ECE 341 ACCELERATED PROJECT

Project Artifacts

Authors: Shelby Westerberg, Trevor Horine, Bryson Goto

December 3, 2020

Instructor: Matthew Shuman

Grading TA: Shane Witsell

Contents

1 Simulation

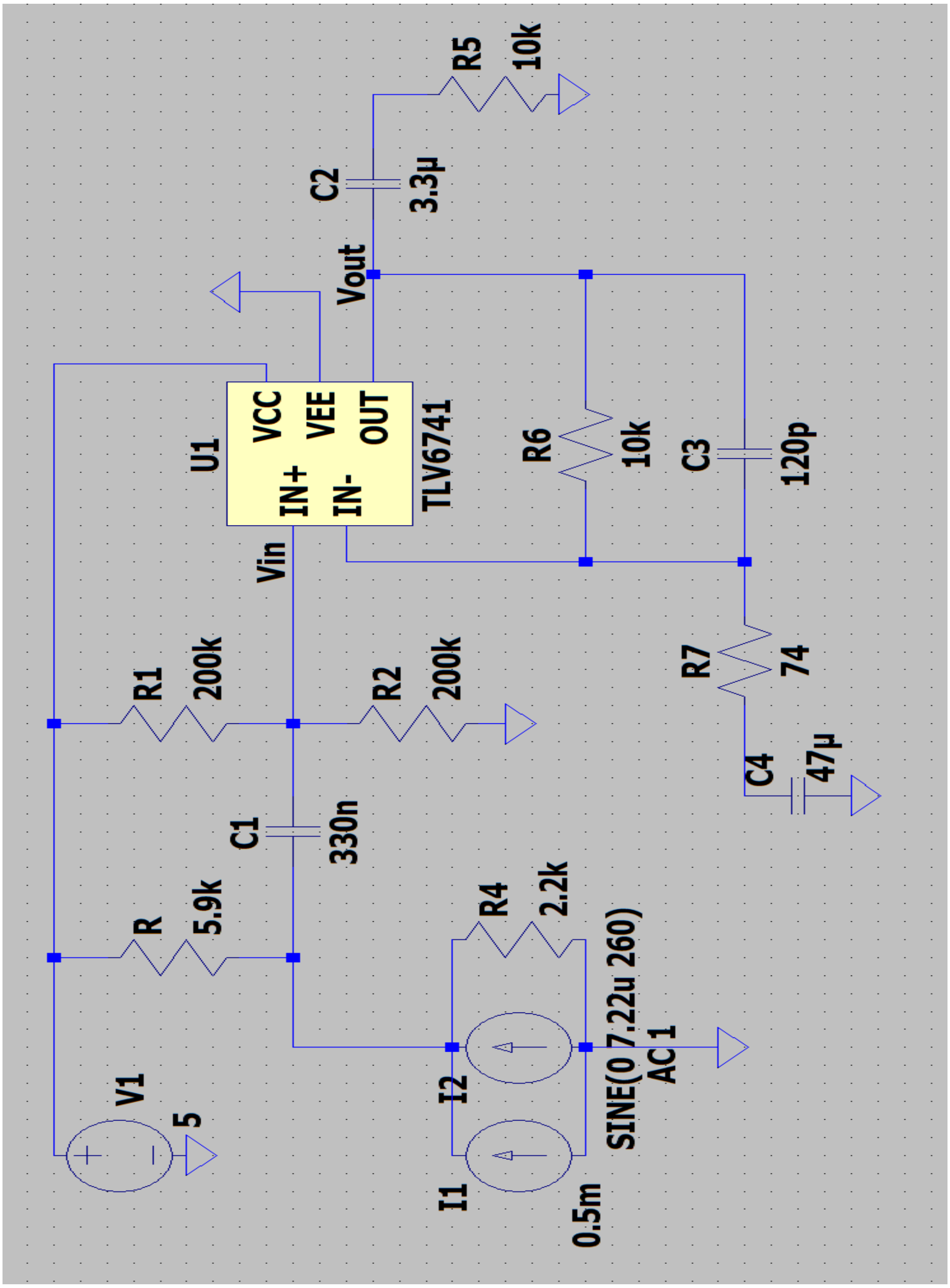

Figure 1: LTSpice Circuit Simulation

