임베디드 시스템 설계 및 실험
텀 프로젝트 최종 보고서_도망가는 미션 알람

분반 : 001 분반 (월)

교수님 : 정 상화 교수님

조교님 : 유 동화 조교님

검사 및 제출일 : 2019-12-24

제작 기간 : 2019-11-18 ~ 2019-12-24

00. 목차

- 01. 프로젝트 주제 및 목적 … p.2
- 02. 프로젝트 과제 … p.2
- 03. 프로젝트 준비 (사용센서, 구성도, 흐름도) … p.2
- 04. 프로젝트 구현 … p.5
- 05. 프로젝트 결과 … p.19
- 06. 결론 … p.32

7 조

장 수현

박 창조

임 다영

이 힘찬

01. 프로젝트 주제 및 목적

1.1 프로젝트 주제

도망가는 미션 알람

1.2 프로젝트 목적

알람의 기능은 잠을 깨우는 것이다. 이러한 알람의 기본적인 기능에 집중하여 사용자가 깊은 잠에서 쉽게 깰 수 있도록 하는 도망가는 미션 알람을 제작하였다.

02. 프로젝트 과제

2.1 알람 및 미션 난이도 설정

- 스마트폰(블루투스 시리얼 통신 어플)을 통해 알람에 시간과 미션 난이도를 전송한다.

* 전송 받은 시간이 지나면 알람이 울리게 되며, 난이도는 수행할 미션의 횟수를 의미한다.

2.2 알람 작동 및 미션 출력

- LCD 모니터의 start 버튼을 누르면 타이머 기능이 수행된다.
- 타이머는 입력 받은 시간이 되면 종료되며 알람과 기상 미션을 작동시킨다.
- 기상 미션으로 1 부터 4 까지의 숫자 값 중 하나를 랜덤으로 LCD 에 출력한다.

* 이 때 1 부터 4 까지의 숫자 값은 사용자 인식이 필요한 센서를 지시함

2.3 자율 주행 알람

- 미션을 모두 수행하기 전까지 도망가는 미션 알람은 끊임없이 움직인다.
- 직진을 하다가 정면에 장애물을 만나면 오른쪽/왼쪽 중 하나를 랜덤으로 선택해 방향전환을 한다.

2.4 미션 수행 및 알람 중지

- 사용자는 입력한 미션 횟수(난이도)만큼 미션을 수행한다.
- 알람이 출력하는 미션 숫자에 해당하는 인체 감지 센서를 인식시킨다.
- 미션 수행이 모두 종료되면 알람은 작동을 중지한다.

03. 프로젝트 준비

3.1 사용 센서

3.1.1 블루투스 모듈

bluetooth 통신을 이용하여 스마트폰으로 알람 시간과 난이도를 설정할 수 있도록 한다.

3.1.2 TFT LCD

타이머와 미션을 출력한다.

3.1.3 RC 카

도망가는 미션 알람의 하드웨어로서 도망가는 기능을 담당한다.

3.1.4 릴레이 모듈

RC 카의 바퀴를 구동 시킨다.

3.1.5 초음파 센서

RC 카의 전면에 위치하며 장애물 인지를 하여 알람의 자율주행을 도와준다.

3.1.6 PIEZO

타이머가 완료되면 알람 소리를 출력한다.

3.1.7 인체감지센서

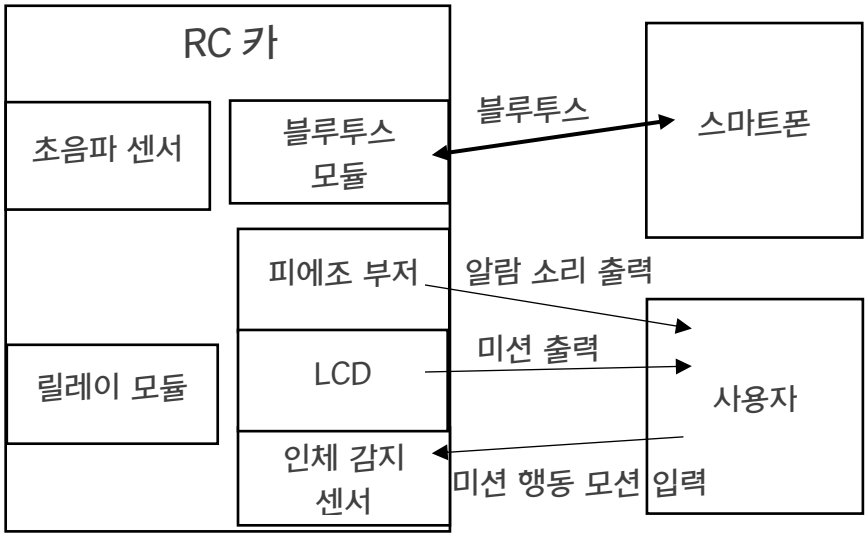
인체감지센서를 통해 미션을 수행한다.

3.1.8 LED

타이머가 종료되고 알람이 작동될 때 점등되어 알람 작동 상태를 알린다.

