Lab 9

PD controller with FreeRTOS

Exercise

- Lab3에서 사용한 PDControl 프로그램을 FreeRTOS를 사용하도록 변경한다.
- PDControl 프로그램에서는 파란색 버튼을 누르면 응답 데이터를 캡쳐해서 시리얼 터미널에 프린트한다. 이를 위해서 파란색 버튼을 누르면 인터럽트가 발생하고 아래와 같은 인터럽트 서비스 루틴에서 data_flag 값을 변경한다.

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
        HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
        data_flag=1;
}
/* USER CODE END 4 */
```

Exercise

- 메인 프로그램에서는 아래와 같은 무한 루프에서 flag 값에 따라서 시리얼 터미널에 프린트 동작을 실행한다.

```
/* USER CODE BEGIN WHILE */
 while (1)
 {
   if(data_done == 1) {
     for (int i=0; i < 4000 ;i++){
     printf("%d %d\r\n",i,data[i]);
         }
     data_flag=0;
     data_done=0;
     HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_14);
  }
   /* USER CODE END WHILE */
```

Exercise

- 이 Lab에서는 이상에서 설명한 것과 동일한 기능을 수행하는 프로그램을 FreeRTOS 를 사용하여 작성한다. 단, 아래의 요구 사항들을 지켜야 한다.

- 2개의 Task를 생성한다.(Task1, Task2)
- 버튼 인터럽트는 사용하지 않는다. 버튼 입력 GPIO에 관련된 설정은 아무것도 하지 않는다. 버튼 입력에 대한 인터럽트 서비스 루틴도 입력하지 않는다.
- Task2는 버튼 입력을 polling 한다. 버튼이 눌리면 아래 함수는 1을, 눌리지 않으면 0을 반환한다.

  HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0)

Exercise

- PDControl프로그램의 메인 루프에서 실행했던 시리얼 터미널에 데이터를 프린트하는 기능은 Task1에서 실행한다.

- 이상의 동작을 구현하여 원래의 PDControl 프로그램과 동일하게 동작하는지 확인한다.

- 배포한 원래 프로그램에서는 전역 변수로 정의한 flag 변수를 사용 했지만, 이를 대신하여 semaphore나 mutex를 사용하여 동일한 기능을 구현할 경우 평가에 유리하도록 고려할 예정임.