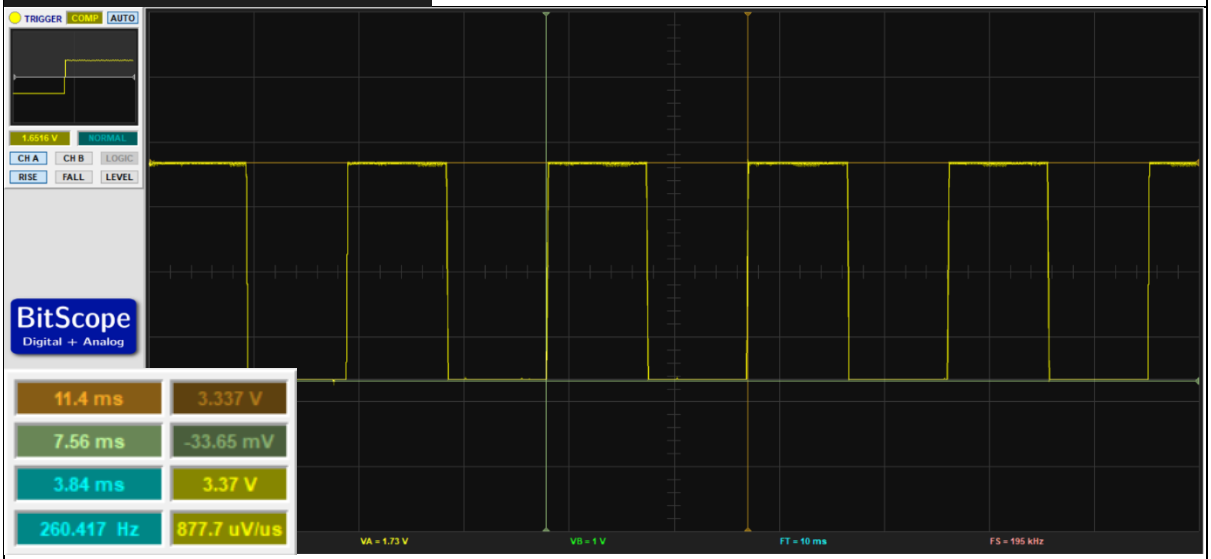
C8 – Timers / Counters and Pulse Width Modulation

3.1 Producing an Audio Frequency Square Wave

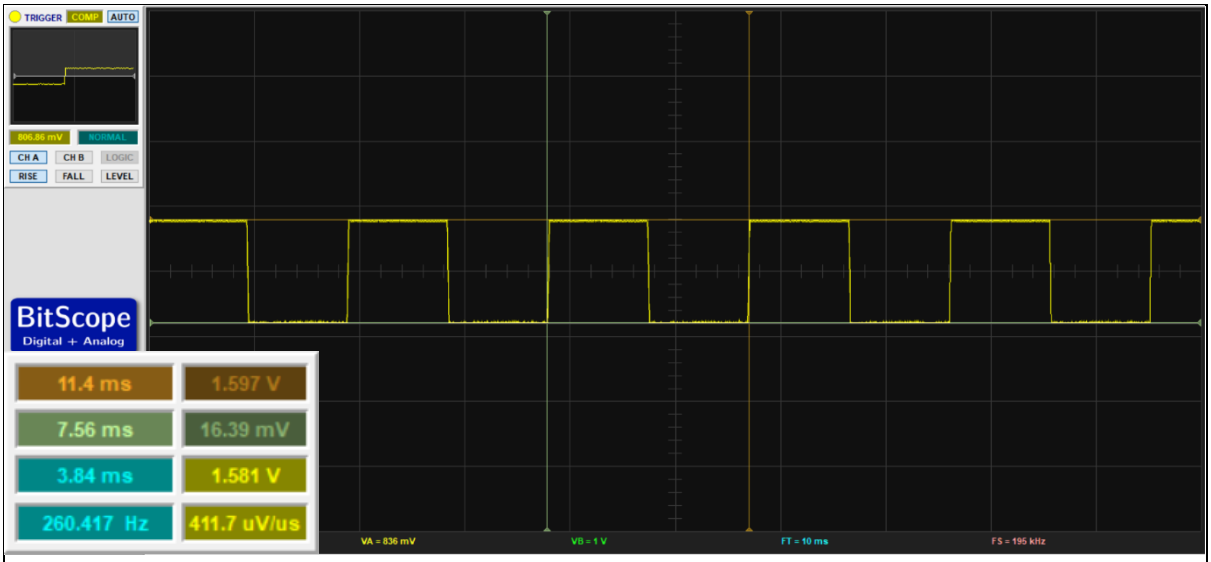
```
void init_tone(void) {
    //D5 as output
    DDRD |= _BV(PD5);

    TCCR1A = _BV(COM1A0) | _BV(WGM10);      //set toggle on compare, phase and frequency

    TCCR1B = _BV(WGM13) | _BV(CS10);        //256 time prescaler (clock = 46,875Hz)
}
```

```
void tone(uint16_t frequency) {

    OCR1A = F_CPU / (frequency * N * 2);
}
```

```
#define MIDDLE_C 262

void init_tone(void);
void tone(uint16_t frequency);

int main() {
    init_tone();
    tone(MIDDLE_C);

    while(1);
}
```



