# Lab 2: Morse Code Decoder

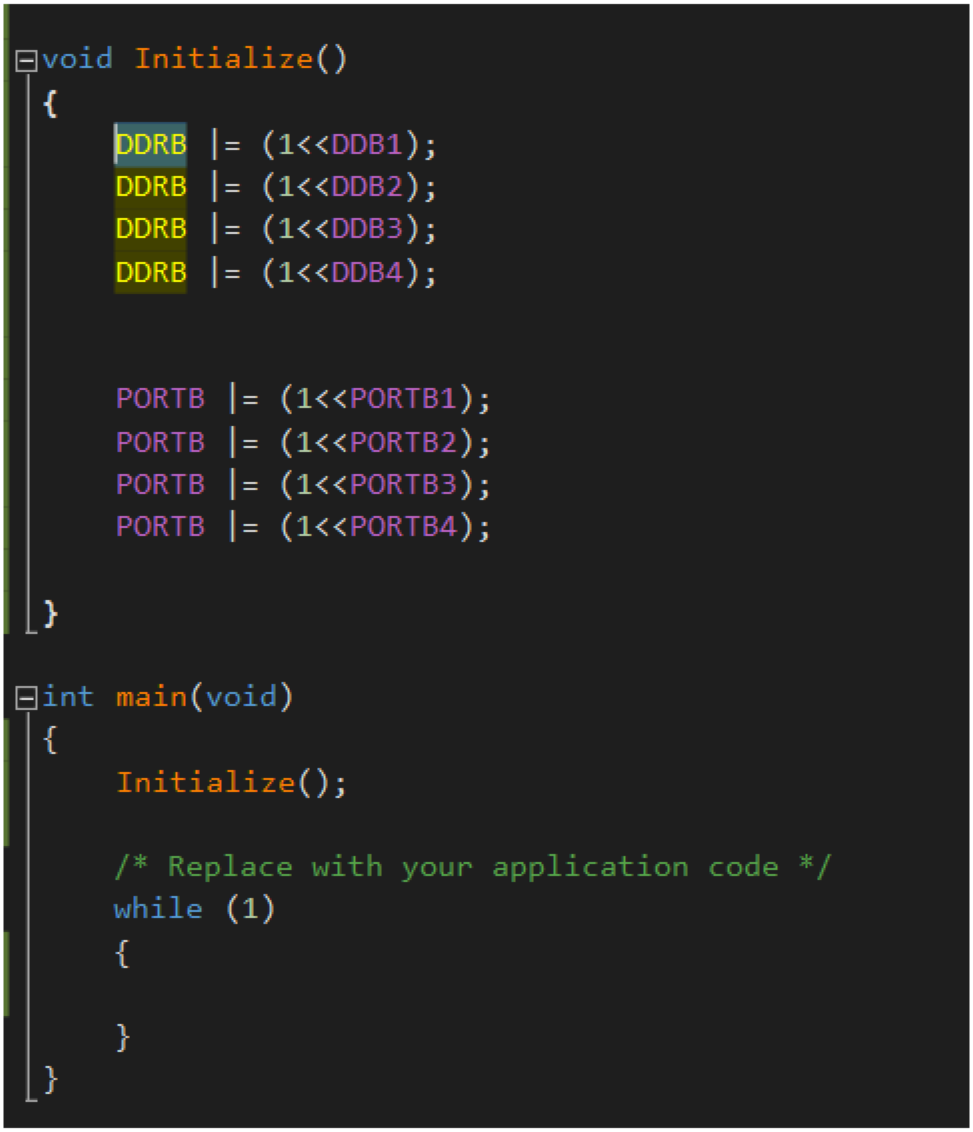
ESE350: Embedded Systems & Microcontroller Laboratory
University of Pennsylvania

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.).

