= HAL_OK )  // 타이머 3 인터럽트 시작
        {
                Error_Handler(); // 오류 처리 함수 호출
        }
```

```
        //각종 초기화
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
        flag_start = 0;
        time_1ns = 0;
        time_1ms = 0;
        time_1s = 0;
        deg = 0; // 초기 각도 설정
        duty = 0; // 초기 듀티 설정
        int_error = 0;
        diff_error = 0;
        error_old = 0;
        error_current = 0;
        timer_entire = 0;
        timer_start = 0;
        //각도 계산을 위한 홀센서 상태 초기화
        if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_8) == GPIO_PIN_SET){
                flag_2 = 1;
        }
        else{
                flag_2 = 0;
        }
        if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10) == GPIO_PIN_SET){
                flag_1 = 1;
        }
        else{
                flag_1 = 0;
        }
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (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_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_MUL16;
  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 = 63;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 999;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  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);

}
```

```c
/**
  * @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};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 63;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 999;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  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();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */

}
```

```
/**
  * @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_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5, GPIO_PIN_RESET);

  /*Configure GPIO pin : PC2 */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PC6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PC8 */
  GPIO_InitStruct.Pin = GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PA10 */
  GPIO_InitStruct.Pin = GPIO_PIN_10;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB4 PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

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

  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

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

/* USER CODE BEGIN 4 */
```

```
//------------------PID
double MotorPIDControl(){
// PID 제어를 위한 함수
// 현재 목표값(property)와 현재 위치(deg)를 기반으로 PID 제어 계산 수행
        //read property
        //read current position
        double K_p = 0.85; // 비례 제어 상수
        double K_i = 0.015; // 적분 제어 상수
        double K_d = 0.08; // 미분 제어 상수
        return (K_p * MotorErrorFunction(property, deg) + K_i * MotorIntegralErrorFunction() + K_d *
MotorDiffErrorFunction());
}



double MotorErrorFunction(double target, double output){
// 목표값과 현재 출력의 차이를 계산하여 에러값 반환
        //property - current
        //store old value to calculate differential
        error_old = error_current;  // 이전 에러값 저장
        error_current = target - output; // 현재 에러값 계산
        return (error_current);
}
double MotorIntegralErrorFunction(){
 // 적분 에러 계산 (누적 에러값)
        //Euler integral?
        if( (time_lms % 500) == 0 && (time_lms / 500)%2 == 1) // 30초부터 1분마다 적분 에러값 초기화
                int_error = 0;
        int_error += error_current; // 에러값 누적
        if (int_error < 0)
                return - int_error;  // 음수일 경우 절댓값 반환
        else
                return int_error;
}


double MotorDiffErrorFunction(){
// 미분 에러 계산 (에러값 변화량)
        //differential
        diff_error = error_current - error_old; // 현재 에러와 이전 에러의 차이
        return diff_error;
}
```

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
        t1=1000; // 1초 단위 주기 설정
        t2=1; // 기본 단위 시간 설정
  if(htim->Instance == TIM3)
        {
                time_lns ++; // 마이크로초 시간 증가
        }
        if( (time_lns % t2) == 0 ) time_lms++;  // 밀리초 시간 증가
        if( (time_lms % t1) == 0 ) time_1s++; // 초 단위 시간 증가


        if(flag_start == 0) // 프로그램 시작 플래그가 0일 때 (신호 오기 전)
        {
         if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_6) == GPIO_PIN_SET) // 시작 신호 감지
         {
// 시간 및 상태 초기화
            flag_start = 1; // 시작 플래그 설정
             time_lns = 0;
             time_lms = 0;
             time_1s = 0;
            deg = 0;
             duty = 0;
             int_error = 0;
             diff_error = 0;
             error_old = 0;
             error_current = 0;
             timer_entire = 0;
             timer_start = 0;
             if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_8) == GPIO_PIN_SET){
                        flag_2 = 1;
             }
             else{
                        flag_2 = 0;
             }
             if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10) == GPIO_PIN_SET){
                        flag_1 = 1;
             }
             else{
                        flag_1 = 0;
             }
         }
      }
        else {
        //---------------------------------property // 목표값(property) 설정
                if(time_1s == 1) property =180;
```

