Music Visualizer Mini Project Report

Ankit Verma, Himanshu Verma, DESE, IISc

ABSTRACT

This report contains the concept and way to implement music visualizer on TIVA-TM4C123G Board with Neo-pixel LEDs array and BOOSTER-K350QVG display.

I INTRODUCTION

Music Visualizer is a very useful tool to analyse the different frequency component present in the sound waves. It is also used to detect the noise component and helps in desigining appropriate filters to remove it. It is also used as a decoration on speakers these days. The algorithm behind the working of this is discrete fourier transform.

II DISCRETE FOURIER TRANSFORM

Discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. The DFT of a sequence x_n is given below:

$$F[k] = \sum_{n=0}^{N-1} x_n . e^{-\frac{i2\pi}{N}kn}$$

We may write this equation in matrix form as:

$$\begin{pmatrix} F[0] \\ F[1] \\ F[2] \\ \vdots \\ F[N-1] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{N-2} \\ 1 & W^3 & W^6 & W^9 & \dots & W^{N-3} \\ \vdots & & & & & \\ 1 & W^{N-1} & W^{N-2} & W^{N-3} & \dots & W \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{pmatrix}$$

where W = $e^{(-j2\pi/N)}$and W = W^{2N}... =1.
But the time complexity of DFT is N^2 which is not good for real time applications.
So, a variant of DFT is used which is called Fast Fourier Transform.

III FAST FOURIER TRANSFORM

The fast Fourier transform is a method that allows $O(nlogn)$ computing the DFT in time. The basic idea of the FFT is to apply divide and conquer. We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT for each of them, and combine the results to compute the DFT of the complete polynomial. So let there be a polynomial with degree n-1 , where n is a power of 2, and n>1,

$$F(x) = a_0 + a_1 x + ... + a_{n-1} x^{n-1}$$

We divide it into two smaller polynomials, the one containing only the coefficients of the even positions, and the one containing the coefficients of the odd positions:

$$F_0(x) = a_0 + a_2 x^2 ...... + a_{n-2} x^{\frac{n}{2}-1}$$

$$F_0(x) = a_1 + a_3 x^2 ...... + a_{n-1} x^{\frac{n}{2}-1}$$

Then it is easy to write

$$F(x) = F_0(x^2) + x_1 F(x^2)$$

If $y_k$is the DFT of F(x) then it can be written as below:
$y_k = y_k^0 + w_n^k y_k^1$
where, k = 0,1,..... $\frac{n}{2} - 1$
and
$y_{k+n/2} = y_k^0 - w_n^k y_k^1$
where, k = 0,1,...... $\frac{n}{2} - 1$
Thus, FFT takes $O(nlogn)$ time.

