ELE 417
EMBEDDED SYSTEMS
TERM PROJECT REPORT

SPECTRUM ANALYZER

Ayşe İdman - 21728366
Kaan Arı - 21628042

Fall 2020-2021

Introduction

This project is about taking the Fast Fourier Transform (FFT) of the audio signal and printing the magnitude spectra of the signal by using MSP430. Taking FFT of an audio can be beginning of the more developed projects such as guitar tuner, guitar pedal or all the other digital signal processing applications. Note that, this project can be further improved by adding digital to analog converter (DAC). As a result, this is a dynamic and improvable project.

In this project, two MSP430's have been used. One MSP controls the FFT process and prints the magnitude spectra on the OLED screen while the other MSP controls the button so that user can travel along the menu and reset the hardware.

The flow of this work starts with researching about communication between MSP430 and OLED screen via Inter-Integrated Circuit (I^2C) protocol. Then we search about the FFT algorithm in low-power embedded systems. After that, we saw that floating point arithmetic is inefficient, so we try to find another technique. This is part when we came across with the fix point arithmetic FFT. With the help of fix FFT, we implemented spectrum screen.

Tools and Resources:

- CCS
- SSD1306 I2C Library
- Git & GitHub
- Several Fix-Point FFT Implementations (Cooley -Tukey FFT, KissFFT, Fix_fft)

Drawing the Intro:

Since we used an OLED screen, at first we designed a bitmap logo. Our screen is 128x32 pixels but it has 128x64 bit GRAM. By using this feature, we were able to create sliding effect on the screen. Also this extra memory location (128x32 bit) can be used as a screen buffer. We can switch between the screens by only changing the starting offset value.

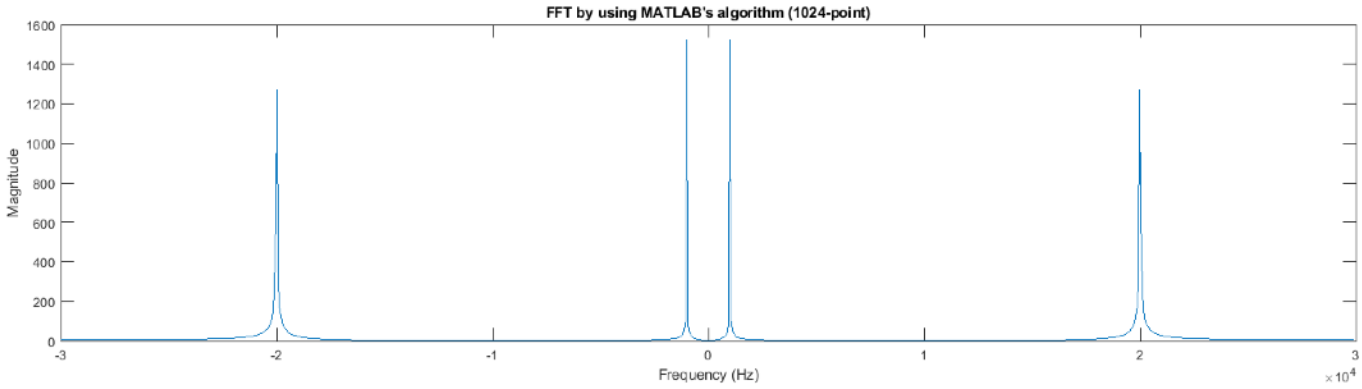
Drawing the Menu:

