```c
#include <avr/io.h>
#include <util/delay.h>


int main(void)
{
    /* Replace with your application code */
    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);

    }
}
```

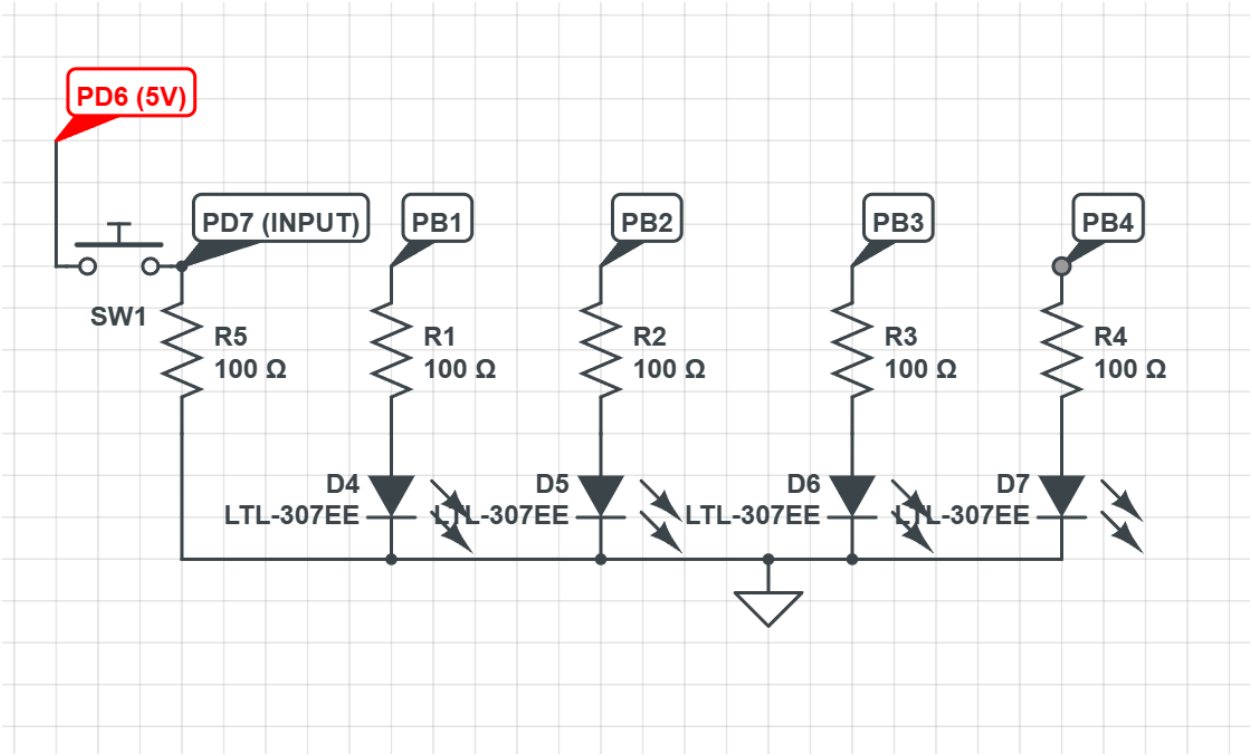
1.

2.

```c
int main(void)
{
    /* Replace with your application code */
    DDRB |= (1<<DDB1); //enable pin as output
    PORTB |= (1<<PORTB1); //PB1 (pin 9) will be the constant 5v source
    DDRD &= ~(1<<DDD7); //set up PD7 as the signaling pin
    DDRB |= (1<<DDB2); //set up PB2 (pin10) as output pin. This will supply 5v to the led



    while (1)
    {
        if (PIND & (1<<PIND7)) {
            PORTB |= (1<<PORTB2);
        } else {
            PORTB &= ~(1<<PORTB2);
        }

    }
}
```

3.

```c
int main(void)
{
    /* Replace with your application code */

    /*
    Setup notes:
    1. Pin 6 (PD6) constantly provides 5 volts (output)
    2. Pin 7 (PD7) checks if 5 volts is there (input)
    3. Pins 9 (PB1), 10 (PB2), 11 (PB3), 12 (PB4) are LED pins (output)
    */
    DDRB |= (1<<DDB1); //enable pin as output
    DDRB |= (1<<DDB2); //enable pin as output
    DDRB |= (1<<DDB3); //enable pin as output
    DDRB |= (1<<DDB4); //enable pin as output
    DDRD |= (1<<DDD6); //enable pin as output

    DDRD &= ~(1<<DDD7); //set up PD7 as the signaling pin (input)
    PORTD |= (1<<PORTD7); //enable pull up resistor

    PORTD |= (1<<PORTD6); //PB6 (pin 6) will be the constant 5v source

    int counter = 1; //initialize
    PORTB |= (1<<PORTB1); //intially on

    while (1)
    {
        if (PIND & (1<<PIND7)) {
            if (counter == 0) {
                PORTB |= (1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB &= ~(1<<PORTB3);
                PORTB &= ~(1<<PORTB4);
                counter ++;
                _delay_ms(2000);
                } else if (counter == 1) {
                PORTB &= ~ (1<<PORTB1);
                PORTB |= (1<<PORTB2);
                PORTB &= ~(1<<PORTB3);
                PORTB &= ~(1<<PORTB4);
                counter ++;
                _delay_ms(2000);
                } else if (counter == 2) {
                PORTB &= ~ (1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB |=(1<<PORTB3);
                PORTB &= ~(1<<PORTB4);
                counter ++;
                _delay_ms(2000);
                } else {
                PORTB &= ~ (1<<PORTB1);
                PORTB &= ~(1<<PORTB2);
                PORTB &= ~(1<<PORTB3);
                PORTB |=(1<<PORTB4);
                counter = 0;
                _delay_ms(2000);

            }


            }
        }
    }
```

4. 

5. Because button presses are considered less frequent events in the order of 1000s of clock cycles, and the ISR is relatively simple to execute (toggling an LED and the edge of the button press to be captured, as well as clearing the input capture flag), ISR will use significantly less CPU power to check if a condition is met compared to polling.
6. In 16mHz clock, each tick is 6.25e-8 seconds
   For 30ms, there are 480000 ticks
   For 200ms there are 3200000 ticks
   For 400ms there are 6400000 ticks
7. A prescaler allows us to slow down the clock by dividing the clocks by a division factor. This can be done either by slowing down the system clock or the timer clock.
8. https://drive.google.com/file/d/1fFft1R94sWXJYl2Fv02vv-ciHhiOCd38/view?usp=sharing
9. Someday I will rule you all