Sebastien Cayo
Physics 201: Analog and Digital Electronics
Professor John Ruhl
Final Project
December 16, 2019

The goal of the circuit is to create a device that can read, process, and then display the numerical value of the sampled frequency to an LCD display. Using a Cylewet Cylindrical Electret Condenser Microphone Pickup as the main component in the frequency sampling process, it is placed in an operational amplifier circuit, using an OPA344 Op-Amp, see **Figure 2** for Op-Amp pinout, to amplify the received signal. The negative voltage supply of the Op-Amp is railed to ground, while the positive voltage supply is railed to +5 volts. Set up the circuit as shown in **Figure 1**. The output of the Op-Amp is plugged into the A0 Analog input of the Arduino Uno, so that the arduino can receive the sampled frequency for the program to run and execute. And using a 7 Segment Display Serial Connection MAX7219 as the display for the entire circuit system, connect the VCC pin hole to the +5 volt rail, the GND pin hole to the ground rail, the DIN pin hole to Pin 13 of the arduino, the CS/LOAD pin hole to pin 11, and the CLK pin hole to pin 9. See **Figure 3** for a better visual.

To run the program, both a Fourier Transformer library and an LED Control library are needed. To download it, in the arduino program, go to Tools in the upper left corner, then Library Manager. In the Library Manager, search ‘fourier’; the library used in this code is from “arduinoFFT” by Enrique Condes Version 1.5.0. Then search for ’ledcontrol’; the one used for this circuit is “LedControl” Eberhard Fahle Version 1.0.6. The code for the fourier transformer comes from an example code, under FILE>EXAMPLES>arduinoFFT>EXAMPLES>FFT_3, then adjusting the samplingFrequency and samples to 950 and 128, respectively, to be able to read frequencies between 80-450 Hz with a margin of error of 1 to 3 when reaching either the low or high end of the range. **Figures 4**, **5**, **6** show the sampled frequency, being inputted into the system and run by the code versus the displayed frequency, from a lower samplingFrequency to

the samplingFrequency of the code that is provided in **Appendix 1** to a higher samplingFrequency respectively.