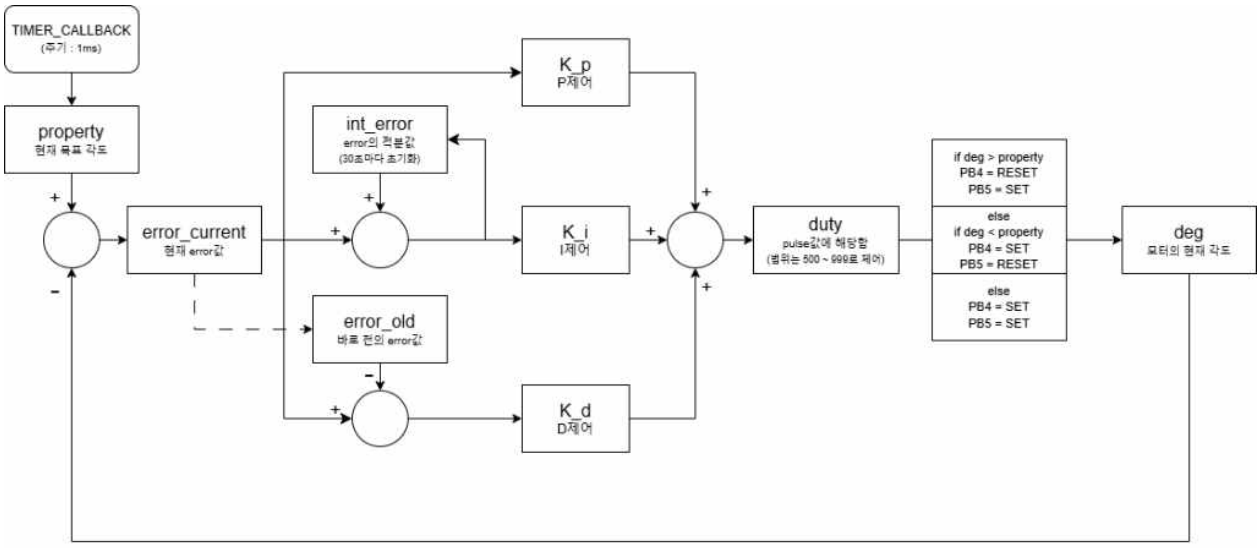
```
                if(time_1s == 2) property =0;
                if(time_1s == 3) property =-180;
                if(time_1s == 4) property =0;
                if(time_1s == 5) property =360;
                if(time_1s == 6) property =0;
                if(time_1s == 7) property =-360;
                if(time_1s == 8) property =0;
                if(time_1s == 9) property =120;
                if(time_1s == 10) property =240;
                if(time_1s == 11) property =360;
                if(time_1s == 12) property =240;
                if(time_1s == 13) property =120;
                if(time_1s == 14) property =0;
                if(time_1s == 15) property =-120;
                if(time_1s == 16) property =-240;
                if(time_1s == 17) property =-360;
                if(time_1s == 18) property =-240;
                if(time_1s == 19) property =-120;
                if(time_1s == 20) property =0;
                if(time_1ns >= 21*t1*t2 && time_1ns < 27*t1*t2){
                        Tim = time_1ns - 21*t1*t2;
                        property = -120*sin(2*PI*Tim/(6*t1*t2));
                }
                if(time_1ns >= 27*t1*t2 && time_1ns < 33*t1*t2){
                        Tim = time_1ns - 27*t1*t2;
                        property = -240*sin(2*PI*Tim/(6*t1*t2));
                }
                if(time_1ns >= 33*t1*t2 && time_1ns < 39*t1*t2){
                        Tim = time_1ns - 33*t1*t2;
                        property = -360*sin(2*PI*Tim/(6*t1*t2));
                }
        //------------------------------PID
// PID 제어를 통해 듀티 계산
                duty = MotorPIDControl() / dt;
 // 듀티 제한 (최대/최소 값 설정)
                if(duty > 999){
                        duty = 999;
                }
                else if(duty < -999){
                        duty = -999;
                }
// 듀티 값에 따라 PWM 출력 설정
                if(duty < 0){
                        TIM2->CCR1 = -duty + 500;
                }
                else{
```

```
                        TIM2->CCR1 = duty + 500;
                }
// 현재 각도와 목표 각도 비교하여 방향 설정
                if(deg > property){
                        debug = 1;
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
                }
                else if(deg == property){
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
                }
                else if(deg < property){
                        debug = -1;
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
                        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_RESET);
                }
        }
}


void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
        //각도계산
         if (GPIO_Pin == GPIO_PIN_10) { // PIN 10에서 인터럽트 발생
                        cnt_1++;
                        if(flag_1 == 0) //flag_1 : 0 -> 1
                        {
                                flag_1 = 1;
                                if(flag_2 == 0) // (0 , 0) -> (1 , 0)
                                {
                                        deg = deg + 36.0f/(4*13) ; //정방향
                                }
                                else // (0 , 1) -> (1 , 1)
                                {
                                        deg = deg - 36.0f/(4*13) ; // 역방향
                                }
                        }
                        else //flag_1 : 1 -> 0
                        {
                                flag_1 = 0;
                                if(flag_2 == 0) // (1 , 0) -> (0 , 0)
                                {
                                        deg = deg - 36.0f/(4*13) ; //역방향
                                }
                                else // (1 , 1) -> (0 , 1)
                                {
```

