LAB: Timer & PWM

Date: 2022-10-28

Author/Partner: 21800447 Jeahyun Oh / 21800805 SeungEung Hwang

Github: https://github.com/Ohjeahyun1/EC-jeahyun-447.git

Demo Video: https://youtu.be/YGgrhryD850

Introduction

In this lab, you are required to create two simple programs: toggling multiple LEDs with a push-button input and displaying the number counting from 0 to 9 at 1 second rate on a 7-segment display.

You must submit

- LAB Report (*.md & *.pdf)
- Zip source files(main*.c, ecRCC.h, ecGPIO.h, ecSysTick.c etc...).
    - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
    - NUCLEO-F411RE
- Actuator
    - 3 LEDs
    - RC Servo Motor (SG90)
- Sensor
    - Button(B1)
- Other
    - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Create HAL library

Create HAL library

Declare and Define the following functions in your library. You must

update your header files located in the directory `EC \lib\`.

ecTIM.h

```
/* Timer Configuration */
```

```
// Timer Period setup
void TIM_init(TIM_TypeDef *timerx, uint32_t msec);     //Timer initlization with
mode(PWM,TIM)

void TIM_period_ms(TIM_TypeDef* TIMx, uint32_t msec);           //Timer period
setting ms
void TIM_period_us(TIM_TypeDef* TIMx, uint32_t msec);           //Timer period
setting us
void TIM_period_test(TIM_TypeDef* TIMx, uint32_t msec);         //Timer period
setting ms for test

// Timer Interrupt setup
void TIM_INT_init(TIM_TypeDef* timerx, uint32_t msec); //Timer interrupt
initlization
void TIM_INT_enable(TIM_TypeDef* timerx);                       //Timer
interrupt enable
void TIM_INT_disable(TIM_TypeDef* timerx);                      //Timer
interrupt disable

// Timer Interrupt Flag
uint32_t is_UIF(TIM_TypeDef *TIMx);                             // Check Timer
interrupt
void clear_UIF(TIM_TypeDef *TIMx);                              // clear Timer
interrupt pending

// Timer value reset
void reset_TIMER(TIM_TypeDef *TIMx);                            //Timer value
reset
```

ecPWM.h

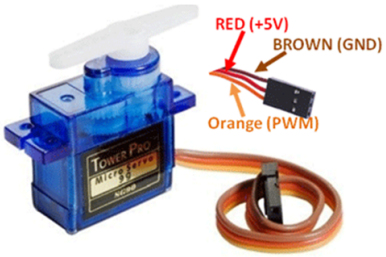
```
/* PWM STRUCTURE */
typedef struct {
    GPIO_TypeDef *port;
    int pin;
TIM_TypeDef *timer;
int ch;
} PWM_t;

/* PWM initialization */

// Default: 84MHz PLL, 1MHz CK_CNT, 50% duty ratio, 1msec period

void PWM_init(PWM_t *pwm, GPIO_TypeDef *port, int pin);

/* PWM PERIOD SETUP */

// allowable range for msec:  1~2,000

void PWM_period_ms(PWM_t *pwm,  uint32_t msec);

// allowable range for usec:  1~1,000

void PWM_period_us(PWM_t *pwm, uint32_t usec);
```

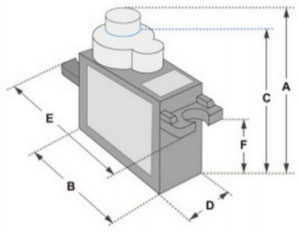
```
/* DUTY RATIO SETUP */
// High Pulse width in msec

void PWM_pulsewidth_ms(PWM_t *pwm, float pulse_width_ms);

// Duty ratio 0~1.0

void PWM_duty(PWM_t *pwm, float duty);
```

Problem 2: RC Servo motor

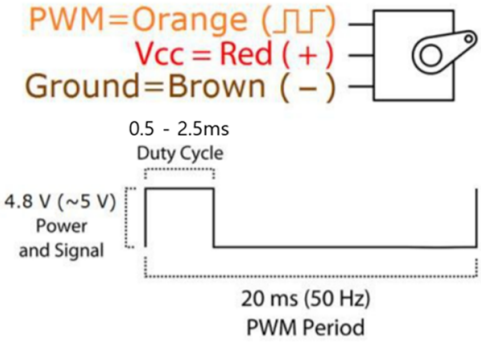
An RC servo motor is a tiny and light weight motor with high output power. It is used to control rotation angles, approximately 180 degrees (90 degrees in each direction) and commonly applied in RC car, and Small-scaled robots.