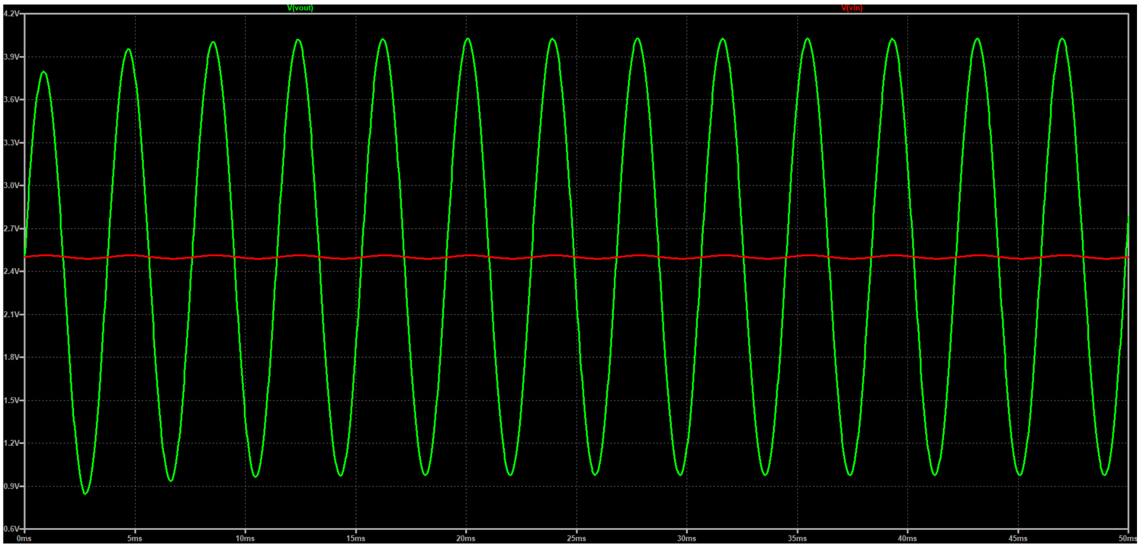

Figure 2: LTSpice Circuit Simulation Voltage Graph

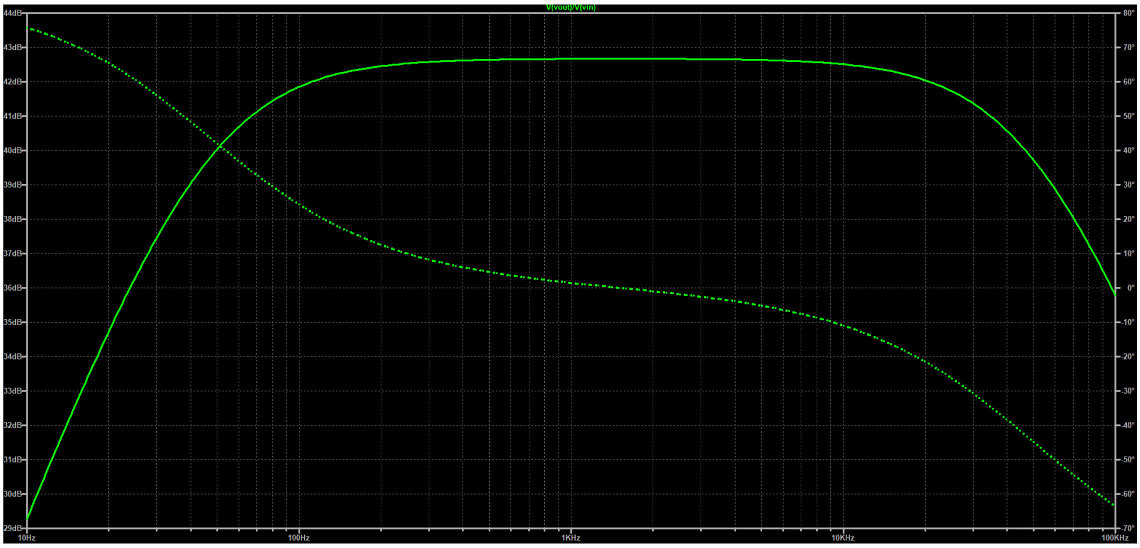

Figure 3: LTSpice Circuit Simulation Gain Graph

2 Schematic

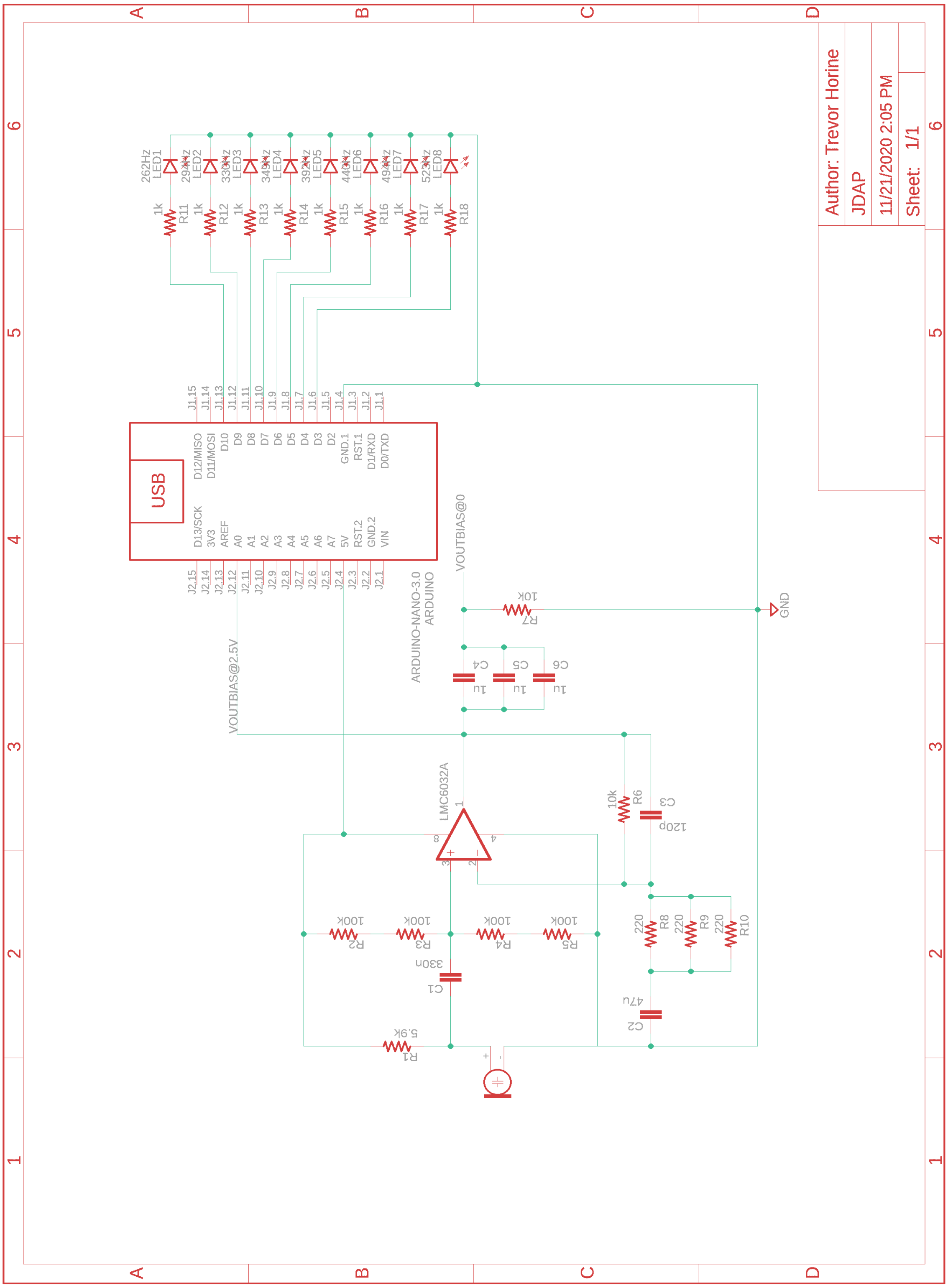

Figure 4: Detailed schematic of audio detection circuit.