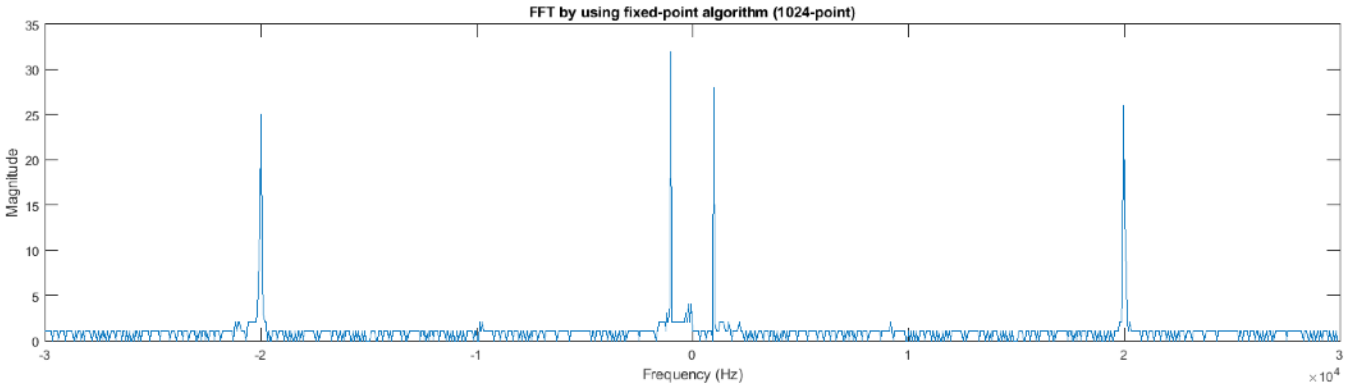
As we have mentioned before, extra screen buffer is useful when user travels along the interface. Different from the drawing the intro, this time we used the buffer for creating multiple pages too. We also add bouncing effect when there is no page to slide further. Other approaches are similar to drawing the intro.

We use 6x8 and 12x16 pixels ASCII fonts for OLED screen. However, there is a memory absence issue for 12x16 pixels font. Because it stores all of the ASCII values hence requires 4 KB of memory space. In order to solve this problem, we decided to add only the necessary ASCII characters manually.

Implementing the fixed point FFT:

First of all, we would like to start with the differences between floating point arithmetic and fixed-point arithmetic. For roughly speaking, fixed point arithmetic is integer mapped version of floating point. All data are fixed-point signed char, in which -128 to +128 represent -1.0 to +1.0 respectively. Fixed point arithmetic is used for speed, instead of the more natural floating-point.



