임베디드시스템 설계 및 실험 보고서

[002 분반 - 2 조 - 11 주차]

조원	202055531 김후겸 202055584 이태경 202155540 김채현 202255535 김진우

실험날짜	2024-11-13

1. 실험 주제

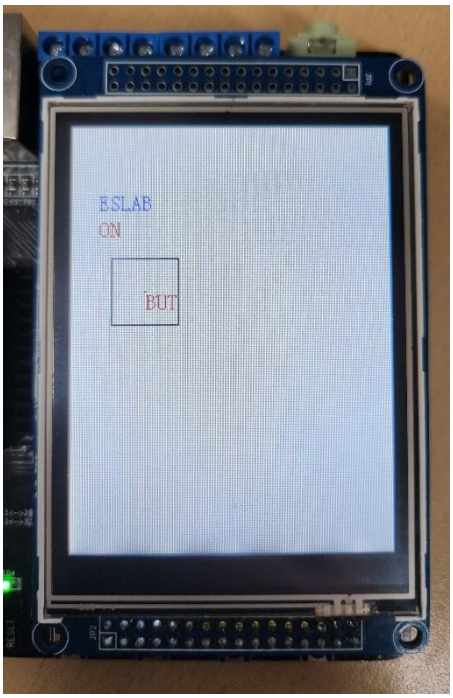
- Timer 와 PWM

2. 실험 목적

- Timer 의 종류와 특징의 이해
- PWM 신호의 이해
- 분주 계산 방법

3. 세부 실험 목적

1. TFT LCD 에 team 이름, LED 토글 ON/OFF 상태, LED ON/OFF 버튼 생성
2. LED ON 버튼 터치 시 TIM2 interrupt, TIM3 PWM 을 활용하여 LED 2 개와 서보모터 제어
3. LED OFF 버튼 터치 시 LED Toggle 동작 해제 및 서보모터 동작 반전



4. 실험 장비

- STM32F107VCT6
- TFT-LCD

5. 실험 과정

1. RCC & GPIO Configuration

```
void RCC_Configure(void){
    // ADC port clock Enable
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //TMI2
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //TMI3
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE); //LED

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE); //LED 1,2
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE); //motor

    /* Alternate Function IO clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
}
```

```
void GPIO_Configure(void){
    GPIO_InitTypeDef  GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0; // TIM2
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0; //TIM3
    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_2; //LED1
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3; //LED2
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

- LED 제어를 위한 타이머로 TIM2 를 사용한다.
- 서보모터 제어를 위한 타이머로 TIM3 를 사용한다.
- 타이머를 위한 GPIOA, GPIOB 를 enable 한다.
- LED 사용을 위해 GPIOD 를 enable 한다.

- 타이머는 APB1 에 속하므로 APB1 주변장치로 타이머를 Enable 한다.

2. Timer 설정

```
#if defined (STM32F10X_LD_VL) || (defined STM32F10X_MD_VL) || (
/* #define SYSCLK_FREQ_HSE    HSE_VALUE */
 #define SYSCLK_FREQ_24MHz  24000000
#else
/* #define SYSCLK_FREQ_HSE    HSE_VALUE */
/* #define SYSCLK_FREQ_24MHz  24000000 */
/* #define SYSCLK_FREQ_36MHz  36000000 */
/* #define SYSCLK_FREQ_48MHz  48000000 */
/* #define SYSCLK_FREQ_56MHz  56000000 */
#define SYSCLK_FREQ_72MHz  72000000
#endif
```

system_stm32f10x.c 파일을 보면 우리가 사용하는 주파수가 무엇인지 알 수 있다. 위 사진을 보면 72MHz 를 사용한다.

