

# 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:** Leon Kabue

**Pennkey:** 89330724

**GitHub Repository:** [https://github.com/e3500/lab2\\_89330724](https://github.com/e3500/lab2_89330724)

1.

```
int main() {  
    DDRB |= (1 << DDB1);  
    DDRB |= (1 << DDB2);  
    DDRB |= (1 << DDB3);  
     DDRB |= (1 << DDB4);  
  
    while(1) {  
        PORTB |= (1 << PORTB1);  
        PORTB |= (1 << PORTB2);  
        PORTB |= (1 << PORTB3);  
        PORTB |= (1 << PORTB4);  
    }  
}
```

2.

```
int main() {  
    DDRB |= (1 << DDB1);  
    DDRD &= ~(1 << DDD7);  
  
    while(1) {  
        if (PIND & (1 << PIND7)) {  
            PORTB |= (1 << PORTB1);  
        } else {  
            PORTB &= ~(1 << PORTB1);  
        }  
    }  
}
```

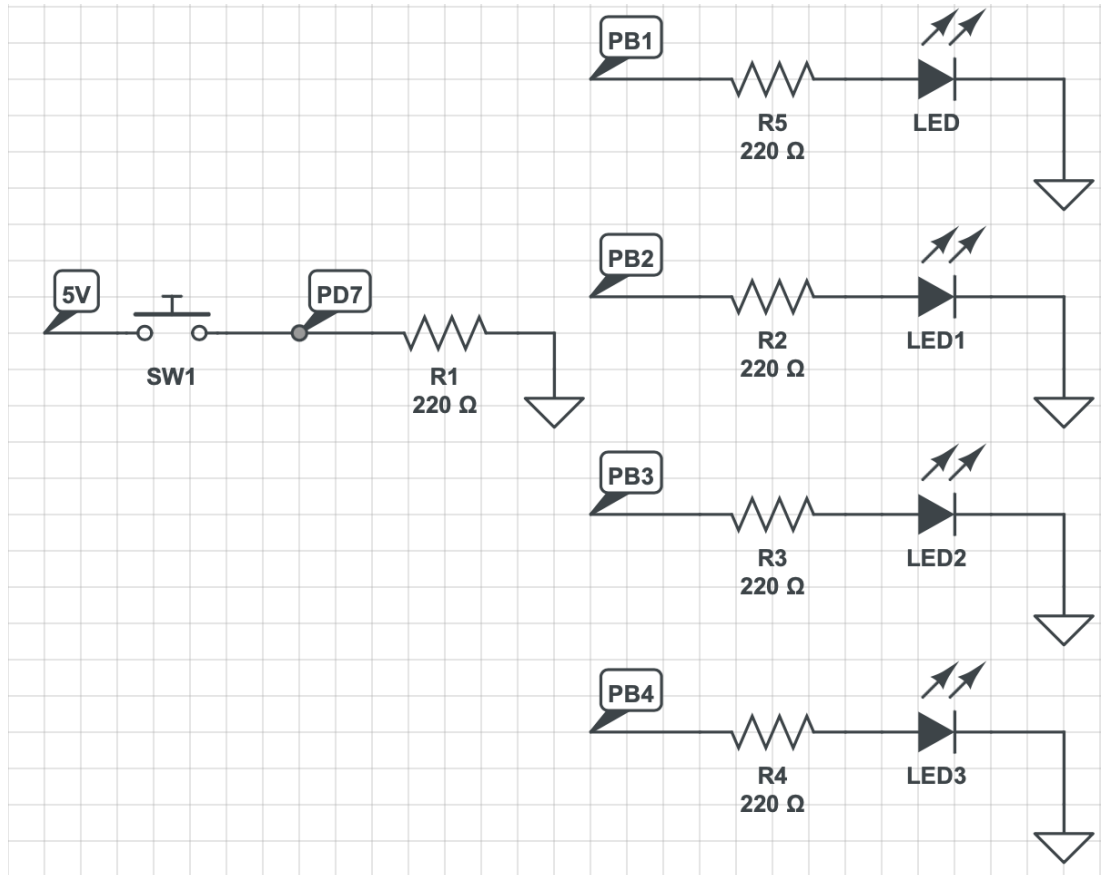
3.

```

int main() {
    DDRB |= (1 << DDB1); //setting PB1 as output
    DDRB |= (1 << DDB2); //setting PB2 as output
    DDRB |= (1 << DDB3); //setting PB3 as output
    DDRB |= (1 << DDB4); //setting PB4 as output
    DDRD &= ~(1 << DDD7); //setting PD7 as input
    int count = 4;
    int remainder = 0;

    while(1) {
        if (PIND & (1 << PIND7)) {
            count = count + 1;
            _delay_ms(1000);
        }
        remainder = count % 4;
        switch (remainder) {
            case 0:
                PORTB |= (1 << PORTB1);
                break;
            case 1:
                PORTB &= ~(1 << PORTB1);
                PORTB |= (1 << PORTB2);
                break;
            case 2:
                PORTB &= ~(1 << PORTB2);
                PORTB |= (1 << PORTB3);
                break;
            case 3:
                PORTB &= ~(1 << PORTB3);
                PORTB |= (1 << PORTB4);
                break;
        }
    }
}

```



4.

5. An advantage of using interrupts for this is that the MCU does not keep querying the button to see whether it is toggled thus saving a lot of cpu cycles.

A disadvantage of using polling for this is that it increases the code since we have to set up the interrupt and keep toggling between checking for a falling edge and checking for a rising edge each time the button is pressed.

6. At 16MHz, 30ms = 480,000 ticks  
 200ms = 3,200,000 ticks  
 400ms = 6,400,000 ticks

7. A prescaler allows us to work with different frequencies by slowing down the MCU clocks, we can either prescale(slow down) the system clock of the MCU by a factor of 2, 4 or 8 meaning that instead of operating at the standard 16MHz, the MCU will be operating at 8MHz, 4MHz, or 2MHz respectively when these prescalers are used; we could also prescale the specific timers on the MCU allowing us to measure signals at different frequencies, timer prescalers are usually powers of two, 8, 64, 256 and 1024, are prescalers that can be used on timer1b and they work similar to the system clock

prescalers but instead of prescaling from 16MHz, they prescale on top of the prescale already applied on the system clock, thus if the system clock has been prescaled to 8MHz using the prescale 2, then if the timer1b is prescaled by 8, the timer will run at 1MHz.

8. I used a prescaler of 256 on the timer to achieve a frequency of 62.5kHz, then set up my code such that a DOT was a press of 30 to 400ms, a DASH a press of 400 to 1200 ms and a space was represented by a release of over 1200ms.

In the video, I have 2 leds, a red led which lights up if button is pressed as a DOT and the yellow led which lights up if the button is pressed as DASH. I demo the decoding of the first 3 letters of my name i.e LEO

<https://drive.google.com/file/d/1nOH5bseHnFsBkTQRqFm7tUfoX0klyk2-/view?usp=sharing>

9. Someday I will rule you all

- 10.