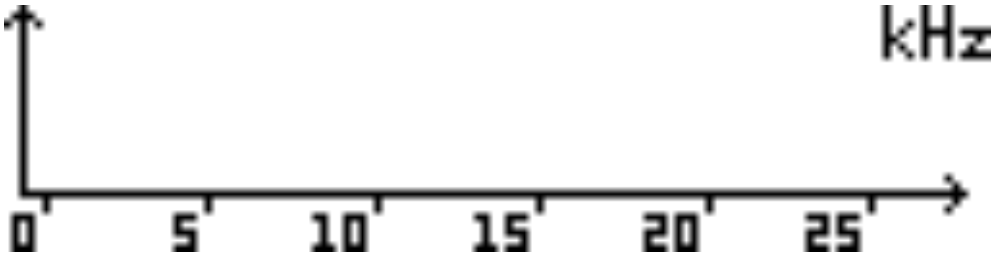


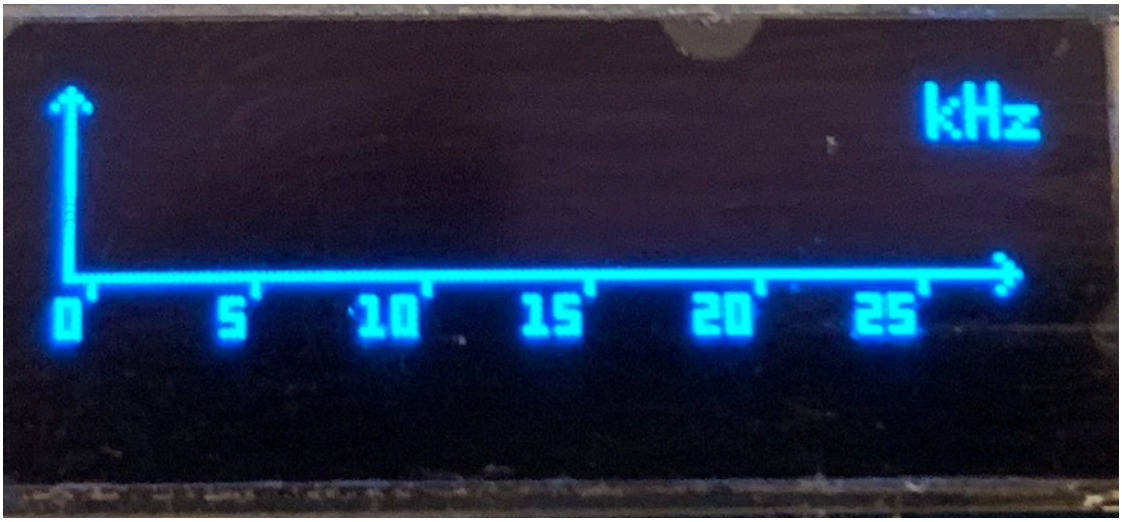
In the beginning, we implemented 1024-point FFT. This is the highest value that we can get without any memory overload. However, drawing the result requires mapping from 512 (0 to $f_{sampling}/2$) to 128. In order to get more simple mapping, we decided to implement 256-point FFT too. Also since the low pass filter at the input (will be discussed later) has cutoff frequency at 25 kHz, there is no need to draw the frequencies larger than 25 kHz (corresponds to first 107 point of result).

With the help of fixed point FFT, we achieved 600-700 ms for calculating 1024-point magnitude spectrum. Timing delay decreased even further when we perform 256-point magnitude spectrum calculation (almost 20ms).

Drawing the Magnitude Spectrum:

At first, we draw the x and y axis of the magnitude spectrum on the OLED screen. We created bitmap image by using graphical image editors.



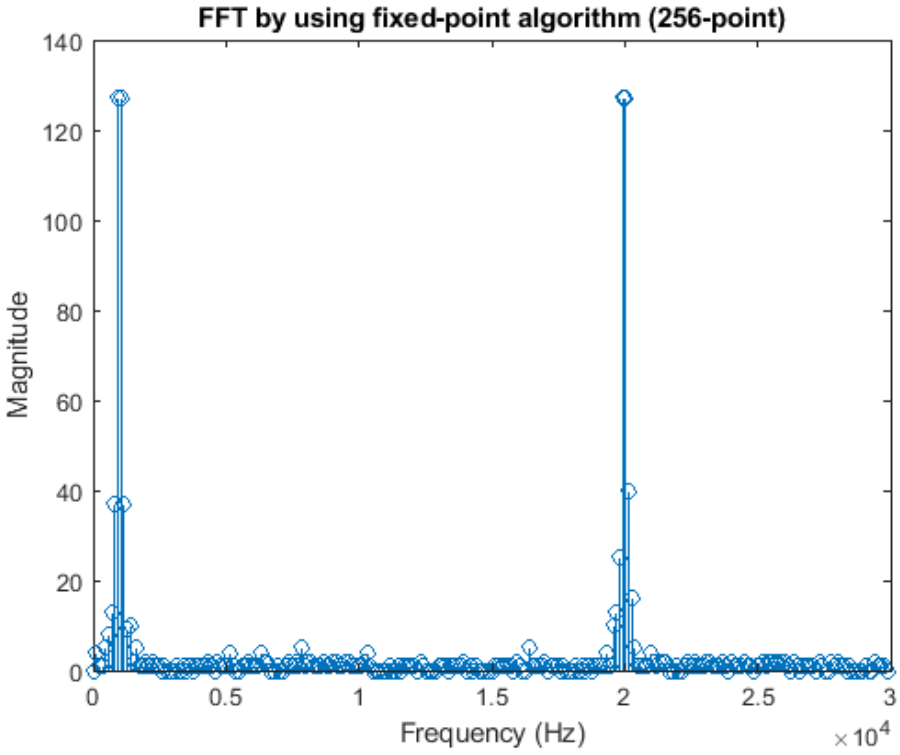


We choose to draw maximum magnitude as 20 pixels. This requires mapping unsigned char (8 bit) to 0-20. To achieve that, we divide the numbers by 8 by shifting it 3 times. Then we add different values according to the interval of the number take place, such as:

1) Add 9 if 24<number<31
2) Add 7 if 16<number<23
3) Add 5 if 8<number<15
4) Add 3 if otherwise

Then divide all the resulting values by 2. In the end they are mapped to 0-20.