3 Block Diagram

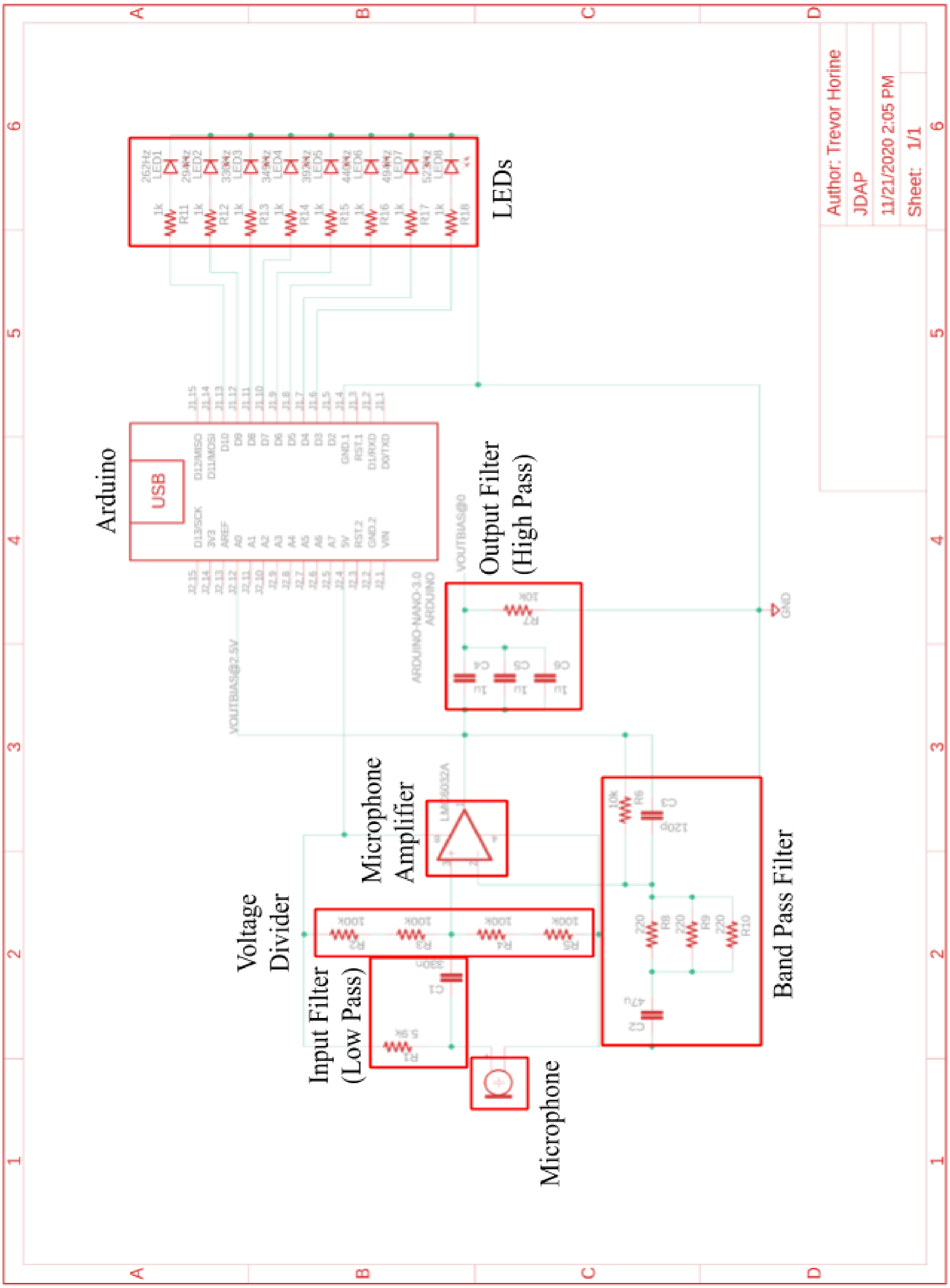

Figure 5: Block Diagram of Audio Detection Circuit

4 PCB Layout