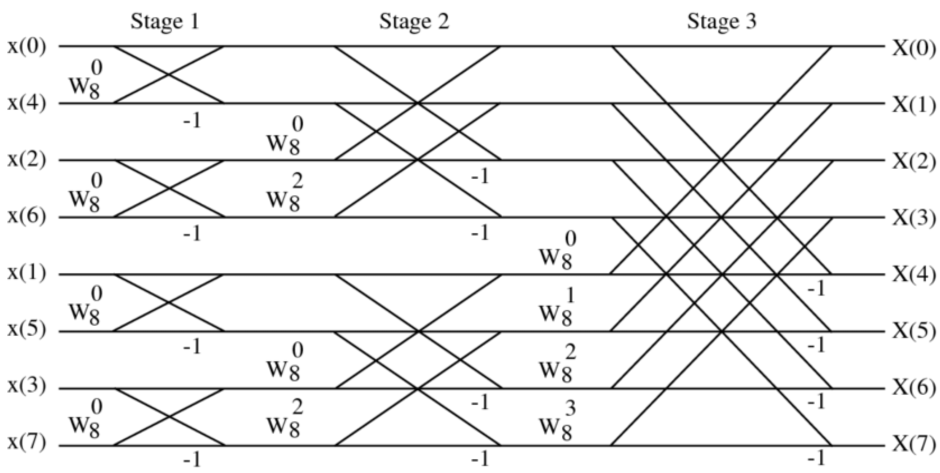

Figure 1: Butterfly Structure in FFT

IV SETUP

In this project, TIVA TM4C123G is used for all the fft computation, sampling and LED matrix driving. LEDs are Neo-Pixel LEDs which are addresseable LEDs. A waveshare mic is used for sampling the audio. Song's 1024 samples are sampled at 8192Hz and then given to 64 points FFT algorithm. The output is then averaged in 8 bins and then given to Neo-Pixel array of 8 lines. The setup configuration is given below:

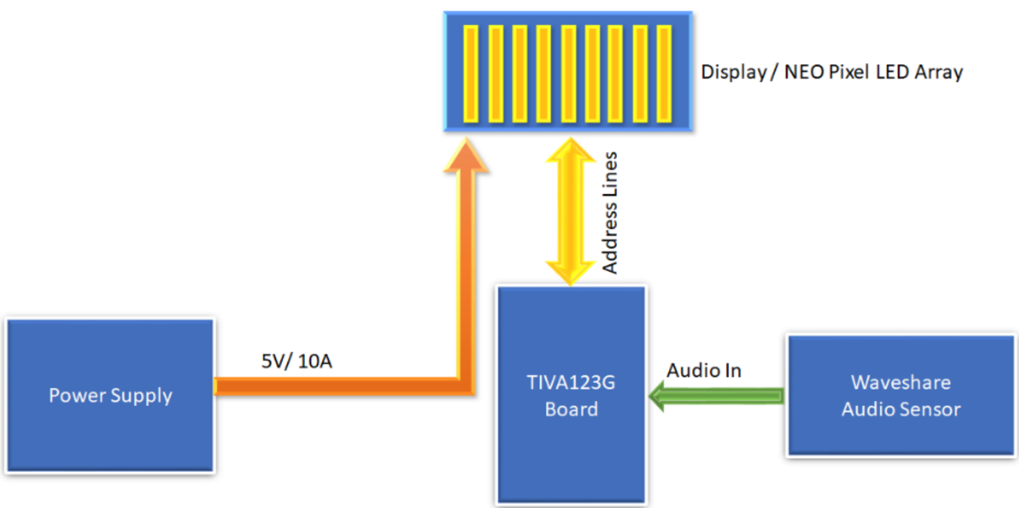

Figure 2: Block Diagram of the setup

4.1 Neo-Pixel LEDs

Neo-pixel LEDs come in form of an array and one LEDs Data_out is connected to Data_in of next LED in the array. The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port.

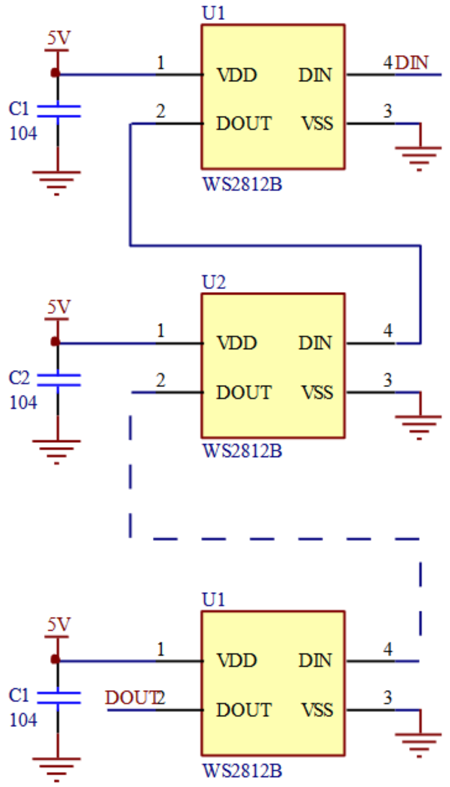

Figure 3: Circuit configuration of Neo-Pixel LEDs

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

Figure 4: Data Frame of Neo-Pixel LEDs

To make these bits one or zero, a pulse with a specific timing has to be given at proper time interval. The timing details are given in following diagrams:

Data transfer time(TH+TL=1.25µs±600ns)

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.8us	±150ns
T0L	0 code , low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.45us	±150ns
RES	low voltage time	Above 50µs	

Figure 5: Timing Details

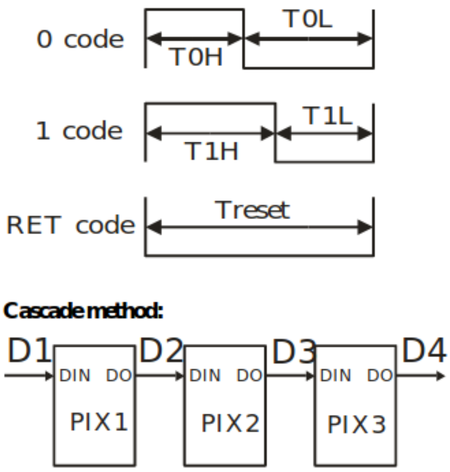

Figure 6: Timing Waveforms

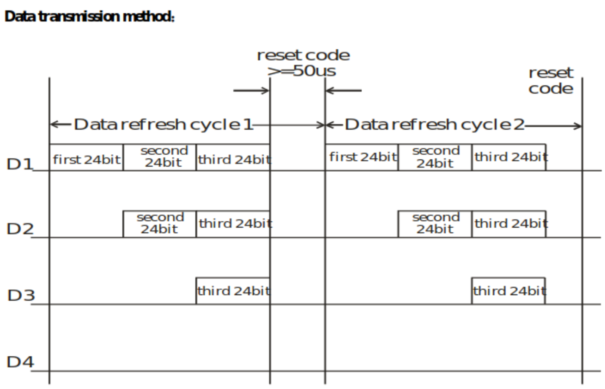

Figure 7: Full Data Frame Time Cycle

4.1.1 Neo-Pixel LEDs wave generation

Waves are generated by writing “nop” after making pulse high and then pulling it to low. All the experiments done and verified on Digital Oscilloscope. Waveforms are given in Figure 8 and Figure 9.

