# Countdown Alarm

Brian Worts (CE) and Chris Jenson (CE)

ELC 343

12/6/19

# Requirements

The Idea: This project is meant to serve as a proof of concept for an “alarm clock” that “closes the blinds” when a countdown is set, counts down from the set time, and upon reaching 0, run a servo to “open the blinds” of the room.

# Devices

- PSOC CY8C5868AXI-LP035 Board
- 1 input buttons on PSOC board
- 2 seven segment displays
- 16 330 ohm resistors
- 1 Parallax Standard Servo
- One character LCD

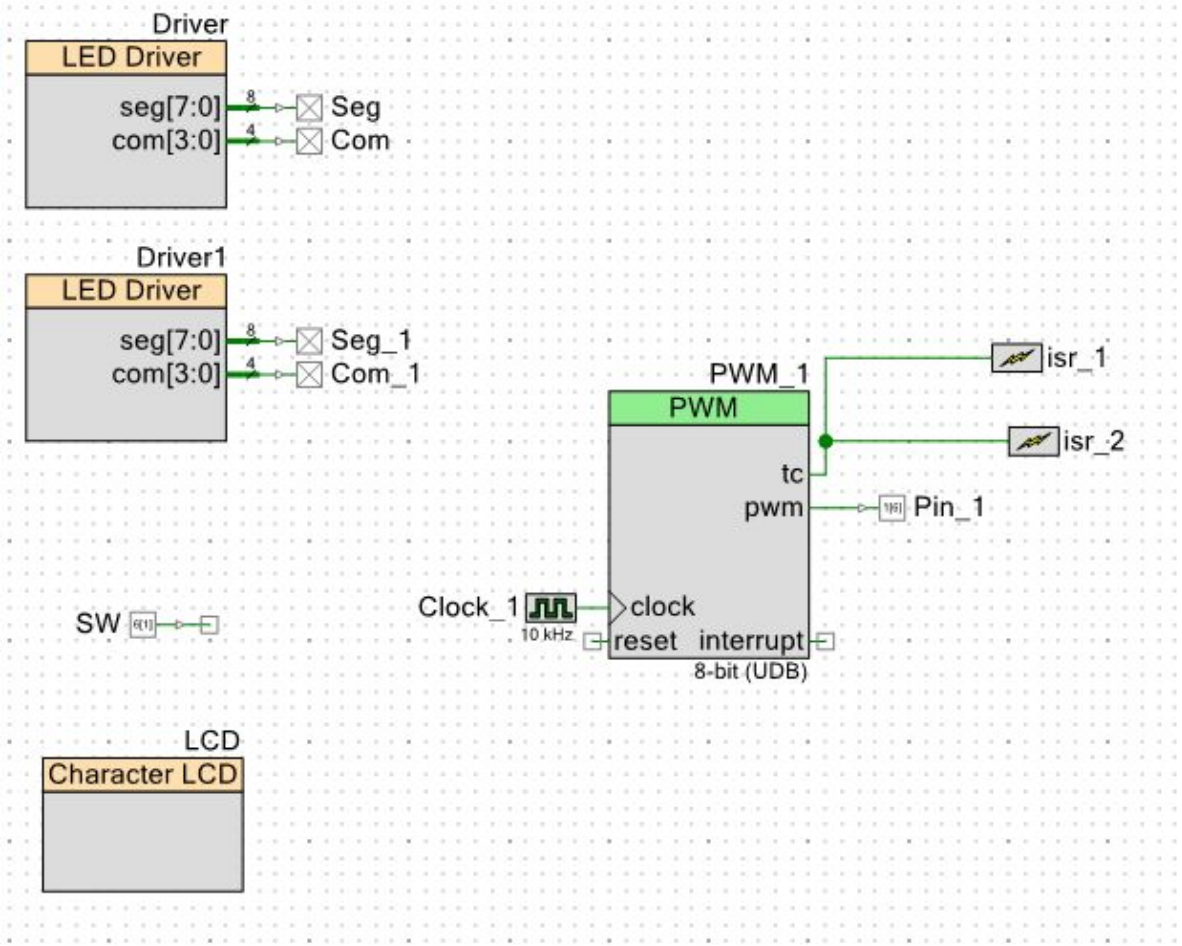
# Points Breakdown

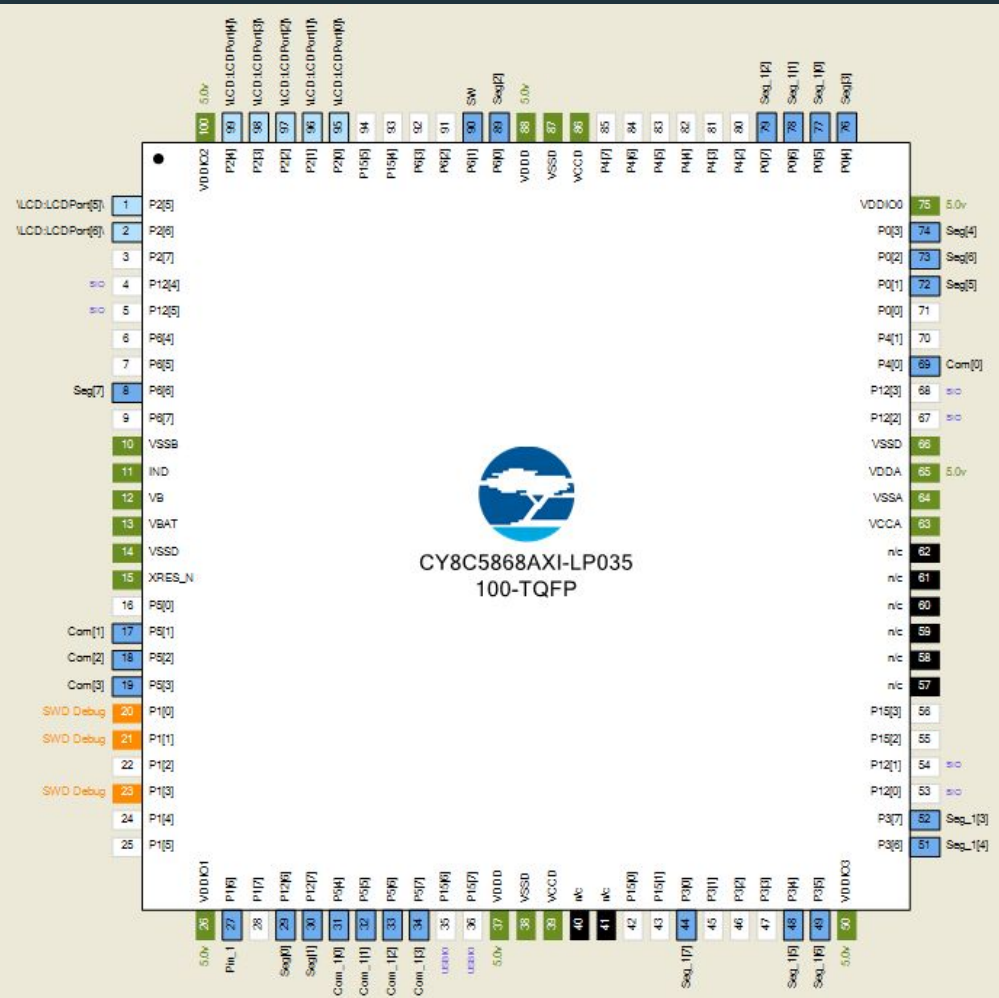
- Servo Motor (30 Points)
- 7 Segment Displays (20 Points)
- Pulse Width Modulator (30 Points)
- Interruptor Service Routines (20 Points)

**Total 100 Points**

# Tasks

- Use the buttons to input time in minutes and seconds
- Wait for end of input
- Turn servo 180 degrees
- Begin counting down
- Display current time on display
- Return servo to beginning state
- Wait for input

