Illuminated Guitar

Oli Thompson

Abbreviations

PLA - Polylactic Acid

ADC- Analogue to digital converter

FHT – Fast Hartley transform

FFT – Fast Fourier transform

SRAM – Static random-access memory

GND – Ground

DPDT – Double pole double throw

PCB – Printed circuit board

SMT - Surface mount technology

Figure 1: A psychedelic light show

Introduction

Our group of three, Ric Zhang, Emily Branson and myself (Oli Thompson) set out to build a project that allowed visualisation of music in real time on the face of an electric guitar. We wanted to create an entertaining and interesting experience for both the audience and guitarist, taking inspiration from the psychedelic light shows in rock concerts (Figure 1). Having overcome several technical challenges, the project was then developed to include a guitar tuner that outputted to the face of the guitar and further refinement is still ongoing. This report documents the project's journey and its technical specification.

Design Considerations

We set out to modify an existing electric guitar and chose to use a Fender Stratocaster as its flat body allowed for a simpler construction. The modifications are all fully reversible. We decided early on that the guitar should be fully self-contained, meaning all the signal processing would be done on an onboard microcontroller requiring only a power cable to connect it to the mains.

We elected to display the amplitude of the different octaves within the human range of hearing using a matrix of 206 digitally addressable LEDs. We chose an external microphone as the project's input, rather than the guitar's output jack for two reasons; using a microphone allowed the spectrum of the whole band's music to be displayed and we wanted to be able to display sounds across the full spectrum of human hearing, whereas the guitar has a limited range of about 4 octaves.

Crucially, the guitar needed to maintain its full functionality which meant designing the LED matrix and its frosted acrylic diffusion panel to be as low profile as possible and also allowing the control knobs, pickup selector, output jack and tremolo arm to be fully functional whilst not impeding the player's hand. As a consequence, all the electronics had to be concealed in the arm cutaway behind the matrix, so that they were hidden from view allowing the guitar to maintain its original shape as much as possible.

We designed a simple user interface consisting of 3 toggle switches (Figure 2) that switch between the various output modes and turn the op-amp (operational amplifier) on and off. This was to allow the performer to discreetly change the settings on the guitar itself.

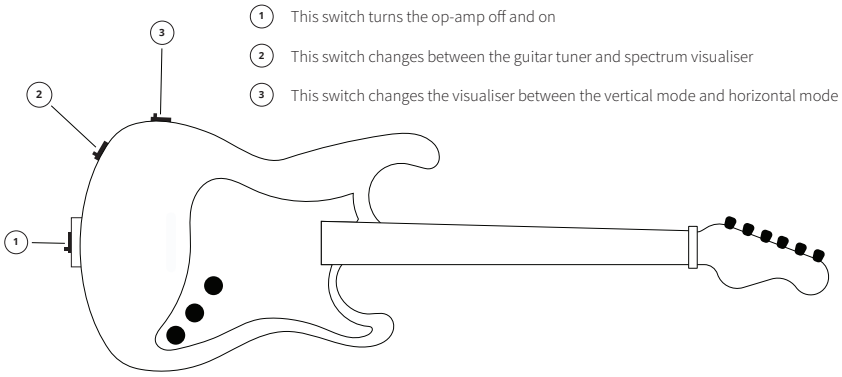

Figure 2: User Interface Explained

Technical Journey

Mechanical Design

A CAD model of the guitar was sourced online (Figure 3) [1] and an acrylic sheet was cut to match the guitar's profile (Figure 4). The addressable LED strips were secured to the acrylic with epoxy resin and soldered together (Figure 5). The existing pick-guard was then removed and all the guitar's hardware was installed into the new acrylic sheet which was screwed into the guitar's body using the existing screw holes. The acrylic sheet was designed to mount the audio jack perpendicular to the face of the guitar (Figure 6), this way fewer LEDs are obscured. A frosted acrylic sheet was mounted over the LEDs using small brass standoffs (Figure 7) and we 3D printed new, custom control knobs out of translucent PLA to allow the light to shine through them (Figure 8).