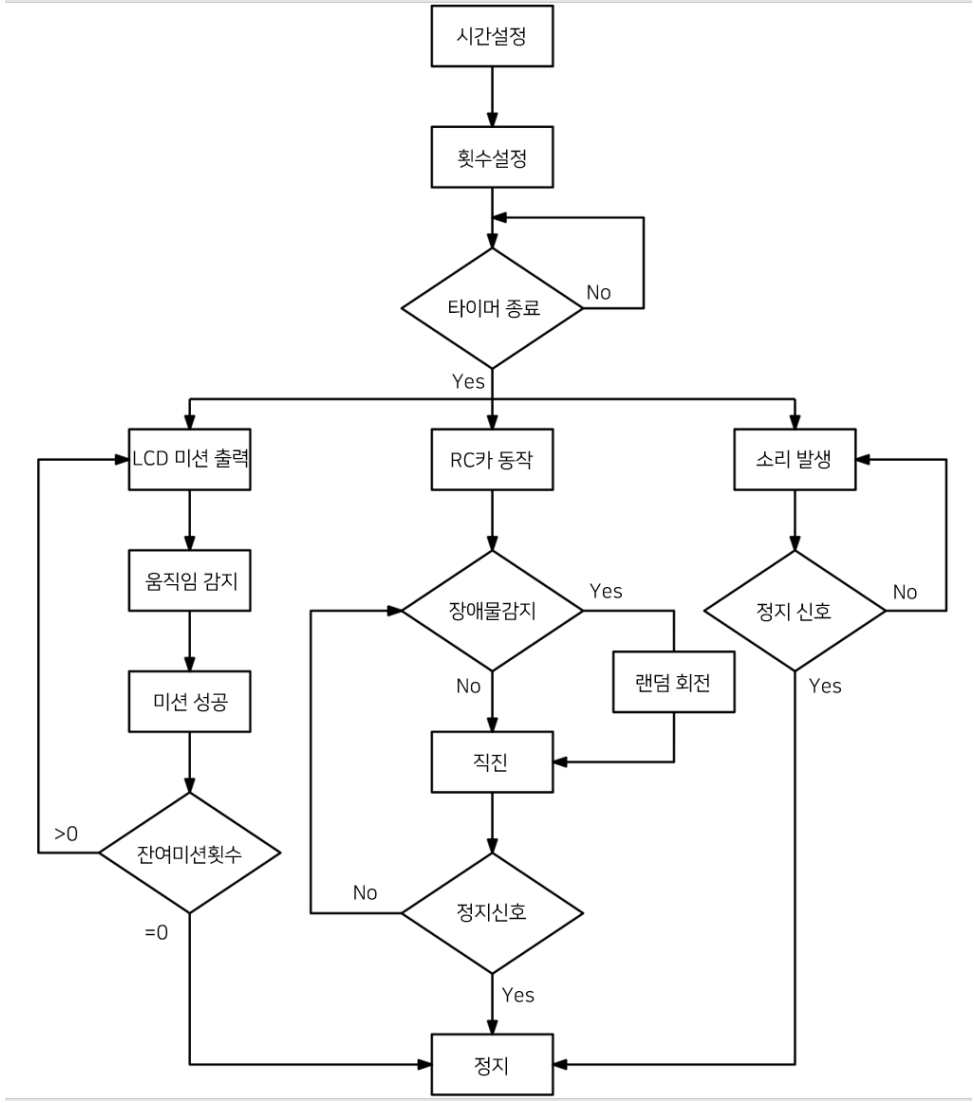
3.2 구성도

도망가는 미션 알람의 구성도는 다음과 같다.



3.1 흐름도

도망가는 미션 알람의 작동 흐름도는 다음과 같다.



04. 프로젝트 구현

4.1 소프트웨어 구현

4.1.1 RCC Configuration

프로젝트에 사용 될 AFIO, TIM, USART, GPIO, ADC 를 ENABLE 해준다.

```
void RCC_Configure() {
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

   RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA       |       RCC_APB2Periph_GPIOC       |
RCC_APB2Periph_GPIOD      |      RCC_APB2Periph_USART1|      RCC_APB2Periph_GPIOE      |
RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1, ENABLE);
}
```

4.1.2 GPIO Configuration

블루투스, 인체 감지 모듈, 초음파 센서, 릴레이, LED, PIEZO 에서 사용할 각 핀을 인가해준다.

```
void GPIO_Configure() {
   GPIO_InitTypeDef GPIO_InitStructure;

   // USART1 TX
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;   //PA9
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   // USART1 RX
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;   //PA10
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   // USART2 TX
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;   //PA2
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   // USART2 RX
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   /* 초음파 */
```

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOB, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* 릴레이관련 */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* LED */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);
GPIO_ResetBits(GPIOD, GPIO_Pin_2); // Set C13 to Low level ("0")

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_ResetBits(GPIOD, GPIO_Pin_2); // Set C13 to Low level ("0")
```

```
    /* 인체감지센서 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /* 피에조 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

4.1.3 TIM Configuration

- TIM2 는 블루투스로 시간을 받아와 알람이 작동하도록 하는 타이머이다. 알람을 위한 타이머는 초단위로 작동해야 하므로 분주를 1 초로 맞추기 위해 Period 에 12000, Prescaler 에 6000 의 값을 주었다
- TIM3 은 초음파 센서를 사용할 때 초음파 값을 받아올 주기를 계산 및 설정하고 그 주기에 맞게 초음파 센서로 측정한 거리 값을 측정하기 위한 타이머이다.

```
void TIM_Configure() {
    RCC_ClocksTypeDef RCC_ClocksStatus;
    uint16_t prescaler;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
    TIM_OCInitTypeDef TIM_OCInitStruct;
    TIM_ICInitTypeDef TIM_ICInitStruct;

    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

    /* relay */
```

```
   RCC_GetClocksFreq(&RCC_ClocksStatus);
   prescaler = RCC_ClocksStatus.SYSCLK_Frequency / 1000000 - 1; //1 tick = 1us (1 tick =
0.165mm resolution)

   TIM_DeInit(TIM3);
   TIM_TimeBaseInitStruct.TIM_Prescaler = prescaler;
   TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
   TIM_TimeBaseInitStruct.TIM_Period = 0xFFFF;
   TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

   TIM_OCStructInit(&TIM_OCInitStruct);
   TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
   TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
   TIM_OCInitStruct.TIM_Pulse = 15; //us
   TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
   TIM_OC3Init(TIM3, &TIM_OCInitStruct);

   TIM_ICInitStruct.TIM_Channel = TIM_Channel_1;
   TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
   TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
   TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
   TIM_ICInitStruct.TIM_ICFilter = 0;

   TIM_PWMIConfig(TIM3, &TIM_ICInitStruct);
   TIM_SelectInputTrigger(TIM3, TIM_TS_TI1FP1);
   TIM_SelectMasterSlaveMode(TIM3, TIM_MasterSlaveMode_Enable);
   TIM_CtrlPWMOutputs(TIM3, ENABLE);
   TIM_ClearFlag(TIM3, TIM_FLAG_Update);

   /* usart */
   TIM_TimeBaseStructure.TIM_Period = 12000;
   TIM_TimeBaseStructure.TIM_Prescaler = 6000;
   TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
   TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

   TIM_ARRPreloadConfig(TIM2, ENABLE);
   TIM_Cmd(TIM2, ENABLE);
   TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}
```

4.1.4 USART Configuration

- 알람 시간 및 난이도 설정을 위한 블루투스 사용을 위해 USART 의 TX / RX 를 인가했다.
- 통신에서 문자의 길이는 8bit 로 설정하였다.
- Parity Bit 는 사용하지 않았고 Stop Bit 는 1 로 설정했다.

```
void USART_Configure() {
   USART_InitTypeDef USART_InitStructure;
   USART_InitTypeDef USART2_InitStructure;

   /*TODO: USART1 configuration*/
   USART_InitStructure.USART_BaudRate = 9600;
   USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
   USART_InitStructure.USART_Parity = USART_Parity_No;
   USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
   USART_InitStructure.USART_StopBits = USART_StopBits_1;

   USART2_InitStructure.USART_BaudRate = 9600;
   USART2_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
   USART2_InitStructure.USART_Parity = USART_Parity_No;
   USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
   USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
   USART2_InitStructure.USART_StopBits = USART_StopBits_1;

   USART_Init(USART1, &USART_InitStructure);
   USART_Init(USART2, &USART2_InitStructure);

   /*TODO: USART1 cmd ENABLE*/
   USART_Cmd(USART1,ENABLE);
   USART_Cmd(USART2,ENABLE);

   /*TODO: USART1 IT Config*/
   USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
   USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
}
```

4.1.5 NVIC Configuration

- USART Handler 와 TIM2 Handler 를 사용하기위해 우선순위를 설정해주었다.

```
void NVIC_Configure() {
   NVIC_InitTypeDef NVIC_InitStructure;

   NVIC_InitTypeDef NVIC_USART_InitTD;
   NVIC_InitTypeDef NVIC_USART2_InitTD;

   NVIC_USART_InitTD.NVIC_IRQChannel = USART1_IRQn;
```

```
    NVIC_USART2_InitTD.NVIC_IRQChannel = USART2_IRQn;

    NVIC_USART_InitTD.NVIC_IRQChannelCmd = ENABLE;
    NVIC_USART2_InitTD.NVIC_IRQChannelCmd = ENABLE;


    NVIC_USART_InitTD.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_USART2_InitTD.NVIC_IRQChannelPreemptionPriority = 0;

    NVIC_USART_InitTD.NVIC_IRQChannelSubPriority = 0;
    NVIC_USART2_InitTD.NVIC_IRQChannelSubPriority = 0;

    NVIC_Init(&NVIC_USART_InitTD);
    NVIC_Init(&NVIC_USART2_InitTD);

    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&NVIC_InitStructure);
}
```

