Embedded Systems Lab 1 Report

Tz-Ching Yu

October 23, 2022

1 Introduction

The purpose of the lab is to construct a system that control the rotation of a servomotor with a joystick through a microcontroller. The system diagram is illustrated as below. The rotation of the servomotor

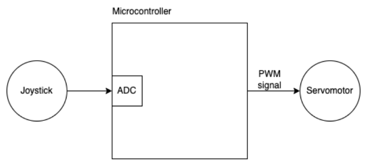

Figure 1: Overall system diagram

is controlled using PWM signal. When duty cycle is above some base value, the servomotor will be positioned to a clockwise degree, and vice versa. A timer is used to generate PWM signal to control the rotation of the servomotor. Analog signal from the joystick is converted to digital signal with ADC of the microcontroller, and digital signal is used to calculate the duty cycle which is then used to adjust PWM signal.

2 Implementation

2.1 Pin Configuration

P4.0 is used as input signal for ADC and is connected to **VRx** pin of joystick. **P2.4** is configured to output PWM signal using **TimerA0** and is connected to input of servomotor.

2.2 Microcontroller Configuration

For this lab, we utilize three peripherals: **TimerA0**, **TimerA1** and **14-bit ADC**, and we will describe the configuration of peripherals of microcontroller in this section.

As mentioned previously, **TimerA0** is used to generate PWM signal. It's configured to use auxiliary clock (**ACLK**) and set to count up mode. In order to achieve PWM functionality, we set the output mode to be Reset/Set mode, which means the timer would output high when the counter is below CCRn and output low when it's above CCRn and below CCR0. With the setup, we can easily set CCRn to period $\times$ duty_cycle and CCR0 to period. For CCRn, we use CCR1 for this purpose. Source code for initializing the timer is listed below.

```c
void init_PWM(int period, float duty_cycle)
{
    TIMER_A0->CTL = TIMER_A_CTL_TASSEL_1 | TIMER_A_CTL_MC_1;
    TIMER_A0->CCTL[1] = TIMER_A_CCTLN_OUTMOD_7;
    TIMER_A0->CCR[0] = ms2ticks(period);
    TIMER_A0->CCR[1] = ms2ticks(period * duty_cycle);
}
```

To interpret the orientation of the joystick, we configured the ADC to use **A13** as input and conversion mode to single-channel-repeat, and we trigger the conversion with software trigger, i.e. setting **ADC14_CTL0_SC** every a predefined time interval. We also utilize ADC interrupt to read out conversion result as soon as the value is written to ADC memory and, in ADC interrupt service routine, we adjust PWM duty cycle according Eq 1. The following code is used to initialize the ADC and setup ISR.

```
void init_ADC(int mem_addr)
{
    /* param mem_addr should be from 0 to 31 */

    // setting ADC input as A13
    ADC14->MCTL[0] = ADC14_MCTLN_INCH_13;

    // ADC14ON & select ACLK & select single channel repeat mode &
    // select sample clock as SAMPCON
    ADC14->CTL0 |= ADC14_CTL0_ON | ADC14_CTL0_SSEL_2 | ADC14_CTL0_SHP |
    ADC14_CTL0_CONSEQ_2;

    // select the output mem register for conversion
    ADC14->CTL1 |= mem_addr << ADC14_CTL1_CSTARTADD_OFS;

    // enable interrupt for mem[0]
    ADC14->IER0 |= ADC14_IER0_IE0;

    // enable conversion
    ADC14->CTL0 |= ADC14_CTL0_ENC;
}

void ADC14_IRQHandler(void)
{
    ADC14->CLRIFGR0 |= ADC14_CLRIFGR0_CLRIFG0;
    adjust_duty(SERVO_PERIOD, calculate_duty(ADC14->MEM[0]));
}
```

$$\text{Duty Cycle} = 0.025 + 0.1 \times \frac{\text{ADC value}}{16384} \tag{1}$$

To trigger the conversion of ADC, an additional timer is needed. We use **TimerA1** to set **ADC14_CTL0_SC** in ISR of **TimerA1** and the time between each conversion is set to 100 ms. The following code is used to configure **TimerA1** accordingly.

```
TIMER_A1->CTL = TIMER_A_CTL_TASSEL_1 | TIMER_A_CTL_MC_1;
TIMER_A1->CCR[0] = ms2ticks(ADC_CONVERSION_PERIOD);
TIMER_A1->CCTL[0] |= TIMER_A_CCTLN_CCIE;

void TA1_0_IRQHandler(void)
{
    TIMER_A1->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
    ADC14->CTL0 |= ADC14_CTL0_SC;
}
```

3 Conclusion

With the setup, the entire project works as expected where servomotor is correctly controlled by the joystick. ADC converts the signal from joystick every 100 ms, and microcontroller takes the conversion result to calculate duty cycle and sets CCR1 of **TimerA0**, therefore, changing duty cycle of PWM signal and causing servomotor to rotate accordingly. While the lab documentation shows that duty cycle of the signal to servomotor should range from 5% to 10%, we set it to be in the range of 2.5% to 12.5% for more accurate rotation. The screenshots of logic analyzer of PWM signal for maximum forward x-axis orientation, zero x-axis orientation and maximum backward x-axis orientation Fig. 2 are shown below.

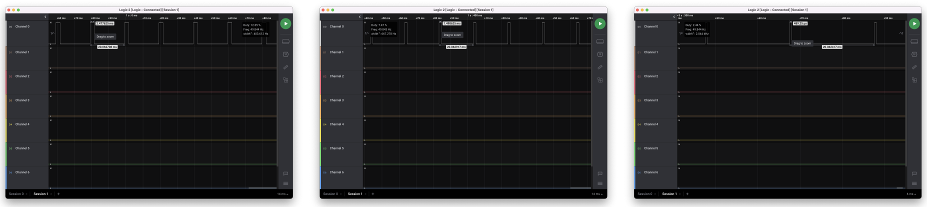

Figure 2: Signal from **P2.4** captured with logic analyzer when joystick pushed forward, at rest and pushed backward