Figure 3: Guitar CAD model and custom acrylic profile

Figure 4: The laser cut acrylic profile

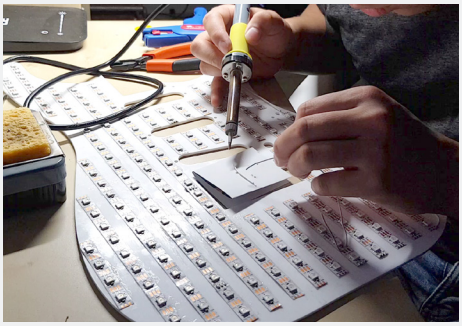

Figure 5: Ric Soldering the LED matrix by hand

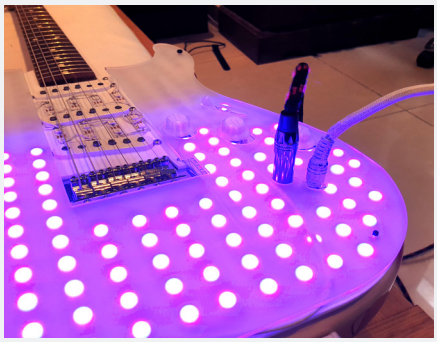

Figure 6: The modified output jack position and DC input

Figure 7: The mechanical construction with the brass stand-offs

Figure 8: The 3D printed control knobs in translucent PLA

Spectrum Visualiser

The 10-bit ADC of an Arduino Nano (chosen for its small form factor) takes the analogue input from a MAX9814 electret microphone board (Figure 9) [2]. This breakout board includes an amplifier module with built-in gain control. We used a gain of 60dB. To detect the frequency spectrum, we implemented an FHT algorithm from Open Music Labs [3]. This is similar to an FFT but does not involve complex numbers and as such is more computationally efficient. Values from the ADC are stored in the SRAM. The data is then reordered and windowed and the FHT returns (in our case) 128 frequency bands. Displaying the amplitude of these bands would not make an effective spectrum visualiser as human hearing works logarithmically. Therefore, we segmented the frequencies into octaves by averaging the contents of the following bands (0, 1, 2:4, 5:8, 9:16, 17:32, 32:64, 65:128) into 8 octave bins. We struggled to get a reliable signal for the lowest octave and were forced to discard the lowest octave bin. This is likely to be a hardware issue with the microphone.

The Arduino executes a reading each time the software loops, therefore as the code grew in complexity the ADC's sampling rate decreased. This could be rectified somewhat by decreasing the resolution, however the sampling rate is likely to be less than 40kHz (twice the maximum frequency of human hearing), meaning that some aliasing may be occurring in accordance with the Nyquist-Shannon sampling theorem. Attempting to measure this slows down the Arduino further and gives inaccurate results, so it is very difficult to determine if this is true, however the visualiser is still pleasing to look at even if the spectral analysis is not scientifically rigorous.

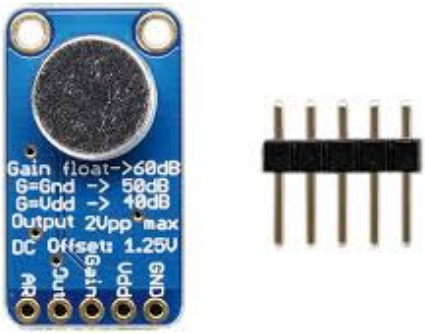

Figure 9:The MAX9814 electret microphone break-out board

Displaying the amplitude of each octave on the LED matrix involved using a lookup table to map the LEDs onto the irregular shape of the guitar (Figure 10). This time-consuming process involved manually counting each LED and assigning it a serialised value and a co-ordinate on the matrix, to which the code would write the RGB value. We used the NeoPixel library [4] along with WS8012 addressable LEDs.

```
byte lookuptable[16][26]={
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  7,  8,  0,  0,  0,  0 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  5,  6,  0,  12, 11, 10, 9,  0,  0 },
      {0,  0,  0,  0,  0,  0,  1,  2,  3,  0,  0,  0,  0,  0,  0,  4,  0,  0,  13, 14 ,15 ,16 ,17, 18, 19, 20, 0  },
      {0,  0,  0,  0,  0,  0,  0,  0,  34, 33, 32, 31, 30, 29, 0,  0,  28, 27, 0,  0,  26, 25, 0,  24, 23, 22, 21 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  69, 68, 67, 66, 65, 64, 63, 0,  62, 61, 60, 59, 58, 57, 56, 55, 54, 53 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  70, 71, 0,  85, 84, 0,  0,  0,  0,  0,  0,  0,  90, 89, 88, 87, 86 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  73, 72, 0,  82, 83, 0,  0,  0,  0,  0,  0,  0,  91, 92, 93, 94, 95 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  74, 75, 0,  81, 80, 0,  0,  0,  0,  0,  0,  0,  100,99, 98, 97, 96 },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  77, 76, 0,  78, 79, 0,  0,  0,  0,  0,  0,  0,  101,102,103,104,105},
      {0,  0,  0,  0,  0,  126,125,124,123,122,121,120,119,118,117,116,115,114,113,112,111,110,109,108,107,106},
      {0,  0,  0,  0,  0,  127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147},
      {0,  0,  0,  0,  169,168,167,166,165,164,163,162,161,160,159,158,157,156,155,154,153,152,151,150,149,148},
      {170,171,172,173,174,175,176,177,0,  0,  0,  0,  0,  181,182,183,184,185,186,187,188,189,190,191,192,193},
      {0,  0,  180,179,178,0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  202,201,200,199,198,197,196,195,194,0  },
      {0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  203,204,205,206,0,  0,  0 }
  };
```