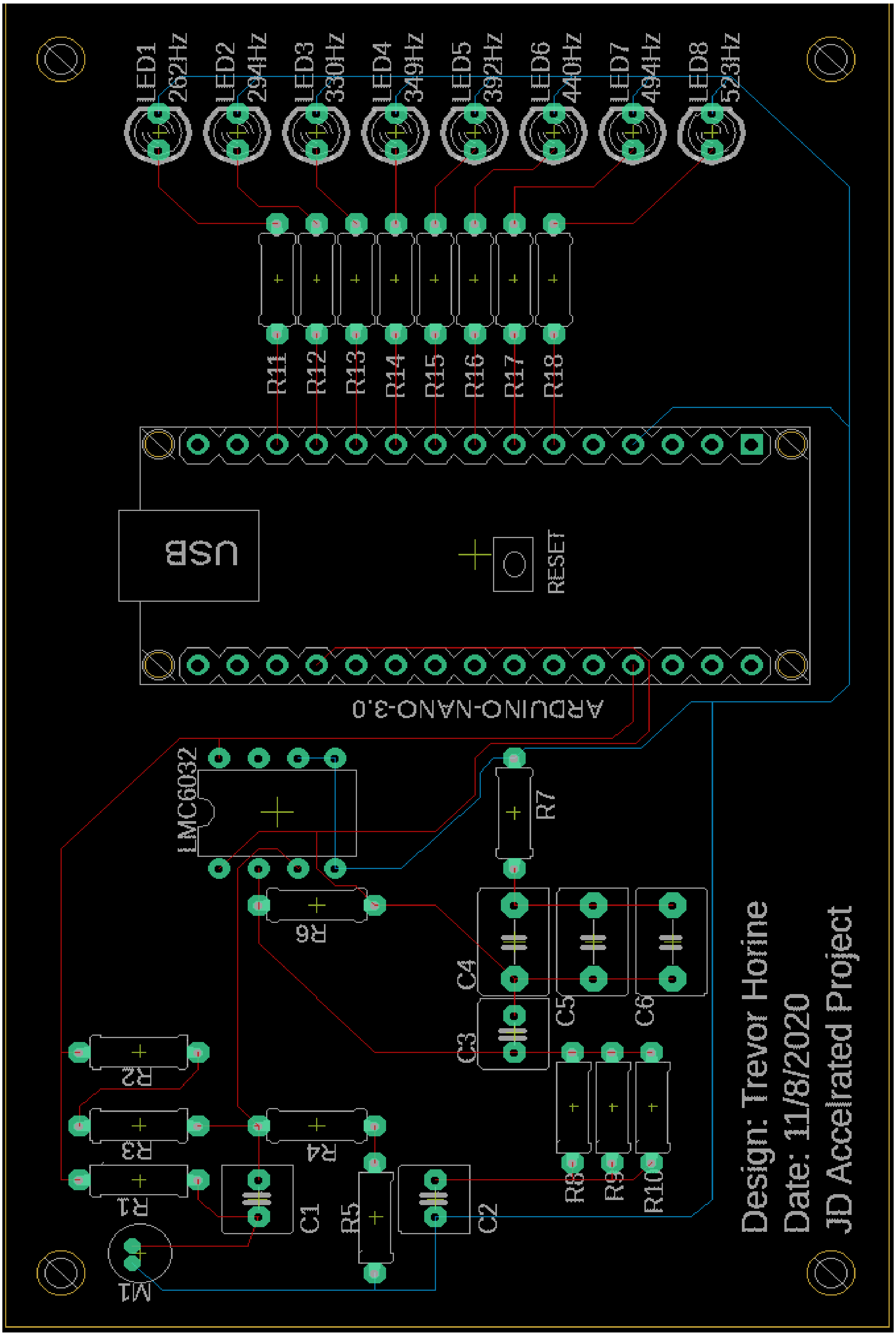

Figure 6: Possible layout of a PCB we designed but did not get printed.