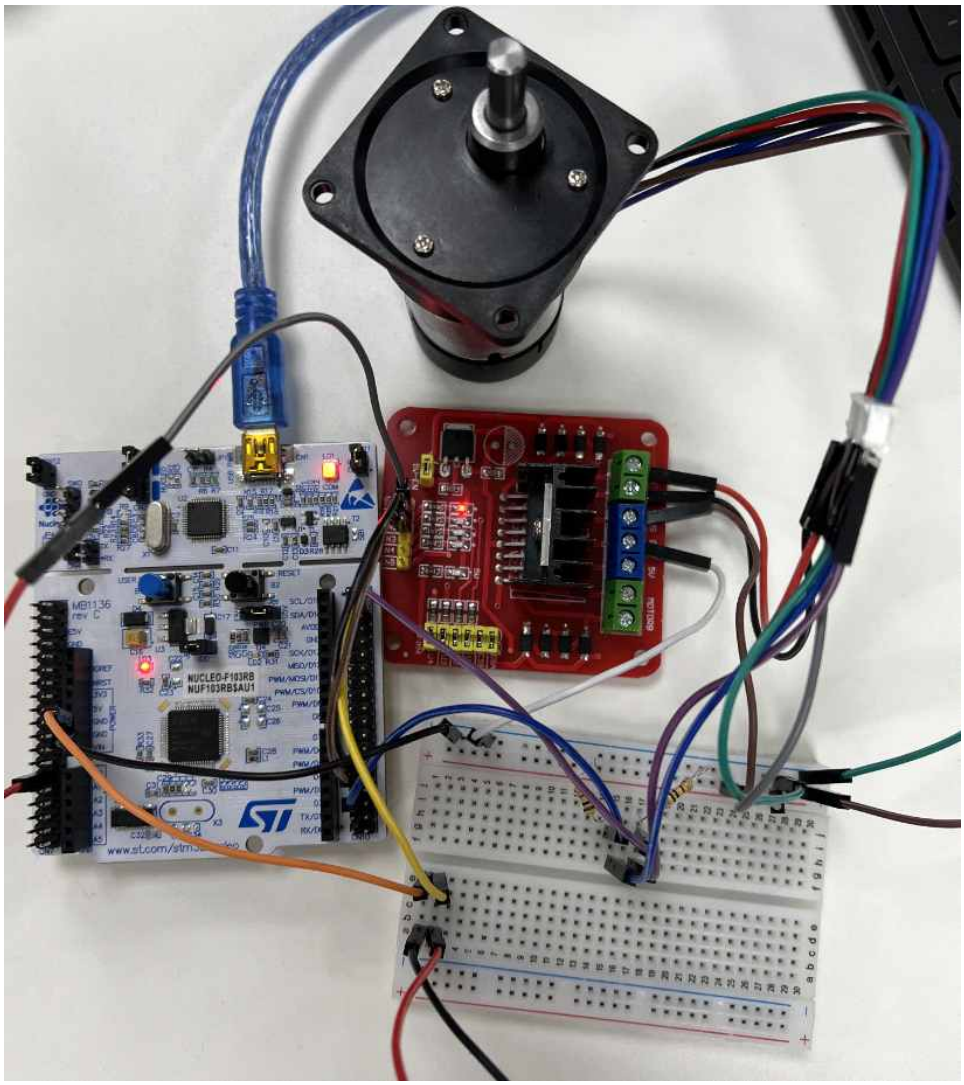
```
                                        deg = deg + 36.0f/(4*13) ; // 정방향
                                }

                        }
                }
                if (GPIO_Pin == GPIO_PIN_8) // PIN 8에서 인터럽트 발생
                {
                         cnt_2++;
                        if(flag_2 == 0) //flag_2 : 0 -> 1
                        {
                                flag_2 = 1;
                                if(flag_1 == 0) // (0 , 0) -> (0 , 1)
                                {
                                        deg = deg - 36.0f/(4*13) ; //역방향
                                }
                                else // (1 , 0) -> (1 , 1)
                                {
                                        deg = deg + 36.0f/(4*13) ; //정방향
                                }
                        }
                        else //flag_2 : 1 -> 0
                        {
                                flag_2 = 0;
                                if(flag_1 == 0) // (0 , 1) -> (0 , 0)
                                {
                                        deg = deg + 36.0f/(4*13) ; //정방향
                                }
                                else // (1 , 1) -> (1 , 0)
                                {
                                        deg = deg - 36.0f/(4*13) ; //역방햔
                                }

                        }
                }

}



/* 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 */
```