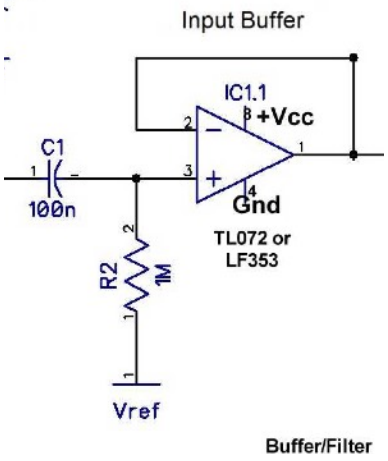
Word length of the I2C communication is one byte minimum, hence we were not able to draw a single pixel at once. So we try to draw every frequency component by sending total of 3 bytes (24 bits). Even though this works, we were not pretty satisfied with the result because the drawing process took more time than we expected. In order to handle this problem, we used block transfer method. In block transfer method, all the frequency component impulses are stored in a 3x107 byte array. Then all of them transferred to OLED screen at once. This improvement reduces the accessing time and pending time of the communication protocol. As a result, we got 30+ fps refreshing on the OLED screen.



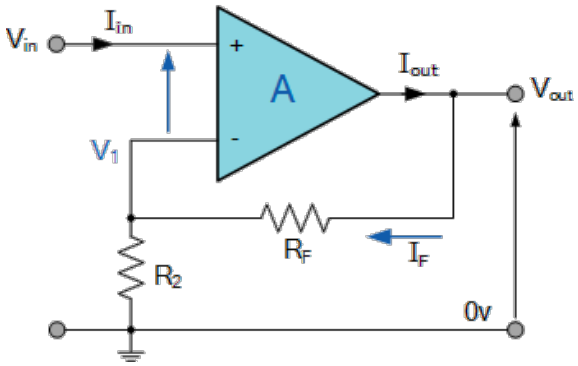
Sampling (ADC10 Module):

We used ADC10 module existing in our MSP430 Launchpad to sample input signals. We tried to set sampling frequency to 60 kHz but we realized that the clock speed is not enough to process such speed. Therefore we pushed forward to our limits and set clock to 16 Mhz. After proper settings for clock, we achieved desired results (f_s = 60+ kHZ). We perform 8 bit quantization. From Nyquist rate, we can measure up to 30kHz without aliasing.

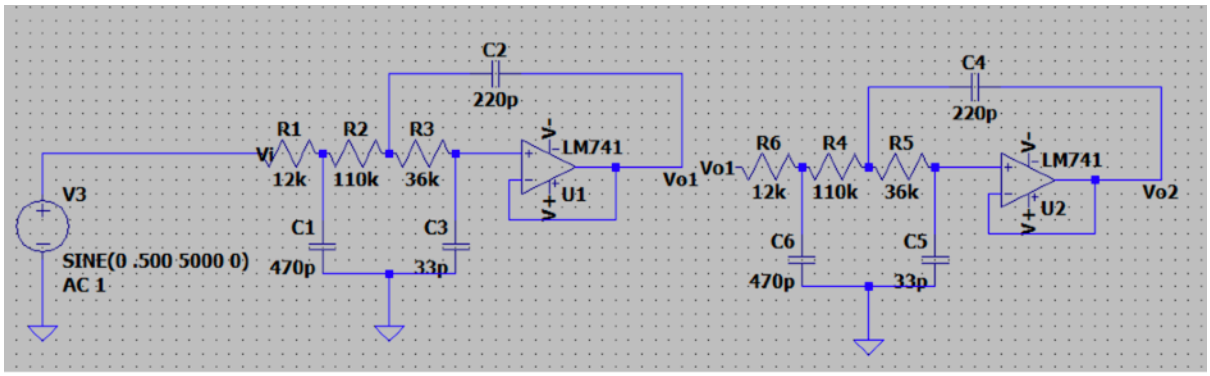
To prevent aliasing, we had to design a low pass filter (LPF). We designed input buffer , 2 stage 3rd order (each stage) Sallen-Key Low-pass Filter and a gain amplifier.



