11주차 보고서

월요일 분반 2조

조원)

201524566 정경호

201724539 이재욱

201724579 정현모

201814331 이해민

실험 목표

1. 타이머 이해 및 실습
2. 모터 이해 및 실습

실험 과정

1. 기본 설정

1. LCD, PWM, LED 사용에 필요한 부분들을 Configure 한다.

```
void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIO_InitStructure3;
    GPIO_InitStructure3.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_7;
    GPIO_InitStructure3.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure3.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure3); // led 활성화
}

void NVIC_Configure(void) {

   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
   NVIC_InitTypeDef NVIC_InitStructure;

   /* Enable TIM2 Global Interrupt */
   NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);
}
```

```
void PWM_Configure()
{
   prescale = (uint16_t) (SystemCoreClock / 10000);
   TIM_TimeBaseStructure.TIM_Period = 10000;
   TIM_TimeBaseStructure.TIM_Prescaler = prescale;
   TIM_TimeBaseStructure.TIM_ClockDivision = 0;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;

   TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
   TIM_ARRPreloadConfig(TIM2, ENABLE);
   TIM_Cmd(TIM2, ENABLE);
}
```

GPIO, NVIC 를 Configure 하는 부분은 기존에 사용하던 방식을 그대로 사용하였고, PWM을 새롭게 선언했다. 1초에 한번씩 작동하도록 설정해주기 위해서 (1/72Mhz) * 10000 * (72M/10000) =1 의 계산식을 통해 1초로 지정해주었다.

2. LED 두개를 동시에 제어하기 위해서 인터럽트에 대한 핸들러 함수를 선언 및 구현한다.

```
void TIM2_IRQHandler(void) {
  if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
  {
      if (ledCount % 2 == 0) {
      GPIO_SetBits(GPIOD, GPIO_Pin_4);
      } else{
      GPIO_ResetBits(GPIOD, GPIO_Pin_4);
      }

      if(ledCount == 0) {
      GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      } else if(ledCount == 5) {
      GPIO_SetBits(GPIOD, GPIO_Pin_7);
      }

      ledCount++;
      ledCount %= 10;

      TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  }
}
```

하나의 인터럽트로 두가지의 LED를 제어한다. 하나의 LED는 1초마다 토글을 반복하고 하나의 LED는 5초에 한번씩 작동하도록 설정해준다.

3. LED ON 버튼 터치를 인식하기 위해 터치값 좌표를 입력받아 좌표값이 사각형 내부면 TIM2 interrupt를 활성화시킨다.

```
  while (1) {
         Touch_GetXY(&pos_temp[0], &pos_temp[1], 1); // 터치 좌표 받아서 배열에 입력
         Convert_Pos(pos_temp[0],pos_temp[1],&pos_temp[0],&pos_temp[1]); // 받은 좌표를 LCD 크기에 맞게 변환
         printf("%d, %d\n", pos_temp[0], pos_temp[1]);

         if ((uint16_t)40 < pos_temp[0] && pos_temp[0] < (uint16_t)80 && \
             (uint16_t)80 < pos_temp[1] && pos_temp[1] < (uint16_t)120) {
            if (flag) {
               LCD_ShowString(90, 50, "    ", RED, WHITE);
```

```
            LCD_ShowString(40, 50, "ON", RED, WHITE);
            TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); // interrupt enable
         } else {
            LCD_ShowString(90, 50, "OFF", RED, WHITE);
            LCD_ShowString(40, 50, "    ", RED, WHITE);
            TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE); // interrupt disable
         }
         flag++;
         flag %= 2;
      }
```

터치한 좌표의 값이 사각형 내부라고 인식되면 ON을 띄우고 TIM2를 실행한다. 좌표값이 사각형 외부라면 ON을 지우고 OFF을 출력한다. 그리고 사각형 외부 좌표값이 인식된다면 인터럽트를 실행시키지 않는다. 따라서 LED 토글도 진행되지 않도록 한다.