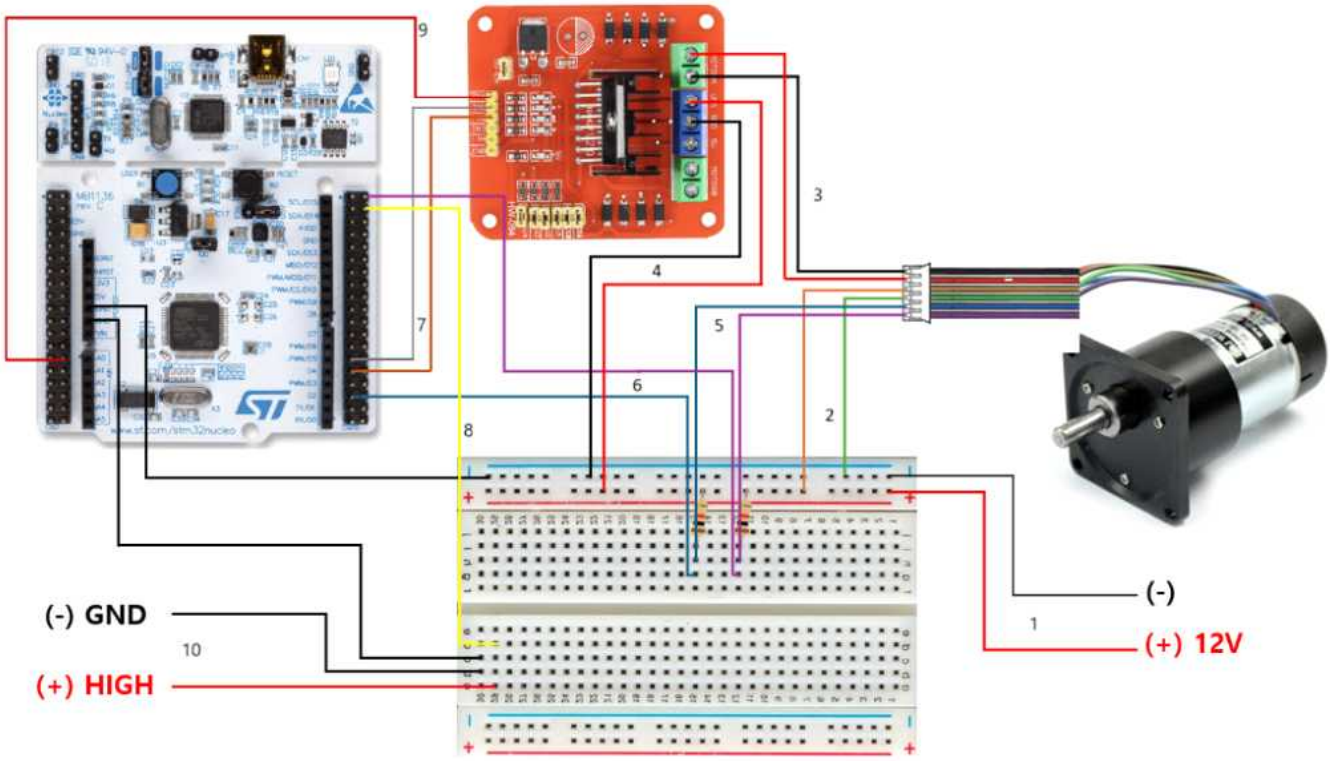
- 설계한 PID 컨트롤러의 블록 다이어그램



- 구성한 시스템 및 회로의 사진 및 설명



**회로 설명:



해당 회로는 먼저 회로 선이 복잡해질 것을 미리 생각하여 보드를 활용하기로 하였다. 보드에서는 모터의 (+) 케이블은 (+) 회로, (-) 케이블은 (-) 회로를 구성하였고, 마찬가지로 모터 컨트롤러와 STM보드의 각 부분도 GND(Ground)는 (-) 케이블로 연결되게 하는 등 (+), (-)에 맞게 연결하였다.