Figure 10: The look-up table array

The amplitudes from the octave bins were not standardized and as such we iterated experimentally with mapping the maximum and minimum amplitudes in different ways, to achieve the most aesthetically pleasing result. We set the noise floor of the microphone to illuminate the first row of LEDs, meaning that there would always be some lighting even in a silent room. This was soft coded for adjustable scaling in noisier environments.

Figure 11: The guitar in the horizontal display mode

Figure 12: The guitar in the vertical display mode

It became apparent that the spectrum visualizer could turn the guitar into an aesthetic static installation when not in use, and hence the code was modified to allow the spectrum visualiser to be displayed either horizontally, for when the guitar is in use (Figure 11), or vertically when it is placed on its stand (Figure 12). We implemented a conditional statement at the end of each loop to check the position of the toggle switch and adjust the display accordingly. This was done instead of using hardware interrupts as these appeared to introduce noise into the ADC readings. The final stage of the spectrum visualiser was a piece of code that iterated through the RGB colour space to allow the spectrum visualiser to change colour.

Guitar Tuner

With the spectrum visualiser finalised we considered other uses for the LED matrix. We decided to design a guitar tuner that displayed the note being played and its variation from the correct frequency value on the LED matrix [6]. This presented 2 major technical challenges: Firstly, the output from an electric guitar has too low of an amplitude (voltage) for the Arduino's ADC to measure and the amplitude oscillates around ground, therefore half of the wave form is a negative voltage that the Arduino cannot read. Secondly, the FHT algorithm does not work well discerning similar individual frequencies from one another, meaning a different time domain method of signal processing would be required.

To amplify the electric guitar's signal, we wired the guitar's output directly to a non-inverting op-amp circuit with a gain of 5.54dB (Figure 13). This figure was determined by measuring the guitar's signal with an oscilloscope (Figure 14). The TL082 op-amp takes both a positive and negative 9V. The negative voltage allows the signal to be amplified below the circuit's ground potential, meaning that two 9V batteries are required. The op-amp only needs to amplify the signal to 5V peak to peak, but 9V is used due to the op-amp's characteristic inefficiency. The gain of the op-amp is set using 2 resistors and a 1:1 potential divider from the Arduino's 5V pin provides a 2.5V DC bias to the op-amp's output. We verified our circuit using the oscilloscope and ensured that the signal would not exceed 5V as this could damage the ADC (note that different guitars will produce different amplitudes at their output jacks). The circuit diagram for the op-amp is shown overleaf (Figure 15). The circuit was manufactured on perf-board and we designed a 3D printed enclosure that would house the circuit board and two 9V batteries. 3D printed covers kept the components in place with an interference fit (Figure 16).