5 Built Circuit

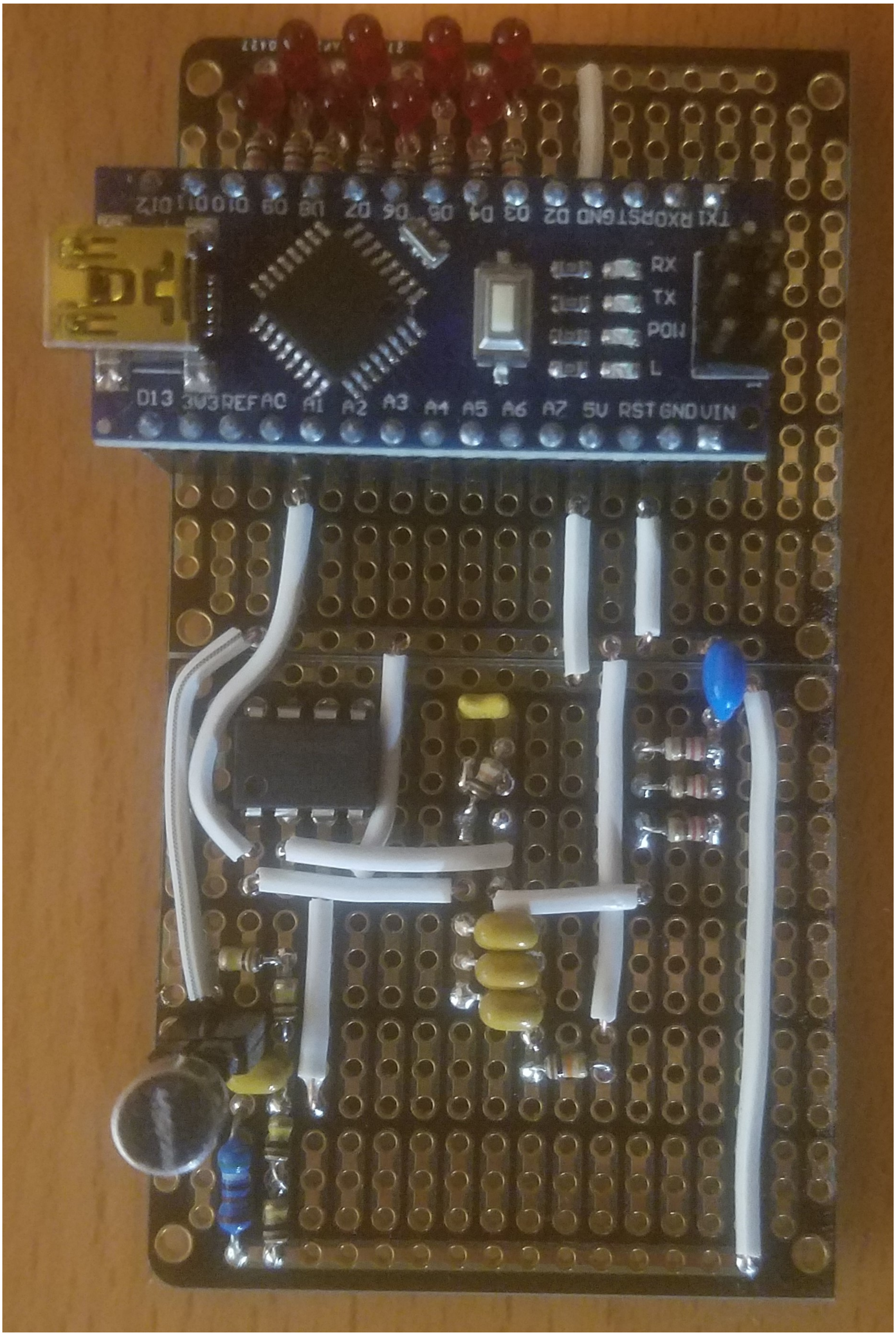

Figure 7: Picture of our project built.

6 Bill of Materials

Part	Value	Package	Description
ARDUINO	ARDUINO-NANO-3.0	ARDUINO-NANO-3.0	Arduino Nano 3.0
C1	330n	Through Hole	CAPACITOR
C2	47u	Through Hole	CAPACITOR
C3	120p	Through Hole	CAPACITOR
C4	1u	Through Hole	CAPACITOR
C5	1u	Through Hole	CAPACITOR
C6	1u	Through Hole	CAPACITOR
LED1		LED3MM	LED
LED2		LED3MM	LED
LED3		LED3MM	LED
LED4		LED3MM	LED
LED5		LED3MM	LED
LED6		LED3MM	LED
LED7		LED3MM	LED
LED8		LED3MM	LED
LMC6032		DIL08	OP AMP
M1	ELECTRET_MICROPHONE	CMC-5042PF-AC	Electret Condenser Microphone
R1	5.9k	Through Hole	RESISTOR
R2	100k	Through Hole	RESISTOR
R3	100k	Through Hole	RESISTOR
R4	100k	Through Hole	RESISTOR
R5	100k	Through Hole	RESISTOR
R6	10k	Through Hole	RESISTOR
R7	10k	Through Hole	RESISTOR
R8	220	Through Hole	RESISTOR
R9	220	Through Hole	RESISTOR
R10	220	Through Hole	RESISTOR
R11	1k	Through Hole	RESISTOR
R12	1k	Through Hole	RESISTOR
R13	1k	Through Hole	RESISTOR
R14	1k	Through Hole	RESISTOR
R15	1k	Through Hole	RESISTOR
R16	1k	Through Hole	RESISTOR
R17	1k	Through Hole	RESISTOR
R18	1k	Through Hole	RESISTOR

7 Time Report

This time report is based on when the report was finished and does not include the hours put in to record the project video submission.

7.1 Hours Spent per Week

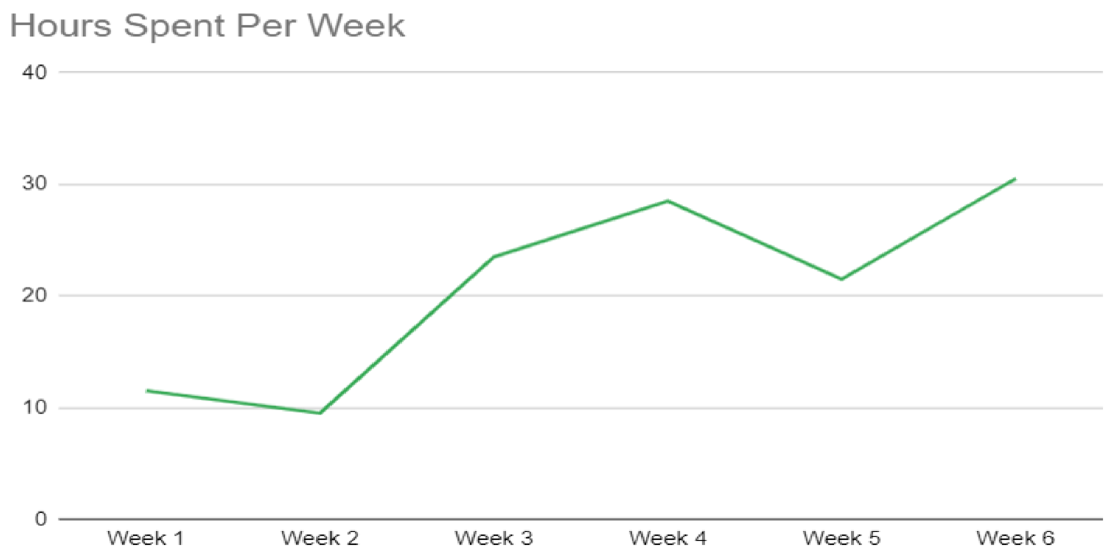

Figure 8: Hours spent by the team each week on the project.