```
void TIM_Configure(void){
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;

    TIM_TimeBaseStructure.TIM_Period = 10000 ;
    TIM_TimeBaseStructure.TIM_Prescaler = 7200;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;

    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
    TIM_Cmd(TIM2, ENABLE);
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);

}
```

타이머의 클락을 우리가 사용하기 쉬운 값으로 맞춰주기 위해 분주를 계산한다. 시스템 클락이 72MHz 이므로 Period 를 10000, Prescaler = 7200 으로 분주하여 1 초로 설정한다. 이렇게 하면 타이머는 1 초마다 한번씩 cnt 값을 변화시킬 수 있다.

설정이 끝났으면 Init, cnt 를 통해 타이머를 활성화 시키고 ITConfig 를 통해 TIM2 인터럽트를 활성화시킨다.

3. PWM 설정

```c
void PWM_Configure(void){
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;

    TIM_TimeBaseStructure.TIM_Period = 20000 ;
    TIM_TimeBaseStructure.TIM_Prescaler = 72;
    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= 2000;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);

    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
    TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);
    TIM_ARRPreloadConfig(TIM3, ENABLE);
    TIM_Cmd(TIM3, ENABLE);

    delay();

    TIM_Cmd(TIM3, DISABLE);
    TIM_OCInitStructure.TIM_Pulse= 1500;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);
    TIM_Cmd(TIM3, ENABLE);

    delay();

    TIM_Cmd(TIM3, DISABLE);
    TIM_OCInitStructure.TIM_Pulse= 1000;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);
    TIM_Cmd(TIM3, ENABLE);
```

- PWM 은 ON 과 OFF 시간의 비율을 변화시켜 전체적인 평균값을 조절해 모터의 속도를 제어할 수 있다. ARR 에 의해 주파수가 결정되고, CRR 에 의해 duty cycle 이 결정된다.
- ARR 은 Auto-reload register 로 상한선을 지정하는 레지스터이다. ARR 이 100 이면 100 까지 카운터가 진행되고 0 으로 돌아가서 자동 반복한다.
- CRR 은 x 번째 타이머의 y 번째 채널 출력을 조절할 수 있는 레지스터이다. CCR 값이 ARR 보다 큰 경우 또는 0 인 경우에는 PWM 기능이 동작하지 않는다.
- CNT 는 타이머의 카운터 값이 저장되어 있다. 설정된 주기에 맞게 1 씩 감소하다 0 을 만나면 인터럽트를 발생시킨다.
- Ouput Compare 는 TIMx_CNT 의 값과 TIMx_CCRx 레지스터 값이 일치할 때 Output Pin 상태를 제어할 수 있는 모드이다. TIM_OCInitTypeDef 구조체로 설정한다.
- PWM1 모드는 CNT 가 CCR 보다 작을 경우 Low, CNT 가 CCR 보다 큰 경우에 High 상태가 된다.
- Polarity 는 출력의 극성(Low, High)를 결정한다.

- OutputState 는 Output 채널의 enable/disable 설정을 담당한다.
- Pulse 는 CCR 레지스터의 값으로 CNT 와 비교하여 일치하면 인터럽트가 발생한다. 즉, Output 이 있는 경우에 해당 Output 상태를 변경 시킨다. 우리는 2us, 1.5us, 1us 로 설정하여 각각 90, 0, -90 의 회전각으로 딜레이를 두고 서보모터를 작동시켰다.
- OC3Preload 설정에서 OC3 의 전처리를 disable 하였다.
- ARRPreload 설정에서 ARR 의 전처리를 enable 하여 실행시 ARR 을 우선적으로 load 하게 하였다.
- Cmd 를 통해 TIM3 블록을 구동시켰다.
- 이 후, 서보모터 동작을 위해 이미 작동하던 TIM3 를 중단시키고 Pulse 값을 바꾼 후 서보모터가 동하게 하였다.

4. TIM2_IRQHandler 설정