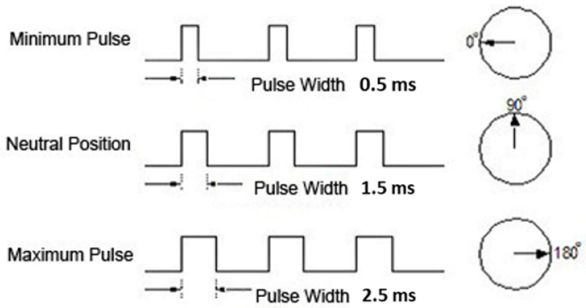
The angle of the motor can be controlled by the pulse width (duty ratio) of PWM signal. The PWM period should be set at **20ms or 50Hz**. Refer to the data sheet of the RC servo motor for detailed specifications.





Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6





Make a simple program that changes the angle of the RC servo motor that rotates with a given period of time and reset by pressing the push button (PC13).

- The button input has to be External Interrupt
- Use Port A Pin 1 as PWM output pin, for TIM2_Ch2.
- Use Timer interrupt of period 500msec.
- The angle of RC servo motor should rotate from 0° to 180° and back 0° at a step of 10° at the rate of 500msec.

You need to observe how the PWM signal output is generated as input button is pushed, using an oscilloscope. You need to capture the Oscilloscope output in the report.

Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_PWM_RCmotor`

- The project name is "**LAB_PWM_RCmotor".**
- Create a new source file named as "**LAB_PWM_RCmotor.c"**

> You MUST write your name on the source file inside the comment section.

2. Include your updated library in `\repos\EC\lib\` to your project.

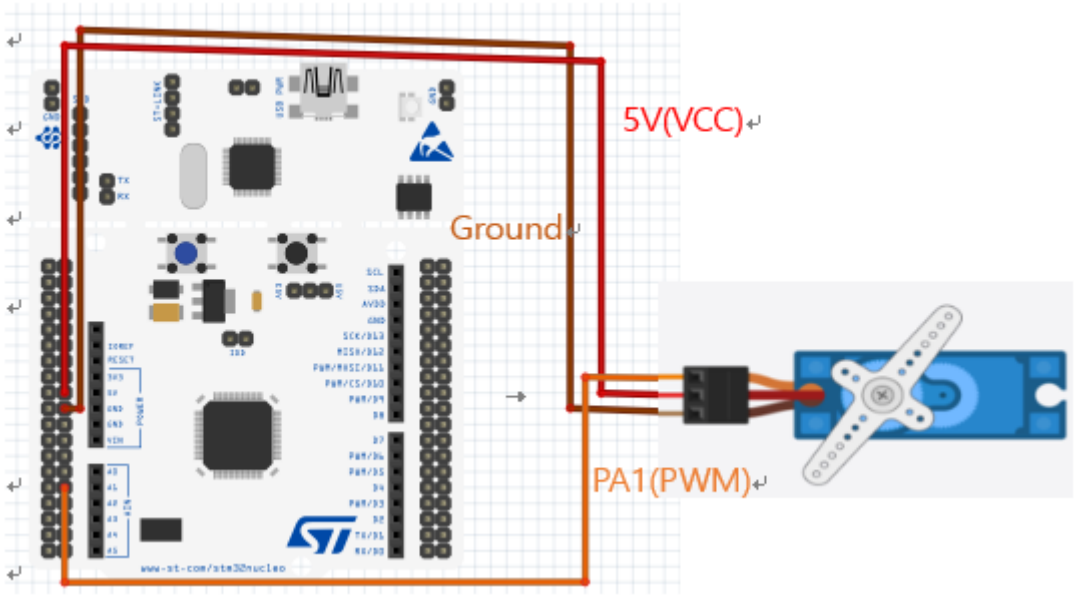
- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**
- **ecEXTI.h, ecEXTI.c**
- **ecTIM.h**, **ecTIM.c**
- **ecPWM.h ecPWM.h**

1. Connect the RC servo motor to MCU pin (PA1) , VCC and GND
2. Increase the angle of RC servo motor from 0° to 180° with a step of 10° every 500msec. After reaching 180°, decrease the angle back to 0°. Use timer interrupt IRQ.
3. When the button is pressed, it should reset to the angle 0° and start over. Use EXT interrupt.

Configuration

Button (B1)	PWM Pin	Timer	PWM
Digital In(PC 13)	AF(PA1)	TIM3	TIM2_CH2(PA1)
PULL-UP	Push-Pull	Timer Period:100usec	PWM period:20msec
	Pull-up, Fast	Timer Interrupt of 500msec	duty ratio: 0.5~2.5msec

Circuit Diagram



Discussion

1. Derive a simple logic to calculate for CRR and ARR values to generate xHz and y% duty ratio of PWM. How can you read the values of input clock frequency and PSC?

