Busan Software Meister High School

MICROPROCESSOR

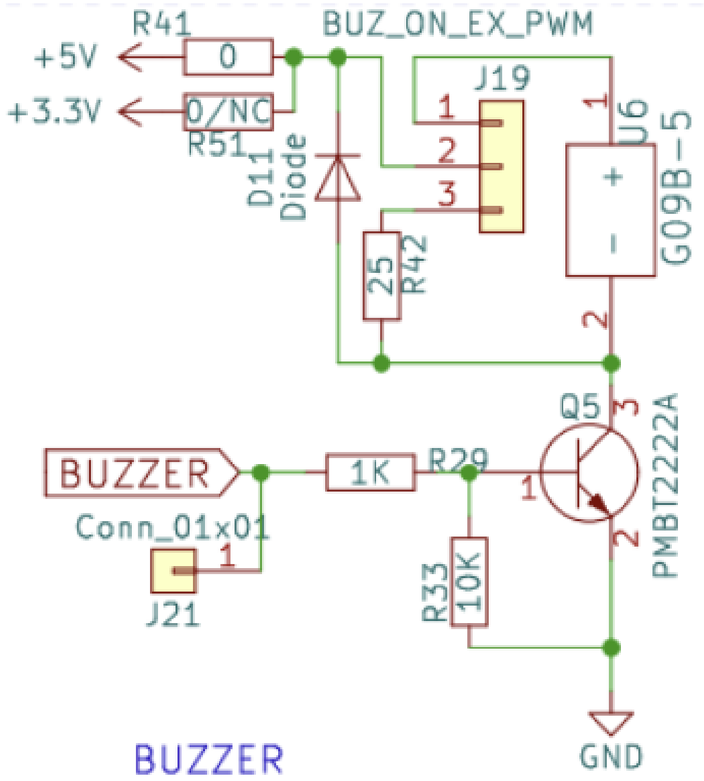
2309 양유빈

20231214 마이크로프로세서

버저 회로 설정

Buzzer circuit settings

- Active buzzer : 일정 전압을 가하면 소리가 나는 버저(단일 소리)
- Passive buzzer : 전압을 일정 주기로 HIGH-LOW-HIGH-LOW…를 반복하는 경우 소리가 나는 버저(음계 소리)
- 트랜지스터 회로를 이용하여 구성, GPIO 포트로 제어
- 전원과 콜렉터 사이에 보통 10Ω 정도의 저항을 직렬로 연결하는 방법을 많이 사용하지만 코일 자체 저항(10Ω 정도)이 있으므로 따로 연결하지 않아도 괜찮음
- 플라이휠(fly feel) 다이오드 버저 양단에 역방향으로 연결하여 트랜지스터 보호
  - 플라이휠(fly feel) 다이오드 : 유도기전력에 의한 순간적인 큰 전류를 빼내기 위한 다이오드



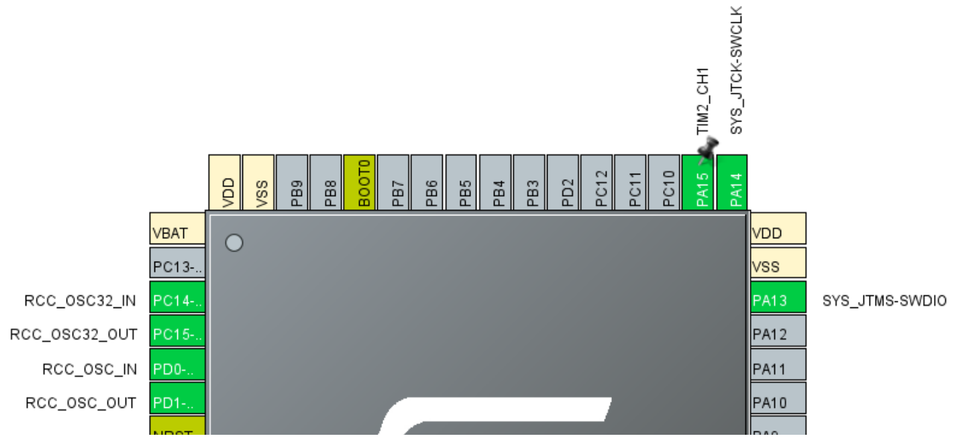
GPIO로 버저 제어

Buzzer control with GPIO

버저 제어 신호(PA15)를 HIGH와 LOW를 번갈아 가면서 사각파를 만들되, 주파수가 500Hz, Pulse(Duty Cycle)는 50%가 되도록 하며(1ms 동안 HIGH, 1ms 동안 LOW) 2초 동안 지속하도록 프로그래밍(2초 유지는 SysTick 활용)

$$\text{주파수 } f = \frac{1}{T} = \frac{f_{CLK}}{Prescaler \times Counter\ Period}$$