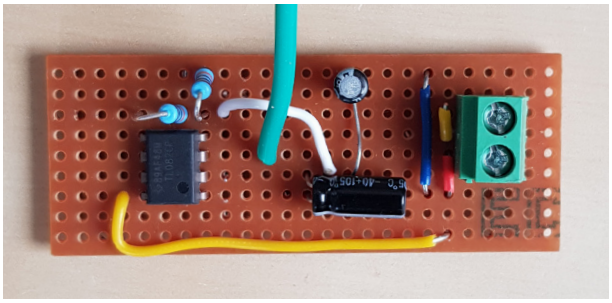

Figure 13: The op-amp prototype on strip-board

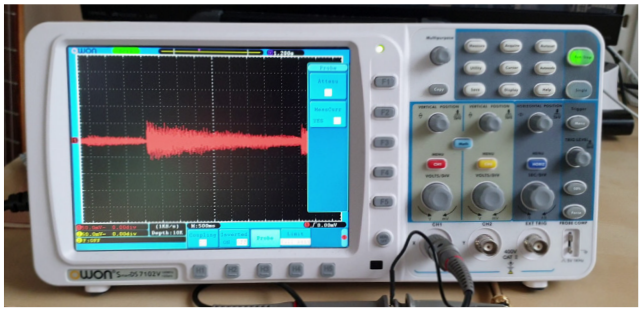

Figure 14: Measuring the output on an oscilloscope

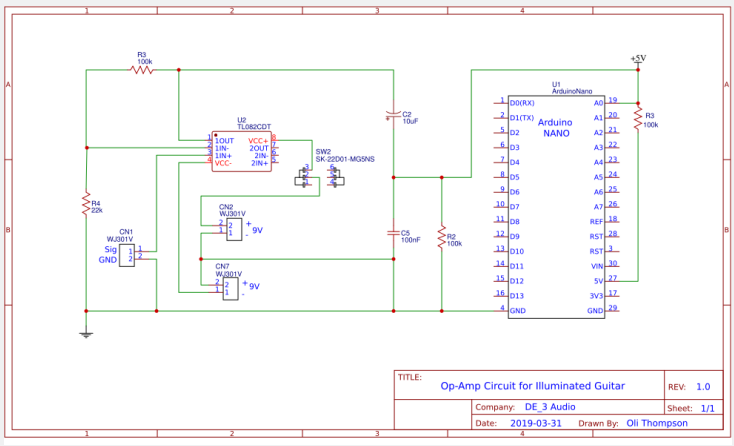

Figure 15: The circuit diagram for the op-amp

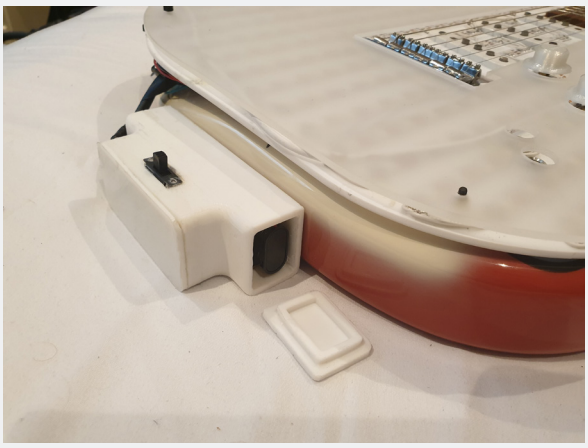

Figure 16: The 3D printed housing for the circuit board and batteries

The guitar tuner works by analysing the frequencies in the time domain using autocorrelation The previous ADC value is stored in a buffer and is compared to the next values measured at a known time interval. By measuring the correlation between the original and delayed signal the periodicity of the signal can be verified, eliminating any unwanted noise. Now a simple peak to peak detection code can measure the time period of the signal and thus find the frequency. The software displays the target note on the matrix (the target note is the one whose frequency is nearest to the current input). Then, by comparing this frequency to the known values (E - 82.4 Hz, A - 110 Hz, D - 146.8 Hz, G - 196 Hz, B - 246.9 Hz, E - 329.6 Hz) the discrepancy is shown on the LED Matrix (Figure 17). One final toggle switch is used to switch between the visualiser and tuner modes.

Electronic Design

There are some aspects of the electronic design that do not fall into the category of spectrum visualiser or guitar tuner. The project is powered by a 5V, 8A mains power supply with a 1000nF de-coupling capacitor across 5V and GND to smooth out any voltage spikes. The toggle switches are brought to GND when switched, as the Arduino's inputs have internal pullup resistors. When the DPDT toggle switch is thrown one circuit brings one of the Arduino's digital pins low, signalling to the Arduino that the switch has been thrown, and the other circuit switches between connecting the microphone or connecting the op-amp output to the Arduino's ADC. This feature is critically important as otherwise the microphone's output would be added to the guitar's output, meaning that both signals would be amplified when the guitar was played through an amplifier.

Figure 17: The tuner in use

Further Developments

After the project was completed and fully functional I decided to develop it further as a personal exercise in electronics and PCB design. The Arduino is bulky and includes superfluous components and the break out boards and perf-board based circuitry are well suited to a prototype, but not a developed product. I designed my own dual layer SMT PCB around the ATmega328p microcontroller [6] and sent the designs off to China for manufacture. At the time of writing the board is still in production but the circuit diagram (Figure 18), top and bottom layer views (Figure 19 and 20) and the PCB render (Figure 21) can be seen to the right and overleaf. The curved shape of the board allows it to fit underneath the arm cutaway and the bootloader and programmer headers allow the bootloader to be burned onto the chip and the chip to be programmed through an FTDI USB to serial interface.