7.2 What We Worked on by Percentage

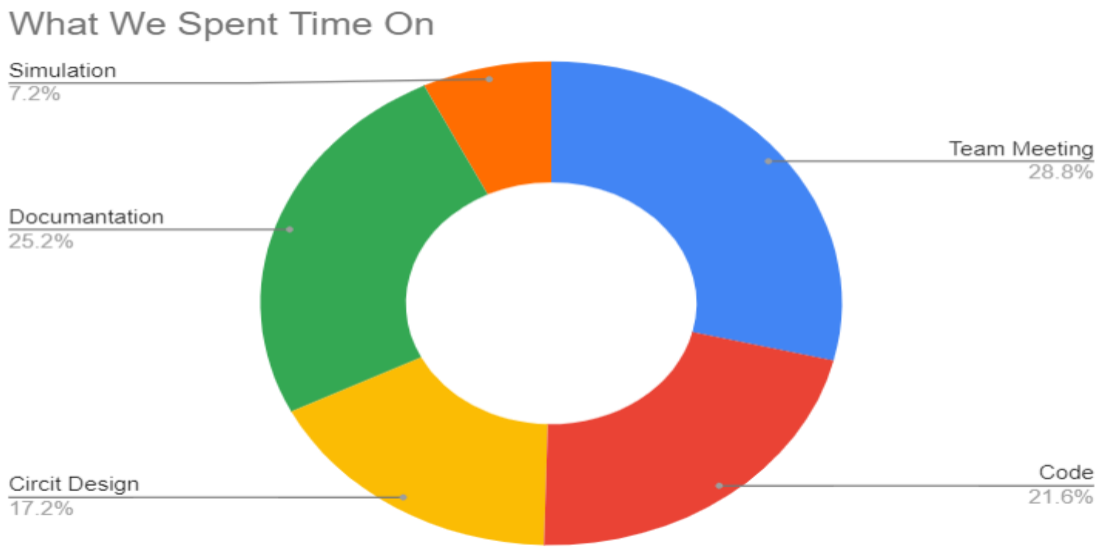

Figure 9: What the team worked on by percentage.

7.3 Hours Spent per Person

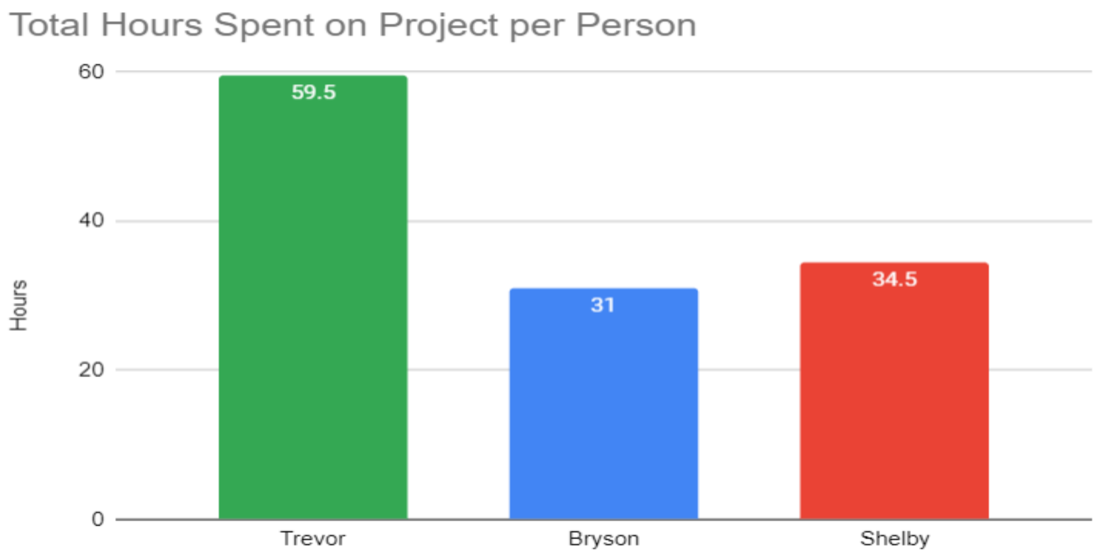

Figure 10: Breakdown of how may hours each team member put in to the project.

7.4 Time sheet

Name	Week	Time spent (Hours rounded to nearist .5)	What did you work on
Trevor	The Beginning of Time	0.5	Set up time sheet
All	10/22/2020	3	Team meeting, Project planning
Trevor	10/22-28	6	Circuit calculations, circuit building, LTSpice, Matlab, Arduino
Shelby	10/22-28	1	Review datasheet, circuit calculations, LTSpice
Bryson	10/22-28	1	Circuit calculations
All	10/29/2020	3	Team meeting, Simulation, circuit calculations
Trevor	11/3	2.5	simulation, circuit calculations, data gathering
Shelby	11/3/2020	1.5	Report introduction and frequency calculations
Bryson	11/3/2020	2.5	Calculations, circuit simulations
Trevor	11/5	6	Questions in lab, debugging/matlab/arduino code
All	11/5/2020	3	Group meeting: simulation and debugging
Trevor	11/6	4	debug and testing, some code "magic"
Bryson	11/8	2	Assembling circuit
Trevor	11/8	2	code commenting and schematics
Bryson	11/9	1.5	Documentation, LTSpice simulations + screenshots
Shelby	11/9	1	Documentation
Trevor	11/11	4	codeing (MATLAB and Arduino) (regerster programming)
Bryson	11/12	3	Group meeting, troubleshooting frequency response
Trevor	11/12	3	Group meeting, troubleshooting frequency response
Shelby	11/12	0.5	Group meeting
Shelby	11/13	2	Circuit assembly, Matlab code
Shelby	11/14	1	Matlab code
All	11/14	12	Group meeting, troubleshoting
Trevor	11/14	0.5	Time sheet formulas
Trevor	11/16	0.5	Troubleshooting, office hour prep
All	11/17	6	Office hours, troubleshooting
Trevor	11/19	4	Coding and troubleshooting
Bryson	11/19	2	Circuit building, video script writing
Shelby	11/19	2	Coding and troubleshooting
Shelby	11/20	0.5	Shielding design
Trevor	11/20	1	Circuit soldering
Shelby	11/21	1	Circuit troubleshooting
All	11/21	7.5	Complete circuit build, documentation
Trevor	11/21	2.5	Circuit Soldering, and testing
Shelby	11/24	1	Report writing
Shelby	11/26	2	Report writing
Shelby	11/29	1.5	Report writing, sampling frequency measurement, SNR setup
Trevor	11/29	3	Demo video
Shelby	11/30	1	Report writing
Trevor	11/30	1.5	Artifacts document
Bryson	12/1	0.5	Report + Artifact
All	12/1	9	Presentation prep
Trevor	12/2	0.5	Graphs and documentation
All	12/3	12	Documentation and report writing
		Total Manhours	
	Percentage per Person	125	
Trevor	47.6	59.5	
Bryson	24.8	31	
Shelby	27.6	34.5	