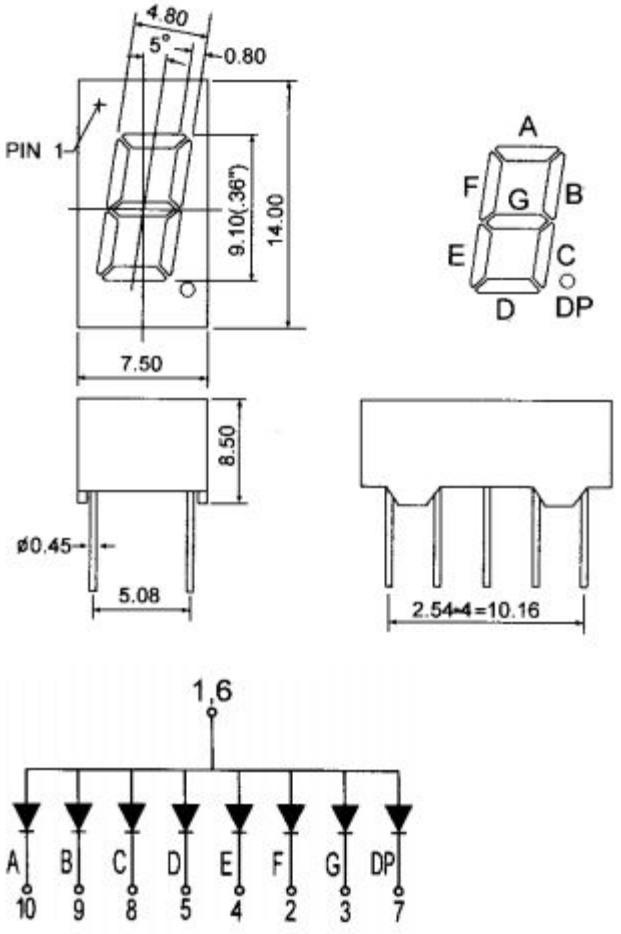

Driver
LED Driver
seg[7:0]
com[3:0]
8
4
Seg
Com
Driver1
LED Driver
seg[7:0]
com[3:0]
8
4
Seg_1
Com_1
PWM_1
PWM
tc
pwm
clock
reset
interrupt
8-bit (UDB)
isr_1
isr_2
Pin_1
Clock_1
10 kHz
SW
LCD
Character LCD