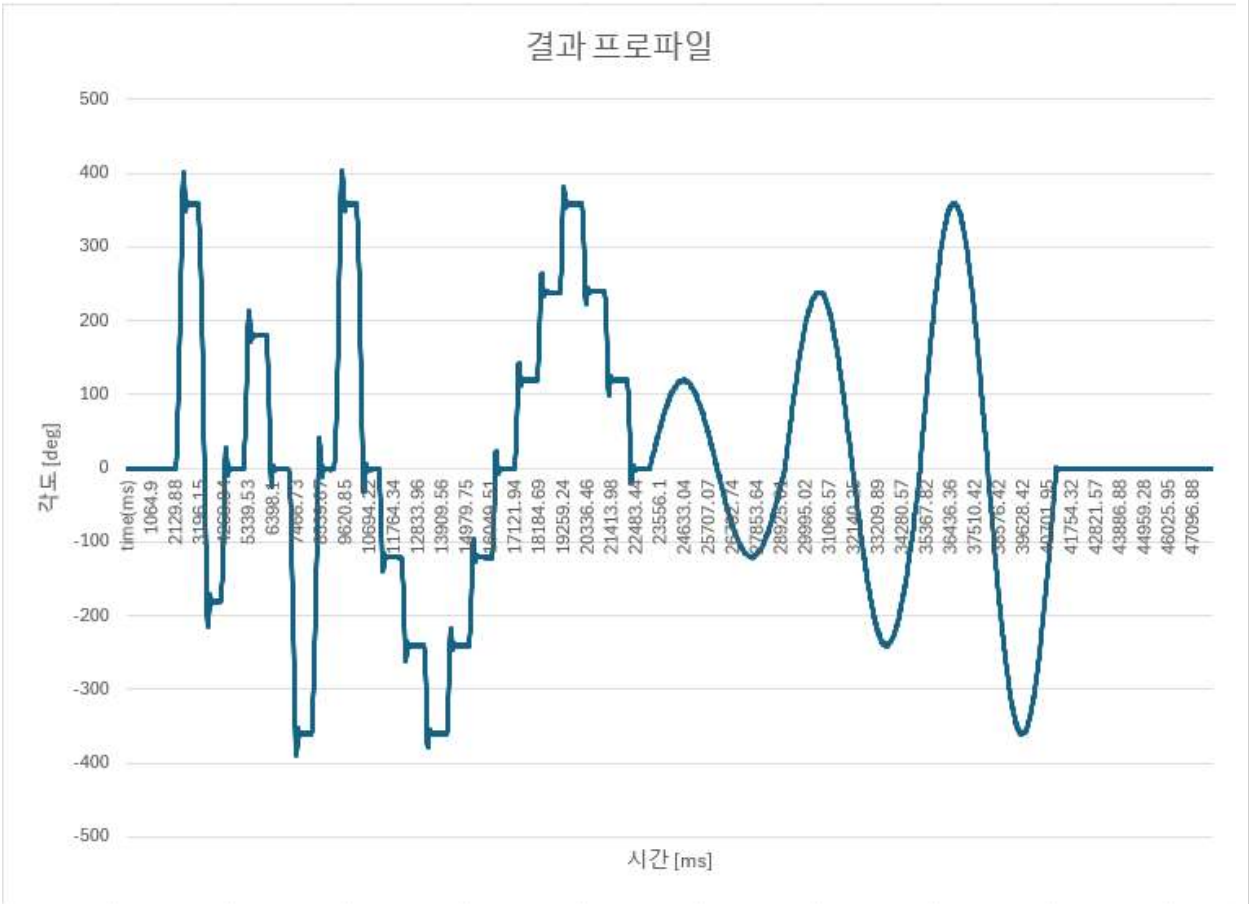
1) 전체 회로에 전력이 들어가는 부분으로 빨간색이 VCC, 검은색이 GND이다.
2) 홀센서에 전원이 들어가는 부분으로 갈색이 VCC, 초록색이 GND이다.
3) 모터에 전원이 들어가는 부분이다.
4) 모터드라이브에 전력이 들어가는 부분으로 빨간색이 VCC, 검은색이 GND이다.
5) 홀센서에서 신호가 나와 저항을 통해 필터링을 거쳐 안정적이며 Pull-Up으로 전환되는 부분이다.
6) 홀센서에서 나온 신호가 HIGH/LOW로 전환되어 PA10과 PC8로 들어간다.
7) PB4와 PB5에서 나온 신호로 모터를 제어한다.
8) 입련된 시작 신호를 PC8로 받아들여 읽는다.
9) 모터의 PWM 신호를 준다.
10) 시작신호가 들어오는 부분으로, 빨간색에 HIGH신호가 들어오고 검은색은 GND와 연결된다.