8 Code

8.1 Arduino Sample Collection Code

```
/*
This program waits for an input in the serial monotor then prints the next 10500 values captred by the arduino to the serial port. It also passes the time it took to collect all the samples the time it took to gather thoes samples to the serial port.
Trevor Horine
11/6/2020
*/

//Pin assinments and global varables
const int analogInPin = A0;  // Analog input pin that is used as input form amplifier
int sensorValue = 0;         // value read from the output of amplifier
const int MAX_SAMPLES = 10500; //number of samples collected before sending
int last;

void setup() {
  Serial.begin(230400); // initialize serial communications at 230400 baud rate
  pinMode(LED_BUILTIN, OUTPUT); //built in led
  digitalWrite(LED_BUILTIN, HIGH);
}

void loop() {
  int values = MAX_SAMPLES+100;
  digitalWrite(LED_BUILTIN, HIGH);
  while(!(Serial.available() > 0)){ //while nothing in serial port (waiting for matlab to reach out)
    Serial.read();
    values = 0;
    last = true;
  }
  digitalWrite(LED_BUILTIN, LOW);//when found something in serial port (matlab has reached out)
  delay(10);
  long before = micros(); //current time in microseconds before collecting data
  while(values < MAX_SAMPLES) { //while less then desired number of samples has been printed to the serial port
    // read the analog in value form amplifier
    sensorValue = analogRead(analogInPin);
    byte buf[2]; //split in to bytes
    buf[0] = sensorValue & 255;
    buf[1] = (sensorValue >> 8)  & 255;
    Serial.write(buf, sizeof(buf)); // write to serial port
    values++;
  }
  if (last) {
    long diff = micros()-before;//calculate and write time it took to get samples to serial port
    byte buf[4];
    buf[0] = diff & 255;
```

```
    buf[1] = (diff >> 8)  & 255;
    buf[2] = (diff >> 16)  & 255;
    buf[3] = (diff >> 24)  & 255;
    Serial.write(buf, sizeof(buf));
    last = false;
  }
}
```

8.2 Matlab FFT and Graphing Single Set of Data

```
%This program will open a serial port, then send a signal to an Arduino. It
%will then read in a set number of samples and how long it took to collect
%them. The arduino ADC values will then be converted to voltages. The
%captured signal will be graphed and run through the MATLAB FFT to
%determine the frequancy of the signal. The Single-Sided Amplitude Spectrum
%is graphed and the highest frequancy in the range between 250 and 540 is
%reported to the command window.
%Trevor Horine
%11/6/2020

%start with a clean workspace
clear

%Number of samples
numSamples = 10500;


%open serial port set this to what port the Arduino is conected to
device = serialport("COM6",500000);
configureTerminator(device,"CR/LF")
%create empty vector to add values of the signal to
y = 0:0:0;
%read in line specified number (50) times from serial port
pause(5);
write(device,"getval","string");
%read in samples from Arduino
tic;
num = read(device,numSamples,"uint8")
toc
%read in time to collect samples from Arduino
timeIn = read(device,1,"uint32");
%convert time to seconds from micro seconds
captureSeconds = timeIn/1000000;
%print time to capture samples
fprintf("Capture Time: %f\n",captureSeconds);
%time per sample in seconds
sampleTime = captureSeconds/numSamples;
%create time vector to graph signal against
%x is the vector of the x-axis and has increments of time that
x = 0:sampleTime:captureSeconds-sampleTime;
%for loop to covert from ADC value and place it in the vector
for i = 0:numSamples-1
    %num is the string that is read in from the serial port
    curnum = num(i+1);
    %add it to the end of the y vector
    y(end +1) = (curnum*5)/256;
end
%release the serial port so it can be used by something else
%clear device
%create figure with a 1x3 array of graphs
figure
```

```
%plot the graph using the x and y vectors
%plot the whole signal
subplot(3,1,1)
plot(x,y)
%title graph and axes
title('Mic voltage')
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
%range of axis
xlim([0 1.05])
ylim([0 5])

%plot small section of signal so can see the period (might not line
    up with
%the window perficly, if not pan up or down)
subplot(3,1,2)
plot(x,y)
%title graph and axes
title('Mic voltage')
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
%range of axis
xlim([.1 .2])
ylim([2 4.5])

%MATLAB FFT
Fs = numSamples/captureSeconds;      % Sampling frequency
T = 1/Fs;          % Sampling period
L = numSamples;              % Length of signal (in milliseconds)
t = (0:L-1)*T;               % Time vector
Y = fft(y);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
%0.74074074074074 is weird factor that we found in testing, dont
    know where
%it came from but the FFT is off by this factor every time.
f = Fs*(0:(L/2))/L;
%f = (1/captureSeconds)*Fs*(0:(L/2))/L;

%SNR computation:
r = snr(y,Fs,3); %outputs snr in decibles, ignoring the first 3
    harmonics
fprintf("Signal to Noise Ratio %f\n", r)


%plot the Single-Sided Amplitude Spectrum (intensity at difrent
    frequancies)
subplot(3,1,3);
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
xlim([200 600])
ylim([0 .075])

%find maximum value in the intensity vector from the FFT and it's
    index
```

```
%ignore anything that is not between 250ish to 530ish
 b = P1(53:120);
 [maximum, maxindex] = max(b);
%if maximum is really small, liklely no signal
feq = round(f(maxindex+52));
if maximum < .001
        fprintf("No Signal\n")
        write(device, "N", "string");
    %print the freqancy of signal
else
    fprintf("Highest frequancy is %i\n", feq)
    if feq >= 260 && feq <= 263
        write(device, "MC","string");
    end
    if feq >= 292 && feq <= 295
       write(device, "D","string");
    end
    if feq >= 327 && feq <= 331
       write(device, "E","string");
    end
    if feq >= 347 && feq <= 351
       write(device, "F","string");
    end
    if feq >= 390 && feq <= 394
        write(device, "G","string");
    end
    if feq >= 437 && feq <= 442
        write(device, "A","string");
    end
    if feq >= 491 && feq <= 496
        write(device, "B","string");
    end
    if feq >= 520 && feq <= 526
        write(device, "HC","string");
    end
end
%clear workspace when done
clear device
```

