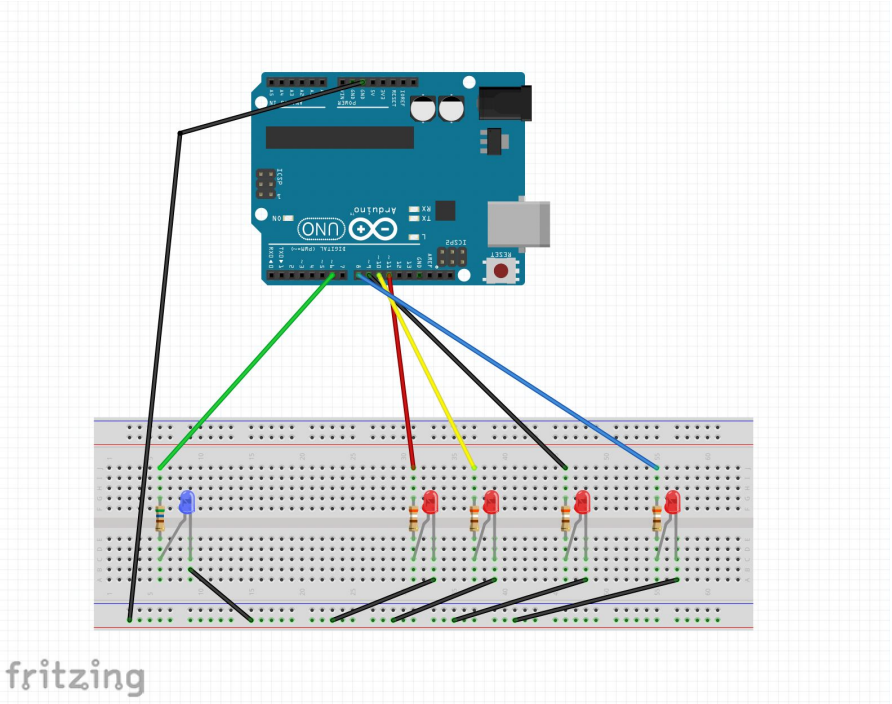
Binary PWM Counter

Zach East

THE BASIC IDEA

- The basic idea of my project is to expand on using the timer interrupt and PWM.
- Previously we've mainly used PWM to pulse a single light or play a tone
  - I wanted to use the pulsing as some sort of indicator so I thought timing something to the pulsing of PWM would be interesting
- I decided I wanted to make a counter that counts how many times an LED controlled by the PWM would pulse
  - Since I don't have unlimited lights , I decided to have 4 LEDs count up visually with binary (it resets after 16 times)

CIRCUIT DESIGN



- This final design requires
  - 11 wires
  - 5 resistors (220 ohm or higher prefered)
  - 1 LED for the PWM (I chose blue)
  - 4 LEDs for the binary counter (I chose red)
  - Digital pin 6 is the output for the PWM
  - Digital pins 8-11 are outputs for the binary counter

CODE

- Based on how we did the PWM for lab 08, I start with setup
  - Enable ability to use delay and interrupts
  - Set up base duty cycle for the PWM
  - Create an integer that keeps track of of the number we're counting
- For the second chunk we have the function binary_check()
  - This function sets the output of the binary counter LEDs based on what number is passed to it (binary_num)
  - I wanted to do this more mathematically but couldn't figure out how to do it bare metal

```
#include<avr/io.h>
#include<avr/delay.h>
#include<avr/interrupt.h>

static int duty_cycle = 200; //Duty cycle starting point of 200
static int delta = 20; //Delta is the amount the duty cycle will change every intterupt
int binary_num = 0;//Binary num is a counter for the current number to be converted to binary
```

```
void binary_check(int bin)
{
  int number = bin;

  if(number == 0){
    PORTB = 0b00000000;
  }
  else if(number == 1){
    PORTB = 0b00000001;
  }
  else if(number == 2){
    PORTB = 0b00000010;
  }
```

CODE CONTINUED

- Our ISR is the TIMER 1 COMPA interrupt
- The basic idea is that every interrupt, we add the delta to our duty cycle and assign that to OCR0A
  - This sets the PWM duty cycle to change to whatever was just assigned
- Every time the duty cycle gets too high and the LED blinks:
  - It updates the binary number by 1 or resets it (depends on what is appropriate)
  - Makes delta negative to switch directions for the PWM
- Every time the PWM duty cycle reaches zero it makes the delta positive to start ramping up the LED again
- After both of these checks, we set the duty cycle and call binary_check() to update the current LED statuses

```
//ISR for the compare match A interrupt of Timer1.
ISR(TIMER1_COMPA_vect) // Interrupt Service Routine for overflow mode
{
  duty_cycle = duty_cycle + delta; //Every time we hit the interrupt add delta to the duty cycle

  //If duty cycle is higher than max of OCR0A (LED BLINK)
  if (duty_cycle >= 0xFF ){

    //Increase the binary counter if within range
    if(binary_num<15){
      binary_num = binary_num+1;
    }
    //Reset when it gets too high
    else{
      binary_num = 0;
    }
    delta = -delta; // reverse direction: increasing <--> decreasing duty cycle
  }
  //When duty cycle is zero
  if (duty_cycle == 0){
    delta = -delta; // reverse direction: increasing <--> decreasing duty cycle
  }

  //Once duty cycle ,delta , and binary_num are updated
  binary_check(binary_num);//Call binary_check to set the binary counter LEDs
  OCR0A = duty_cycle; //Set brightness of pulsing LED using PWM duty cycle
  _delay_ms(100);
}
```