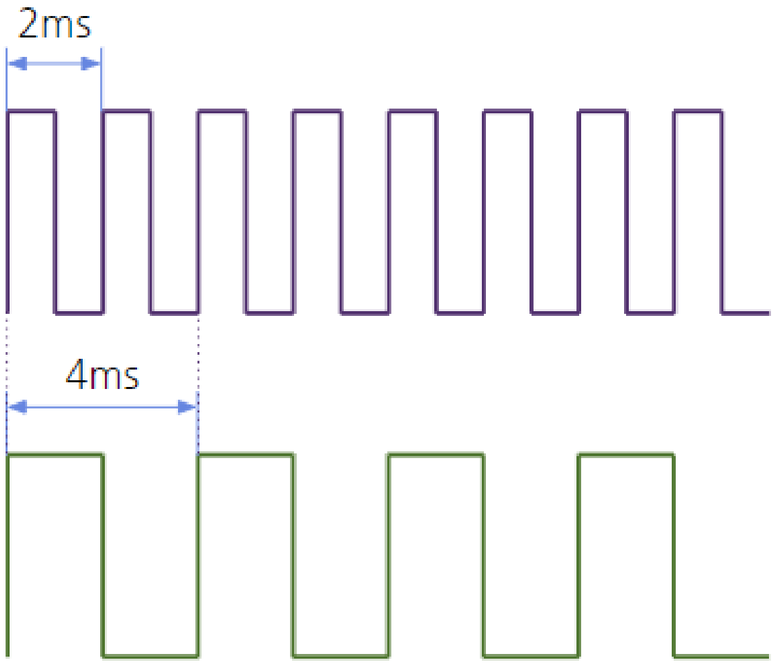
(f_{CLK} :클럭주파수, Prescaler:분주기, Counter Period(ARR):카운터 주기)




```
/* USER CODE BEGIN WHILE */
while (1)
{
    if(HAL_GetTick() < 2000){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 1);
        HAL_Delay(1);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 0);
        HAL_Delay(1);
    }
    else
        while(1);
```

GPIO로 1 옥타브 낮은 소리 재생

Play sound 1 octave lower with GPIO




```
for(int i=0;i<3;i++){
    for(int j=0;j<125;j++){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 1);
        HAL_Delay(2);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, 0);
        HAL_Delay(2);
    }
    HAL_Delay(500);
}
while(1);
```