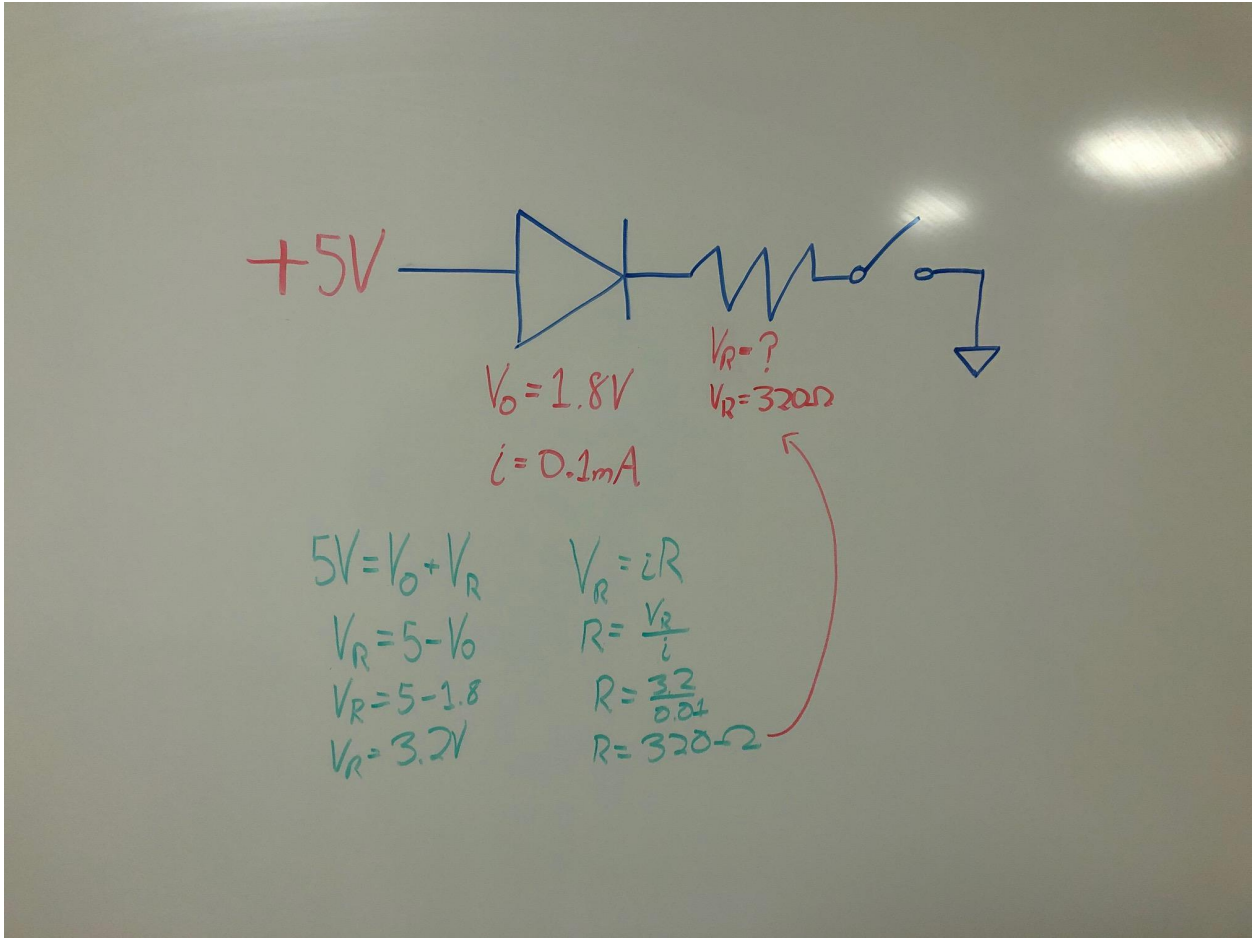



| Name | Port |
|---|---|
| \LCD:LCDPort[6:0]\ | P2[6:0] |
| Com[0] | P4[0] |
| Com[1] | P5[1] |
| Com[2] | P5[2] |
| Com[3] | P5[3] |
| Com_1[0] | P5[4] |
| Com_1[1] | P5[5] |
| Com_1[2] | P5[6] |
| Com_1[3] | P5[7] |
| Pin_1 | P1[6] |
| Seg[0] | P12[6] |
| Seg[1] | P12[7] |
| Seg[2] | P6[0] |
| Seg[3] | P0[4] |
| Seg[4] | P0[3] |
| Seg[5] | P0[1] |
| Seg[6] | P0[2] |
| Seg[7] | P6[6] |
| Seg_1[0] | P0[5] |
| Seg_1[1] | P0[6] |
| Seg_1[2] | P0[7] |
| Seg_1[3] | P3[7] |
| Seg_1[4] | P3[6] |
| Seg_1[5] | P3[4] |
| Seg_1[6] | P3[5] |
| Seg_1[7] | P3[0] |
| SW | P6[1] |

# 3161BS 7 Segment Display

- Only 2 were used due to space limitations on the PSOC board. Thus it can only display 0-60 seconds.




+5V
$V_D = 1.8V$
$i = 0.1mA$
$V_R = ?$
$V_R = 320\Omega$
$5V = V_D + V_R$
$V_R = 5 - V_D$
$V_R = 5 - 1.8$
$V_R = 3.2V$
$V_R = iR$
$R = \frac{V_R}{i}$
$R = \frac{3.2}{0.01}$
$R = 320\Omega$

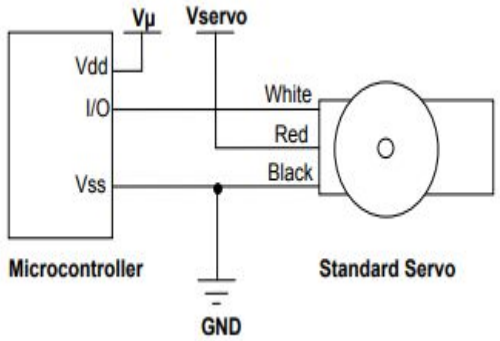
# Parallax Standard Servo (#900-00005)

Power requirements: 4 to 6 VDC*;

Maximum current draw is 140 +/- 50 mA at 6 VDC when operating in no load conditions, 15 mA when in static state

The Parallax Standard Servo is controlled through pulse width modulation, where the position of the servo shaft is dependent on the duration of the pulse. In order to hold its position, the servo needs to receive a pulse every 20 ms