   Ck_CNT f = (Clock frequency /(PSC+1))

   xHz = Ck_CNT f/ (ARR+1)

   y% duty ratio = (CRR/(ARR+1))*100

   Read the system CLK

   read RCC->CFGR(clock configuration register) bit 3:2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

Bits 3:2 **SWS:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as the system clock.
00: HSI oscillator used as the system clock
01: HSE oscillator used as the system clock
10: PLL used as the system clock
11: not applicable

Read the PSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

Read Timx _> PSC (Timx = timer we use(Tim1,Tim2~5,Tim9~11))

2. What is the smallest and highest PWM frequency that can be generated for Q1?

   PWM frequency = 1/x

   So, the highest PWM frequency = (1+65535)*Ck_CNT_Period

   (ARR 16bits)

   the smallest PWM frequency = Ck_CNT_Period

Code

Your code goes here: https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/main/lab/LAB_PWM_RCmotor.c , https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/98436454e2a3247ed053afbcf25e371e2acae617/include/ecPWM.c , https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/98436454e2a3247ed053afbcf25e371e2acae617/include/ecTIM.c

Explain your source code with necessary comments.

Description with Code

- Code Initialization

```
// Initialization
void setup(void)
{
    RCC_PLL_init();                         // 84Mhz clock
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable() and button
pin mode -> input
    GPIO_pupdr(GPIOC, BUTTON_PIN, EC_PU); // GPIOC button pin pupdr -> pull up
  EXTI_init(GPIOC,BUTTON_PIN,FALL,0);   //EXTI button PIN -> trigger
type(falling),propriority(0)
    TIM_INT_init(TIM3,500,TIM);           // TIM3 Timer period: 100us ->
interrupt 500ms
    PWM_init(&pwm,GPIOA,1,UP,SFAST,1);    // TIM2_CH2(PA1) DOWN clock,FAST,1ms
clock
  PWM_period_ms(&pwm,20);                 // set PWM period 20ms
    GPIO_pupdr(GPIOA,1,EC_PU);            // GPIOA1 pull up
}
```

- Time interrupt

```
//Code about motor angle 0 to 180 degree and back angle 0 step of 10 degree
// Timer interrupt 500ms
void TIM3_IRQHandler(void){
    if(is_UIF(TIM3)){                       //interrupt occur
        if(dir == 0){                       //dir = forward
    PWM_duty(&pwm,0.025+0.05/9.0*i);      // motor angle 0~180
            i++;                              // angle value update
            }else if(dir == 1){               // if dir = backward
        PWM_duty(&pwm,0.125-0.05/9.0*i);    // motor angle 180~0
                i++;                            // angle value update
            }
        if(i>18.0){                         // not over 180 degree
        dir ^= 1;                           // dir update
        i=1.0;                              // motor angle reset
        }
     clear_UIF(TIM3);                    // clear by writing 0
    }
}
```

- EXT interrupt

```
// button pressed reset angle 0 and start over
void EXTI15_10_IRQHandler(void) {
    if (is_pending_EXTI(BUTTON_PIN)){     //when button pressed
      PWM_duty(&pwm,0.025);               // set motor angle 0
        i = 1.0;                          // reset angel value
        dir = 0;                          // forward
        reset_TIMER(TIM3);                // reset TIMer value
        clear_pending_EXTI(BUTTON_PIN);     // cleared by writing '1'
    }
}
```

