Ethan Nagelvoort
821234668
Lab 7
Video:
https://drive.google.com/file/d/14k_onOrVDlNZJRM0YETite9_lPhwcnfr/view?usp=sharing

Description:

I first set all the necessary inputs, outputs, and pullups. Then I use cli() to clear the global flags. Then I have two functions, timer0() and timer2(). Timer0() sets TCCR0A to CTC mode and OCR0A to the necessary value to fit 7ms, since my x = 6. I also set TCCR0B to the correct prescale. I also use TIMSK0 |= (1<<OCIE0A); to use the ISR for this timer. This ISR uses another function called keyboard(). I use timer 0 because it is 8 bit. Then timer2() uses timer2 since it is 8 bit and TCCR2A to CTC mode. Then I set OCR2A to the necessary value needed for 900Hz since my z=8. I then use TIMSK2 |= (1<<OCIE2A) |(1<<OCIE2B); so this timer can connect to 2 ISRs. These two ISRs regulate duty cycle with one turning on LED and one turning off LED. The keyboard() function is the same keyboard function used in all the other labs that use the keypad. In this function, when a button is pressed, another function is called taking that button as an input. This function, called scan, uses a switch statement that sets the OCR2B value at a given percentage of the OCR2A value. So for example in case 3, button 3 would have been pressed and so OCR2B is set to 30% of OCR2A. In each case I also set the prescale. Since this function is also called through another function that is called in the ISR connected to timer0, scan() is called every 7ms.

Result:

I connected my keypad to the correct ports on the AVR, Then I ran the code and when I pressed the numbers on the keypad, the LED brightness either increased or decreased corresponding to the percentage determined in the code.

Code:

```
/*
 * Lab 7.c
 *
 * Created: 4/09/2020 5:02:45 PM
 * Author : Ethan
 */

#include <avr/io.h>
#define F_CPU 16000000
#define LEDON PORTB |= (1<<PORTB5);
#define LEDOFF PORTB &=~ (1<<PORTB5);
#include <avr/interrupt.h>
//redID is 821234668
//x=6
```

```
//y=6
//z=8
void scan(int x);
int main(void)
{
        DDRB |= (1<<5); //set LED as output
        DDRD |= (1<<4)|(1<<5)|(1<<6)|(1<<7);//set D4-D7 as outputs
        DDRB &=(~(1<<0)& ~(1<<1) &~(1<<2) &~(1<<3));//set B0-B3 as inputs
        PORTB|=(1<<0)|(1<<1)|(1<<2)|(1<<3);//set B0-B3 as pullups
   cli();//clear global interrupt flag
        timer0();//timer0 initialization
        timer2();//timer2 initialization
        sei(); //enable global interrupts
        PORTB &=~(1<<5);//turn LED off
   while (1)
   {
   }
}

void keyboard()
{
        int r, c; // row and column
        int keys[4][4] = { {1, 2, 3, 10}, {4, 5, 6, 10}, {7, 8, 9, 10}, {10, 0, 10, 10}}; // 10 for invalid
characters
        for(r=4; r<8; r++)
        {
                PORTD |= (1<<4) | (1<<5) | (1<<6) | (1<<7); // set all rows to 1
                PORTD &= ~(1<<r); // set row r to 0
                for(c = 0; c<4; c++)//loop through column
                {
                        if(!(PINB & (1<<c))) // find the column that is low to determine which key
was pressed
                        {
                                scan(keys[r-4][c]);
                        }
                }
        }
}

void timer0()
{
        TCCR0A &= ~(1<<WGM00) &~(1<<WGM02);
        TCCR0A |= (1<<WGM01); // set timer to CTC mode
```

```
    OCR0A= 0x1B5; // 7ms , x=6, OCRA value is 437
    TCCR0B |= (1<<CS02) | (1<<CS00);
    TCCR0B &= ~(1<<CS01);//set prescale to 1024
    TIMSK0 |= (1<<OCIE0A);//enable compare match for A
}

void timer2()
{
    TCCR2A &= ~(1<<WGM20)&~(1<<WGM22);
    TCCR2A |= (1<<WGM21); // set timer to CTC mode
    OCR2A= 0x44;//z=8, 9*100 = 900Hz
    TIMSK2 |= (1<<OCIE2A) |(1<<OCIE2B);//enable compare match A & B
}

void scan(int x)
{
    switch(x)
    {
        case 0:// 0% duty cycle
        OCR2B = 0x00;
        break;

        case 1:// 10% duty cycle
        OCR2B = 0x6;
        TCCR2B|= (1<<CS22);
        TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
        break;

        case 2:// 20% duty cycle
        OCR2B = 0xE;
        TCCR2B|= (1<<CS22);
        TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
        break;

        case 3:// 30% duty cycle
        OCR2B = 0x14;
        TCCR2B|= (1<<CS22);
        TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
        break;

        case 4:// 40% duty cycle
        OCR2B = 0x1B;
        TCCR2B|= (1<<CS22);
```

```
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;

                case 5:// 50% duty cycle
                OCR2B = 0x22;
                TCCR2B|= (1<<CS22);
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;

                case 6:// 60% duty cycle
                OCR2B = 0x29;
                TCCR2B|= (1<<CS22);
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;

                case 7:// 70% duty cycle
                OCR2B = 0x30;
                TCCR2B|= (1<<CS22);
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;

                case 8:// 80% duty cycle
                OCR2B = 0x36;
                TCCR2B|= (1<<CS22);
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;


                case 9:// 90% duty cycle
                OCR2B = 0x3D;
                TCCR2B|= (1<<CS22);
                TCCR2B &= ~(1<<CS21) &~(1<<CS20);//set prescale to 256
                break;
        }
}

ISR(TIMER0_COMPA_vect)
{
        keyboard();
}
ISR(TIMER2_COMPA_vect)
{
        LEDON;
```

```
}
ISR(TIMER2_COMPB_vect)
{
	LEDOFF;
}
```