8.3 Matlab Repeated Calculations and LED Communication

```
%This program will open a serial port, then send a signal to an Arduino. It
%will then read in a set number of samples and how long it took to collect
%them. The arduino ADC values will then be converted to voltages. The
%captured signal will be graphed and run through the MATLAB FFT to
%determine the frequancy of the signal. The Single-Sided Amplitude Spectrum
%is graphed and the highest frequancy in the range between 250 and 540 is
%reported to the command window.
%Trevor Horine
%11/6/2020

%start with a clean workspace
clear

%Number of samples
numSamples = 10500;
%open serial port set this to what port the Arduino is conected to
device = serialport("COM6",500000);
configureTerminator(device,"CR/LF")

while 1
    pause(2);
    %create empty vector to add values of the signal to
    y = 0:0:0;
    write(device,"getval","string");
    %read in samples from Arduino
    tic;
    num = read(device,numSamples,"uint8");
    toc
    %read in time to collect samples from Arduino
    timeIn = read(device,1,"uint32");
    %convert time to seconds from micro seconds
    captureSeconds = timeIn/1000000;
    %print time to capture samples
    fprintf("Capture Time: %f\n",captureSeconds);
    %time per sample in seconds
    sampleTime = captureSeconds/numSamples;
    %create time vector to graph signal against
    %x is the vector of the x-axis and has increments of time that
    x = 0:sampleTime:captureSeconds-sampleTime;
    %for loop to covert from ADC value and place it in the vector
    for i = 0:numSamples-1
        %num is the string that is read in from the serial port
        curnum = num(i+1);
        %add it to the end of the y vector
        y(end +1) = (curnum*5)/256;
    end
    %release the serial port so it can be used by something else
    %clear device
    %create figure with a 1x3 array of graphs
    % figure
```

```matlab
    % %plot the graph using the x and y vectors
    % %plot the whole signal
    % subplot(3,1,1)
    % plot(x,y)
    % %title graph and axes
    % title('Mic voltage')
    % xlabel('Time (seconds)')
    % ylabel('Voltage (Volts)')
    % %range of axis
    % xlim([0 1.05])
    % ylim([0 5])
    %
    % %plot small section of signal so can see the period (might not line up with
    % %the window perficly, if not pan up or down)
    % subplot(3,1,2)
    % plot(x,y)
    % %title graph and axes
    % title('Mic voltage')
    % xlabel('Time (seconds)')
    % ylabel('Voltage (Volts)')
    % %range of axis
    % xlim([.1 .2])
    % ylim([2 4.5])

    %MATLAB FFT
    Fs = numSamples/captureSeconds;      % Sampling frequency
    T = 1/Fs;          % Sampling period
    L = numSamples;               % Length of signal (in milliseconds)
    t = (0:L-1)*T;                % Time vector
    Y = fft(y);
    P2 = abs(Y/L);
    P1 = P2(1:floor(L/2+1));
    P1(2:end-1) = 2*P1(2:end-1);
    %0.74074074074074 is weird factor that we found in testing, dont know where
    %it came from but the FFT is off by this factor every time.
    f = Fs*(0:(L/2))/L;
    %f = (1/captureSeconds)*Fs*(0:(L/2))/L;

    %plot the Single-Sided Amplitude Spectrum (intensity at difrent frequancies)
    % subplot(3,1,3);
    % plot(f,P1)
    % title('Single-Sided Amplitude Spectrum of X(t)')
    % xlabel('f (Hz)')
    % ylabel('|P1(f)|')
    % xlim([200 600])
    % ylim([0 .075])

    %find maximum value in the intensity vector from the FFT and it's index
    %ignore anything that is not between 250ish to 530ish
     b = P1(50:120);
     [maximum, maxindex] = max(b);
     feq = round(f(maxindex+49));
    %if maximum is really small, liklely no signal
```

```
    if maximum < .001
        fprintf("No Signal\n")
        write(device, "N", "string");
    %print the freqancy of signal
    else
        fprintf("Highest frequancy is %i\n", feq)
        if feq >= 260 && feq <= 263
            write(device, "MC","string");
        end
        if feq >= 292 && feq <= 295
            write(device, "D","string");
        end
        if feq >= 327 && feq <= 331
            write(device, "E","string");
        end
        if feq >= 347 && feq <= 351
            write(device, "F","string");
        end
        if feq >= 390 && feq <= 394
            write(device, "G","string");
        end
        if feq >= 437 && feq <= 442
            write(device, "A","string");
        end
        if feq >= 491 && feq <= 496
            write(device, "B","string");
        end
        if feq >= 520 && feq <= 526
            write(device, "HC","string");
        end
    end
end
%clear device when done
clear device
```