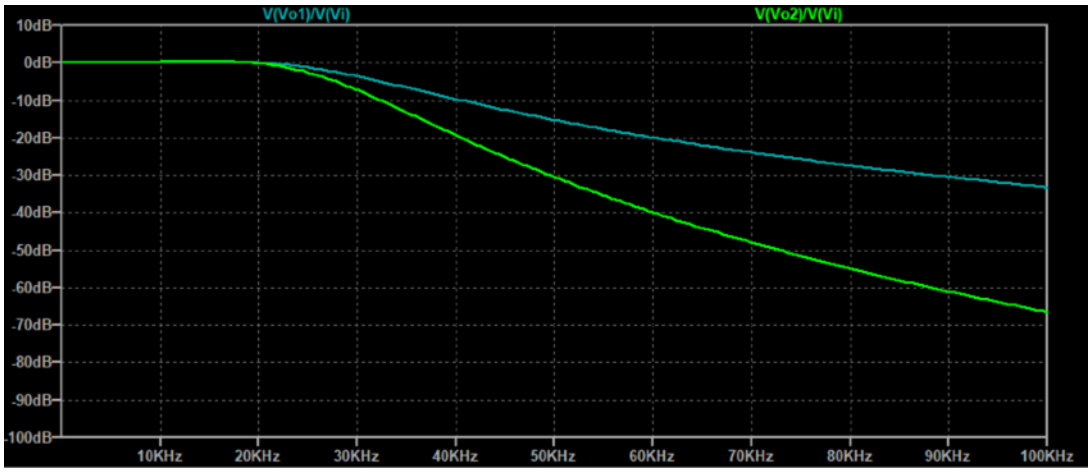
Buffer/Filter



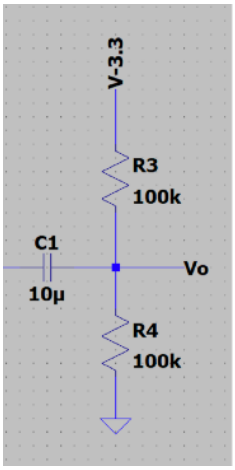
Amplifier



2 stage 3rd order Sallen-Key Low-pass Filter



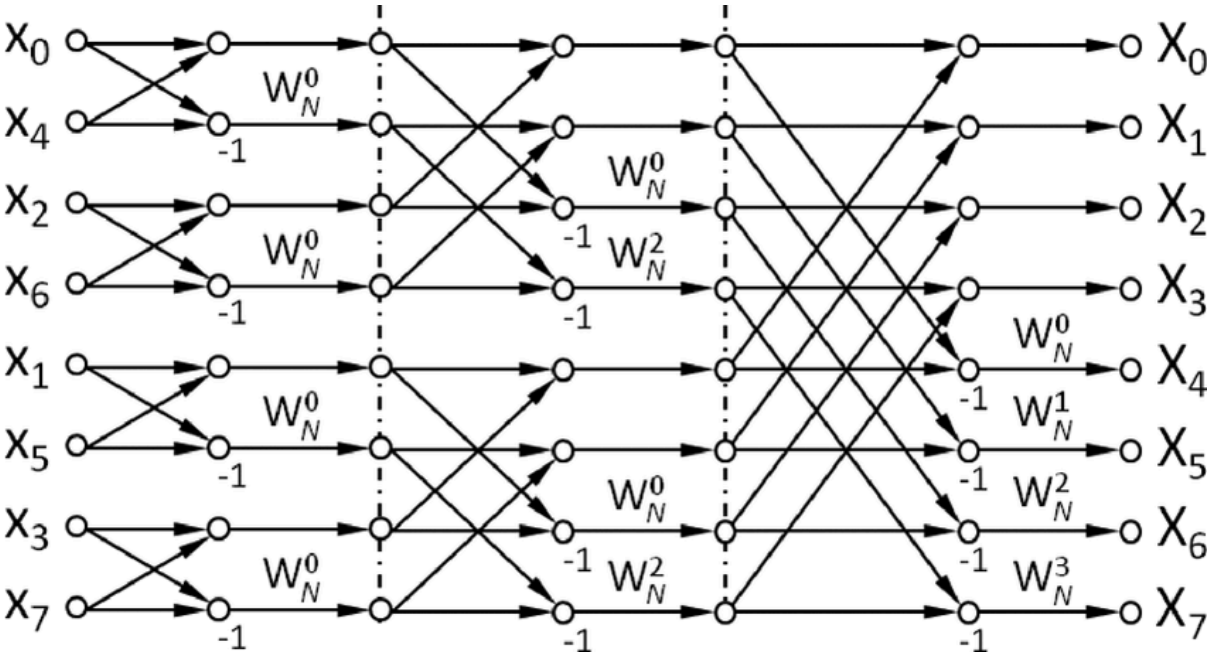
Low Pass Filter Magnitude Response



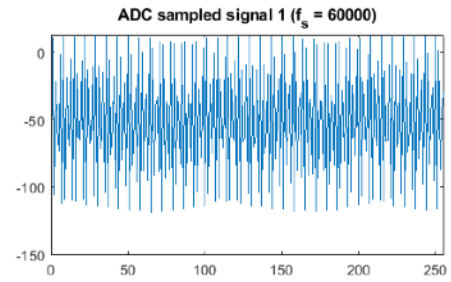
DC offset for ADC

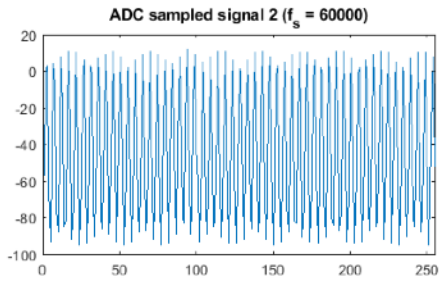
Time Decimation:

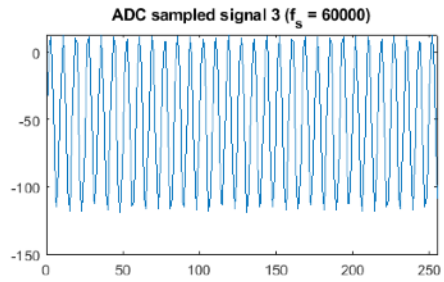
To compute the FFT, we have to rearrange the sampled data. There are two approach for rearranging: time decimation or frequency decimation. We used time decimation because it is more