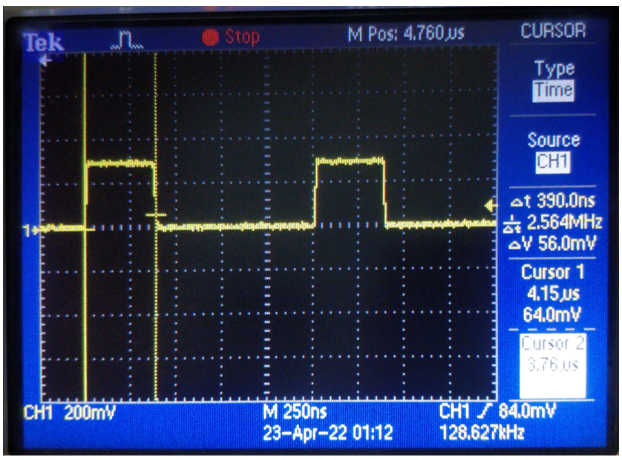

Figure 8: Zero Pulse Timing

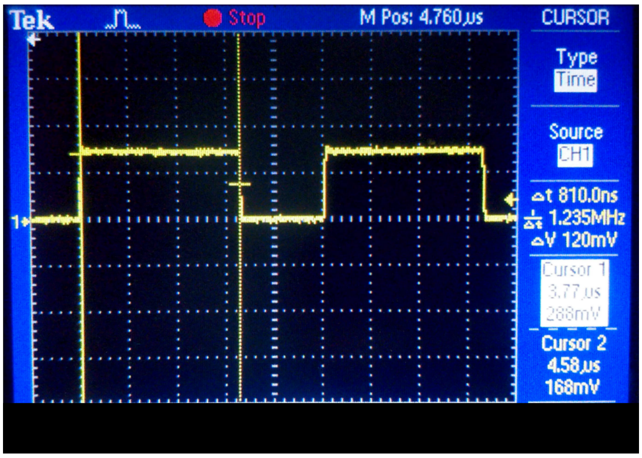

Figure 9: One Pulse Timing

The sample code for generation of this is given below(actual code is in form of macro):

```
Write_0()
{
        GPIO_PORTF_DATA_R= 0x02;
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        GPIO_PORTF_DATA_R= 0x00;
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
}
Write_1()
{
        GPIO_PORTF_DATA_R= 0x02;
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        GPIO_PORTF_DATA_R= 0x00;
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
        __asm   ("   NOP   \n");
}
```

4.2 Waveshare Mic

Waveshare mic provides analog and digital output. There are separate pins for both options.

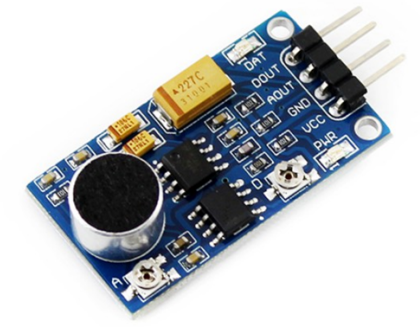

Figure 10: Waveshare Mic

The pin configuration is given below:

VCC ↔ 3.3V ~ 5.3V

GND ↔ power supply ground

AOUT ↔ MCU.IO (analog output)

DOUT ↔ MCU.IO (digital output)

4.3 Tiva TM4C123G Board

The TM4C123G LaunchPad Evaluation Kit is a low-cost evaluation platform for Arm Cortex-M4F based microcontrollers. Featuring a 80-MHz Arm Cortex-M4F CPU, 256kB of flash, and 32kB of SRAM, the TM4C123GH6PM MCU provides integrated USB 2.0 support for USB Host/Device/OTG and two 12-bit ADC modules. The TM4C123GH6PM also includes a multitude of serial communication channels such as UART, SPI, I2C, and CAN. The design of the TM4C123G LaunchPad highlights the TM4C123GH6PM USB 2.0 device interface and additional device features such as the hibernation and PWM modules.

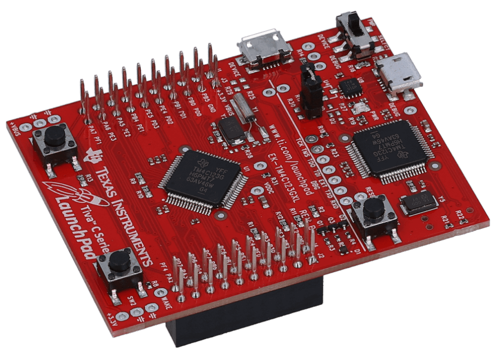

Figure 11: TM4C123G

The final setup is shown in Figure 12.

4.4 Booster-K350QVG Display by Kentec

This booster lcd display works on SPI communication, and is very handy for touch screen applications. There is a library provided by Tivaware called Grlib.