260.417Hz ≈ 262Hz = Middle C



1.581V ≈ ½ x 3.3V

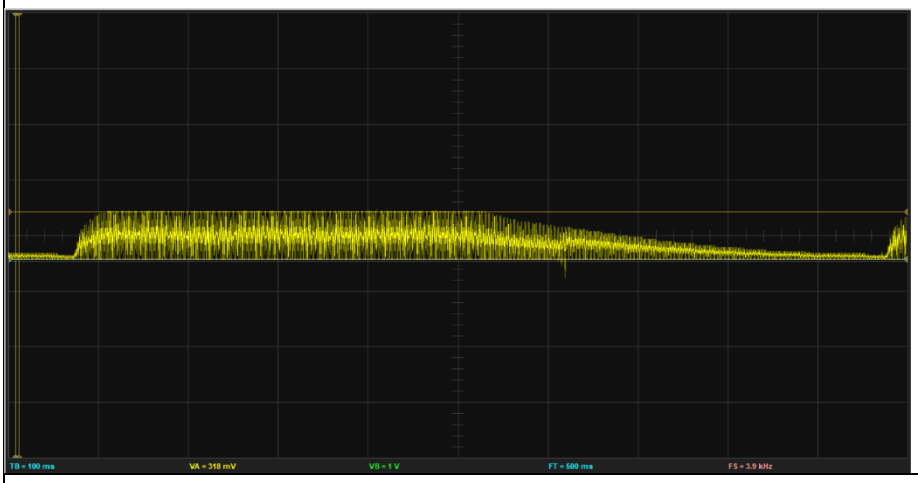
3.2 Generate a Tone Scale

```
int main() {
    init_tone();
    int i=0;

    while(1) {

        //scale up (including top and bottom note)
        for(i=ET_SCALE_C; i<ET_SCALE_TOP; i++) {
            tone(et_scale[i]);
            _delay_ms(BEAT_T120);
        }

        //scale down (-2 so dont repeat last note and > so dont repeat bottom note)
        for(i=ET_SCALE_TOP-2; i>ET_SCALE_C; i--) {
            tone(et_scale[i]);
            _delay_ms(BEAT_T120);
        }
    }
}
```

Performs chromatic scale ascending and descending as expected.

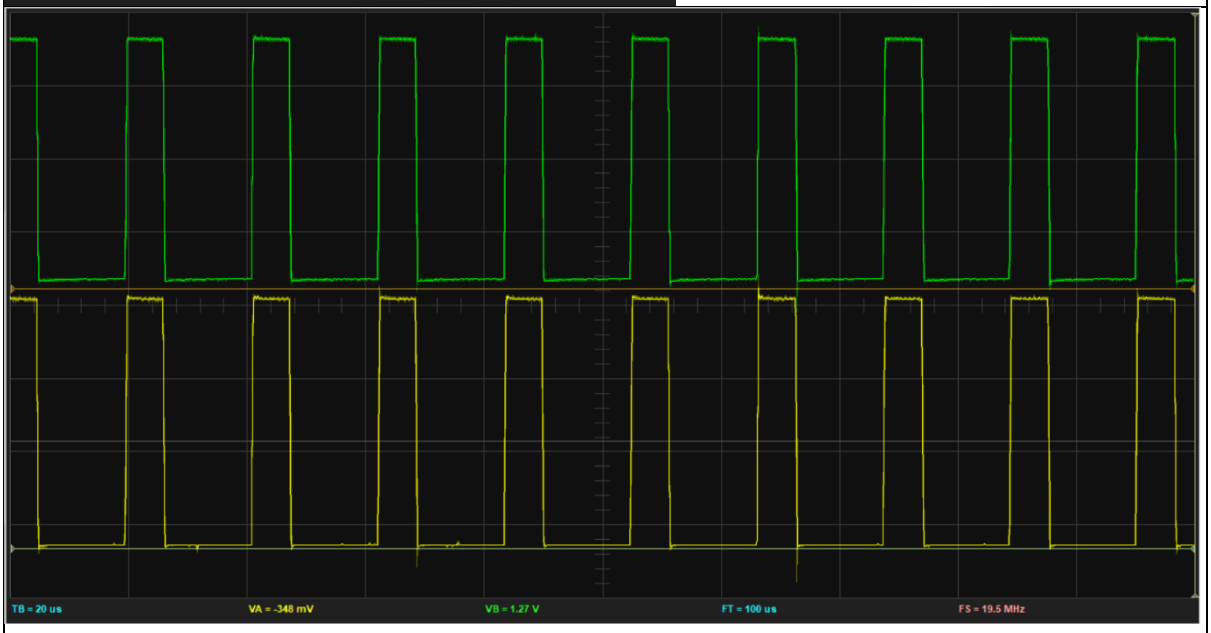
The diode is to control the charge in the capacitor. When the trigger pin is high the capacitor charges, then when it goes low the diode prevents the change stored in the capacitor from going through the trigger pin to ground and instead goes through the rest of the circuit.

