Ethan Nagelvoort
821234668

Lab5 report

Video of LED working:

https://drive.google.com/file/d/14oqpNYe-Z1TRwO90KBjT3GGPZcGo_AJ_/view?usp=sharing

Description:

In this lab, I had to use a timer to manipulate the brightness of the LED using duty cycle. Thus, when I press the button on the AVR, the LED would slowly increase in brightness and when I let go, the LED would slowly decrease in brightness. To do this, I first set B5(LED) to output and B7(button) to input and pullup. Then I set register TCCR0A to CTC mode and clear OCR0A and OCR0B, which will be used to represent 10ms and duty cycle. I set OCR0A to 0x9B or 155 which is the number we want the timer to count to. To get this number, I used the calculations presented in the lecture that involved the prescales 1, 8, 64, 256, 1024, 16 MHz, and a desired timer of 10000 us. Then I set OCR0B to 0 which initializes the duty cycle to 0. Then I set register TCCR0B to have a prescaler of 1024 and to start the timer. In the infinite while loop, I set another while loop while the button is pressed. Then there is another while loop if OCR0B is less than 0x9B. In this loop, OCR0B is incremented, thus duty cycle is incremented. Then the LED turns on and then a busy while loop is used to detect overflow in OCF0B . Once detected, a flag is placed to stop the busy while loop. We also do the same thing with OCF0A except we turn the LED off before detection. If OCR0B is over 0x9B, then the LED remains on since it is already at full brightness. Then when the button is let go, the program will go through another while loop that operates similar to the other while loop except it loops as long as OCR0B is greater than 0 and it decrements OCR0B, thus decreasing the duty cycle. This loop decreases the brightness. Outside of this loop, we turn the LED off since the brightness is already 0.

Results:

After building and running my code, I pressed the button on my AVR. The LED slowly increased in brightness. Then when I let go of the button, The LED slowly decreased in brightness.

Source code:

```
/*
 * LAB5.c
 *
 * Created: 3/8/2020 9:09:20 PM
 * Author : Ethan
 */
```

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#define LEDON PORTB|=(1<<5)
#define LEDOFF PORTB&=~(1<<5)

#define BAUD 9600


#define MYUBBR ((F_CPU)/(BAUD * 16UL) - 1) //Use UBBR for USART baud rate

void init_uart() {
        UBRR0H = (MYUBBR >> 8); //set UBBR high
        UBRR0L = (MYUBBR); //set UBBR low
         UCSR0B |= (1 << TXEN0) | (1 << RXEN0) | (1 << RXCIE0) | (1 << TXCIE0);
}

void delay_time(int a)//function that lets us use any amount of delay
{
        int count =0;
        while(count < a)
        {
                _delay_us(1);
                count++;
        }
}


int main(void)
{
        init_uart();
   DDRB |= (1<<5);//set LED pin as output
        DDRB &=~ (1<<7);//set button pin to input
        PORTB |= (1<<7);//set b7 as pullup
        TCCR0A |=  (1<<WGM01) | (1<<COM0A1) | (1<<COM0B1); // set timer to CTC mode
through (1<<WGM01), use (1<<COM0A1) to clear OCOA, use(1<<COM0B1) to clear OCOB
        LEDOFF;
```

```
	OCR0A = 0x9B; // represents 10ms timer/ number we want timer to count to which is
155
	OCR0B = 0x00; // represents 0% duty cycle
	TCCR0B |= (1<<CS02) | (1<<CS00); // set timer to 1024 prescaler and start timer
  while (1)
  {
		while((PINB&(1<<7))==0)//if button is pressed
		{
			while(OCR0B<0x9B)//if statment instead of while, so if button isn't
pressed loop will end
			{
				LEDON;
				while((TIFR0 & (1<<OCF0B))==0){};//wait for overflow event of
OCF0B
					TIFR0 |= (1 << OCF0B); // reset overflow flag of OCF0B
				LEDOFF;
				while((TIFR0 & (1<<OCF0A))==0){};//wait for overflow event of
OCF0A
					TIFR0 |= (1 << OCF0A); // reset overflow flag of OCF0A
				OCR0B++; // increment OCR0B
			}
			LEDON;//once at full brightness, leave LED on
				}
		while(OCR0B>0)//while loop that decreases brightness
		{
			OCR0B--;//decrement OCR0B
			LEDON;
			while((TIFR0 & (1<<OCF0B))==0){};//wait for overflow event of
OCF0B
				TIFR0 |= (1 << OCF0B); // reset overflow flag of OCF0B
			LEDOFF;
			while((TIFR0 & (1<<OCF0A))==0){};//wait for overflow event of
OCF0A
				TIFR0 |= (1 << OCF0A); // reset overflow flag of OCF0A
			}
			LEDOFF;//once at lowest brightness, leave LED off
```

```
        //checkpoint code below
        /*int count = 0;
    while((PINB&(1<<7))==0)//if button is pressed
        {
            if(count<1000)//if count is less than a second
            {       //our period is one second which is 1000 ms
                count++;//increment to increase brightness
                LEDON;
                delay_time(count);
                LEDOFF;
                delay_time(1000-count);//brightness is controlled by duty cycle
            }
            else//if greater than period then LED is left on at full brightness
            {
                LEDON;
            }
        }

        while(count>0)
        {
            count--;//decrement count so brightness will go down
            LEDOFF;
            delay_time(1000-count);
            LEDON;
            delay_time(count);
        }
        LEDOFF;//after While statement, LED stays off*/


   }
return(0);
}
```