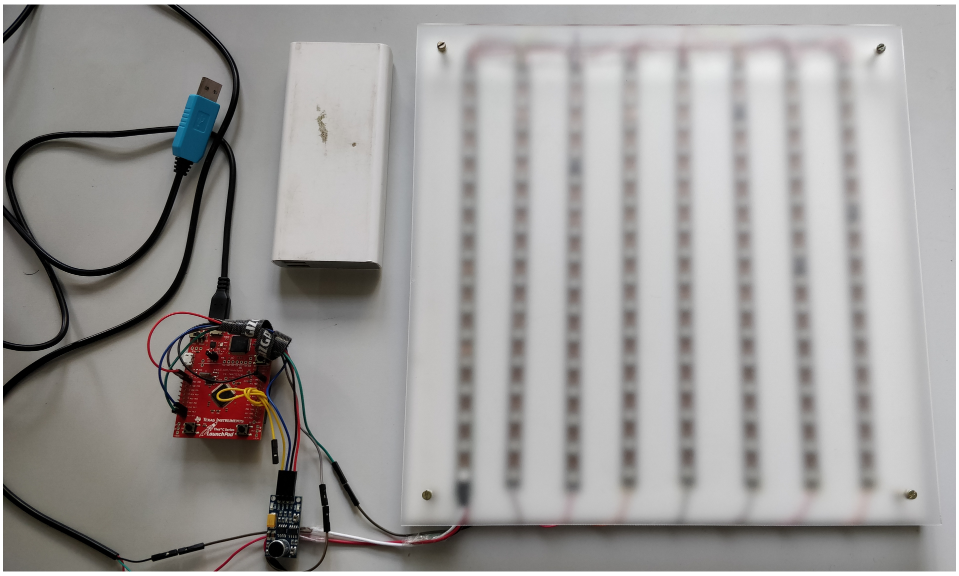

Figure 12: Final Setup

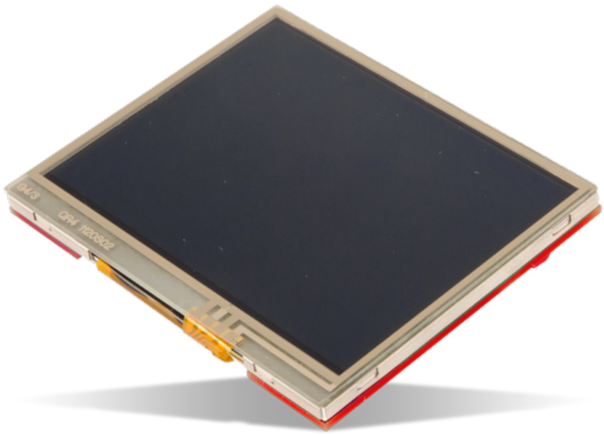

Figure 13: Kentec Display

V FFT CODE

The code for FFT is given below:

```
void fft(complex float *v, int n, complex float *tmp)
{
    if (n >1)
{
        int k,m;
        complex float z, *vo, *ve;
        ve=tmp;
        vo=tmp+n/2;

        for (k=0;k<n/2;k++)
        {
        ve[k] = v[2*k];
        vo[k] = v[2*k+1];
        }
        fft(ve, n/2, v);
```

```
        fft(vo, n/2, v);
        for (m=0;m<n/2;m++)
        {
        z = power(W[Log2n(n)],m)*vo[m];
        v[m] = ve[m] + z;
        v[m +n/2] = ve[m] -z;
        }
        }
        return;
}
```

VI RESULTS

The FFT is algorithm is written for any number of n(multiple of 2). The result of this FFT algorithm is accurate and compared with MATLAB's FFT algorithm as shown in figure 14.

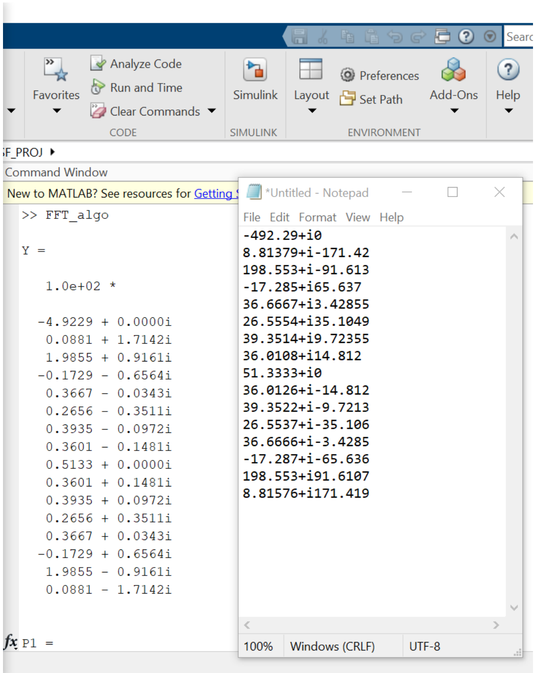

Figure 14: FFT comparision with MATLAB

The output shown on LED matrix is shown in the figure 15.