- PWM_duty

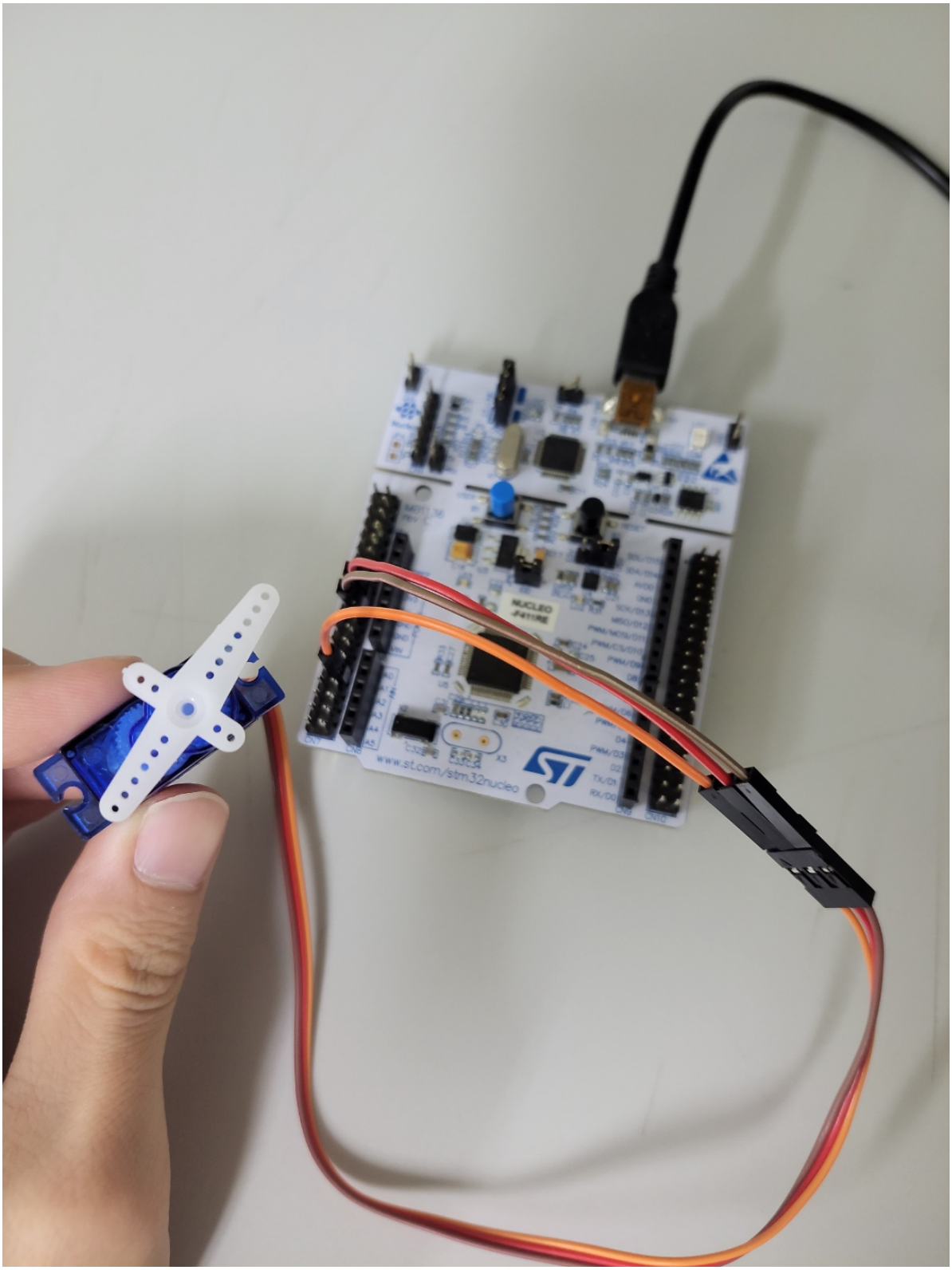
set PWM duty (0~1) (change the ccval)

```

```

```
void PWM_duty(PWM_t *pwm, float duty) {                 //  duty=0 to 1
        TIM_TypeDef *TIMx = pwm->timer;
    uint32_t arr= pwm->timer->ARR;

      float ccval = (arr+1)*duty-1;                                  //
(ARR+1)*dutyRatio - 1
        int CHn = pwm->ch;

        switch(CHn){
            case 1: TIMx->CCR1 = ccval; break;
            case 2: TIMx->CCR2 = ccval; break;
            case 3: TIMx->CCR3 = ccval; break;
            case 4: TIMx->CCR4 = ccval; break;
            default: break;
        }
}
```

Results

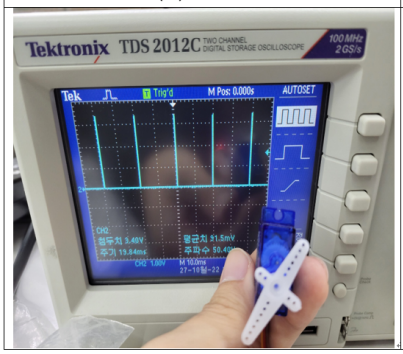
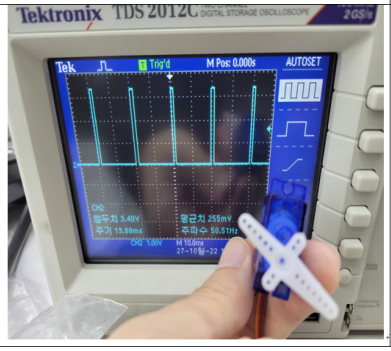
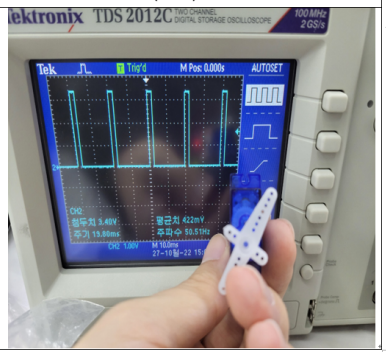
Experiment images and results



RC servo motor from 0° to 180° with a step of 10° every 500msec. After reaching 180°, decrease the angle back to 0°.

When button is pressed, reset angle 0° and start over.

Angle picture

Pulse Width 0.5ms(0°)	Pulse Width 1.5ms(90°)	Pulse Width 0.5ms(180°)
		

Add demo video link: https://youtu.be/YGgrhryD850

Reference

Complete list of all references used (github, blog, paper, etc)