4.1.6 USART2_IRQHandler

- 알람 시간과 난이도 설정을 위한 USART Handler 이다.
- 핸드폰에서 입력한 시간과 난이도를 USART_ReceiveData(USART2)로 받아와 Input 에 저장해준다.
- 받아오는 값은 char 형인데 시간과 난이도에는 int 형이 필요하므로 my_atoi 함수를 만들어 형 변환에 이용하였다.
- 핸드폰에서 시간과 난이도를 받아올 때 한 문자씩 들어오므로 아스키 코드 '0'~'9' 사이에 있는 문자일 경우 그 값을 형 변환 후 배열에 저장해준다.
- 배열에 저장한 값들을 구현한 Make Time 함수를 이용해 Hour, Min, Sec 값에 넣어준다.

```
int Array_Input[7];
int idx = 0;
int Hour, Min, Sec;

void Make_Time(void) {
    Hour = Array_Input[0]*10 + Array_Input[1];
    Min = Array_Input[2]*10 + Array_Input[3];
    Sec = Array_Input[4]*10 + Array_Input[5];
}

int my_atoi(char c) {
    int value_int;
    value_int = c - 48;
```

```
   return value_int;
}

void USART2_IRQHandler(void) {
   if(USART_GetITStatus(USART2, USART_IT_RXNE ) != RESET) {
      char Input = USART_ReceiveData(USART2);
      if(48 <= Input && Input <= 57){
         Array_Input[idx] = my_atoi(Input);
         idx++;
         if(idx == 7){
            Make_Time();
            idx = 0;
         }
      }
   }
   USART_ClearITPendingBit(USART2,USART_IT_RXNE);
}
```

4.1.7 TIM2_IRQHandler 와 main 일부

- 알람 실행을 위한 TIM Handler 이다.
- LCD 화면에 Start 버튼을 누르면 controlflag 가 0 으로 바뀐다.
- controlflag 가 0 으로 바뀌면 TIM2_IRQHandler 로 들어와 타이머가 시작된다.
- 타이머가 증가하면서 블루투스로 설정한 알람 시간과 같아지면 controlflag 가 1 로 바뀌면서 알람 및 RC 카가 작동한다.

```
uint16_t pos_x, pos_y, con_x, con_y;
int t1, t2, t3;
int controlflag=2;      // 초기화

void TIM2_IRQHandler() {
   if(TIM_GetITStatus(TIM2, TIM_IT_Update)!=RESET) {
      if(controlflag==0) {
         t1++;
         if(t1==60) {
            t2++;
            t1 = 0;
         }
         if(t2==60) {
            t3++;
            t2 = 0;
         }
         if(t1==Sec && t2==Min && t3==Hour){
            controlflag=1;
         }
      } else if(controlflag == 1) {
```

```
            t1 = 0, t2 = 0, t3 = 0;
         }

      LCD_ShowNum(100, 50, Hour, 2, BLACK, WHITE);
       LCD_ShowNum(130, 50, Min, 2, BLACK, WHITE);
       LCD_ShowNum(160, 50, Sec, 2, BLACK, WHITE);
       LCD_ShowNum(200, 50, Array_Input[6],1, BLACK, WHITE);

        // 시간 출력
       LCD_ShowNum(100, 10, t3, 2, BLACK, WHITE);
       LCD_ShowNum(130, 10, t2, 2, BLACK, WHITE);
       LCD_ShowNum(160, 10, t1, 2, BLACK, WHITE);

      }
      TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}
int main() {
    while(controlflag != 0) {
         Touch_GetXY(&pos_x, &pos_y, 1);
         Convert_Pos(pos_x, pos_y, &con_x, &con_y);
         if( con_x > 10 && con_x < 70 && con_y > 200 && con_y < 230 ) {
            controlflag = 0;
         }
      }
}
```

4.1.8 딜레이 함수

- 인체 감지 센서에서 값을 인지한 후 다음 인식까지 딜레이를 주기 위한 함수

```
void __attribute__((weak)) osa_mdelay(unsigned int msec)
{
   uint32_t temp;
   SysTick->LOAD=(uint32_t)msec*(HSE_VALUE/1000);
   SysTick->VAL =0x00;      // clear Count flag
   SysTick->CTRL=0x01;
   do
   {
      temp=SysTick->CTRL;
   }
   while((temp&0x01)&&!(temp&(1<<16)));   // wait Count flag set
   SysTick->CTRL=0x00;
   SysTick->VAL =0X00;
}
```

4.1.9 초음파 센서의 거리 측정 함수

- TIM_Configure 에서 설정한 TIM3 을 이용하여 초음파 센서가 측정한 거리를 측정한다.
- 초음파 센서에서 처음 신호를 보낸 TIME 과 신호가 다시 돌아올 때의 TIME 을 계산해 해당 시간 동안의 거리를 측정하여 반환한다.

```
int32_t HCSR04GetDistance() {
   (TIM3)->CNT = 0;
   TIM_Cmd(TIM3, ENABLE);
   while(!TIM_GetFlagStatus(TIM3, TIM_FLAG_Update));
   TIM_Cmd(TIM3, DISABLE);
   TIM_ClearFlag(TIM3, TIM_FLAG_Update);
   return (TIM_GetCapture2(TIM3)-TIM_GetCapture1(TIM3))*165/1000;
}
```