3. 연구과제의 수행 결과 및 목표 달성 정도

1) 프로파일 추종 누적 오차

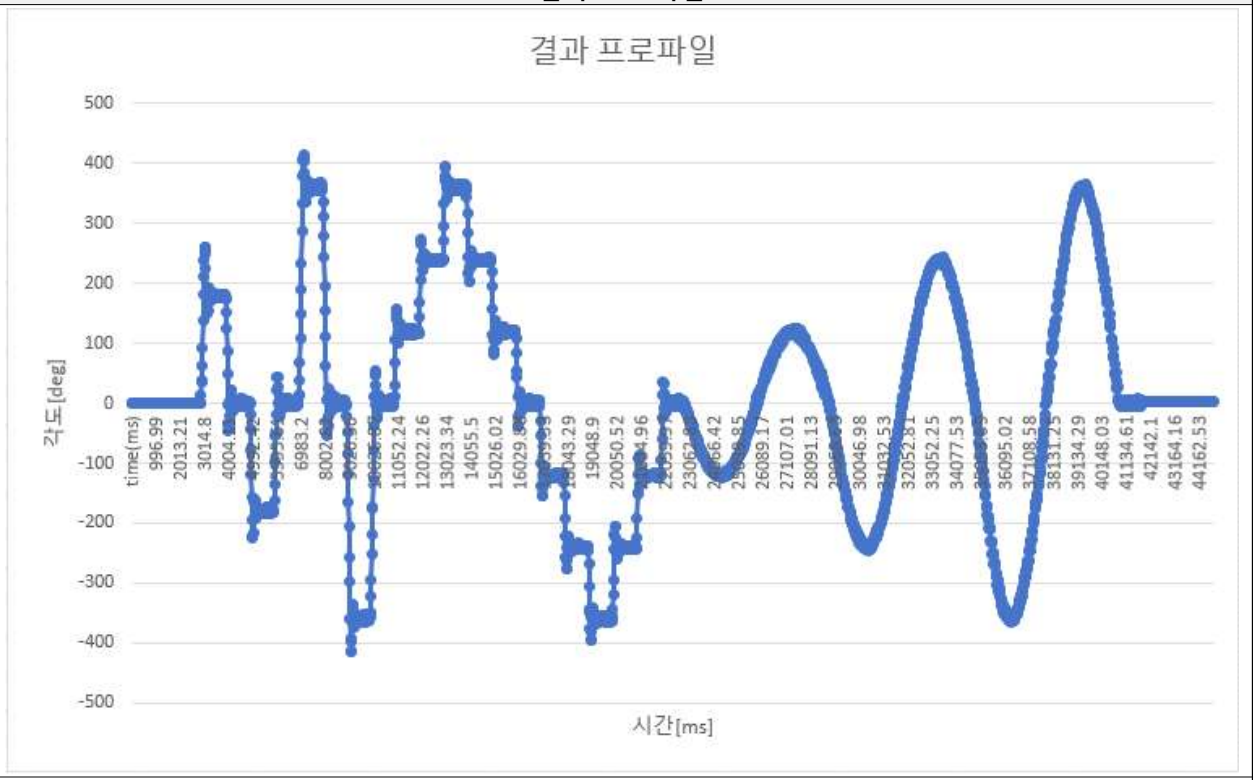
