2021년 2학기 임베디드 시스템 설계 및 실험

- 11주차 보고서 -

004분반 9조

개요

이번 실험에서는 LCD와 타이머를 통하여 LED 및 서보모터를 제어하는 실습을 진행하였다.

목표

- TIM 및 PWM을 사용한 LED와 서보모터 제어

세부실험내용

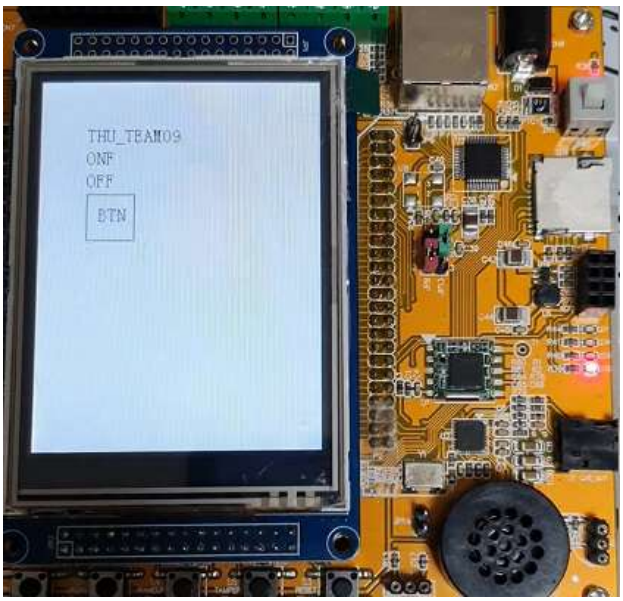
- LCD에 team 이름, LED 토글 ON/OFF 상태, LED ON/OFF 버튼 생성
- LED ON 버튼 터치 시 TIM2 interrupt를 활용하여 LED 2개 제어 (1초마다 LED1, 5초마다 LED2 Toggle)
- LED OFF 버튼 터치 시 LED Toggle 동작 해제
-서보 모터 제어

개념

- 타이머 : 주기적 시간 처리에 사용하는 디지털 카운터 회로 모듈
- Systick Timer : 24bit down counter , Counter가 0에 도달하면 설정에 따라 인터럽트가 발생
- WATCHDOG(WDG) : 특수 상황에서 CPU가 올바르게 작동하지 않을 시 강제로 리셋시키는 기능
- General-purpose timers (TIM2 to TIM5) : prescaler를 이용해 설정 가능한 16-bit up, down, up/down auto-reload counter를 포함 및 입력 신호의 펄스 길이 측정(input capture) 또는 출력 파형 발생(output compare and PWM) 등 다양한 용도로 사용
- PWM : 일정한 주기 내에서 Duty ratio를 변화 시켜서 평균 전압을 제어하는 방법

