Lab 2: Morse Code Decoder

ESE3500: Embedded Systems & Microcontroller Laboratory
University of Pennsylvania

In this document, you'll fill out your responses to the questions listed in the Lab 2 Manual. Please fill out your name and link your Github repository below to begin. Be sure that your code on the repo is up-to-date before submission!

For all the questions that require a video, provide a link to the video (e.g. youtube, google drive, etc.).

Student Name: Krystal Li
Pennkey: lkrystal
GitHub Repository: lkrystal31

1.

```
int main(void)
{
    /* Replace with your application code */
    DDRB |= (1<<DDB1); // set as output pin
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);

    PORTB |= (1<<PORTB1);
    PORTB |= (1<<PORTB2);
    PORTB |= (1<<PORTB3);
    PORTB |= (1<<PORTB4);

}
```

2.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    while(1)
    {
        DDRB |= (1<<DDB1); // set output pin b1
        PORTB &= ~(1<<PORTB1); // set b1 low
        DDRD &= ~(1<<DDD7); //enable input pin d7
        PORTD &= ~(1<<PORTD7); // set d7 low initially

        //PORTD |= (1<<PORTD7); // pull up resistor
        if (PIND & (1<<PIND7))  // check if value at port high
        {

            //PORTB |= (1<<PORTB1);// drive output high to turn ON
            PORTB ^=(1<<PORTB1); //toggle led
            _delay_ms(100); // delay
            // wait until button released
            while(PIND & (1<<PIND7));
        }
        else {
            PORTB &= ~(1<<PORTB1);
        }
    }

}
```

3.

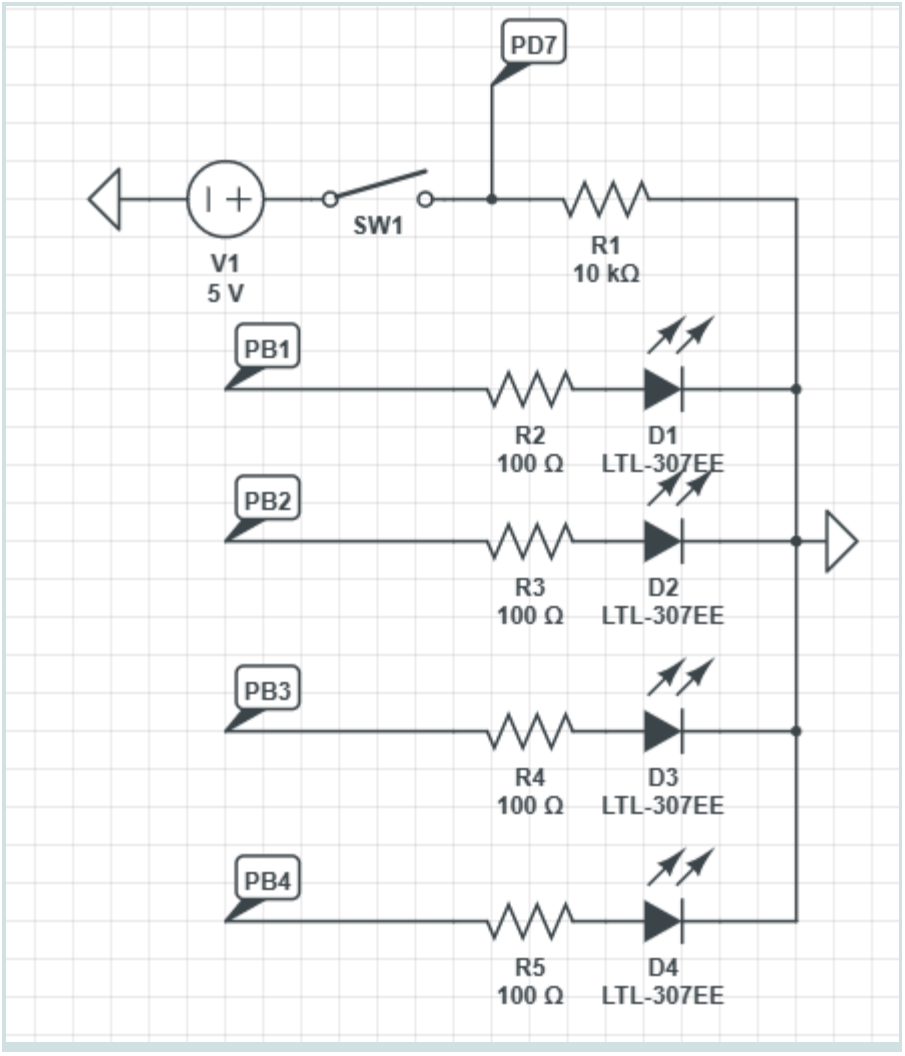
```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
            DDRB |= (1<<DDB1); // set output pin b
            DDRB |= (1<<DDB2); // b2
            DDRB |= (1<<DDB3); // b3
            DDRB |= (1<<DDB4); // b4
            DDRD &= ~(1<<DDD7); // input pin d7
            int list[] = {1, 2, 3, 4};
            int count = 0;
            int state;
    while(1)
    {
        //if (!(PIND & (1<<PIND7))) {}
        if (PIND & (1<<PIND7)) // if d7 high
        {
            count++;
            state = list[count%4];
            if (state == 1)
            {
                PORTB ^= (1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB &= ~(1<<PORTB3);
                PORTB &= ~(1<<PORTB4);
                _delay_ms(500);
            }
```

```
            if (state == 2)
            {
                PORTB ^=(1<<PORTB2);
                PORTB &= ~(1<<PORTB1);
                PORTB &= ~(1<<PORTB3);
                PORTB &= ~(1<<PORTB4);
                _delay_ms(500);
            }
            if (state == 3)
            {
                PORTB ^= (1<<PORTB3);
                PORTB &= ~(1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB &= ~(1<<PORTB4);
                _delay_ms(500);

            }
            if (state==4)
            {
                PORTB ^= (1<<PORTB4);
                PORTB &= ~(1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB &= ~(1<<PORTB3);
                _delay_ms(500);
            }

        }
    }
}
```

4. 

5. Interrupts vs polling:

Polling is easier to implement, as the program can be written in one main function. However, polling is inefficient since the program is constantly checking if certain conditions are met, which takes up CPU power that can be allocated somewhere else. Interrupts are usually preferred to polling since the ISR will only run and executed if certain conditions are met. Polling will constantly check on the status of a flag or input. In addition, interrupts generally have faster response times. A disadvantage of interrupts could be that they are more difficult to implement in code, as it requires setting a few other registers and specifying rising and falling edges.

6.

30ms = 0.03 s; 0.03*16,000,000*1/62500 = 480,000 ticks

200 ms: 3,200,000 ticks

400 ms: 6,400,000 ticks

7. Prescalars help change notion of speed on the microcontroller. Since timer 1 can count up to 65535, we want to time things so that they don't go over that amount.

The prescalar will decrease the frequency of which the timer increments, allowing longer frequencies to be measured. For example, the timer can be prescaled by powers of 2, so when it is prescaled to 256, the resulting frequency will change from 16 MHz to 62.5 kHz.

8. Microchip studio stopped working and giving error when flash to uno: avrdude OS error: file Debug\.hex is not readable: No such file or directory
9. SOMEDAY I WILL RULE YOU ALL