3.3 Play a Tune

Melody plays as expected when function are replaced with my written tone functions.
One using the CD4066 chip, the notes the ease in and the die away rather then the hard on off sound in the previous circuit. As seen in the waveform below. 
I used a slightly longer pulse time of 20ms. Having too short of a trigger length results in the capacitor not being charged enough and the note only sounding for a very short amount of time which is not ideal.

3.4 Volume Control with PWM

```
void init_volume() {

    //pins pd7 as output
    DDRD |= _BV(PD6) |_BV(PD7);

    //Fast PWM Mode 7 - set WGM20 and WGM21
    //set two outputs non-inverting - set COM2A1, COM2B1
    TCCR2A = _BV(WGM20) | _BV(WGM21) | _BV(COM2A1) | _BV(COM2B1);

    //no clock divide so frrequency of 47kHz
    TCCR2B =  _BV(CS20);

    //OCR2A = 64;
    //OCR2B = 96;

    //50% volume
    volume(127);

}
void volume(uint8_t x) {

    //sets the pulse width of x/255 * 100 %
    OCR2A = x;
    OCR2B = x;
}
```



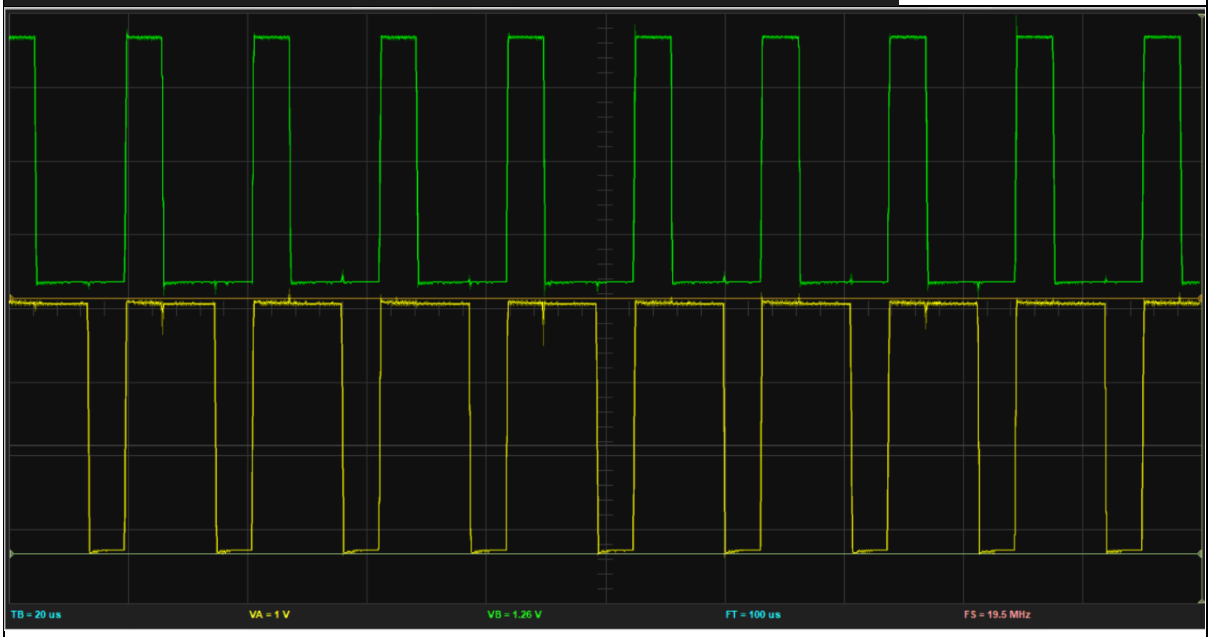
When connecting PD7 to the trigger input on the circuit it controls the volume of the tone as expected.

When a couter i is implemented to increment each time a note is played and therefore increase the volume, once it gets to the maximum value of 255 it then overflows and go back down to 0. This is because i is a uint8_t which means it only takes positive values of an 8 bit length so after 255 it goes back to 0, ignoring the 9th bit which would be set to 1 for 256.

4 Stereo Panning

Since I already had both PWM channels A and B set up for init_volume, I implemented the new function pan(uint8_t l, uint8_t r). Where the left and right channel can be set at volumes independently. To get this to a single value I put in my main loop pan(i, 255-i);

```
void pan(uint8_t l, uint8_t r) {

    //sets the pulse width for channels A and B (L and R)
    OCR2A = l;
    OCR2B = r;
}
```



As we can see the high pulse of channel A is equal to the low pulse of channel B, as expected for panning.