suitable for in-place decimation when we sample a new data. We did not want to spend extra computational load by rearranging N-point array. Therefore we put new data directly to the corresponding decimation index. Also we benefit from circularity property of FFT. We used the memory like a circular buffer such that we calculate the modulus of the current sample index then the sampled values are properly placed to corresponding memory locations.

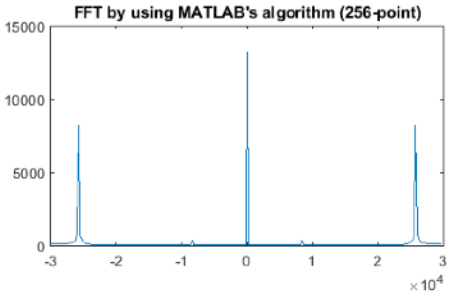


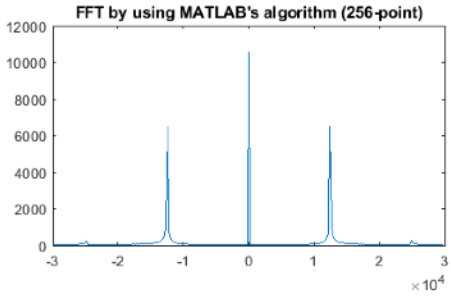
Results of ADC sampling and fixed point FFT:

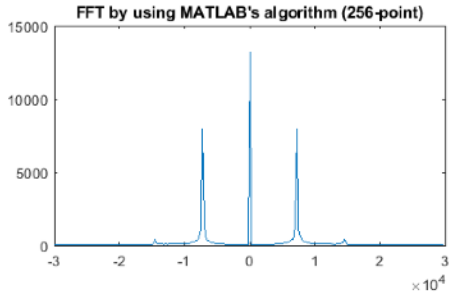


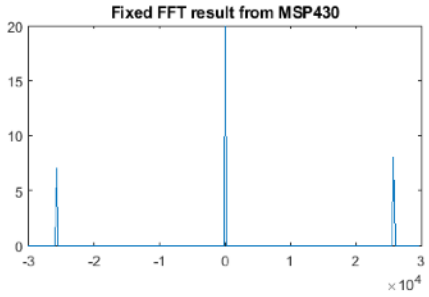


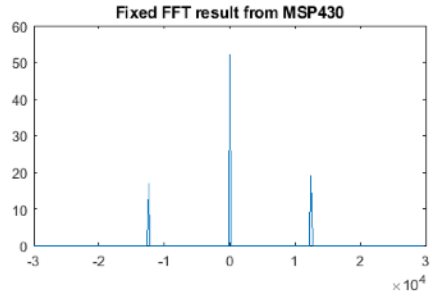


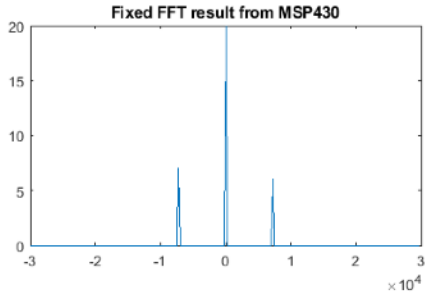








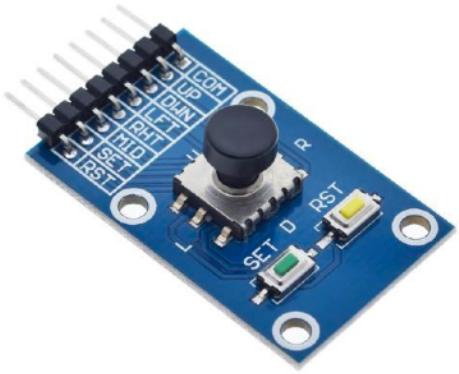




User Inputs (Button Control):

We used 5 axis joystick module for user inputs. With this module, user can switch between Tuner and Spectrum.

We used second MSP430 (slave) as a controller of module. Master MSP430 initiates the communication process and slave MSP430 transmits the input taken from module.