Specifications: Pulse-width modulation, 0.75–2.25 ms high pulse, 20 ms intervals

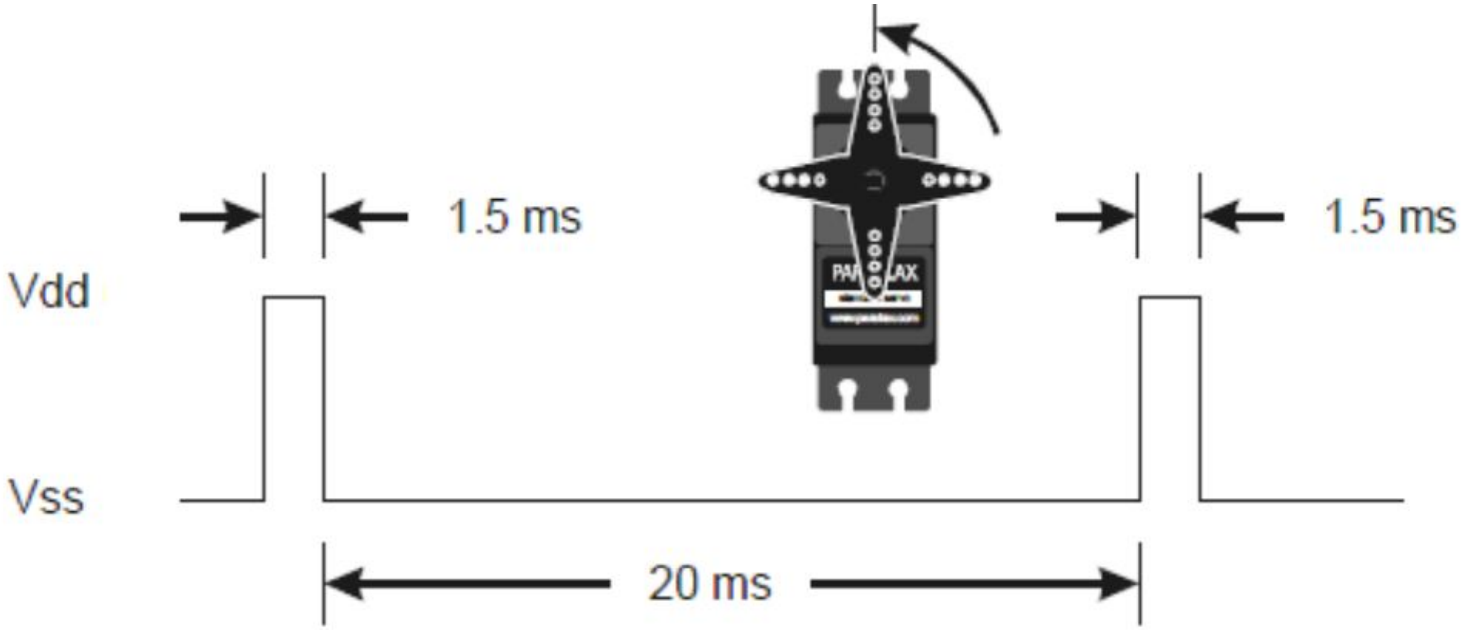




**Vμ** = microcontroller voltage supply

**Vservo** = 4 to 6 VDC, regulated or battery

**I/O** = PWM TTL or CMOS output signal