4.1.10 main 함수

- array[15][12]는 난이도에 따른 미션을 저장하고 있다. -1 은 end of line 의 의미의 값이고 1, 2, 3, 4 는 해당 번호에 맞는 인체감지센서를 의미한다. Loop_num 은 col 값에 해당한다.
- modulo 값은 랜덤하게 row 값을 정한 값이고, dist 는 초음파 센서가 측정한 거리 값을 받아오고, rest_count 는 미션이 남은 횟수를 말하고 random_going 은 자동차가 우회전할지 좌회전할지를 랜덤하게 정해진 값이 들어간 변수이다.
- 먼저 타이머가 시작되면 난이도에 따라 미션 횟수를 정하고 start 버튼을 눌렀을 때의 좌표값을 활용해 랜덤하게 row 값을 정해 array 에 있는 미션을 한 줄 고른다.
- 타이머가 끝나면 릴레이 모듈이 모두 돌아가고 피에조도 전원 인가가 된다. 초음파 센서로 받은 값이 일정 값 이하일 때 랜덤하게 좌회전 또는 우회전을 실행하며 다시 일정 값 이상이 되면 전진한다.
- 인체감지센서를 미션으로 화면에 출력되는 번호와 맞는 감지 센서를 인지시키면 미션 카운트가 하나씩 감소한다. 주어진 미션 카운트를 모두 완료해 0 으로 만들면 릴레이 모듈과 피에조가 OFF 된다.

```
int loop_num;
int array[15][12] = {
      /* LEVEL 1 */
      {1, 4, 2, 3, -1, -1, -1, -1, -1, -1, -1, -1 },
      {1, 2, 4, 3, -1, -1, -1, -1, -1, -1, -1, -1 },
      {4, 3, 1, 2, -1, -1, -1, -1, -1, -1, -1, -1 },
      {3, 4, 2, 1, -1, -1, -1, -1, -1, -1, -1, -1 },
      {1, 1, 2, 4, -1, -1, -1, -1, -1, -1, -1, -1 },

      /* LEVEL 2 */
      {2, 3, 3, 3, 2, 3, 4, 2, -1, -1, -1, -1 },
      {1, 1, 2, 2, 3, 2, 4, 4, -1, -1, -1, -1 },
      {1, 2, 3, 3, 4, 4, 4, 4, -1, -1, -1, -1 },
      {4, 2, 1, 2, 3, 2, 3, 2, -1, -1, -1, -1 },
      {3, 1, 1, 3, 4, 2, 2, 4, -1, -1, -1, -1 },
```

```
      /* LEVEL 3 */
      {1, 1, 2, 4, 2, 3, 1, 3, 2, 4, 2, 1},
      {2, 3, 2, 1, 3, 4, 4, 4, 2, 1, 1, 2},
      {2, 2, 2, 2, 2, 3, 3, 3, 4, 1, 1, 4},
      {1, 4, 2, 3, 4, 2, 3, 4, 2, 1, 1, 1},
      {1, 2, 3, 4, 2, 3, 4, 4, 2, 1, 2, 3}
};


int main() {
   uint32_t dist;
   int modulo;
   int rest_count = 0;
   int random_going = 0;

   SystemInit();
   LCD_Init();
   Touch_Configuration();
   Touch_Adjust();
   LCD_Clear(WHITE);

   RCC_Configure();
   GPIO_Configure();
   USART_Configure();
   TIM_Configure();
   NVIC_Configure();

   LCD_ShowString(20,10,"Time: ", BLACK, WHITE);
   LCD_ShowString(20,50,"Set: ", BLACK, WHITE);
   LCD_ShowString(20,210,"Start", BLACK, WHITE);
   LCD_ShowString(20,100,"Mission: ", BLACK, WHITE);
   LCD_ShowString(20,150,"Count: ", BLACK, WHITE);
   LCD_DrawRectangle(10, 200, 70, 230);

   GPIO_ResetBits(GPIOC, GPIO_Pin_0);
   GPIO_SetBits(GPIOC, GPIO_Pin_8);
   GPIO_SetBits(GPIOC, GPIO_Pin_9);

   while(controlflag != 0) {
      Touch_GetXY(&pos_x, &pos_y, 1);
      Convert_Pos(pos_x, pos_y, &con_x, &con_y);
      if( con_x > 10 && con_x < 70 && con_y > 200 && con_y < 230 ) {
         controlflag = 0;
      }
   }
```

```
if(Array_Input[6] == 1) {
   rest_count = 4;
   modulo = (con_x * con_y)%5;
} else if(Array_Input[6] == 2) {
   rest_count = 8;
   modulo = (con_x * con_y)%5 + 5;
} else {
   rest_count = 12;
   modulo = (con_x * con_y)%5 + 10;
}

while(1) {
   if(controlflag == 1) {
      LCD_ShowNum(150, 150, rest_count, 2, BLACK, WHITE);
      LCD_ShowNum(150, 100, array[modulo][loop_num], 2, BLACK, WHITE);

      GPIO_SetBits(GPIOC, GPIO_Pin_0);
      GPIO_ResetBits(GPIOC, GPIO_Pin_8);
      GPIO_ResetBits(GPIOC, GPIO_Pin_9);
      dist = HCSR04GetDistance();
      random_going = dist%2;

      if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_0) != 0) {
         GPIO_ResetBits(GPIOC, GPIO_Pin_0);
         osa_mdelay(30);
         GPIO_SetBits(GPIOC, GPIO_Pin_0);
         osa_mdelay(30);
         GPIO_ResetBits(GPIOC, GPIO_Pin_0);
         osa_mdelay(30);
         GPIO_SetBits(GPIOC, GPIO_Pin_0);
      }

      if( dist < 600 && random_going == 0 ) {
         while(dist < 600) {
            GPIO_ResetBits(GPIOC, GPIO_Pin_9);
            GPIO_SetBits(GPIOC, GPIO_Pin_8);
            dist = HCSR04GetDistance();
         }
      } else if( dist < 600 && random_going == 1 ) {
         while(dist < 600) {
            GPIO_ResetBits(GPIOC, GPIO_Pin_8);
            GPIO_SetBits(GPIOC, GPIO_Pin_9);
            dist = HCSR04GetDistance();
         }
      }
```

```
         if(      (GPIO_ReadInputDataBit(GPIOD,      GPIO_Pin_12)      !=      0)      &&
(array[modulo][loop_num] == 1) ) {
            loop_num = loop_num + 1;
            rest_count = rest_count - 1;
            osa_mdelay(500);
         }   else   if(   (GPIO_ReadInputDataBit(GPIOD,   GPIO_Pin_11)   !=   0)   &&
(array[modulo][loop_num] == 2) ) {
            loop_num = loop_num + 1;
            rest_count = rest_count - 1;
            osa_mdelay(500);
         }   else   if(   (GPIO_ReadInputDataBit(GPIOB,   GPIO_Pin_6)   !=   0)   &&
(array[modulo][loop_num] == 3) ) {
            loop_num = loop_num + 1;
            rest_count = rest_count - 1;
            osa_mdelay(500);
         }   else   if(   (GPIO_ReadInputDataBit(GPIOC,   GPIO_Pin_7)   !=   0)   &&
(array[modulo][loop_num] == 4) ) {
            loop_num = loop_num + 1;
            rest_count = rest_count - 1;
            osa_mdelay(500);
         } else {
            GPIO_SetBits(GPIOD, GPIO_Pin_2);
            GPIO_SetBits(GPIOD, GPIO_Pin_3);
            GPIO_SetBits(GPIOD, GPIO_Pin_4);
            GPIO_SetBits(GPIOD, GPIO_Pin_7);
//            osa_mdelay(500);
         }

         if(rest_count == 0) {
            GPIO_ResetBits(GPIOC, GPIO_Pin_0);
            GPIO_SetBits(GPIOC, GPIO_Pin_8);
            GPIO_SetBits(GPIOC, GPIO_Pin_9);
            GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            GPIO_ResetBits(GPIOD, GPIO_Pin_3);
            GPIO_ResetBits(GPIOD, GPIO_Pin_4);
            GPIO_ResetBits(GPIOD, GPIO_Pin_7);
            LCD_ShowNum(150, 150, rest_count, 2, BLACK, WHITE);
            break;
         }
      }
   }

   return 0;
}
```