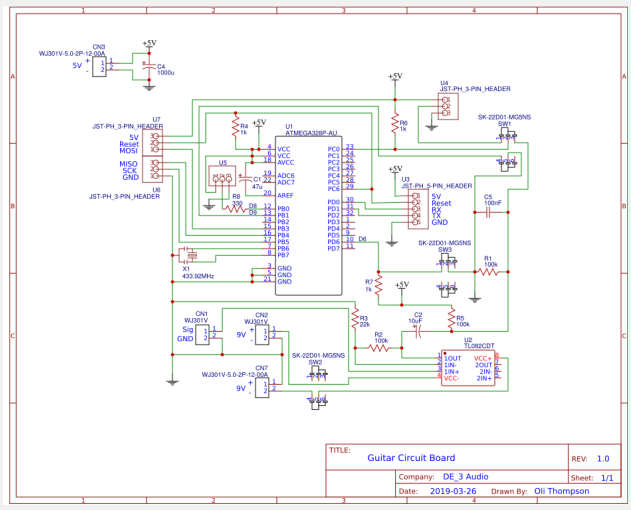

Figure 18: My circuit diagram for the whole project

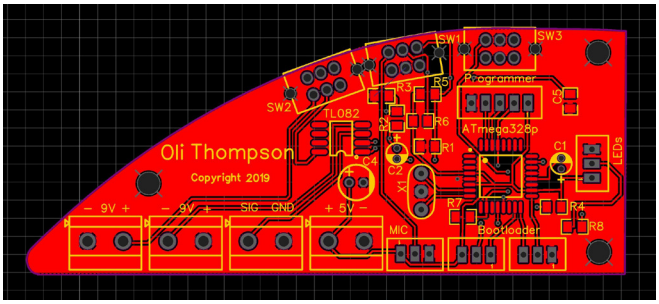

Figure 19: The top layer of the PCB (red is copper)

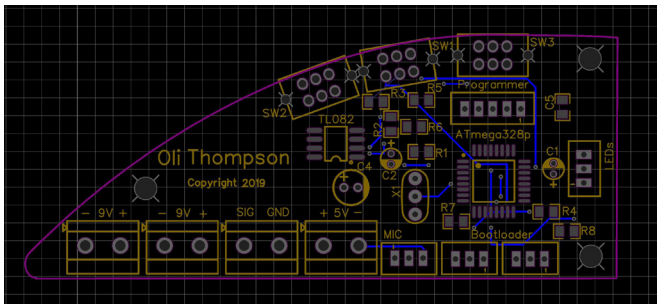

Figure 20: The bottom layer of the PCB (blue is copper)

Concluding Remarks

Overall, I am very pleased with how the project turned out and it has been a valuable learning experience. The project works reliably and uses just under 500 lines of code. The main issue to resolve is the high frequency noise that can be heard when the guitar is heavily distorted. This is likely to be caused by the Arduino's quartz crystal oscillator or some other component interfering with the pickups and could be mitigated either by shielding the electronics, not using single coil pickups that are prone to electro-magnetic interference or using a bandstop filter on the guitar's output.

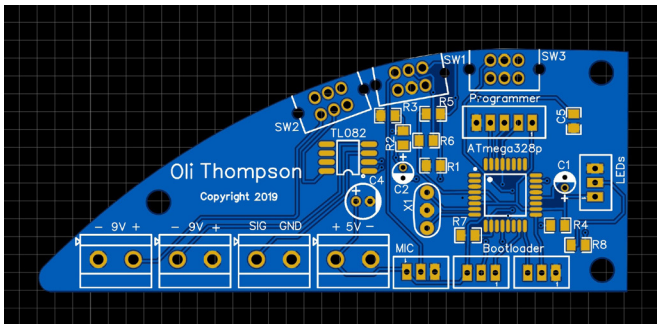

Figure 21: The rendered view of the PCB [7]

References

[1] https://grabcad.com/library/fender-stratocaster-5
[2] https://www.maximintegrated.com/en/products/analog/audio/MAX9814.html
[3] http://wiki.openmusiclabs.com/wiki/ArduinoFHT
[4] https://github.com/adafruit/Adafruit_NeoPixel
[5] https://www.instructables.com/id/Arduino-Guitar-Tuner/
[6] https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf
[7] https://easyeda.com/