from microcontroller: 3.3 to 5 V, not to exceed Vservo + 0.2 V

# Timing Diagram



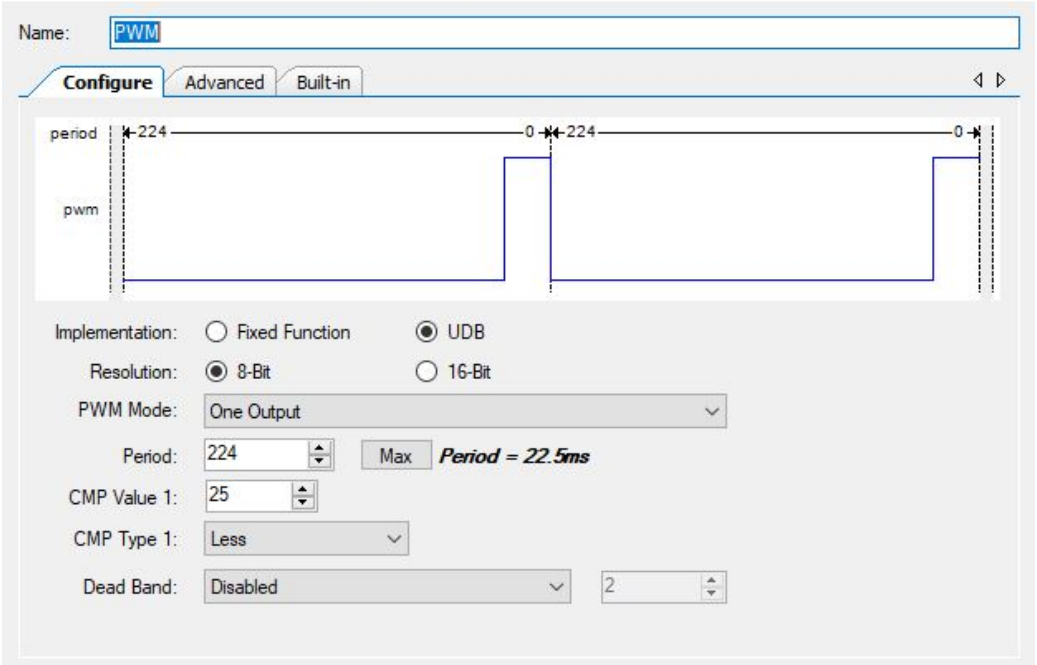
# PWM Calculations

| BASIC Stamp Module | 0.75 ms | 1.5 ms (center) | 2.25 ms |
|---|---|---|---|

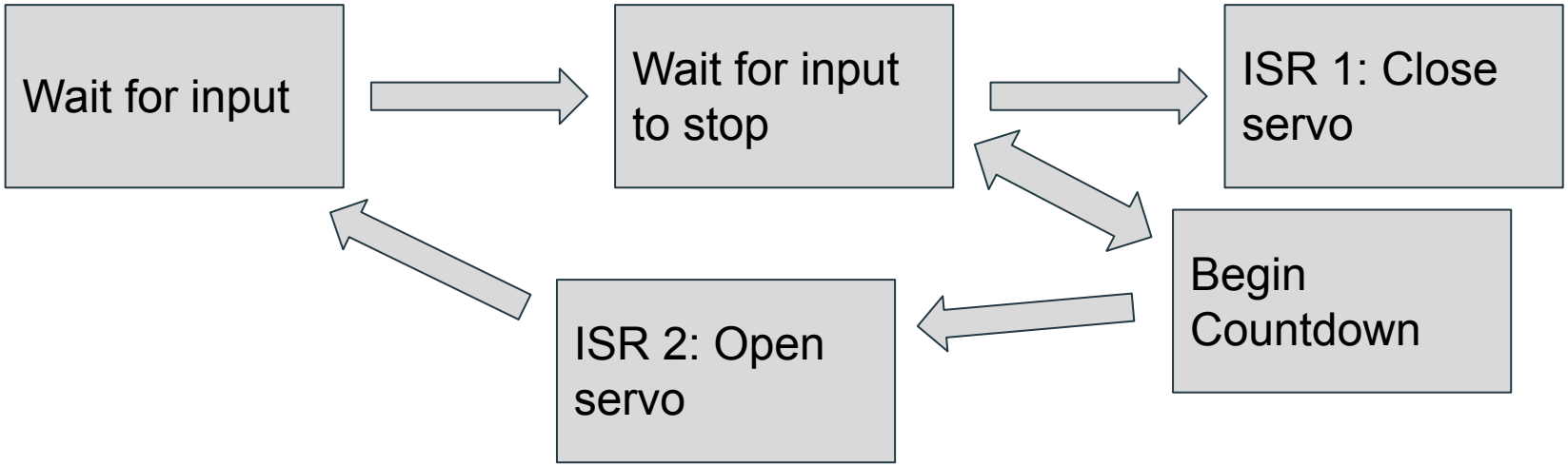