```
void TIM2_IRQHandler(){
    if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET){
      if(btn){
          if(Tim2_cnt%2 == 0){
              GPIO_SetBits(GPIOD,GPIO_Pin_2);
          }
          else{
            GPIO_ResetBits(GPIOD, GPIO_Pin_2);
          }
          if(Tim2_cnt%5 == 0){
            if(flag_cnt5 == 0){
              GPIO_SetBits(GPIOD,GPIO_Pin_3);
              flag_cnt5 = 1;
            }
            else{
              GPIO_ResetBits(GPIOD, GPIO_Pin_3);
              flag_cnt5 = 0;
            }
          }
          Tim2_cnt++;
      }
      else{
        Tim2_cnt = 0;
        flag_cnt5 = 0;
      }
    }
    TIM_ClearITPendingBit(TIM2,TIM_IT_Update);
}
```

- LCD 버튼이 눌렸을 경우 1 초마다 LED 를 토글 시키기 위해서 Tim2 cnt 를 2 로 Modulation 연산을 하여 0,2,4... 초마다 LED1 이 켜지고 1,3,5 초마다 LED1 이 꺼짐으로 toggle 동작을 하도록 하였다.

- LCD 버튼이 눌렸을 경우 5 초마다 LED 를 toggle 시키기 위해서 Tim2_cnt 를 5 로 Modulation 연산을 하여 0,5,10,15...초마다 flag_cnt5 값이 0 이면 5 초동안 LED2 를키고, flag_cnt 값이 1 이면 5 초동안 LED 를 끈다. 그다음 onoff 동작을 위해 다시 if 문에들어오면 flag 값을반전시킨다.
- 위의 동작들을 한 번 진행하면, Tim2_cnt 값을 1 증가시킨다.
- 만약 버튼이 눌리지 않았을 때는 모든 LED 를 off 하기위해서 Tim2_cnt 값과 flag_cnt5 값을 모두 0 으로 설정한다

5. main

```
int main() {
  uint16_t x, y,nx,ny;

  SystemInit();
  RCC_Configure();
  GPIO_Configure();
  TIM_Configure();
  PWM_Configure();
  NVIC_Configure();

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

  char str[11] = "WED_Team02";

  LCD_ShowString(0, 0, str, BLACK, WHITE);
  LCD_DrawRectangle(50, 50, 100, 100);
  LCD_ShowString(60,65,"BUT",RED, WHITE);
  LCD_ShowString(0,20,"OFF",RED, WHITE);

  while(1){

      Touch_GetXY(&x,&y,1);
      Convert_Pos(x,y,&nx,&ny);
      if(nx >= 50 && nx <=100 && ny >=50 && ny <=100){
        if(btn == 0){
            btn = 1;
            LCD_ShowString(0,20," ON",RED, WHITE);
        }
        else{
            btn = 0;
            LCD_ShowString(0,20,"OFF",RED, WHITE);
            GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            GPIO_ResetBits(GPIOD, GPIO_Pin_3);

        }
      }
```

- 지난주 실습에서 진행한 LCD 를 활용하여 LCD 에 표시될 문자들을 설정하였다.(팀명, ON/OFF, 버튼)

- 버튼이 눌러졌을 때, 전역변수인 bnt 값을 1 로 바꾸어 핸들러가 이를 감지하도록 하였다. 또한 on,off 상태를 on 으로 바꿔준다.
- 버튼 이외의 영역이 눌러졌을 때는 아무 동작도 하지 않고, OFF 를 표시한다.

6. 실험 결과

7. 분석 및 결론

이번 실험에서는 PWM 신호를 이용한 서보 모터제어와 타이머 인터럽트를 활용한 LED 제어를 배웠다. 서보 모터의 각도를 조정하기 위해 PWM 신호의 듀티 사이클을 조절하는 과정을 실습하면서, 하드웨어 장치가 어떻게 신호를 인식하고 동작하는지에 대해 이해할 수 있었다. 또한, 타이머 인터럽트를 활용하여 LED 의 점멸 주기를 설정함으로써, 타이머가 보드 내에서 어떤 방식으로 동작하며, 이를 통해 정확한 타이밍으로 장치를 제어할 수 있다는 것을 확인하였다.