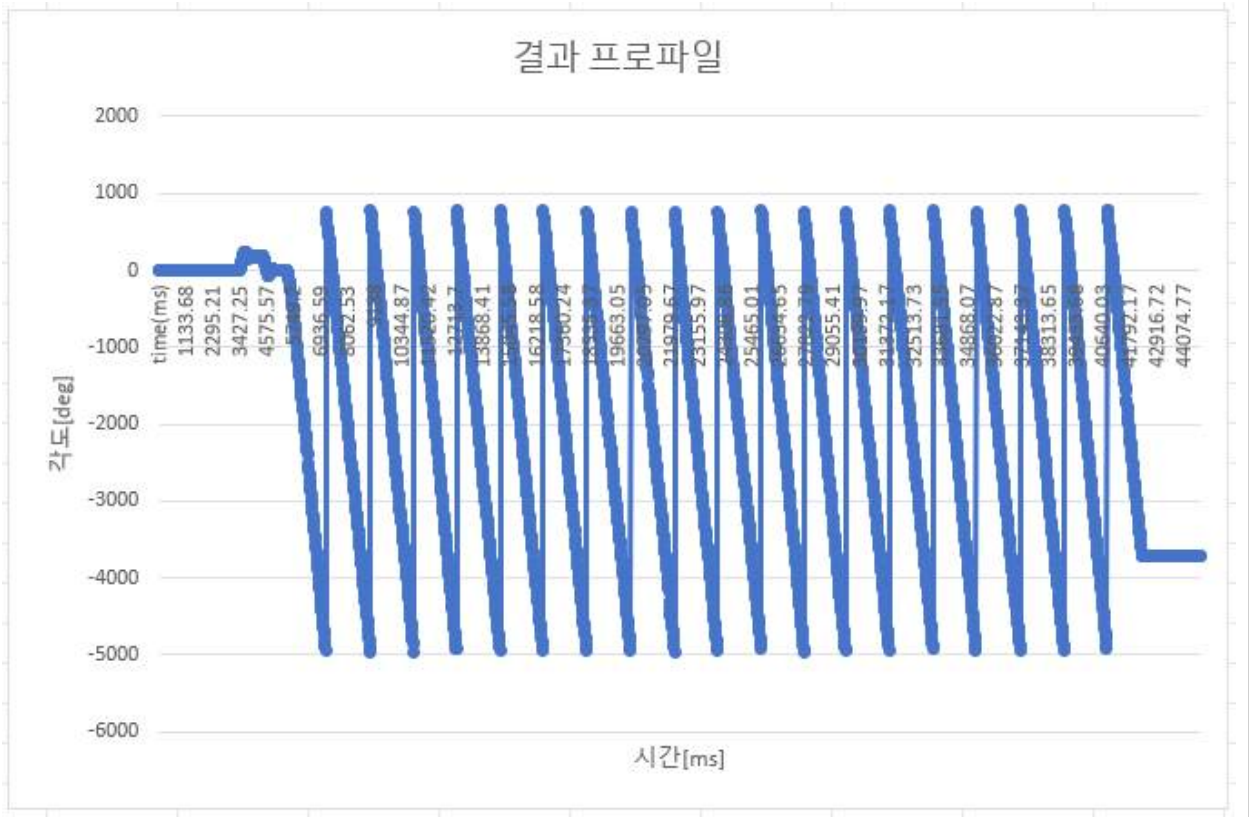
	1차 평가	2차 평가
누적 오차 [deg·ms]	5991612.00	624533.40