Desired Period = 22.5 ms

Pulse Width = 2.5 ms

Duty cycle = 11%



# Software Architecture



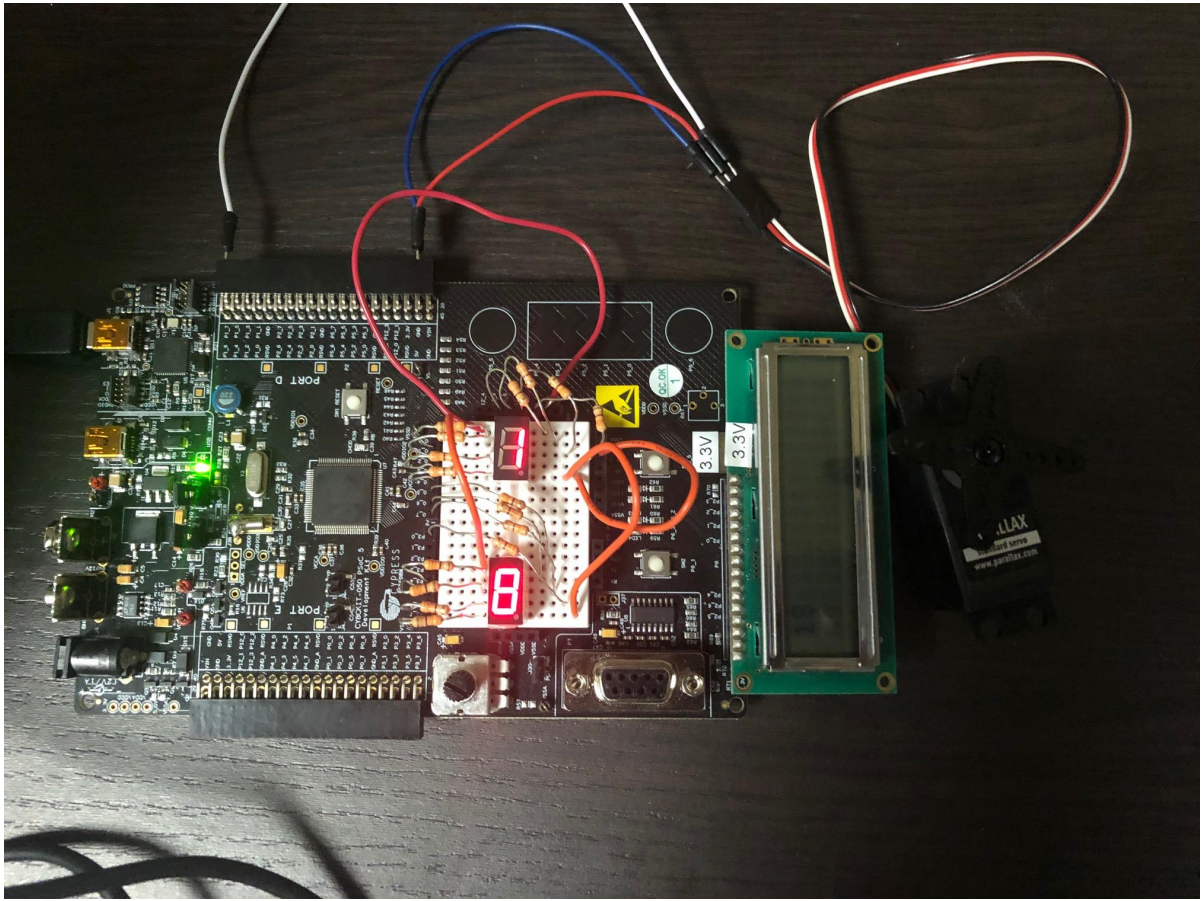

| Instance Name | Interrupt Number | Priority (0 - 7) |
|---|---|---|
| isr_1 | 0 | 7 |
| isr_2 | 1 | 7 |