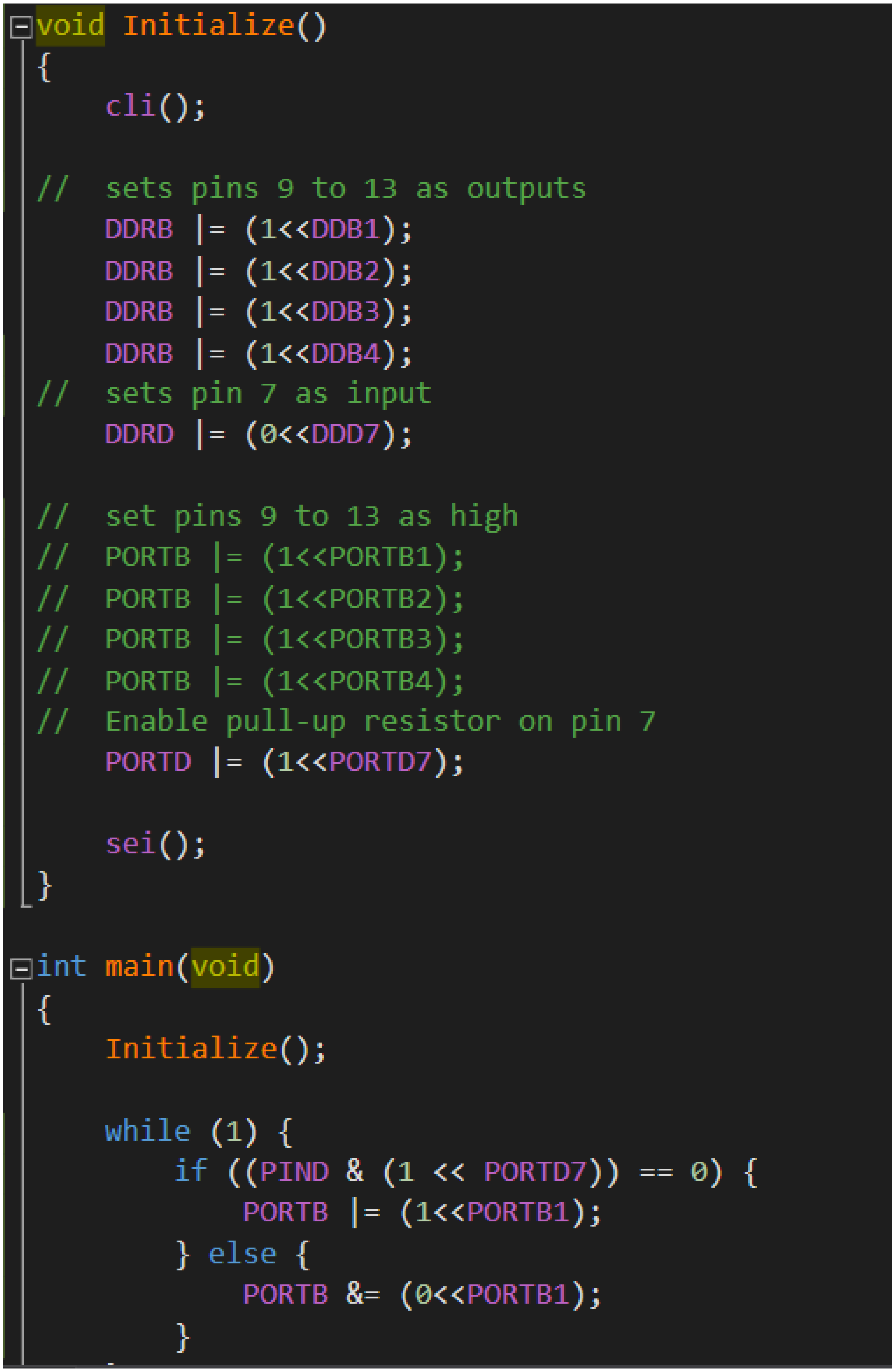
**Student Name:** John Colavito
**Pennkey:** colavijs
**GitHub Repository:** [https://github.com/ese3500/lab2-colavijs](https://github.com/ese3500/lab2-colavijs)