2) 1차 평가 결과 프로파일 분석

	결과 프로파일
1차 -1	1차-1회 코드 및 디버깅 이슈로 진행하지 못하였음.
1차-2	

- 1차 - 1회: 코드 및 디버깅 이슈로 진행하지 못하였음.
- 1차 - 2회: 모터 구동은 성공하였지만, PID 제어 알고리즘에도 문제가 발생하였고, 방향을 잘못 설정하여 그래프가 반대로 그려져 오차 값이 매우 커지는 현상이 발생하였음. -> 방향을 잡고 안정적인 PID 제어 알고리즘 설계가 필요하였다.

3) 2차 평가 결과 프로파일 분석

	결과 프로파일
2차 -1	
2차- 2	

- 2차 - 1회: 약간의 아쉬운 오차값이 발생하였지만, 성공적으로 모터 제어를 진행하였다.
- 2차 - 2회: 성능 향상을 위한 PID 게인 튜닝 과정에서 모터 전류 제어 오류가 발생하여 안정적인 구동 시스템을 구현하지 못하였다.

3) 최종 PID 컨트롤러의 GAIN(이득) 값 및 최종 결과에 대한 논의

- 1차 평가에서는 회전방향 및 PID 제어의 구현이 완벽하지 못해 상당한 누적 오차 값을 도출하였다.
- PID 컨트롤러의 Gain 값 -> K_P: 0.85, K_I: 0.015, K_D: 0.08
- 위 PID 상수를 시행착오를 통해 값을 도출하였고, 2차-1회 평가에서 약 60만의 오차값을 도출하여 상당히 안정적인 시스템을 구현하였다. 비록 1차에서 아쉬운 오차값을 도출하였지만, PBL의 의의를 생각하며, 끝없는 논의와 반복 시행을 거쳐 좋은 결과값을 도출하였다.

- 참고문헌(Reference)

[1] 박성진, 강만원, 이규영, 이 환, 소상균, "2자유도 PID 제어시스템의 지능형 튜닝에 대한 연구", 2000.5.1, 한국퍼지및지능시스템학회 2000년도 춘계학술대회 학술발표 논문집, pp. 138~141

[2] 김민, 윤석암, 최장균, 임중열, 윤형상, 차인수, "PWM 구동방식을 적용한 전기 자전거용 DC 모터의 속도 제어에 관한 연구", 1999.7, 전력전자학회 1999년 학술대회논문집, pp. 143~146

[3] 이필성, "STM32 제어기를 이용한 회전형 도립전자 스윙업 제어에 관한 연구", 한세대학교 대학원 석사 학위 논문, 2019.12, pp 4, 9~11, 16~20

[4] 박치태, "PID 제어기를 이용한 DC서보모터의 속도제어에 관한 연구", Chungju National University 학위 논문(석사), 2006.02, pp 22~23