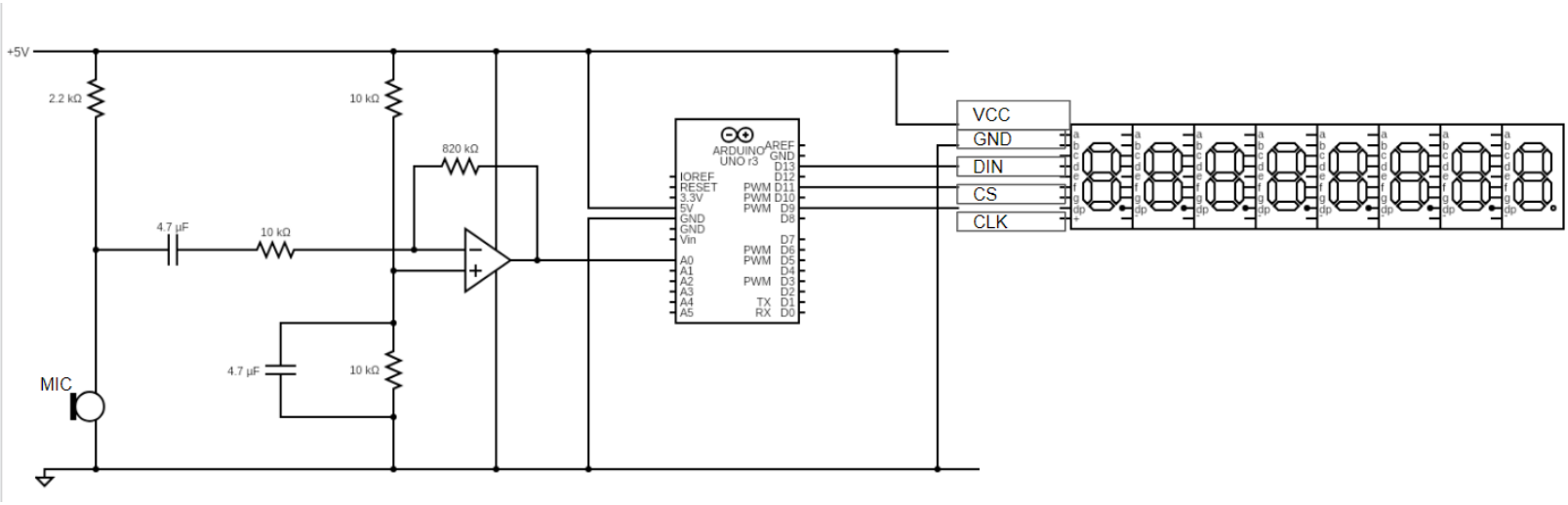

Figure 1: Circuit diagram

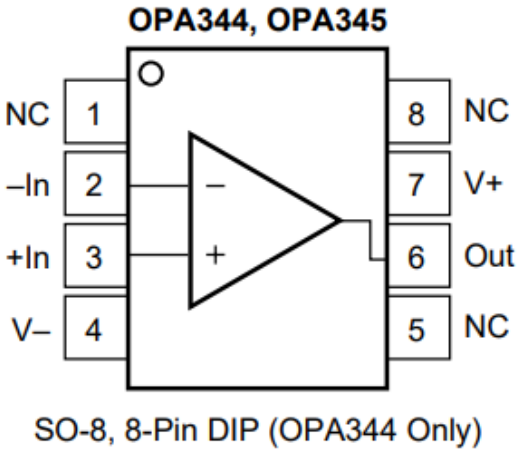

Figure 2: OPA344 Op-Amp Pinout Diagram

Figure 3: 7 Segment Display Serial Connection MAX7219

Making the code for the LCD Display was the most difficult part of the process of making this frequency reader. By checking under TOOLS>Serial Monitor the accuracy can be tested while connecting an Arbitrary Wave Function Generator directly into Analog Input pin A0, and sending in frequencies to the arduino to process. The fourier transform code reads the frequency pretty well, but as stated before, the closer the frequency is set to the outer bounds of its range, the less accurate it becomes. In the Serial Monitor the frequency that shows up is slightly off by a couple decimals. And to more accurately read higher frequencies, simply increase the ‘samplingFrequency’ part of the code. The problem lied with programing the LCD display. Even after figuring out how to get the LCD display to show the frequency that the arduino was reading, it is unable to display the decimal significant figures. But seeing as how the frequency the arduino reads is off anyway, it was not unreasonable to leave out those significant figures.

samplingFrequency = 128	
Input Frequency (Hz)	Displayed Frequency (Hz)
1	too inaccurate and varying
5	too inaccurate and varying
15	too inaccurate and varying
17	17
20	20
25	25
30	30
35	35
40	40
45	45
50	50
55	55
60	60
61	61
62	62
63	63
64	64
65	63
66	62
67	61
70	58
80	48

90	38
100	28

Figure 4: low samplingFrequency

samplingFrequency = 950	
Input Frequency (Hz)	Displayed Frequency (Hz)
1	60
5	60
10	60
15	59
20	20 but inconsistent (jumping from 20 to 60)
25	24 but inconsistent (jumping from 24 to 60)
30	30 but inconsistent (jumping from 30 to 60)
50	50s
60	60
70	65 through 71
80	81
90	90
100	100
120	120
140	141
150	150
200	201
250	252

300	302
350	352
400	403
440	443
450	454
500	453

Figure 5: samplingFrequency of code in Appendix 1*

samplingFrequency = 4250	
Input Frequency (Hz)	Displayed Frequency (Hz)
15	too inaccurate and varying
20	too inaccurate and varying
100	104
120	118
150	150
200	201
250	252
300	302
400	401
500	502
600	603
700	703
800	804
900	905

1000	1005
1100	1106
1200	1206
1300	1307
1500	1508
1700	1711
1900	1910
2000	2010
2100	2126
2200	2071

Figure 6: high samplingFrequency

The goal of this project was to have a circuit that would detect and send frequency to an arduino, make a code to read that analog signal, calculate it then monitor it as its numerical frequency, and then have that code send it to an LCD display. The project was successful, but it could use a bit more work to be more accurate. If this project had to be done again, it would be desired that the system as a whole be more compact.

Appendix 1

```
#include "arduinoFFT.h"
#include "LedControl.h"

String inString = "";
int pos = 0;
int i;

// Arduino Pin 13 to DIN, 9 to Clk, 11 to LOAD, no.of devices is 1
LedControl lc = LedControl(13,9,11,1);
arduinoFFT FFT = arduinoFFT(); /* Create FFT object */

#define CHANNEL A0
const uint16_t samples = 128; //This value MUST ALWAYS be a power of 2
const double samplingFrequency = 950; //Hz, must be less than 10000 due to ADC
unsigned int sampling_period_us;
unsigned long microseconds;

/*
These are the input and output vectors
Input vectors receive computed results from FFT
*/

double vReal[samples];
double vImag[samples];

#define SCL_INDEX 0x00
#define SCL_TIME 0x01
#define SCL_FREQUENCY 0x02
#define SCL_PLOT 0x03
```

```
void setup()
{
	sampling_period_us = round(1000000*(1.0/samplingFrequency));
	Serial.begin(115200);
	// Initialize the MAX7219 device
	lc.shutdown(0, false); // Enable display
	lc.setIntensity(0, 3); // Set brightness level (0 is min, 15 is max)
	lc.clearDisplay(0); // Clear display register
	for (int i = 0; i < 8; i++) {
		lc.setDigit(0, i, i + 1, false);
	}
}

void loop()
{
	microseconds = micros();
	for(int i=0; i<samples; i++)
	{
		vReal[i] = analogRead(CHANNEL);
		vImag[i] = 0;
		while(micros() - microseconds < sampling_period_us){
			//empty loop
		}
		microseconds += sampling_period_us;
	}

	FFT.Windowing(vReal, samples, FFT_WIN_TYP_HAMMING, FFT_FORWARD);  /*
	Weigh data */
	FFT.Compute(vReal, vImag, samples, FFT_FORWARD); /* Compute FFT */
	FFT.ComplexToMagnitude(vReal, vImag, samples); /* Compute magnitudes */
```

```
        double x = FFT.MajorPeak(vReal, samples, samplingFrequency);
        Serial.println(x, 6); //Print out what frequency is the most dominant.

                float y = x ;
                int divvy = 1;
                for (i = 1; i < 9; i++)
                {
                        int num = int(y / divvy) % 10;
                        lc.setDigit(0, i-1, num, false);
                        Serial.print(i-1);
                        Serial.print(" : ");
                        Serial.println(num);
                        divvy *= 10;
                }
        Serial.println("---");
        delay(1000); /* Repeat after delay */
}
```