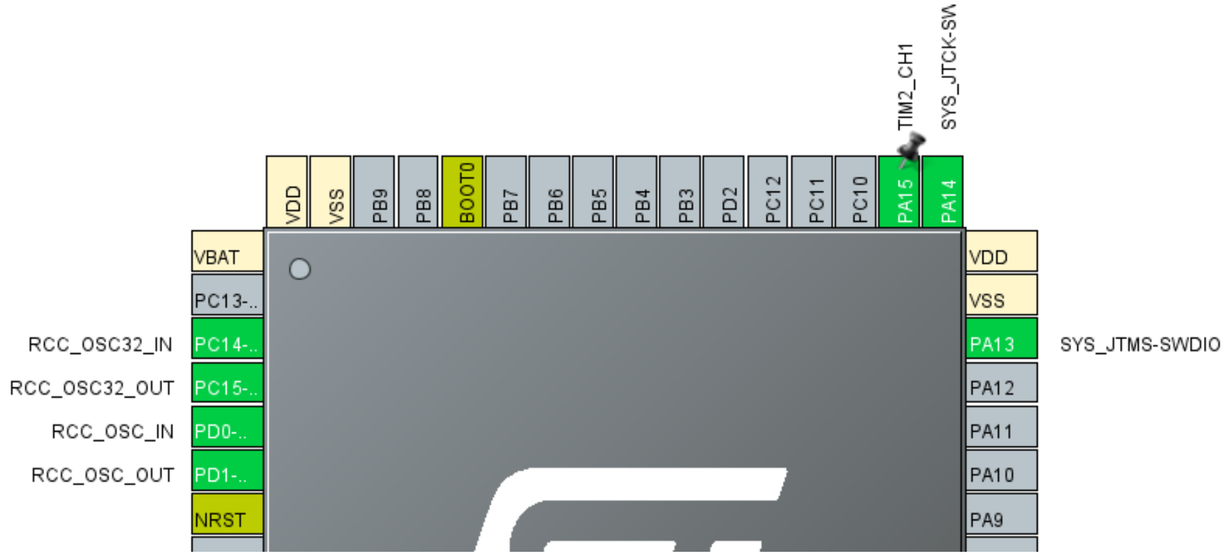
PWM 이용하여 버저 제어

Buzzer control using PWM

```
/* USER CODE BEGIN 2 */
__HAL_TIM_SET_AUTORELOAD(&htim2, 1000);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 500);
/* USER CODE END 2 */
```

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
while(HAL_GetTick() < 2000);
HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1);
while(1);
```

```
/* USER CODE END WHILE */
```



도레미파솔라시도 연주

Do-re-mi-fa-solassido performance

하나의 음계는 16분 음표 길이로 연주하고, 음계 사이에는 16분 음표 길이 만큼 쉼

음계주파수 $f_m = \frac{1}{T} = \frac{f_{CLK}}{Prescaler \times Counter\ Period}$ [Hz]

$Counter\ Period = \frac{f_{CLK}}{Prescaler \times f_m}$

$Duty\ Cycle = \frac{Counter\ Period}{2}$ [%] (f_{CLK} :클럭주파수, Prescaler:분주기, Counter Period(ARR):카운터 주기)

4분 음표의 주기(1분 동안 96번 연주되는 길이) $T_m = \frac{60 \times 1000}{96}$ [ms]

TIM2_CHANNEL1 : Clock = 72MHz, PSC = 71(72-1)

```
// 음계
#define DO_5    261.63
#define RE_5    293.66
#define MI_5    329.63
#define FA_5    349.23
#define SO_5    392.00
#define LA_5    440.00
#define TI_5    493.88
#define DO_6    523.25
#define REST    0       // 쉼표
#define END     -1      // 노래끝
#define MHZ_1   1000000 // 주파수
// 박자
#define N4      60*1000/96
#define N2      (N4*2)
#define N2N4    (N2+N4)
#define N8      (N4/2)
#define N4N8    (N4+N8)
#define N16     (N8/2)
/* USER CODE END PD */
```

음계 연주 +SysTick

Playing scales +SysTick

```
//int song_scale[] = {DO_5, RE_5, MI_5, FA_5, SO_5, LA_5, TI_5, DO_6, END};
int song_scale[] =
  {SO_5, SO_5, LA_5, LA_5, SO_5, SO_5, MI_5, SO_5, SO_5, MI_5, MI_5, RE_5, REST,
    SO_5, SO_5, LA_5, LA_5, SO_5, SO_5, MI_5, SO_5, MI_5, RE_5, MI_5, DO_5, REST, END };
int song_beat[] =
{ N4, N4, N4, N4, N4, N4, N2, N4, N4,N4, N4, N2N4, N4,
  N4, N4, N4, N4, N4, N4, N2, N4, N4, N4, N4, N2N4, N4, END };
int pre_tick;
/* USER CODE END PV */
```

```
void delay(int beat){
    pre_tick = HAL_GetTick();
    while(HAL_GetTick() - pre_tick < beat);
}
```

```
/* USER CODE BEGIN WHILE */
while (1)
{

  for(int i=0;song_scale[i] != END; i++){
      __HAL_TIM_SET_AUTORELOAD(&htim2, MHZ_1 / song_scale[i]);
      __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, ((MHZ_1 / song_scale[i])/2));
      HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
      delay(N16);
      HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1);
      delay(N16);
  }
  while(1);
```

학교종 연주

Playing the school bell

```
for(int i=0; song_scale[i] != END; i++){
    if(song_scale[i] == REST){
        delay(song_beat[i]);
    }
    else {
        __HAL_TIM_SET_AUTORELOAD(&htim2, MHZ_1 / song_scale[i]);
        __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, ((MHZ_1 / song_scale[i])/2));
        HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
        delay(song_beat[i]);
        HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1);
        delay(1);
    }
}
while(1);
```