2. LCD에 team 이름, LED토글 ON/OFF 상태, LED ON/OFF 버튼 생성

1. lcd.c의 LCD_ShowString 함수와 LCD_DrawRectangle 함수를 사용해 팀명, 사각형, 글자를 각각 출력한다.

```
LCD_ShowString(40, 10, "MON_Team02", MAGENTA, WHITE); // 팀명 출력
LCD_ShowString(90, 50, "OFF", RED, WHITE); // 디폴트로 OFF 출력
LCD_DrawRectangle(40, 80, 80, 120); // 사각형 출력
LCD_ShowString(60,100, "Button", MAGENTA, WHITE); // "Button" 글자 출력
```

3. LED ON 버튼 터치 시 TIM2 interrupt를 활용하여 LED 2개 제어 및 LED OFF 버튼 터치 시 LED Toggle 동작 해제

```
  GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
  GPIO_ResetBits(GPIOD, GPIO_Pin_13);   // LCD_RS(0);
  GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);

  GPIO_Write(GPIOE, LCD_Reg);

  GPIO_SetBits(GPIOD, GPIO_Pin_14); // LCD_WR(1);
  GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
```

LCD_CS, LCD_RS, LCD_WR를 Low로 둔 뒤 Command를 전송, 그 후에 다시 LCD_CS와 LCD_WR를 High로 둔다.

4. TIM3 타이머를 이용한 모터 제어

1. PWM을 사용하기 위해서 Configure 부분에 추가적으로 선언을 해준다.

```
void PWM_Configure()
{
   prescale = (uint16_t) (SystemCoreClock / 1000000);
   TIM_TimeBaseStructure.TIM_Period = 20000;
   TIM_TimeBaseStructure.TIM_Prescaler = prescale;
   TIM_TimeBaseStructure.TIM_ClockDivision = 0;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
   TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
   TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
   TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
   TIM_OCInitStructure.TIM_Pulse = 1500; // us
   TIM_OC3Init(TIM3, &TIM_OCInitStructure);
   TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
   TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);
   TIM_ARRPreloadConfig(TIM3, ENABLE);
   TIM_Cmd(TIM3, ENABLE);
}
```

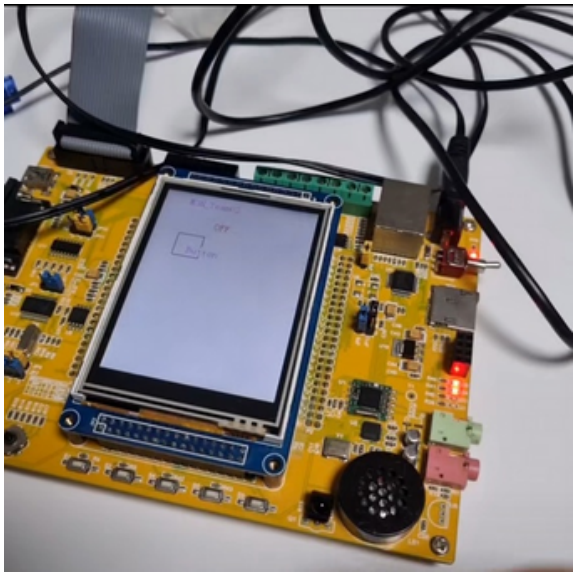
2. main 함수에서 모터를 순차적으로 움직이도록 코드를 작성한다.

```
int main(void)
{

    SystemInit();

    RCC_Configure();

    GPIO_Configure();

    PWM_Configure();

    uint16_t rotation[2] = {700, 2300};
    int i = 0;
    while (1) {
      printf("%d", TIM_OCInitStructure.TIM_Pulse);
      TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
      TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
      TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
      TIM_OCInitStructure.TIM_Pulse = rotation[i]; // us
      TIM_OC3Init(TIM3, &TIM_OCInitStructure);
      i++;
      i %= 2;
      delay();
    }
    return 0;
}
```

rotation 배열을 이용해 pulse값을 700과 2300을 반복하도록 코드를 작성해서 모터가 반복적으로 움직이도록 한다.

실험 결과

1. 처음 실행시켰을 때 팀이름이 출력되고 OFF 상태임. 박스와 버튼이 출력됨을 확인.



2. 박스 내부를 터치한 결과 OFF가 사라지고 ON이 출력됨과 동시에 LED 토글이 진행.