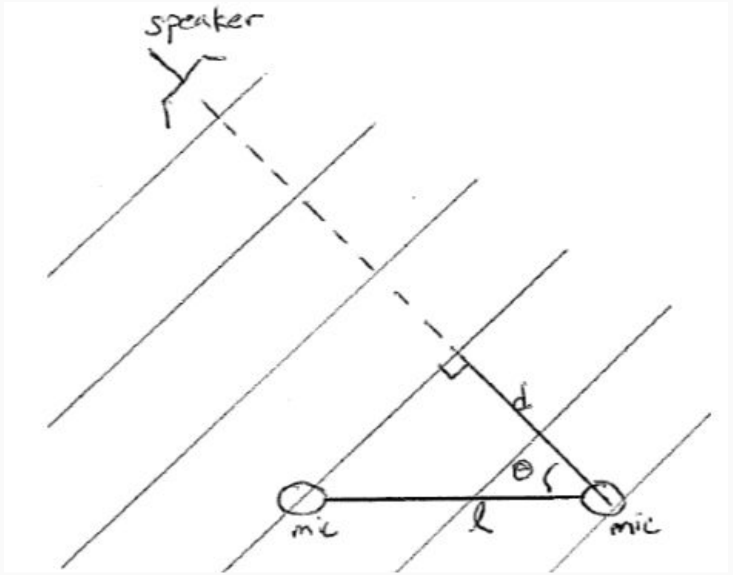
Presentation 2

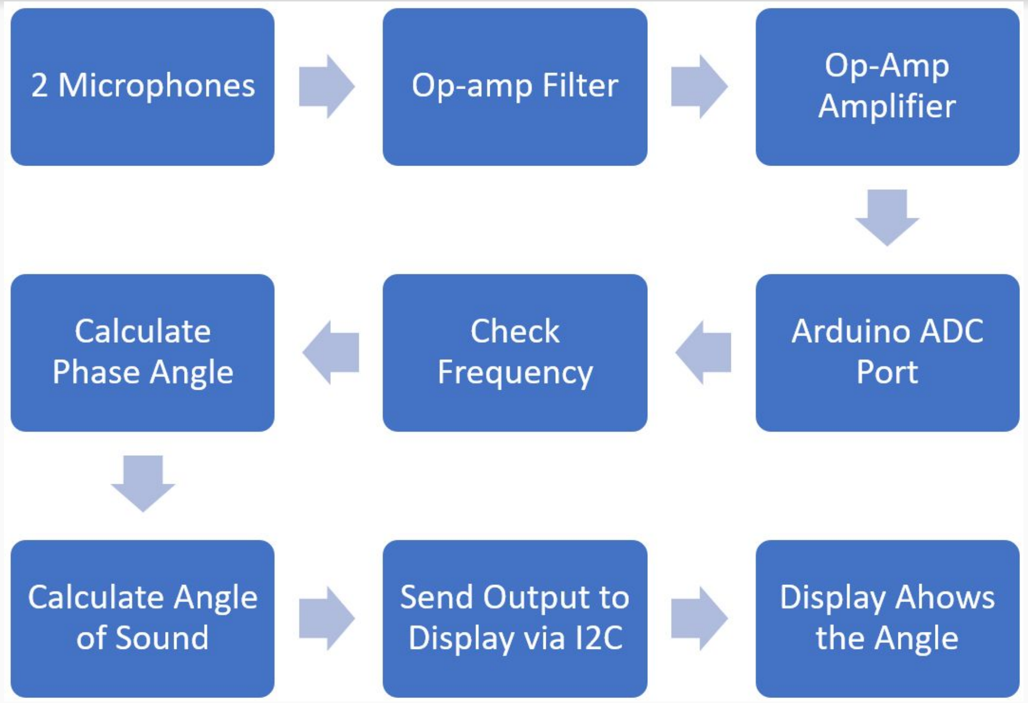
Saksham Goyal and Jeffrey Kedda

Introduction

- Our goal was to be able to find the direction from which a signal is emitted.
- We have 2 microphones which we use to find the source.
- The source will be a 900 Hz signal
- We have a screen where we will display the calculated direction



Block Diagram



Approach to Solving The Problem

Saksham Goyal

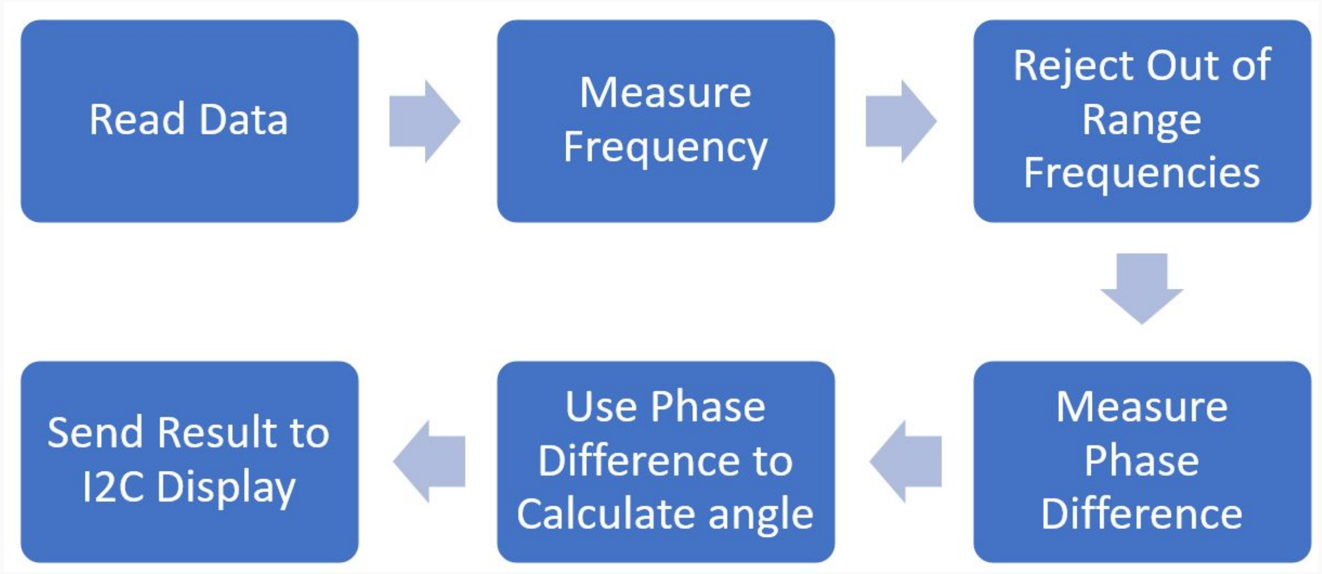
Filtering the Input

- The project wanted us to limit our signals so we only accept a range of 800 - 1000 Hz.
- This means that we want to accept a small band of frequencies.
- This led us to use the Narrow Multiple Feedback Band-Pass filter.
- The requirements wanted a hard cutoff at 800-1000 Hz so we implemented a digital filter

Amplification

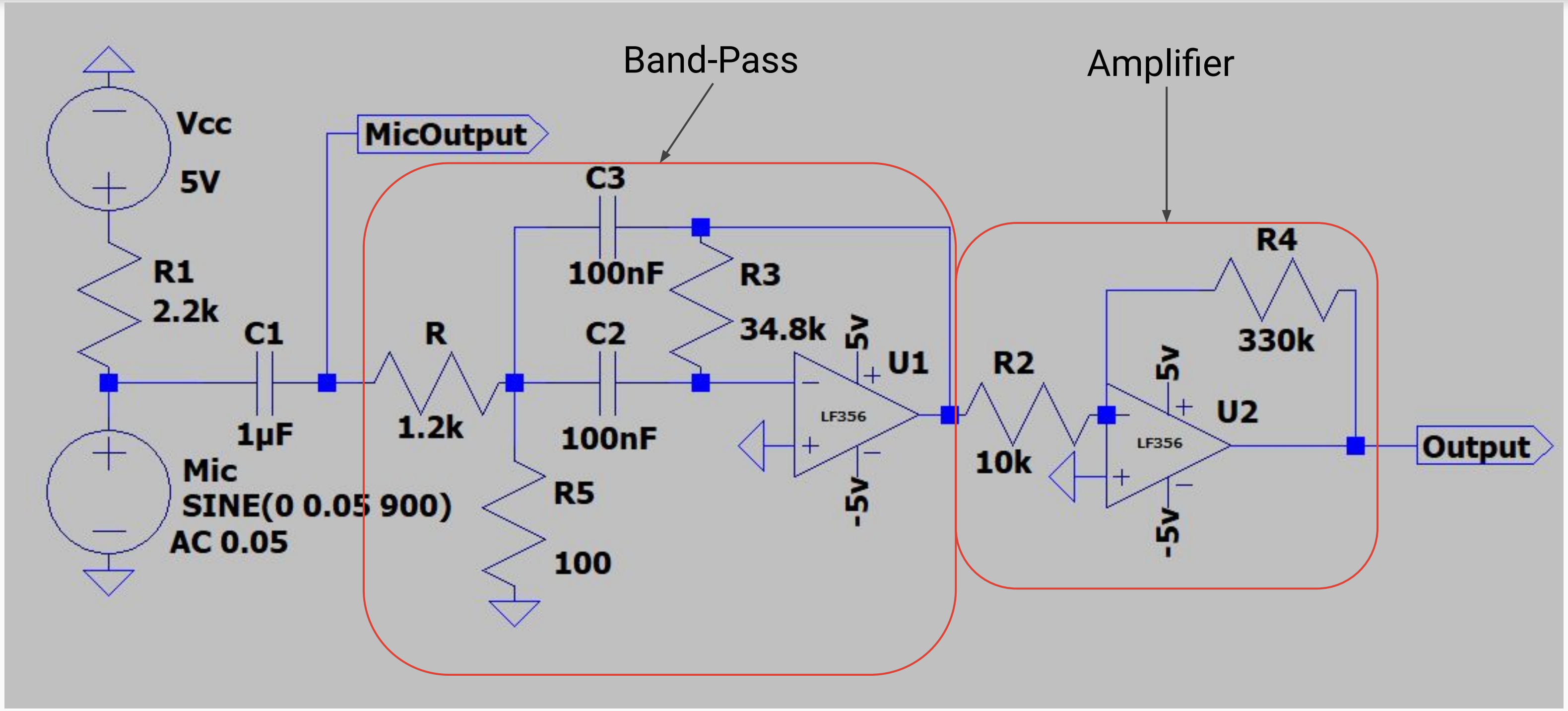
- The amplitude of the signal directly from the mic is about 2.5 mV
  - Using a 2.2kΩ biasing resistor as per the documentation of the Mic
- After passing the signal through the Band-Pass filter, which had a gain of about 12x
  - The signal reached about 30mV
- We added a amplification op-amp stage to increase the voltage more
  - The amplifier had a gain of 33 V/V
  - The output reached approximately 1V with this

Code

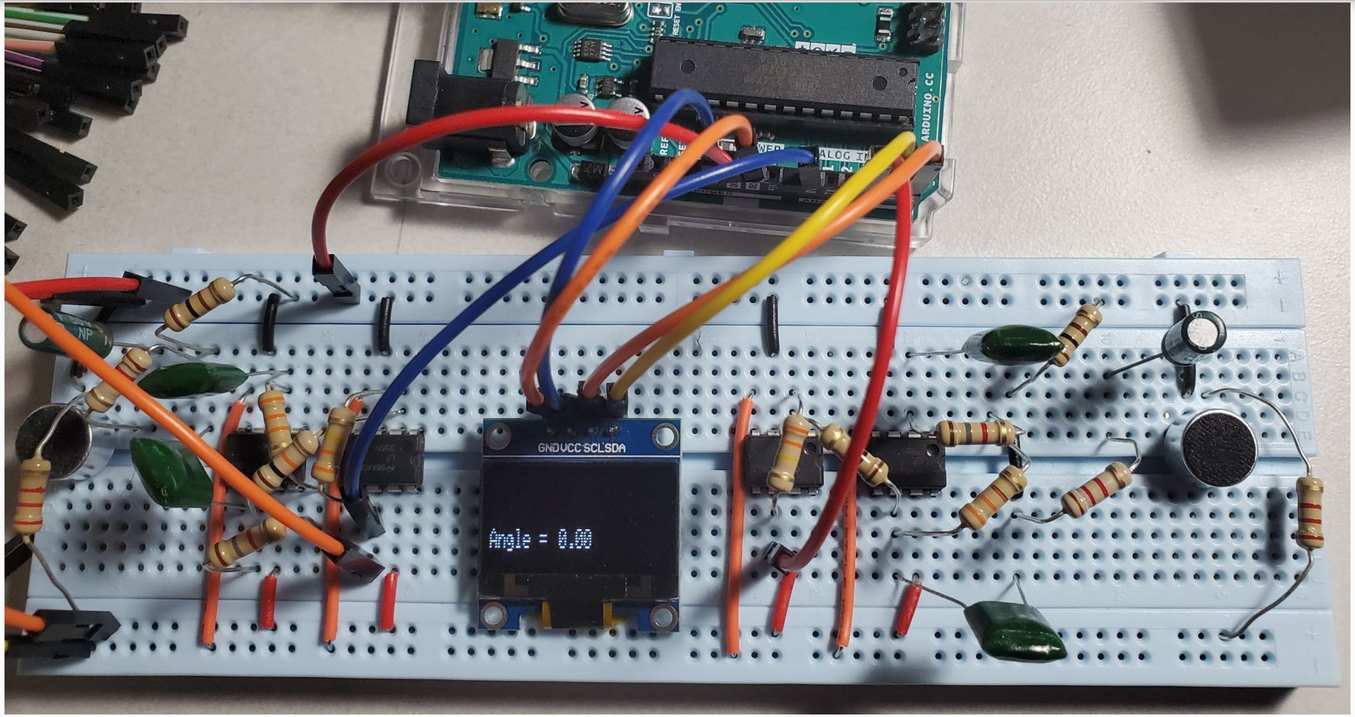


Read Data
Measure Frequency
Reject Out of Range Frequencies
Measure Phase Difference
Use Phase Difference to Calculate angle
Send Result to I2C Display

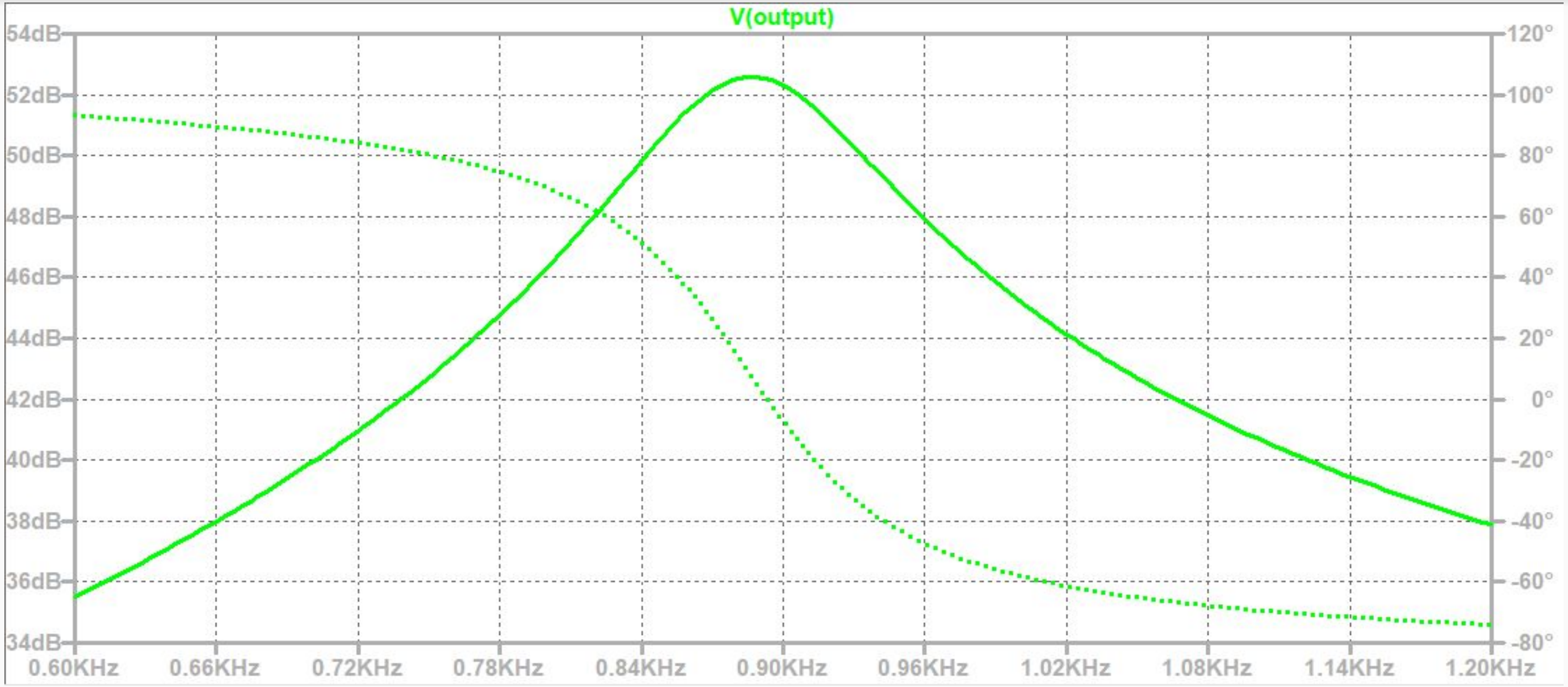
Final Circuit Design



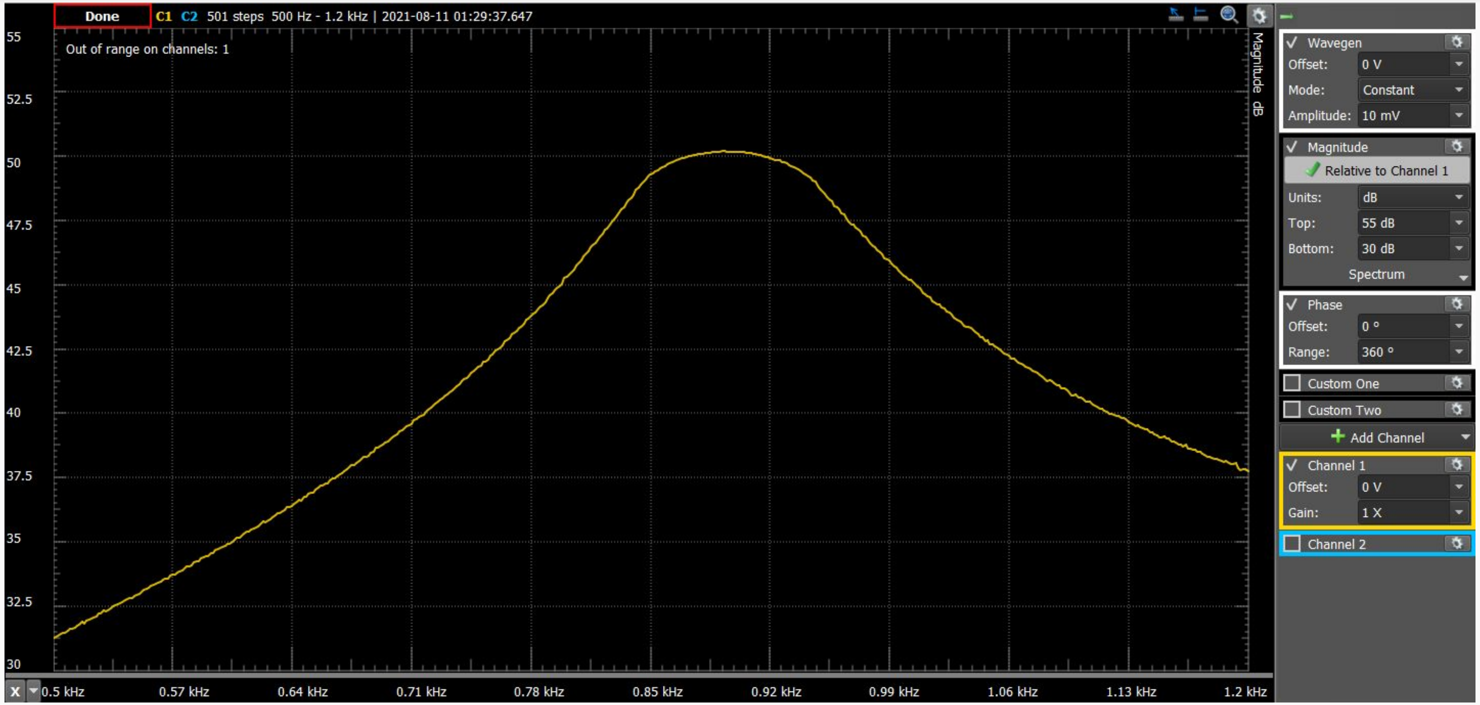
Picture of Circuit



Testing the Filter/Amplification



Testing the Filter/Amplification


Done C1 C2 501 steps 500 Hz - 1.2 kHz | 2021-08-11 01:29:37.647

Out of range on channels: 1

✓ Wavegen
Offset: 0 V
Mode: Constant
Amplitude: 10 mV

✓ Magnitude
Relative to Channel 1
Units: dB
Top: 55 dB
Bottom: 30 dB
Spectrum

✓ Phase
Offset: 0 °
Range: 360 °

Custom One
Custom Two
Add Channel

✓ Channel 1
Offset: 0 V
Gain: 1 X

Channel 2

Code

Jeffrey Kedda

Code

- In the code we have to read the signal and measure the time
  - analogRead()   --> to read the signal
  - micros()   --> measure the time in microseconds
- Then we need to measure certain aspects of it so we can filter out frequencies that we do not want (<800 or >1000)
  - Measure frequency, if out of range, don't do anything with the signal
  - We used zero-crossing method to make it more reliable compared to finding the peaks
- We need to then calculate the difference in phase between the signals
  - Measure the time difference between the point where the signals cross the x-axis (y=0)
- Then use the phase difference to convert that into the angle of the source
  - Use arcCosine to find the angle of the sound

Measuring Frequency

- We first try to find the time when the signal has a rising edge
  - Check the time when we detect it
- Then find the falling edge
  - Convert to seconds from microSeconds
  - Use difference between the 2 times
  - 1/period = frequency
- We can then also implement a simple digital filter
  - If the measured frequency is out of range (within a tolerance of 10Hz), then we can simple stop the calculation of the angle

```
if (oldV1 < 5 && newV1 > 5) {
  timerStart = micros();
}
```

```
if (oldV1 > 5 && newV1 < 5){
  frequency = 1000000.0 / (micros() - timerStart);
  periodMS =  1.0 / frequency;
}
```

```
if (frequency < 790 || frequency > 1010) {
  frequency = 0;
}
```

Measuring the Difference

- Find rising edge of the 2nd signal
  - Measuring the time here will give us the time delay between the 2 rising edges

```
if (oldV2 == 0 && newV2 >= 5) {
  timerStop = abs(micros() - timerStart) / 1000000.0;
}
```

- We can then finally find the direction of the sound
  - We use the speed of sound, the distance between the 2 mics, and the difference in time

```
Direction = acos((SOUNDSPEED * timerStop) / micDistance);
```

Using the Display

- We had to use the OLED display to show the angle we calculated.
  - We print text on the display “Angle = XX”
- The display works over the I2C interface onboard the Arduino
  - I2C interface uses A4 and A5 pins
- The code for the display does not need to be done from scratch since Adafruit has a library for this display
  - We used the <Adafruit_GFX.h> and <Adafruit_SSD1306.h> libraries
  - Documentation for the library was used to see how to use the display

Using the Display

//Display Initialization

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(4);
```

//In the loop() function

```
//The of statement lets us use the
//screen every 3000 loops to avoid
//slowing the code down

if (counter % 3000 == 0) {
display.setCursor(50, 20);
  display.print(frequency);
  display.print("   ");
  display.display();
}
```

//In the setup() function

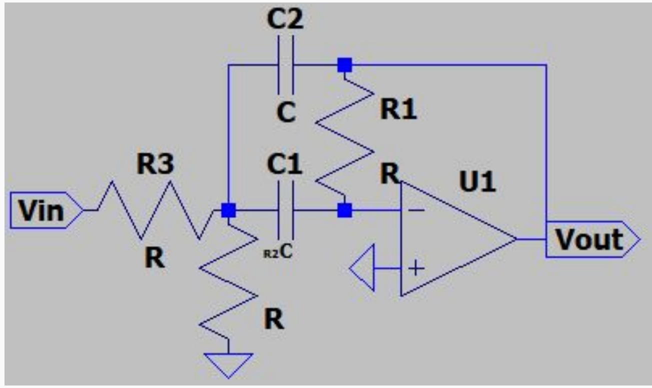
```
//The following code only needs to be run once

//Gives the I2C interface the address of the display
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

display.setTextSize(1);
display.setTextColor(WHITE, BLACK);
display.setCursor(0, 0);
display.println("Find Audio Direction");
display.setCursor(0, 20);
display.println("Angle = ");
display.display();
display.setCursor(50, 20);
```

Sub-System 1 - Filter

Saksham Goyal

Narrow Band Pass Filter

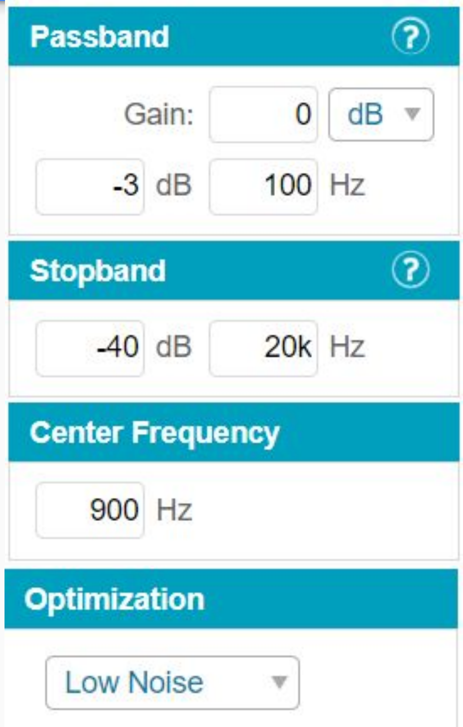
- We used a Multiple Feedback Narrow Band-Pass Filter
  - Narrow band pass filters are similar to combining a low-pass and a high-pass filter into one
- Pass band is 800 Hz - 1000 Hz
  - the passband needs to be very narrow to ensure a steep rolloff
- Filter and Amplifier are both inverting designs
  - This makes it so that the output is still positive
  - This also significantly simplifies the circuit
- We were also able to source LTspice files for our Op-Amps (LF365N)
  - The simulation would be more accurate



Choosing Values

We used the Filter Wizard to find out which components we needed to use.

- We chose a gain of 0db since we amplify the signal later
    - We still got a gain of about 12 because the values we picked were not the same as the ones we got from the wizard
- 20k for the stop band was chosen
    - Avoid having 2 filter stages which could distort the signal/phase too much
- Low noise optimisation was chosen
    - Ensure a clean signal for the Arduino



Passband

Gain: 0 dB

-3 dB 100 Hz

Stopband

-40 dB 20k Hz

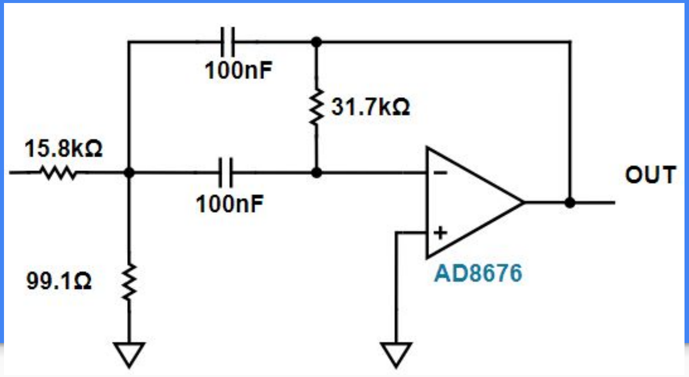
Center Frequency

900 Hz

Optimization

Low Noise

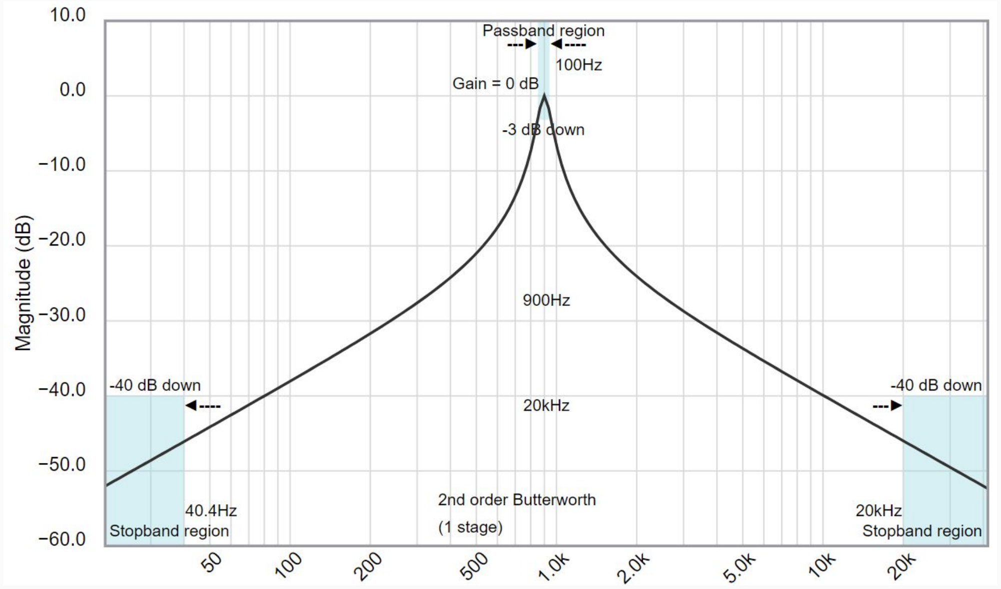
Choosing Values



We had to change the values of the resistors to accurately reflect our circuit requirements, input voltages, and the ones we had available in the kit.

The following changes were made:

- 15.8k --> 1.2k (resulted in 12x gain)
- 99.1 --> 100 (closest value in kit)
- 31.7k --> 34.8k (shift center of pass band back to 900 Hz)

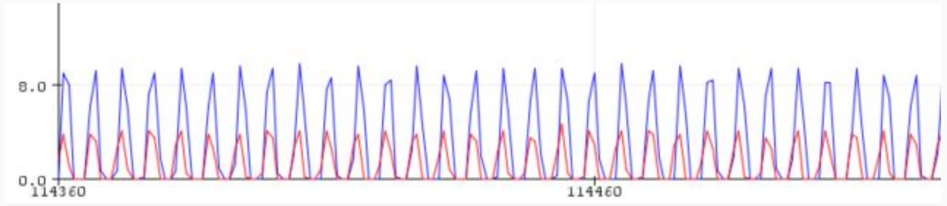


Validation

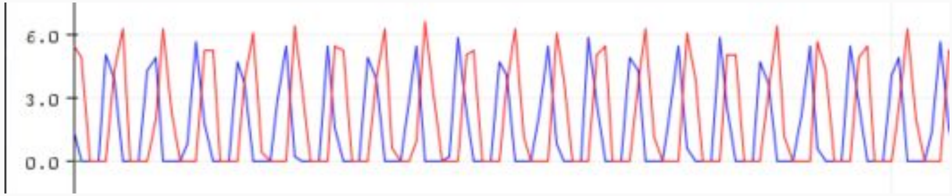
Correct Phase Angles

- To test our code, we used the AD2 to send a signal directly to the Arduino to eliminate the variable of the circuit.
- We sent a 900 Hz signal on the 2 different mics and varied the phase to see if the serial monitor would be able to detect the differences.
- As you can see, it was working as intended

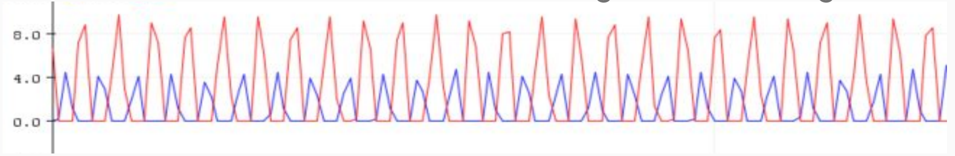
0 degree Phase Angle



90 degree Phase Angle



180 degree Phase Angle



Video