4.2 납땜 및 연결 구현

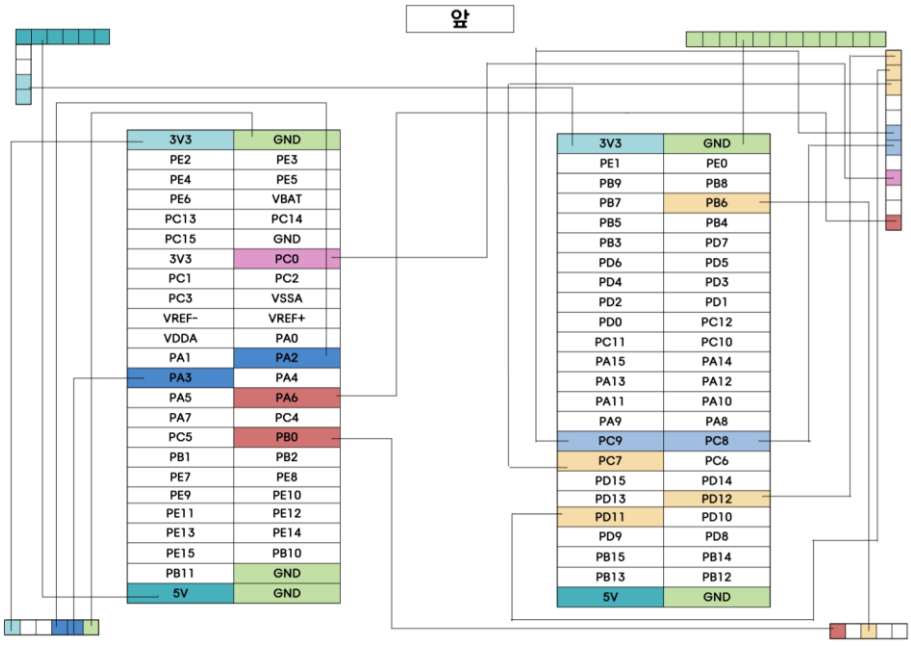
4.2.1 납땜한 핀

HC-SR501 인체 감지 센서 (4)	- PB 6, PC7, PD11, PD12 - GND (4) - VCC[5V] (4)
릴레이 모듈 (2)	- PC8, PC9 - GND (2) - VCC [3.3V] (2)
HC-SR04 초음파 센서 (1)	- PA6(Echo) - PB0(Trigger) - GND (2) - VCC [5V] (2)
SM-1205C PIEZO 부저 (1)	- PC0 - GND - VCC [5V]
블루투스 모듈 (1)	- PA2(USART2 TX) - PA3(USART2 RX) - GND VCC [3.3V] * 괄호 안의 숫자는 수량을 의미

4.2.2 실험에 사용한 핀 (별도의 납땜 및 연결 X)

LCD 모니터	- PA5. PC(5 ~ 12), PD(13 ~15), PE(0 ~ 15)
LED (4)	- PD2, PD3, PD4, PD7
USART1	- PA9(USART1 TX) - PA10(USART1 RX)

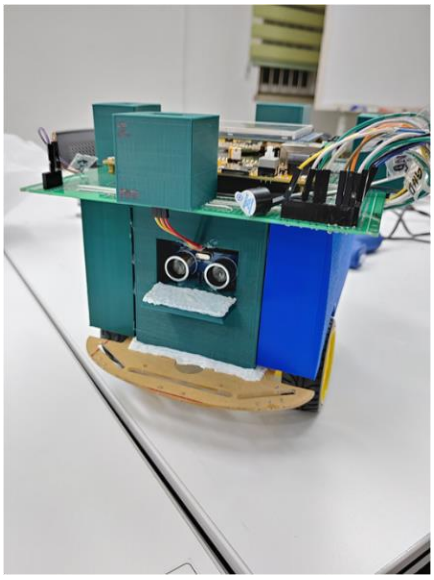
4.2.3 납땜 회로도



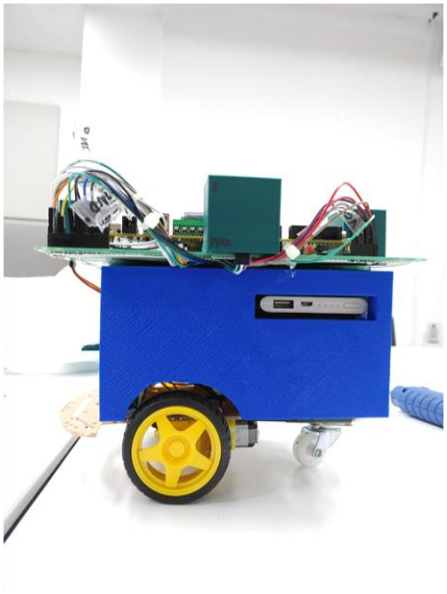
4.3 하드웨어 구현

다음과 같은 이유로 하드웨어를 3D 프린팅으로 제작하였다.

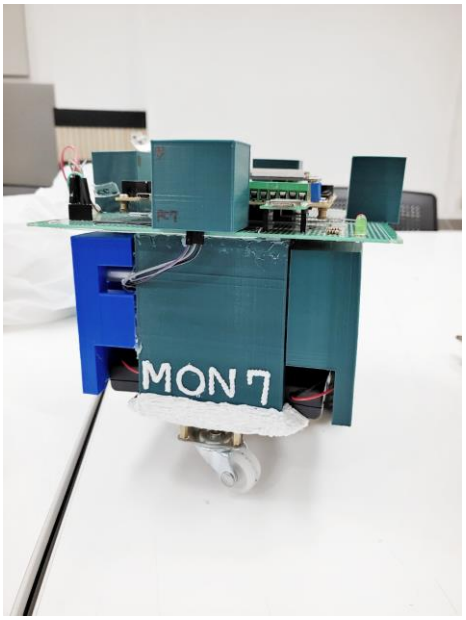
- 보드를 RC 카 위에 올리기 위해
- 민감한 인체감지센서를 감싸기 위해
- 복잡한 전선 연결을 가리기 위해



