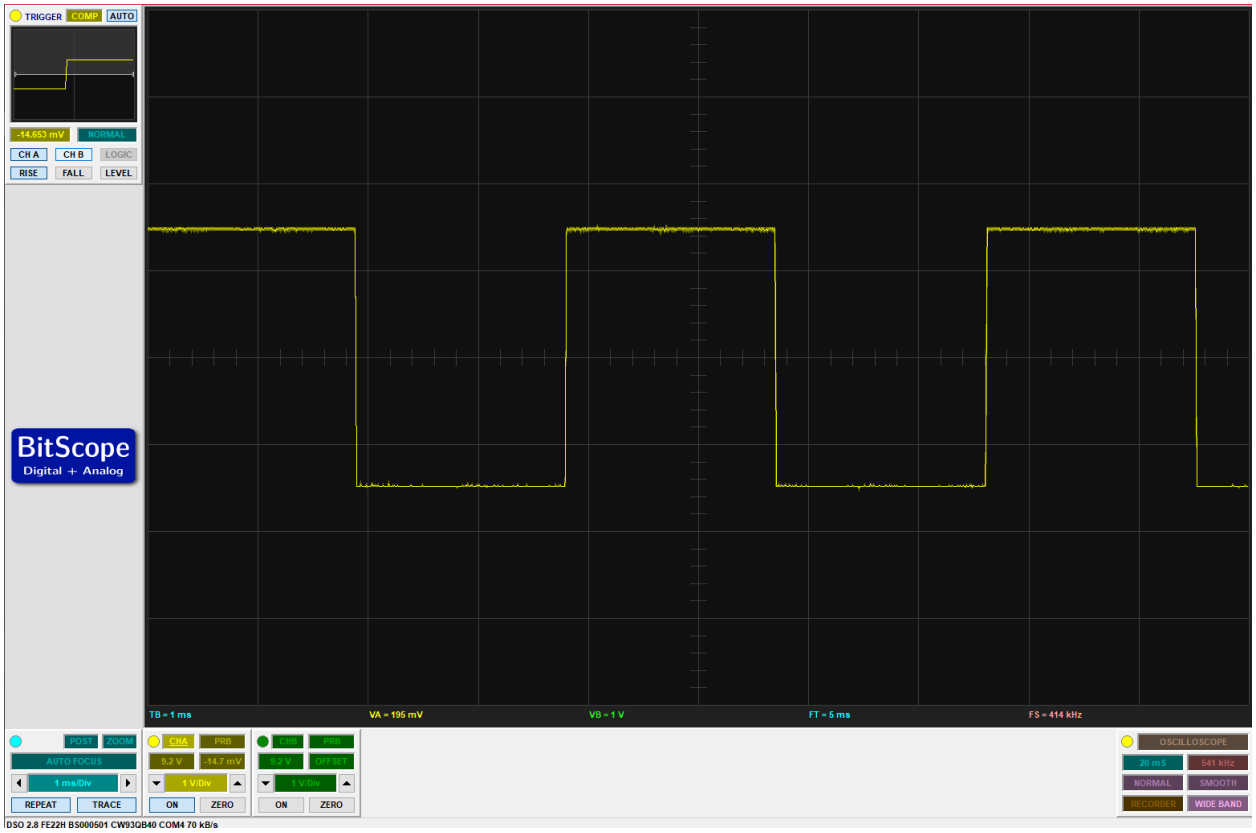
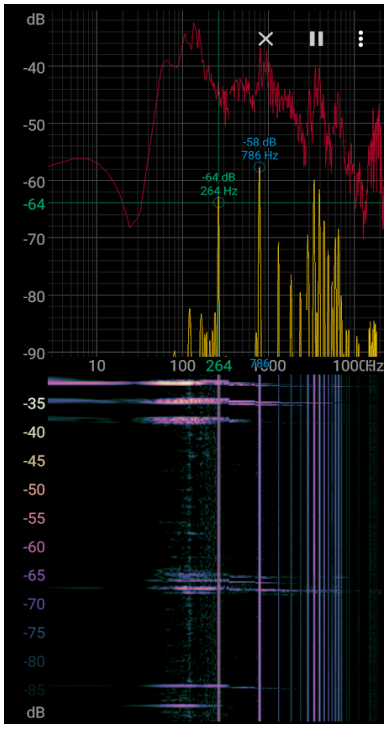
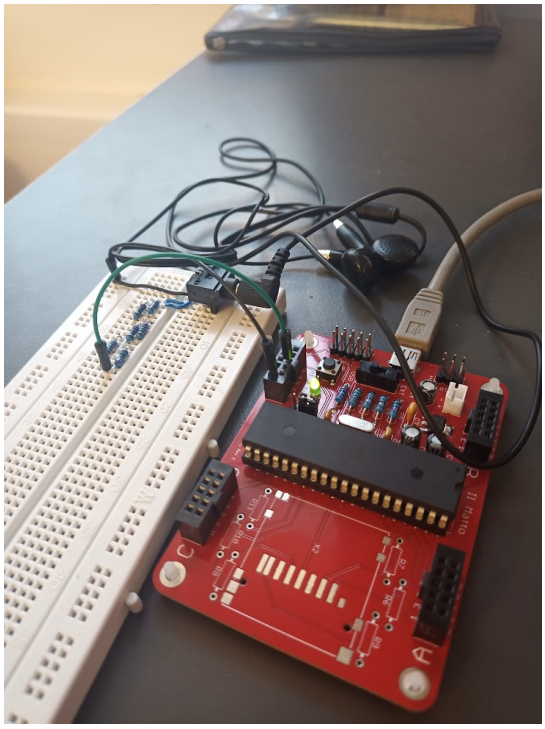
25/11/20