CODE CONTINUED

- For our main function it's mostly setup:
  - We clear the interrupt data
  - Set the output pins and set the Binary LEDs to be off at the start
  - We set up the timer and comparison behavior
  - Make sure that our PWM is active
  - Our prescaler isn't super fast or super slow but you could modify it to be that way

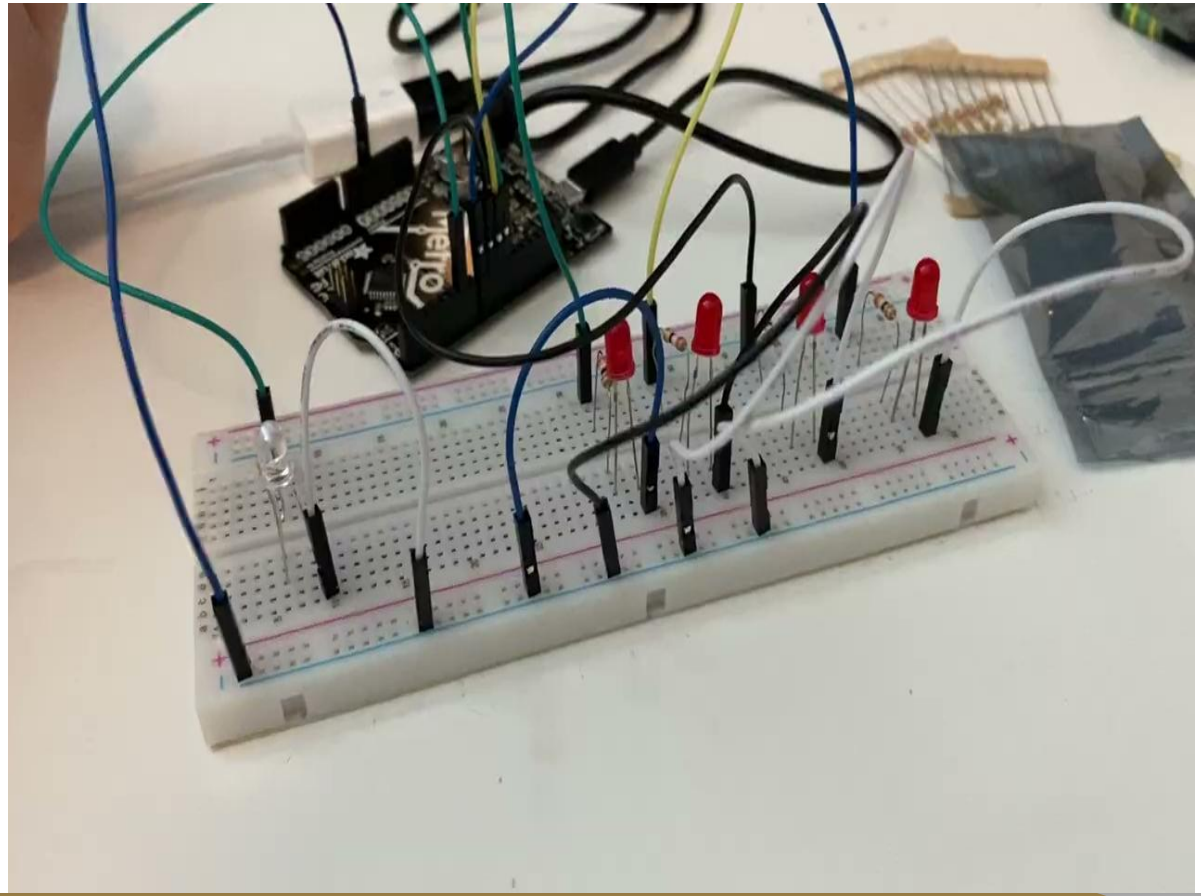
```
//MAIN FUNCTION
int main(){
  cli(); //Clear interrupt data
  DDRD = (1<<6); //Set port D6 as output
  DDRB = 0b00001111; // Set pins 8-11 as output pins
  PORTB = 0b00000000; //set the pin logic to 0 // pull up register inactive

  OCR0A = 10; //Controls PWM
  TCCR0A = 0b10000011;
  TIMSK0 = 0b00000000; //No need for interrupt
  TCCR0B = 0b00000100;

  //Controls separate timer for interrupt
  OCR1A = 47999; //What point the comparison happens
  TCCR1A = 0x00; //No pin behavior
  TCCR1B = 0b00001001; //WGM bit compare to OCR1A to timer and set prescale
  TIMSK1 = 0b00000010; // only set up compa

  sei();
  while(true){
  }
}
```

Demonstration:



Questions?