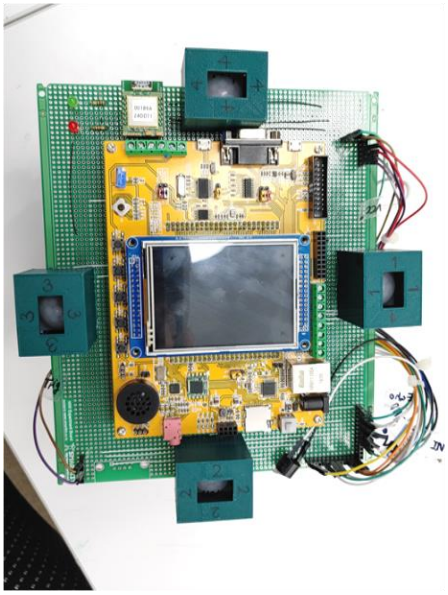
〈 앞 면 〉



〈 옆 면 〉



〈 뒷 면 〉



〈 윗 면 〉

05. 프로젝트 결과

5.1 전체 소스코드

도망가는 미션 알람의 전체 소스코드는 다음과 같다.

```
/*
flash load "C:\longlife3\flashclear.axf"
flash load "C:\longlife3\Debug\longlife3.axf"
*/
#include "misc.h"
#include "core_cm3.h"
#include "stm32f10x.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include <stm32f10x_tim.h>
#include <stm32f10x_adc.h>
#include "lcd.h"
#include "touch.h"

uint16_t pos_x, pos_y, con_x, con_y;
int t1, t2, t3;
int Array_Input[7];
int controlflag=2;      // 초기화
int idx = 0;
int Hour, Min, Sec;

int loop_num;
int array[15][12] = {
        /* LEVEL 1 */
        {1, 4, 2, 3, -1, -1, -1, -1, -1, -1, -1, -1 },
        {1, 2, 4, 3, -1, -1, -1, -1, -1, -1, -1, -1 },
        {4, 3, 1, 2, -1, -1, -1, -1, -1, -1, -1, -1 },
        {3, 4, 2, 1, -1, -1, -1, -1, -1, -1, -1, -1 },
        {1, 1, 2, 4, -1, -1, -1, -1, -1, -1, -1, -1 },

        /* LEVEL 2 */
        {2, 3, 3, 3, 2, 3, 4, 2, -1, -1, -1, -1 },
        {1, 1, 2, 2, 3, 2, 4, 4, -1, -1, -1, -1 },
        {1, 2, 3, 3, 4, 4, 4, 4, -1, -1, -1, -1 },
        {4, 2, 1, 2, 3, 2, 3, 2, -1, -1, -1, -1 },
        {3, 1, 1, 3, 4, 2, 2, 4, -1, -1, -1, -1 },
```

```
	/* LEVEL 3 */
	{1, 1, 2, 4, 2, 3, 1, 3, 2, 4, 2, 1},
	{2, 3, 2, 1, 3, 4, 4, 4, 2, 1, 1, 2},
	{2, 2, 2, 2, 2, 3, 3, 3, 4, 1, 1, 4},
	{1, 4, 2, 3, 4, 2, 3, 4, 2, 1, 1, 1},
	{1, 2, 3, 4, 2, 3, 4, 4, 2, 1, 2, 3}
};


void __attribute__((weak)) osa_mdelay(unsigned int msec)
{
	uint32_t temp;
	SysTick->LOAD=(uint32_t)msec*(HSE_VALUE/1000);
	SysTick->VAL =0x00;	// clear Count flag
	SysTick->CTRL=0x01;
	do {
		temp=SysTick->CTRL;
	}
	while((temp&0x01)&&!(temp&(1<<(16))));	// wait Count flag set
	SysTick->CTRL=0x00;
	SysTick->VAL =0X00;
}

void Make_Time(void) {
	Hour = Array_Input[0]*10 + Array_Input[1];
	Min = Array_Input[2]*10 + Array_Input[3];
	Sec = Array_Input[4]*10 + Array_Input[5];
}

void RCC_Configure() {
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
				RCC_APB2Periph_GPIOD | RCC_APB2Periph_USART1|
RCC_APB2Periph_GPIOE |
				RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1, ENABLE);
}
```

```
void GPIO_Configure() {
	GPIO_InitTypeDef GPIO_InitStructure;

	// USART1 TX
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;   //PA9
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	// USART1 RX
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;   //PA10
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	// USART2 TX
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;   //PA2
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	// USART2 RX
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	/* 초음파 */
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
	GPIO_Init(GPIOB, &GPIO_InitStructure);

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
	GPIO_Init(GPIOA, &GPIO_InitStructure);

	/* 릴레이관련 */
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
GPIO_Init(GPIOC, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

/* LED */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);
GPIO_ResetBits(GPIOD, GPIO_Pin_2); // Set C13 to Low level ("0")

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7; //LED
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_ResetBits(GPIOD, GPIO_Pin_2); // Set C13 to Low level ("0")

/* 인체감지 */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

```
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_Init(GPIOC, &GPIO_InitStructure);

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_Init(GPIOB, &GPIO_InitStructure);

	/* 피에조 */
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
	GPIO_Init(GPIOC, &GPIO_InitStructure);
}

void TIM_Configure() {
	RCC_ClocksTypeDef RCC_ClocksStatus;
	uint16_t prescaler;
	TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
	TIM_OCInitTypeDef TIM_OCInitStruct;
	TIM_ICInitTypeDef TIM_ICInitStruct;

	TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

	/* relay */
	RCC_GetClocksFreq(&RCC_ClocksStatus);
	prescaler = RCC_ClocksStatus.SYSCLK_Frequency / 1000000 - 1; //1 tick = 1us (1 tick = 0.165mm
resolution)

	TIM_DeInit(TIM3);
	TIM_TimeBaseInitStruct.TIM_Prescaler = prescaler;
	TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
	TIM_TimeBaseInitStruct.TIM_Period = 0xFFFF;
	TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
	TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

	TIM_OCStructInit(&TIM_OCInitStruct);
	TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
	TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
```

```
	TIM_OCInitStruct.TIM_Pulse = 15; //us
	TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
	TIM_OC3Init(TIM3, &TIM_OCInitStruct);

	TIM_ICInitStruct.TIM_Channel = TIM_Channel_1;
	TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
	TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
	TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
	TIM_ICInitStruct.TIM_ICFilter = 0;

	TIM_PWMIConfig(TIM3, &TIM_ICInitStruct);
	TIM_SelectInputTrigger(TIM3, TIM_TS_TI1FP1);
	TIM_SelectMasterSlaveMode(TIM3, TIM_MasterSlaveMode_Enable);
	TIM_CtrlPWMOutputs(TIM3, ENABLE);
	TIM_ClearFlag(TIM3, TIM_FLAG_Update);

	/* usart */
	TIM_TimeBaseStructure.TIM_Period = 12000;
	TIM_TimeBaseStructure.TIM_Prescaler = 6000;
	TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
	TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
	TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

	TIM_ARRPreloadConfig(TIM2, ENABLE);
	TIM_Cmd(TIM2, ENABLE);
	TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}

void USART_Configure() {
	USART_InitTypeDef USART_InitStructure;
	USART_InitTypeDef USART2_InitStructure;

	/*TODO: USART1 configuration*/
	USART_InitStructure.USART_BaudRate = 9600;
	USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
	USART_InitStructure.USART_Parity = USART_Parity_No;
	USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART_InitStructure.USART_WordLength = USART_WordLength_8b;
	USART_InitStructure.USART_StopBits = USART_StopBits_1;

	USART2_InitStructure.USART_BaudRate = 9600;
```

```
	USART2_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
	USART2_InitStructure.USART_Parity = USART_Parity_No;
	USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
	USART2_InitStructure.USART_StopBits = USART_StopBits_1;

	USART_Init(USART1, &USART_InitStructure);
	USART_Init(USART2, &USART2_InitStructure);

	/*TODO: USART1 cmd ENABLE*/
	USART_Cmd(USART1,ENABLE);
	USART_Cmd(USART2,ENABLE);

	/*TODO: USART1 IT Config*/
	USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
	USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
}

void NVIC_Configure() {
	NVIC_InitTypeDef NVIC_InitStructure;

	NVIC_InitTypeDef NVIC_USART_InitTD;
	NVIC_InitTypeDef NVIC_USART2_InitTD;

	NVIC_USART_InitTD.NVIC_IRQChannel = USART1_IRQn;
	NVIC_USART2_InitTD.NVIC_IRQChannel = USART2_IRQn;

	NVIC_USART_InitTD.NVIC_IRQChannelCmd = ENABLE;
	NVIC_USART2_InitTD.NVIC_IRQChannelCmd = ENABLE;


	NVIC_USART_InitTD.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_USART2_InitTD.NVIC_IRQChannelPreemptionPriority = 0;

	NVIC_USART_InitTD.NVIC_IRQChannelSubPriority = 0;
	NVIC_USART2_InitTD.NVIC_IRQChannelSubPriority = 0;

	NVIC_Init(&NVIC_USART_InitTD);
	NVIC_Init(&NVIC_USART2_InitTD);

	NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
```

```
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);
}

int32_t HCSR04GetDistance() {
	(TIM3)->CNT = 0;
	TIM_Cmd(TIM3, ENABLE);
	while(!TIM_GetFlagStatus(TIM3, TIM_FLAG_Update));
	TIM_Cmd(TIM3, DISABLE);
	TIM_ClearFlag(TIM3, TIM_FLAG_Update);
	return (TIM_GetCapture2(TIM3)-TIM_GetCapture1(TIM3))*165/1000;
}

int my_atoi(char c) {
	int value_int;
	value_int = c - 48;
	return value_int;
}

void USART2_IRQHandler(void) {
	if(USART_GetITStatus(USART2, USART_IT_RXNE ) != RESET) {
		char Input = USART_ReceiveData(USART2);
		if(48 <= Input && Input <= 57){
			Array_Input[idx] = my_atoi(Input);
			idx++;
			if(idx == 7){
				Make_Time();
				idx = 0;
			}
		}
	}
	USART_ClearITPendingBit(USART2,USART_IT_RXNE);
}

void TIM2_IRQHandler() {
	if(TIM_GetITStatus(TIM2, TIM_IT_Update)!=RESET) {
		if(controlflag==0) {
			t1++;
			if(t1==60) {
```