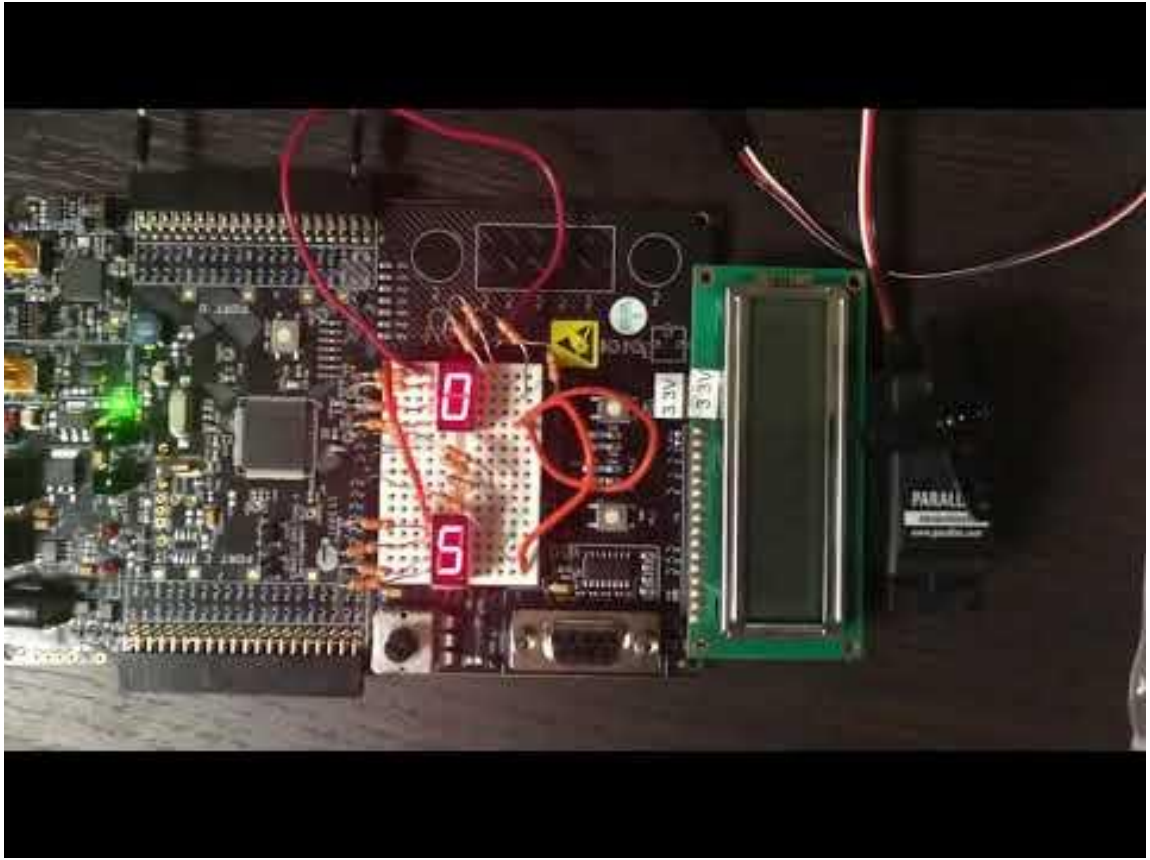
```
CY_ISR(isr_1_Interrupt)
{
    #ifdef isr_1_INTERRUPT_INTERRUPT_CALLBACK
        isr_1_Interrupt_InterruptCallback();
    #endif /* isr_1_INTERRUPT_INTERRUPT_CALLBACK */

    /*  Place your Interrupt code here. */
    /* `#START isr_1_Interrupt` */
    int x = 0;
    char strl[15];

    while (x<25)
    {
        PWM_1_WriteCompare(x);
        x = x + 1;

        LCD_ClearDisplay();
        LCD_Position(0,0);
        sprintf(strl, "%d",x );
        LCD_PrintString(strl);
        CyDelay(10);
    }
    isr_1_Stop();
    /* `#END` */
}
```

```
CY_ISR(isr_2_Interrupt)
{
    #ifdef isr_2_INTERRUPT_INTERRUPT_CALLBACK
        isr_2_Interrupt_InterruptCallback();
    #endif /* isr_2_INTERRUPT_INTERRUPT_CALLBACK */

    /*  Place your Interrupt code here. */
    /* `#START isr_2_Interrupt` */
    int x = 25;
    char strl[15];

    while (x>0)
    {
        PWM_1_WriteCompare(x);
        x = x - 1;
        LCD_ClearDisplay();
        LCD_Position(0,0);
        sprintf(strl, "%d",x );
        LCD_PrintString(strl);
        CyDelay(10);
    }

    isr_2_Stop();
    /* `#END` */
}
```

```
//setting the inital display to show all zeros
Driver_Write7SegDigitDec(0, 0);
Driver1_Write7SegDigitDec(0, 0);

for(;;)
{
    //increment seconds by 1 when button two is pressed
    if (SW_Read() == 0)
    {
        LCD_ClearDisplay();
        LCD_Position(0,0);

        if (seconds < 59)
        {
            seconds = seconds + 1;
            sprintf(str1, "%d", seconds);
            LCD_PrintString(str1);

            if (seconds < 10)
            {
                Driver_Write7SegDigitDec(0, 0);
                Driver1_Write7SegDigitDec(seconds, 0);
            }
            else //when double digits
            {
                Driver_Write7SegDigitDec((seconds / 10), 0); //divid
                Driver1_Write7SegDigitDec((seconds % 10), 0); //modu
            }
        }
    }
    CyDelay(500);
```

```
//this is the count down section of the code
if ((SW_Read() != 0) && (seconds > 0))
{
    isr_1_Start();

    //this decrements the minutes every 60 seconds
    while (seconds > 0)
    {
        //begin countdown
        CyDelay(1000);

        seconds = seconds - 1;
        LCD_ClearDisplay();
        sprintf(str1, "%d", seconds);
        LCD_PrintString(str1);

        if (seconds < 10)
        {
            Driver_Write7SegDigitDec(0, 0);
            Driver1_Write7SegDigitDec(seconds, 0);
        }
        else //when double digits
        {
            Driver_Write7SegDigitDec((seconds / 10), 0); //shi
            Driver1_Write7SegDigitDec((seconds % 10), 0); //mo
        }
    }
    isr_2_Start();
}
```


PORT D
PORT E
3.3V
3.3V


0
5
3.3V
3.3V

# Challenges

- Having the count-down timer count down based on an interrupt
- Making sure the servo is in the starting position
- Learning the new components

# Conclusion

- We were successful in accomplishing our goals
- We learned a significant amount about PWM, Servos, ISRs, 7-Segment Displays, the C Programming Language, and PSOC Microcontrollers

# Questions?