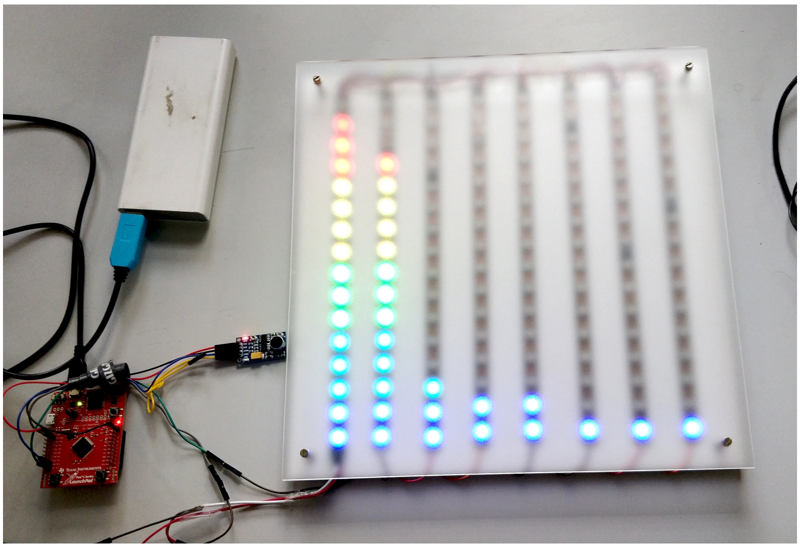

Figure 15: Final Result

VII KENTEC DISPLAY SETUP AND RESULT

For this demo, 100 arrays each having 16 elements is stored in the microcontroller. Then, it is passed to the FFT function and 16 values generated by FFT function is then converted to absolute values. These values are displayed on the display in form of vertical slider with no active touch option. The demo snapshot is shown in figure 16.

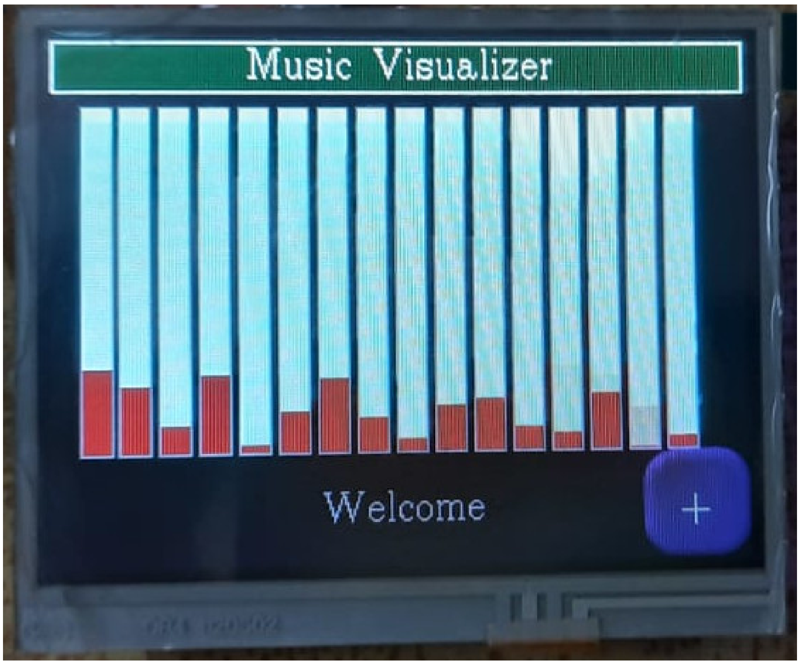

Figure 16: FFT result on Kentec Display

VIII CONCLUSIONS

There is a timing problem in LED Matrix at high sampling rate which can be solved by increasing the operating frequency of the microcontroller. So, with faster microcontroller board, bigger FFT can be done and shown on the LED matrix. Further the LED matrix can be controlled using SPI protocol by controlling its clock frequency which matches 1 and 0 of the neo-pixel timing.