1. The below code lights up 4 LEDs on a breadboard



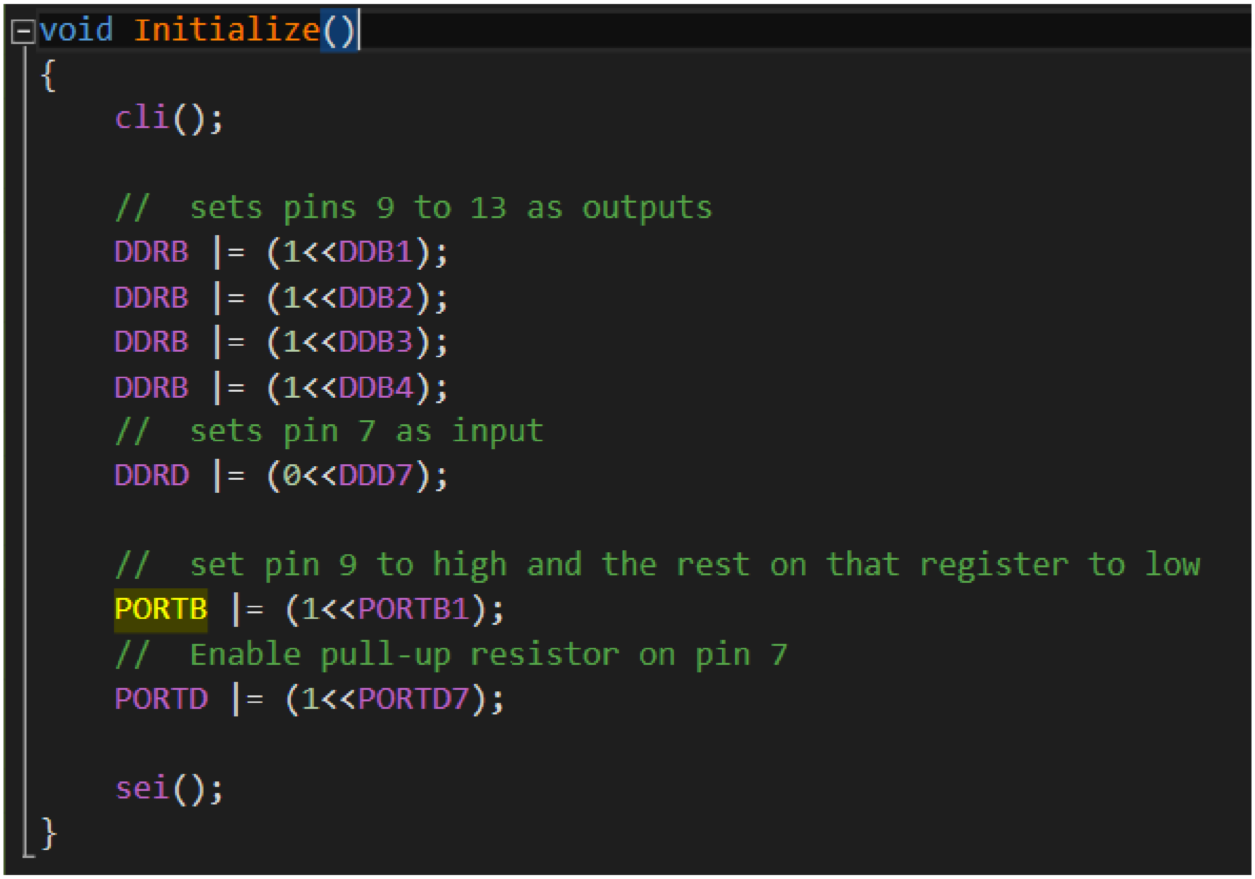
```
void Initialize()
{
    DDRB |= (1<<DDB1);
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);


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

}

int main(void)
{
    Initialize();

    /* Replace with your application code */
    while (1)
    {

    }
}
```