```
                        t2++;
                        t1 = 0;
                }
                if(t2==60) {
                        t3++;
                        t2 = 0;
                }
                if(t1==Sec && t2==Min && t3==Hour){
                        controlflag=1;
                }
        }
        else if(controlflag == 1) {
                t1 = 0, t2 = 0, t3 = 0;
        }

        LCD_ShowNum(100, 50, Hour, 2, BLACK, WHITE);
        LCD_ShowNum(130, 50, Min, 2, BLACK, WHITE);
        LCD_ShowNum(160, 50, Sec, 2, BLACK, WHITE);
        LCD_ShowNum(200, 50, Array_Input[6],1, BLACK, WHITE);

        // 시간 출력
        LCD_ShowNum(100, 10, t3, 2, BLACK, WHITE);
        LCD_ShowNum(130, 10, t2, 2, BLACK, WHITE);
        LCD_ShowNum(160, 10, t1, 2, BLACK, WHITE);
    }
    TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}

int main() {
    uint32_t dist;
    int modulo;
    int rest_count = 0;
    int random_going = 0;

    SystemInit();
    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    LCD_Clear(WHITE);

    RCC_Configure();
```

```
GPIO_Configure();
USART_Configure();
TIM_Configure();
NVIC_Configure();

LCD_ShowString(20,10,"Time: ", BLACK, WHITE);
LCD_ShowString(20,50,"Set: ", BLACK, WHITE);
LCD_ShowString(20,210,"Start", BLACK, WHITE);
LCD_ShowString(20,100,"Mission: ", BLACK, WHITE);
LCD_ShowString(20,150,"Count: ", BLACK, WHITE);
LCD_DrawRectangle(10, 200, 70, 230);

GPIO_ResetBits(GPIOC, GPIO_Pin_0);
GPIO_SetBits(GPIOC, GPIO_Pin_8);
GPIO_SetBits(GPIOC, GPIO_Pin_9);

while(controlflag != 0) {
        Touch_GetXY(&pos_x, &pos_y, 1);
        Convert_Pos(pos_x, pos_y, &con_x, &con_y);
        if( con_x > 10 && con_x < 70 && con_y > 200 && con_y < 230 ) {
                controlflag = 0;
        }
}

if(Array_Input[6] == 1) {
        rest_count = 4;
        modulo = (con_x * con_y)%5;
}
else if(Array_Input[6] == 2) {
        rest_count = 8;
        modulo = (con_x * con_y)%5 + 5;
} else {
        rest_count = 12;
        modulo = (con_x * con_y)%5 + 10;
}

while(1) {
        if(controlflag == 1) {
                LCD_ShowNum(150, 150, rest_count, 2, BLACK, WHITE);
                LCD_ShowNum(150, 100, array[modulo][loop_num], 2, BLACK, WHITE);
```

```
                        GPIO_SetBits(GPIOC, GPIO_Pin_0);
                        GPIO_ResetBits(GPIOC, GPIO_Pin_8);
                        GPIO_ResetBits(GPIOC, GPIO_Pin_9);
                        dist = HCSR04GetDistance();
                        random_going = dist%2;

                        if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_0) != 0) {
                                GPIO_ResetBits(GPIOC, GPIO_Pin_0);
                                osa_mdelay(30);
                                GPIO_SetBits(GPIOC, GPIO_Pin_0);
                                osa_mdelay(30);
                                GPIO_ResetBits(GPIOC, GPIO_Pin_0);
                                osa_mdelay(30);
                                GPIO_SetBits(GPIOC, GPIO_Pin_0);
                        }

                        if( dist < 600 && random_going == 0 ) {
                                while(dist < 600) {
                                        GPIO_ResetBits(GPIOC, GPIO_Pin_9);
                                        GPIO_SetBits(GPIOC, GPIO_Pin_8);
                                        dist = HCSR04GetDistance();
                                }
                        }
                        else if( dist < 600 && random_going == 1 ) {
                                while(dist < 600) {
                                        GPIO_ResetBits(GPIOC, GPIO_Pin_8);
                                        GPIO_SetBits(GPIOC, GPIO_Pin_9);
                                        dist = HCSR04GetDistance();
                                }
                        }

                        if( (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_12) != 0) && (array[modulo][loop_num]
== 1) ) {
                                loop_num = loop_num + 1;
                                rest_count = rest_count - 1;
                                osa_mdelay(500);
                        }
                        else if( (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_11) != 0) &&
(array[modulo][loop_num] == 2) ) {
                                loop_num = loop_num + 1;
                                rest_count = rest_count - 1;
                                osa_mdelay(500);
```

```
                        }
                        else if( (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_6) != 0) &&
(array[modulo][loop_num] == 3) ) {
                                loop_num = loop_num + 1;
                                rest_count = rest_count - 1;
                                osa_mdelay(500);
                        }
                        else if( (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_7) != 0) &&
(array[modulo][loop_num] == 4) ) {
                                loop_num = loop_num + 1;
                                rest_count = rest_count - 1;
                                osa_mdelay(500);
                        }
                        else {
                                GPIO_SetBits(GPIOD, GPIO_Pin_2);
                                GPIO_SetBits(GPIOD, GPIO_Pin_3);
                                GPIO_SetBits(GPIOD, GPIO_Pin_4);
                                GPIO_SetBits(GPIOD, GPIO_Pin_7);
                                // osa_mdelay(500);
                        }

                        if(rest_count == 0) {
                                GPIO_ResetBits(GPIOC, GPIO_Pin_0);
                                GPIO_SetBits(GPIOC, GPIO_Pin_8);
                                GPIO_SetBits(GPIOC, GPIO_Pin_9);
                                GPIO_ResetBits(GPIOD, GPIO_Pin_2);
                                GPIO_ResetBits(GPIOD, GPIO_Pin_3);
                                GPIO_ResetBits(GPIOD, GPIO_Pin_4);
                                GPIO_ResetBits(GPIOD, GPIO_Pin_7);
                                LCD_ShowNum(150, 150, rest_count, 2, BLACK, WHITE);
                                break;
                        }
                }
        }