실험과정

- 실험



-그림1) 버튼 누를 시 1초마다 LED1 Toggle

-그림2) 5초마다 LED2 Toggle

-그림3) LED 동시에 ON

-그림4) LED와 서보 모터 동시 동작

- 코드

```
12  /* function prototype */
13  void RCC_Configure(void);
14  void GPIO_Configure(void);
15  void NVIC_Configure(void);
16  void TIM_Configure(void);
17  void changePWM(uint16_t pulse);
18
19  void TIM2_IRQHandler(void);
20
21  uint16_t prescale;
22  uint16_t motorFlag = 3;
23  uint16_t ledPowerFlag = 0;
24  uint16_t led1ToggleFlag = 0;
25  uint16_t led2ToggleFlag = 0;
26  uint16_t led1Counter = 0;
27  uint16_t led2Counter = 0;
28  int color[12] = {WHITE, CYAN, BLUE, RED, MAGENTA, LGRAY, GREEN, YELLOW, BROWN, BRRED, GRAY};
29  char* ledStatus[2] = {"OFF", "ON"};
30
31  //-------------------------------------------------------------------------------------------
32
33  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
34  TIM_OCInitTypeDef TIM_OCInitStructure;
35
36  void RCC_Configure(void) // stm32f10x_rcc.h 참고
37  {
38      // TIM2 clock enable
39      RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
40      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
41
42      RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
43      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
44      /* LED pin clock enable */
45  }
46
47  void GPIO_Configure(void) // stm32f10x_gpio.h 참고
48  {
49      GPIO_InitTypeDef GPIO_InitStructure;
```

```
50
51      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3;
52      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
53      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
54      GPIO_Init(GPIOD, &GPIO_InitStructure);
55
56      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
57      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
58      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
59      GPIO_Init(GPIOB, &GPIO_InitStructure);
60  }
61
62  void TIM_Configure(void) {
63      TIM_TimeBaseStructure.TIM_Period = 10000;
64      TIM_TimeBaseStructure.TIM_Prescaler = 7200;
65      TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
66      TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
67      TIM_TimeBaseStructure.TIM_RepetitionCounter = 0;
68      TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
69      TIM_ARRPreloadConfig(TIM2, ENABLE);
70
71      TIM_TimeBaseStructure.TIM_Prescaler = (uint16_t) (SystemCoreClock / 1000000) - 1;
72      TIM_TimeBaseStructure.TIM_Period = 20000-1;
73      TIM_TimeBaseStructure.TIM_ClockDivision = 0;
74      TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
75      TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
76      TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
77      TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
78      TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
79      TIM_OCInitStructure.TIM_Pulse = 1500; // us
80      TIM_OC3Init(TIM3, &TIM_OCInitStructure);
81      TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);
82      TIM_ARRPreloadConfig(TIM3, ENABLE);
83
84      TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
85      TIM_Cmd(TIM2, ENABLE);
86      TIM_Cmd(TIM3, ENABLE);
87  }
88
89  void NVIC_Configure(void) { // misc.h 참고
90      NVIC_InitTypeDef NVIC_InitStructure;
91      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
92
93      //TIM2
94      NVIC_EnableIRQ(TIM2_IRQn);
95      NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
96      NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0;
97      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x1;
98      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
```

```
    NVIC_Init(&NVIC_InitStructure);
}

void TIM2_IRQHandler(void) {
    if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET) {
        led2Counter = (led2Counter+1) % 5;
        if(motorFlag++ == 13){ motorFlag = 3; }
        led1ToggleFlag = !led1ToggleFlag;
        led2ToggleFlag = led2ToggleFlag ^(!led2Counter);
        TIM_ClearITPendingBit(TIM2,TIM_IT_Update);
    }
}

void change(int per)
{
    int pwm_pulse;
    pwm_pulse = per * 20000 / 100;
    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 = pwm_pulse;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);
}

int main(void)
{
    SystemInit();
    RCC_Configure();
    GPIO_Configure();
    TIM_Configure();
    NVIC_Configure();

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

    uint16_t rawTouchX = 0;
    uint16_t rawTouchY = 0;
    uint16_t touchX = 0;
    uint16_t touchY = 0;

    while (1) {
        change(motorFlag);
        LCD_ShowString(40, 40, "THU_TEAM09", BLACK, WHITE);
        LCD_ShowString(40, 60, ledStatus[led1ToggleFlag & ledPowerFlag], BLACK, WHITE);
        LCD_ShowString(40, 80, ledStatus[led2ToggleFlag & ledPowerFlag], BLACK, WHITE);
        LCD_DrawRectangle(40, 100, 80, 140);
```

```
148         LCD_ShowString(50, 110, "BTN", BLACK, WHITE);
149         GPIO_Write(GPIOD, ((GPIO_Pin_2 * led1ToggleFlag) | (GPIO_Pin_3 * led2ToggleFlag))*ledPowerFlag);
150 
151         Touch_GetXY(&rawTouchX, &rawTouchY, 0); //Wait until Touched
152         Convert_Pos(rawTouchX, rawTouchY, &touchX, &touchY);
153         if(touchX >= 40 && touchX <= 100 && touchY >= 80 && touchY <= 140) {
154             ledPowerFlag = !ledPowerFlag;
155             touchX = 0;
156             touchY = 0;
157         }
158     }
159     return 0;
160 }
161 
```

결론

이번 실험에서는 LCD를 통하여 TIM를 통한 LED Toggle 및 서보모터를 제어하는 실험을 진행하였다. 실험을 진행하면서 LED Toggle 부분까지는 큰 문제없이 진행하였으나 서보 모터 부분에서 차질을 겪였다. 타이머의 RCC에 직접 값을 할당하는 방식으로 진행하였었는데 이와 달리 타이머의 펄스 값을 수정하여 재할당 함으로써 해결할 수 있었다. 또한 서보 모터의 회전속도를 낮추기 위해서 딜레이를 사용한다면 간단하겠지만 TIM2의 인터럽트에 영향이 끼칠 수 있기 때문에 딜레이 함수를 사용하는 것이 아닌 다른 방법을 사용하는 것(MAP 등)을 고려해보아야겠다.