2. The below code turns a LED on when a button is pressed and turns the LED off when the button is released

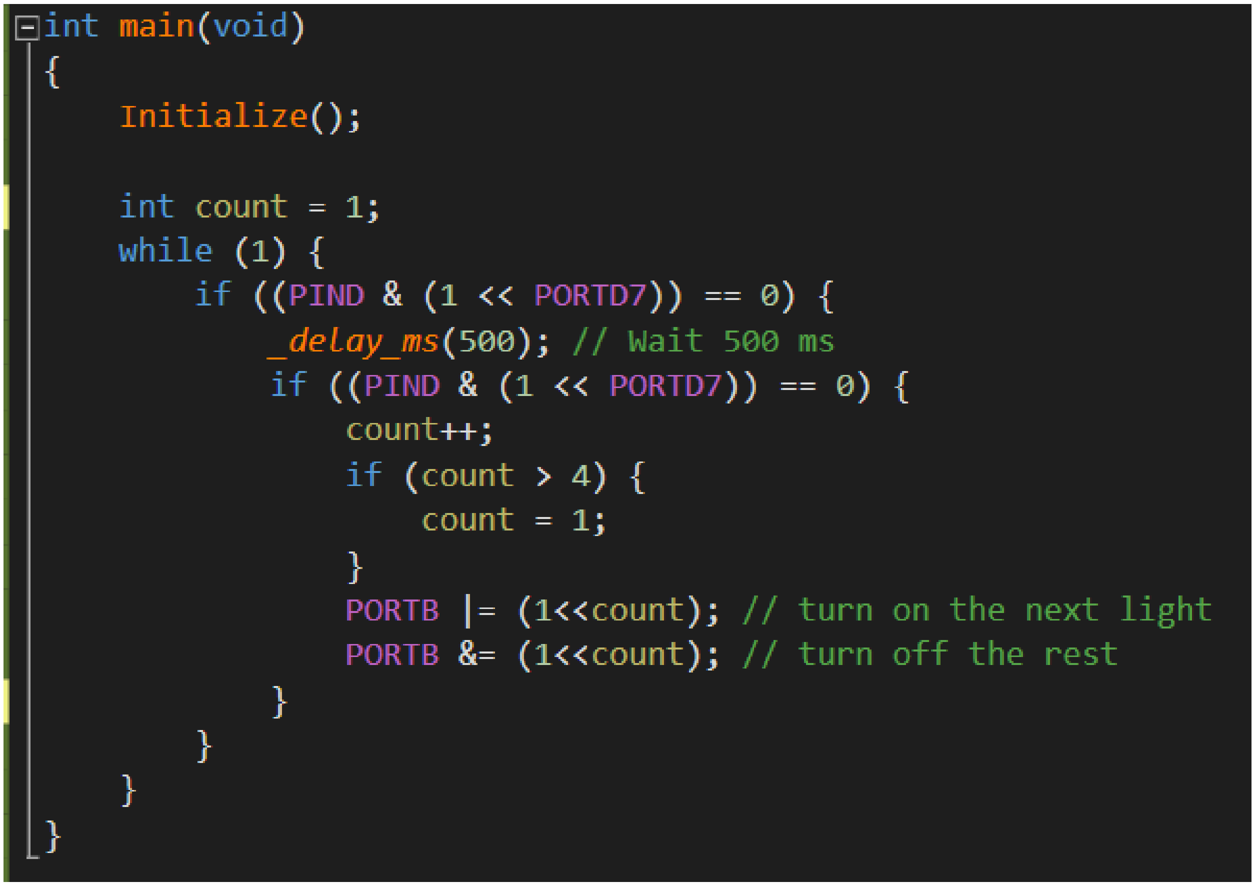


```
void Initialize()
{
    cli();

//  sets pins 9 to 13 as outputs
    DDRB |= (1<<DDB1);
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);
//  sets pin 7 as input
    DDRD |= (0<<DDD7);

//  set pins 9 to 13 as high
//  PORTB |= (1<<PORTB1);
//  PORTB |= (1<<PORTB2);
//  PORTB |= (1<<PORTB3);
//  PORTB |= (1<<PORTB4);
//  Enable pull-up resistor on pin 7
    PORTD |= (1<<PORTD7);

    sei();
}

int main(void)
{
    Initialize();

    while (1) {
        if ((PIND & (1 << PORTD7)) == 0) {
            PORTB |= (1<<PORTB1);
        } else {
            PORTB &= (0<<PORTB1);
        }
```