        return 0;
}
```

5.2 결과 및 동영상

5.2.1 결과

결론적으로 말하면, 제안서의 내용대로 대부분 구현하였다. 구체적인 구현을 하는 과정에서 조금 달라진 점도 있다. 제안서에서는 LED 를 통해 미션 문제를 출력하기로 계획했지만 실제 구현할 때에는 LCD 화면에서 값을 직접적으로 보여주는 것으로 미션을 출력하였다. LED 는 단순히 타이머가 종료되고 미션 수행 상태일 때 알리는 용도로 점등하는 것으로 수정하였다. 그리고 제안서에는 구체적으로 언급하지 않았지만 블루투스 모듈에서 시간을 전송하는 즉시 타이머를 작동시키는 것으로 계획하였었다. 그러나 실용적인 측면에서 고려해볼 때 미리 일정 시간을 정해두고 사용자가 원할 때 타이머를 시작하게 하는 것이 더 합리적이라고 판단하였다. 그래서 스마트폰으로 시간을 전송한 후 LCD 화면의 start 버튼을 눌러야 타이머가 작동되도록 수정하였다.

이 외에는 대부분의 기능들이 제안서대로 구현되었다. 스마트폰의 블루투스 통신을 통해 도망가는 미션 알람 본체에 시간과 난이도를 보내고, start 버튼을 누르면 타이머가 작동한다. 입력한 시간이 흐르고 나면 타이머는 멈춤과 동시에 알람이 시작된다. LED 는 점등되고, PIEZO 는 시끄러운 소리를 낸다. 릴레이 모듈들은 두 바퀴를 구동시키고, 직진하던 알람의 초음파에 장애물이 탐지되면 임의의 바퀴를 정지시켜 방향을 전환한다. LCD 에는 감지시켜야 할 인체감지센서 번호가 미션으로 출력되고, 사용자가 정해진 횟수만큼 인체감지센서를 모두 감지시키면 도망가는 미션 알람은 모든 작동을 중지한다. 여기까지 시행하고 나면 프로젝트가 목표했던 바와 같이 사용자는 깊은 잠에서 깨 있을 것이다.

5.2.2 시연 동영상

아래의 QR 코드를 찍어서 확인할 수 있다.

* 영상을 바로 클릭하면 파일 다운로드가 되니 링크를 통해 보는 것을 권장한다.

06. 결론

6.1 장 수현

7 조의 조장을 맡아 실험과 텀 프로젝트를 진행했다. 텀 프로젝트에서는 약 한달정도 힘찬 햄과 블루투스 통신과 알람 기능 부분을 구현했는데 한달 째 안될 때는 너무 막막했었다. 그래도 힘찬 햄이랑 계속 디버깅을 했고, 조교님의 힌트로 결국 해결했다. 거기서부터 차근차근 하다보니 우리 텀 프로젝트는 누구보다도 빠르게 마무리 되어있었다. 또 프로젝트를 진행하면서 크게 맡은 역할은 납땜이었다. 납땜을 여러 번에 걸쳐서 했는데 조원들은 내가 납땜할 게 많아서 힘들었다고 생각하는 것 같지만 난 솔직히 재밌었다. 그리고 역할 분담을 나눠 적으려고 생각해보니 우리 조는 모든 것을 같이 해서 사실 크게 나눌 수가 없더라. 보드 연결부터, 코딩, 디버깅, 납땜 등 네 명이 모두 열심히 참여했다. 진짜 납땜을 조원 다 같이 한 조는 우리 조밖에 없었을 거라 자부한다. 누구 하나 빠져서는 안됐었던 프로젝트를 진행했고, 완성했다고 생각한다. 그래서 다 함께 만든 결과물이 매우 만족스럽고 정이 간다. 그리고 보고서 쓰는 것에 조금 집착했는데 그게 좀 미안하다. 어쨌든 조장을 잘 이끌어준 조원들에게 너무 고맙다.

6.2 박 창조

실험을 할 때도 주로 코드를 짜는 역할을 맡아서 프로젝트를 진행할 때도 코드를 주로 만들었습니다. 그리고 특히 인체감지센서, 초음파센서, 릴레이모듈, 피에조와 관련된 코딩을 맡았습니다. 릴레이 모듈은 실험을 하면서 구현해본 센서이지만 나머지는 센서들은 처음 사용하는 센서라 처음에는 두려움도 많았습니다. 그러나 조원과 함께 코드를 만들고 하드웨어를 다루면서 진행하다보니 그렇게 두렵고 어려운 일이지는 않았습니다. 그리고 다른 실험들과 달리 C 언어를 사용하여 코딩하다 보니 익숙하고 조금만 적응하니 하드웨어 센서와 연결, 설정을 다루는 방법도 능숙하게 코딩할 수 있었습니다. 그러나 이번 실험은 하드웨어를 사용하여 진행하는 실험인 만큼 그에 따른 어려움도 있었습니다. 저희가 생각한대로 코딩을 하고 하드웨어 연결도 했는데 원하는 결과가 나오지 않을 때 디버깅이 어려웠습니다. 소프트웨어의 문제인가, 하드웨어의 문제인가 정확하게 알지 못하고 그에 따른 변수를 하나 하나씩 따져보며 디버깅을 진행하니 다른 때의 디버깅보다 훨씬 많은 시간이 걸리고 또 이 과정에서 실수가 발생하거나 하면 더 많은 시간이 걸렸습니다. 그래도 저는 지금까지의 실험들 중에 가장 좋은 조원들을 만났던 것 같고 가장 즐겁게 실험을 진행했던 것 같습니다. 감사합니다.

6.3 임 다영

주로 맡은 역할은 실험에 사용될 포트와 핀을 분배하고, 센서들의 데이터시트를 참고하여 하드웨어와 연결하고 테스트 해보는 일이었다. 처음에는 막연하게 '지금까지 사용하지 않은 핀을 사용할 센서에 분배하면 되겠지' 하고 시작했는데 실험을 할 때 왜 데이터시트와 schematic 을 주셨는지 뒤늦게 깨닫게 되었다. 별다른 연결이나 납땜이 없더라도 LCD 나 USART 통신 같은 경우에는 코드와 schematic 을 보고 사용하는 기능과 새로 배치할 기능이 겹치지 않도록 핀 분배를 했어야 했는데, 그 점을 고려하지 않고 핀을 분배해서 로직상에 문제도 없고 납땜에도 문제가 없는데 LCD 모니터에 출력이 제대로 되지 않는 일이 발생했다. 센서들을 테스트 해 볼 때에도 schematic 을 읽고 해당 기능을 수행하는 핀으로 테스트를 진행했어야 했는데 그렇지 못해 초반에 시간이 지체되었다. 실수 때문에 납땜을 추가적으로 해야 했고 로직을 짜신 분 역시도 문제가 없는데 계속 문제를 찾아야 하는 일이 생겨서 정말 죄송했다. 앞으로 다른 프로젝트를 하게 된다면 데이터시트와 schematic 을 꼼꼼히 읽고 충분한 고민을 거친 후에 진행해야겠다는 생각을 했다. 실수로 인해 시간이 지체되었음에도 아무 내색 없이 팀을 이끌어준 조장님과 조원님들께 정말 감사하다.

6.4 이 힘찬

프로젝트를 진행하면서 맡은 역할은 블루투스를 이용한 알람 및 난이도 설정 부분 구현과 코딩 보조였다. 블루투스 부분을 구현하면서 핸드폰에서 보낸 문자 값이 제대로 들어오지 않아서 몇 주 동안 고생했는데 조원분들이랑 상의하면서 해결할 수 있었다. 이후에 코드를 합치면서 겹치는 핀 때문에 고생했지만 최종적으로 완성했을 때 잘 작동하는 모습을 보면서 뿌듯했다. 사실 처음에 프로젝트 제안서를 만들 땐 이걸 구현 할 수 있을까라는 의문이 있었는데 유능한 조원 분들 덕분에 수월하게 프로젝트를 진행 할 수 있었던 거 같다. 좋은 분들은 만나서 좋았고 이때까지 진행한 프로젝트 중에서 시간은 가장 많이 들었지만 가장 배운 게 많고 재밌었던 프로젝트였다. 많은 도움을 주신 조교님 그리고 조원 분들에게 다시 한 번 감사의 말씀을 전하고 싶다.