Setting my defined tone to 262 Hz, a middle C tone. I connected my bitscope to the output at pin D5 (pin for Timer1). I measured the output frequency as a clear square wave.



$$f = \frac{1}{T}$$
$$f = \frac{1}{3.9\times10^{-3}}$$
$$f = 256.4\ Hz$$

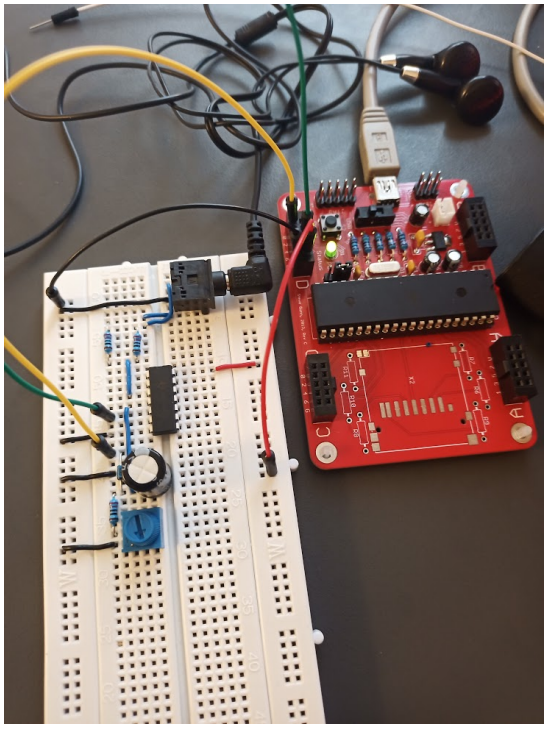


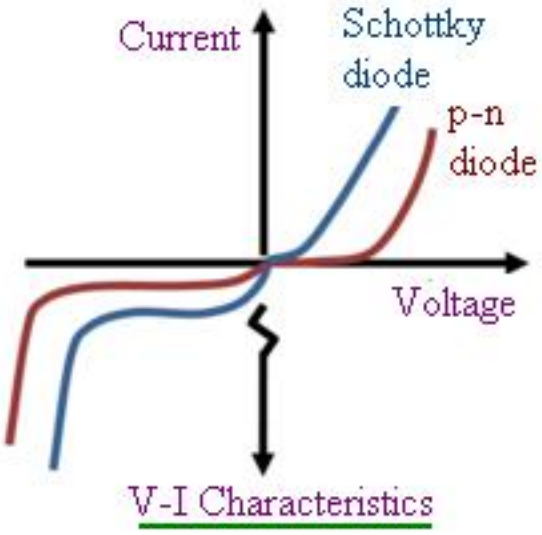


I then modified the main function of my code so that it would scale up and down the frequency range in “et_scale.h”.

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

   for (;;)
   {
      int i;
      for (i = 0; i <= ET_SCALE_TOP; i++)
      {
         tone(et_scale[i]);
         _delay_ms (100);
      }
      for (i = ET_SCALE_TOP; i >= 0; i--)
      {
         tone(et_scale[i]);
         _delay_ms (100);
      }
   }
}
```





The schottky diode used on the trigger will limit the current supplied to the capacitor and then the capacitor will slowly discharge over the duration of the note. This creates a decaying amplitude which will make the voltage supplied to the headphones more like the wave packet when an instrument creates sound.

Making the pulse time significantly shorter than the length of the note makes the note cut off, making the overall sound more electronic.

I then modified the code so that it would briefly pulse a trigger. This charged the capacitor and let it decay over the note to give it a more natural sound.

```
int main()
{
    uint16_t f;

    DDRB |= _BV(PB7); /* LED */
    DDRD |= _BV(PD0); /* Trigger*/
    init_tone();

    for (;;)
    {
        melody2freq(melody); /* initialise  */
        while ((f = melody2freq(NULL)) != M2F_END)
```

```
        {
            if (f == M2F_UNKOWN)
            {
                continue; /* skip unknown symbols */
            }

            tone(f);
            PORTD |= _BV(PD0);      /* set Trigger */
            _delay_ms(10);
            PORTD &= _BV(PD0);      /* Clear Trigger  */
            _delay_ms(STEP_DELAY_MS);
            PORTB ^= _BV(PB7);      /* toggle LED */

        }
        _delay_ms(STEP_DELAY_MS);
        _delay_ms(STEP_DELAY_MS);
    }
}
```

I then added a volume control using a second PWM pulse to control the analog switch chip.

```
void init_volume(void)
{
    DDRD |= _BV(PD6);
    DDRD |= _BV(PD7);

    TCCR2A = _BV(COM2A1) /* toggle OC2A on match */
         | _BV(WGM21)    /* fast PWM, */
         | _BV(WGM20);   /* fast PWM, */
    TCCR2B = _BV(CS20);  /* prescaler set to 1 */
}

void volume(uint8_t x)
{
    uint8_t vol = (256*(x/100));
    OCR2A = vol;
}
```

I finally modified my code so that it would increment the volume every time a note was played. Once the volume had reached max it would cause an overflow and the volume would start counting from zero again.

```
uint8_t vol = 1;

    for (;;)
    {
        melody2freq(melody); /* initialise  */
        while ((f = melody2freq(NULL)) != M2F_END)
        {
            if (f == M2F_UNKOWN)
            {
                continue; /* skip unknown symnols */
            }

            tone(f);
            volume(vol);
            _delay_ms(STEP_DELAY_MS);
            PORTB ^= _BV(PB7);      /* toggle LED */

        }
        _delay_ms(STEP_DELAY_MS);
        _delay_ms(STEP_DELAY_MS);

        vol++;
    }
```