3. The below code turns the sequentially next LED on when a button is pressed

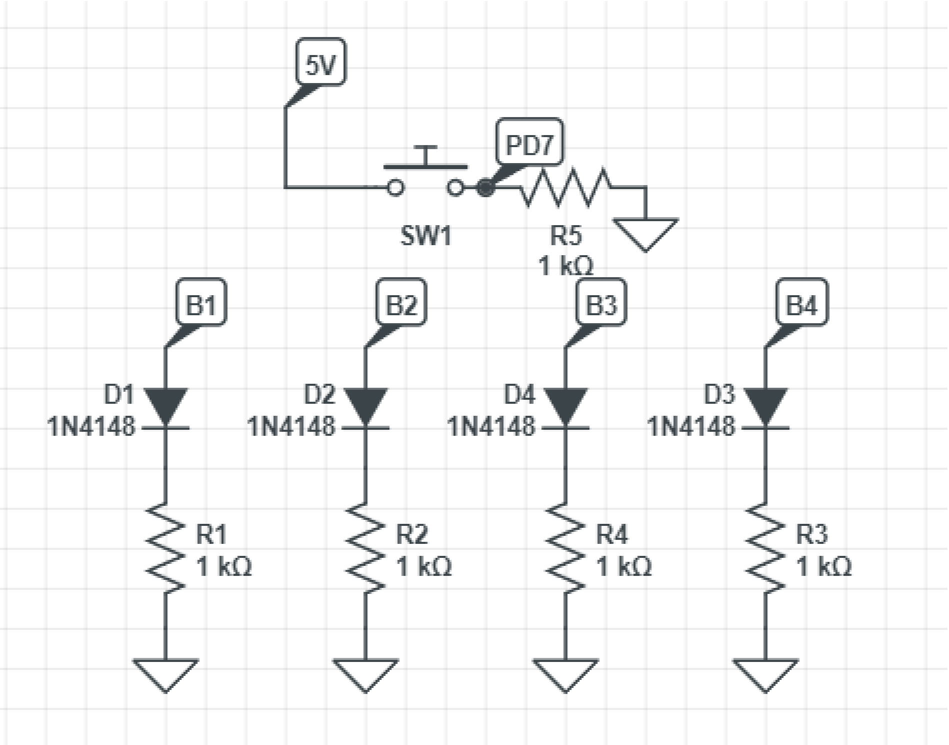


```
void Initialize()
{
    cli();

    //  sets pins 9 to 13 as outputs
    DDRB |= (1<<DDB1);
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);
    //  sets pin 7 as input
    DDRD |= (0<<DDD7);

    //  set pin 9 to high and the rest on that register to low
    PORTB |= (1<<PORTB1);
    //  Enable pull-up resistor on pin 7
    PORTD |= (1<<PORTD7);

    sei();
}
```



```
int main(void)
{
    Initialize();

    int count = 1;
    while (1) {
        if ((PIND & (1 << PORTD7)) == 0) {
            _delay_ms(500); // Wait 500 ms
            if ((PIND & (1 << PORTD7)) == 0) {
                count++;
                if (count > 4) {
                    count = 1;
                }
                PORTB |= (1<<count); // turn on the next light
                PORTB &= (1<<count); // turn off the rest
            }
        }
    }
}
```

4. The below image shows the circuit schematic built on the breadboard.



5. One advantage of using interrupts over polling for a system that turns a LED on with a button input in a case such as this, where the time until the button is pressed is unknown, is that the interrupt does not have to keep continuously checking for the event that the button is pressed, while polling does. This means that we do not have to constantly check for an interrupt, which will reduce overhead and improve the performance of the device. One disadvantage is that it stops the flow of the program, so if something critical was running when the interrupt calls then the critical task could potentially not be completed in the amount of time required.
6. To find these values, we know that we have a 16Mhz clock, so there are 16000000 ticks every second. We multiply this by .03 seconds to find the number of ticks in 30 ms. Since my clock is prescaled by 8, this must additionally be divided by 8 to find the number of ticks that my clock counts in 30 ms.
    a. 30 ms -> 480000 ticks
    b. 200 ms -> 3200000 ticks
    c. 300 ms -> 6400000 ticks
7. A prescaler reduces the frequency that a clock ticks, which allows one to reduce the frequency from very high values to various lower values. For example, if the input frequency is 16 MHz and the prescaler is set to divide by 8, the output frequency will be 2 MHz. The prescaler can be set to various powers of 2.
8. https://drive.google.com/file/d/1tTw108J-TfbnvJjjw-oUM84--uHJ4Op8/view?usp=sharing
9. Someday i will rule you all