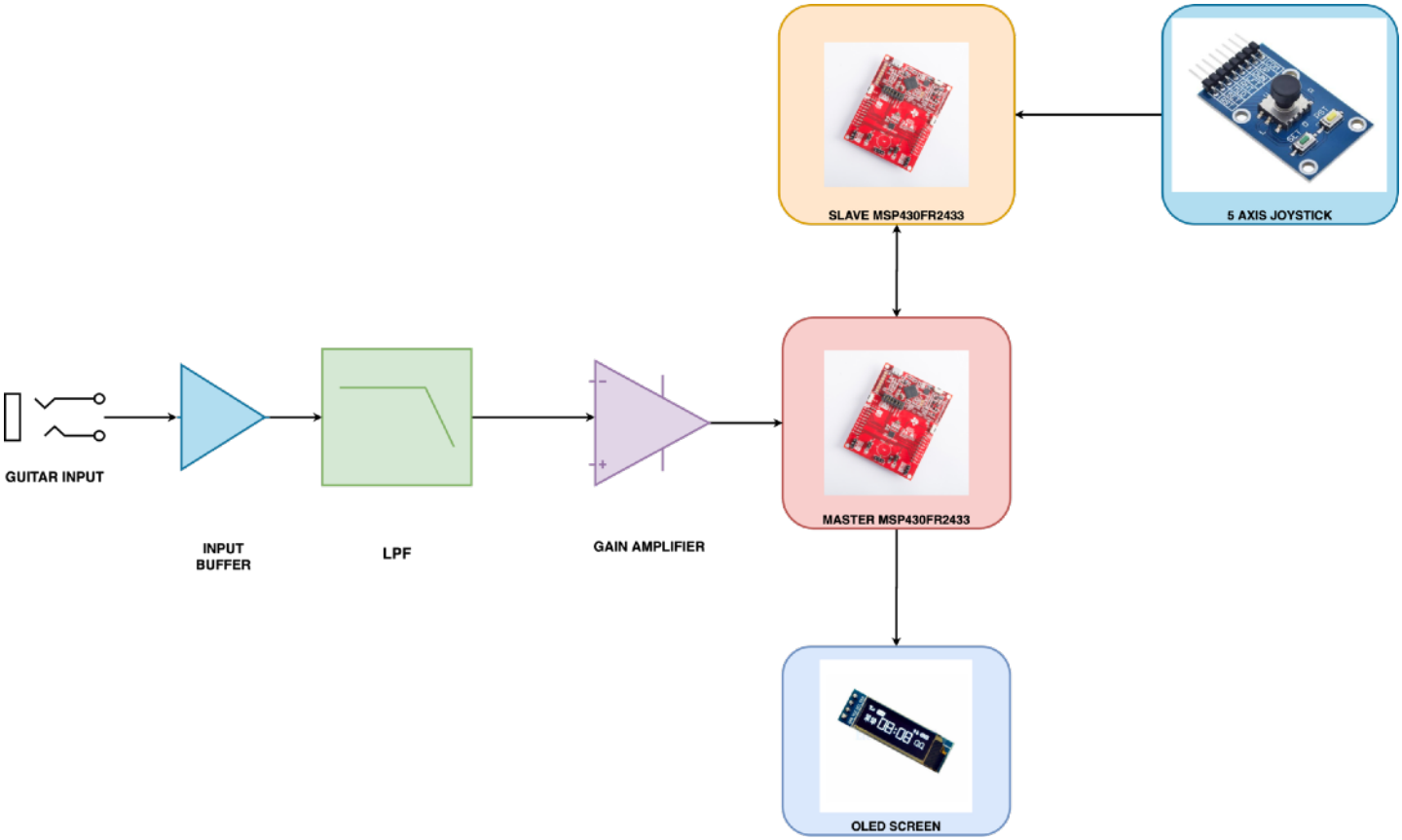
Also we used RST button on module to reset master MSP430 by using Watchdog Timer password violation.

Conclusion:

We have learned a lot of fundamental topics in embedded systems like I2C communication, low level programming, ADC usage, interrupts and timers. Also we wanted to enhance our knowledge about Digital Signal Processing course. Therefore we wanted to find a project that covers both Embedded Systems and Digital Signal Processing fundamentals. In addition playing guitar is our common interest.

On the other hand we experienced multi disciplinary work. Even though it was challenging in the beginning, observing real life results from the theoretical knowledge got us excited.

This project is really open to improvement, considering that the frequency is the base of the universe. For example, making a guitar tuner or a guitar pedal can be next steps of this project.



	Quantity	Price
MSP430FR2433	2	272.58 ₺
128x32 OLED	1	21.95 ₺
5 Axis Joystick Module	1	13.10 ₺
Electronic Components (Opamp etc)	-	20.27 ₺
	Total Price	**327.90 ₺**

Distribution of the Task:

Ayşe İdman: DSP Research, ADC configs, screen operations, FFT Implementation.

Kaan Arı: Button Control, screen operations, I2C communication.

Project Video:

https://www.youtube.com/watch?v=P-Latt5orT4&feature=youtu.be

Project Source Codes:

We uploaded the source codes to GitHub. There is a link for source codes:

https://github.com/kaanaritr/ELE417_Term_Project-MSP430-Spectrum-Analyzer

Input Filter Stage: