McGill Physical Journal

An Experimental Verification of the Doppler Effect and the Determination of the Speed of Sound in Air through an Experiment Automation Using an Arduino/Python Program Pair

William Eaton (260989306), Loïc Miara (260989992) and Aaron Desrochers (260988492)

McGill University Department of Physics

August 24, 2022

Abstract

In this paper, we propose an experimental way to demonstrate and verify the Doppler effect using an Arduino/Python program pair to control an apparatus consisting of a 3D printed pinion-rack system, a stepper motor and a speaker controlled by a function generator. We successfully demonstrate the linear relationship between the relative velocity between the emitter and observer and the relative frequency shift. In addition, this experiment allows for an experimental determination of the speed of sound in air, which we determined to be $328 \pm 24\ m/s$. This value is 4% away from the theoretical value of 343.2 m/s.

1 Introduction

The Doppler shift is a phenomenon that describes the difference between the wave frequency seen from an observer and the emitted frequency of an emitter based on their relative velocities [1]. The Doppler shift is something that must be considered when trying to observe something that is moving fast, such as trying to detect planes with radar or measure the speed of recession of distant galaxies. It is given by the equation:

$$f_o = \frac{c + v_o}{c + v_s} f_s, \tag{1}$$

where f_o is the frequency observed, f_s is the emitted frequency, v_o is the relative velocity of the observer, v_s is the velocity of the emitter, and c is the speed of the wave. Therefore, if the relative velocity between the observer and the emitter is positive (moving away from each other), we expect a decrease in the observed frequency, and conversely. This is due to the fact that the observed waves are being compressed or stretched due to the movement of both the observer and the emitter, as the wavefronts emitted take more or less time to reach the observer.

The Doppler shift equation can be rewritten as:

$$\Delta f = \frac{\Delta v}{c} f_e, \tag{2}$$

where Δf and Δv are simply the difference between both frequencies and velocities. Here, we can see that we expect a linear relationship between the frequency emitted and the difference in velocities at a constant emitted frequency.

The purpose of this experiment is to demonstrate and verify the Doppler effect through a computer controlled experiment. By varying the travelling velocity of a microphone sensor recording a pure sine-wave frequency and plotting the change in frequency for the different velocities, we expect to get a slope of $1/c$ where c is the speed of sound. This can then be compared against the true value of the speed of sound which is given by:

$$20.05 \cdot \sqrt{T}\ m/s, \tag{3}$$

Where T is the room temperature [2].

When detecting frequencies, a Fast-Fourier Transform (FFT) will have to be used in order to pick out and detect the most prominent frequency. A Fourier transform is a tool that is capable of taking any waveform and decomposing it into a sum of sinusoidal waves [3]. The equation by given by:

$$F_n = \frac{1}{T}\int_{T/2}^{T/2} f(t)e^{-inwt}dt, \tag{4}$$

where T is the total time sampling and $f(t)$ is the original waveform of the sound.

Therefore, by passing a complicated wave form through the Fourier transform, all the different frequencies that make up this waveform can be found. An FFT uses this principle but for discrete sampling rate. However, it's important to note Nyquist's Theorem which tells us that for some given waveform, the sampling rate for the FFT algorithm must be at least two times the highest expected frequency or else it won't be able to properly compute the correct peak frequency [4]. The Arduino code that governs the workings of the microphone is inspired by an experiment from C. Lettsome [5]. The code has been modified to detect higher frequencies and continuously write the results of the FFT to the serial port, which is then picked up by the Python program. An example of the recorded data can be seen later in Figure 5.

2 Materials and Methods

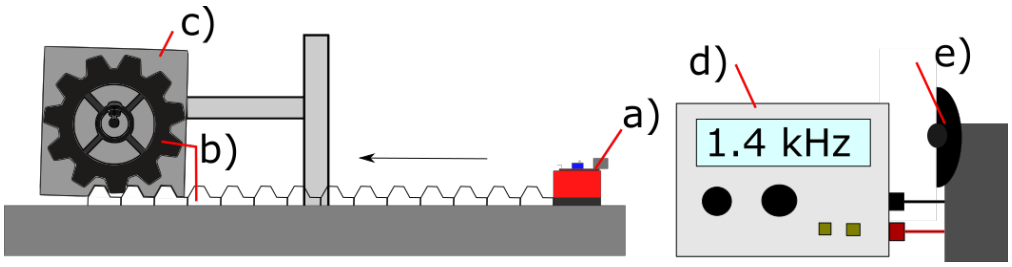

Figure 1: The setup used for this experiment. Here we used: **a)** A high sensitivity sound sensor, **b)** a gear and rack system. The rack is secured by a channel and the gear is secured to the motor using a screw in order to preserve the gear. **c)** A stepper motor, **d)** a frequency generator and **e)** a speaker. Click here to see an example of the left part of the setup (**a)-c)**) in action.

Figure 1 shows the setup for the following experiment. We used a rack and pinion sytem to move a high sensitivity sound sensor [6] towards and away from a speaker.

The speaker was connected to a frequency generator in order to precisely create a sinusoidal wave. The frequency was set to 1.4 kHz and the speaker was placed at approximately half a meter away from the rack and pinion system and pointed towards the microphone.

The sound sensor was attached to the end of the rack and placed facing towards the speaker. The potentiometer on the sensor was calibrated until the sound intensity from the speaker was detectable. The speaker itself does not record frequencies but rather the intensity of the sound. However, by recording the intensity over time with a sampling frequency of 3048 Hz, an FFT can be used to find the most prominent frequency which is the frequency emitted by the speaker, from this intensity readings. It was also found that setting a sampling frequency to target a specific range of frequencies (3048 Hz to target frequencies in the 1k-2k Hz range in this case) made the microphone unable to detect frequencies outside of that range. This worked in our favor, as background noise and voices of people around the microphone did not interfere with the experiment.

It was noted that the frequency emitted by the speaker and the frequency recorded by the sensor while stationary were not the same. Therefore, a calibration was performed for varying emitted frequencies in order to acquire a relationship for emitted and detected frequencies

(see subsection 3.1).

The gear was secured to the axle of the stepper motor using a screw and nut (see Figure A2). This was done because the high speeds used by the stepper motor tended to deform and carve out the inner plastic of the gear. The gear was capable of moving the rack by 25.9cm per rotation and had a maximum speed of $207.2\frac{cm}{s}$ before the torque of motor was unable to move the whole system. It was noted that at high speeds the gear had a tendency to skip steps since it didn't have enough time to accelerate and decelerate. Therefore, a calibration was performed to see how accurate the inputted speeds and distances compared to the actual speeds and distances traveled by the motor (see subsection 3.2). The channel was also lubricated in order to remove as much friction as possible.

3 Results

Before any data was taken, two calibrations of the apparatuses were performed, i.e. one for the microphone sensor and one for the motor.

3.1 Microphone Calibration

The first step in calibration was to measure the difference between the set input frequency and the frequency recorded through the microphone. To do this, varying known frequencies were inputted through the speaker and the output from the microphone was recorded. An example of raw data and its distribution in frequency space can be seen in Figure 2.

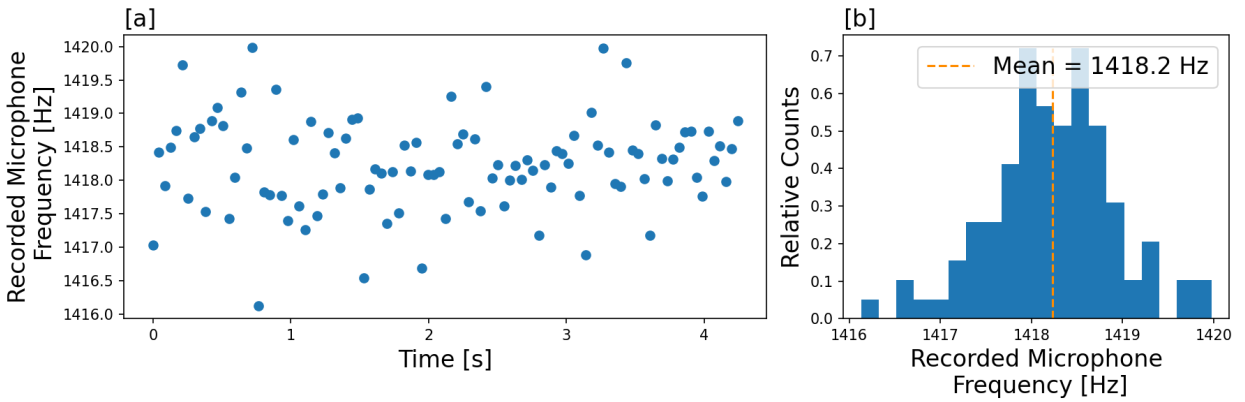

Figure 2: An example of the recorded frequencies by the microphone. [**a**] shows all of the outputs of the FFT in the Arduino program for the microphone against time. [**b**] Shows the same data binned up in a histogram. This particular run was done with an input frequency of 1380Hz. We notice that the mean recorded frequency by the microphone is not exactly near the input frequency, which justifies the need for the calibration.

From these distributions, the means were found and plotted against 9 different input frequencies as shown in Figure 3. Here, the standard deviations for each frequency recording were taken as the error for each data point. The results appear to be linear and follow the equation

$$1.0377 \cdot f_{in} - 12.4Hz. \tag{5}$$

From this we see that the recorded frequencies appear to be slightly greater than the emitted ones and that the difference increases for for larger frequencies. The associated cumulative probability for such a χ^2_r of 1.63 is $P = 88\%$, or that fluctuations alone will create such a value or lower 88% of the time. This indicates a good fit for a linear calibration and a good estimation of our errors.

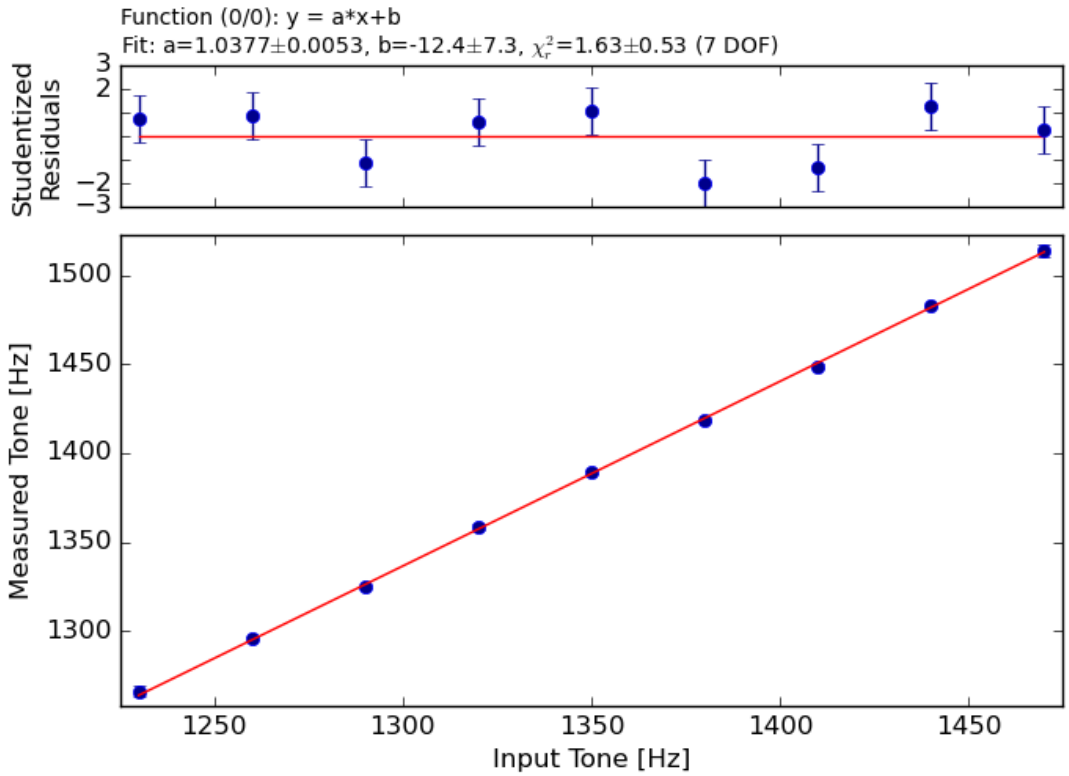

Figure 3: Calibration of the microphone sensor. Different known input frequencies are played on the speaker through a function generator. We then record the frequency on the microphone through a Fourier transform and plot the measured frequency against the input frequency.

3.2 Motor Calibration

Secondly, the speed of the stepper motor was calibrated. At high speeds the motor was suspected to skip steps. To do this, a video recording of 60 frames/second was taken of the setup in action. The motor was set to rotate at varying RPS. By counting the frames of the rotating gear, the difference between the set and true velocity can be found. The results of this are shown in Figure 4. Since the slope is slightly lower than 1, it shows that at higher speeds the motor tends to skip steps and give a different distance travelled then desired. Therefore, this difference can be used for the error of the velocity. Again, the χ_r^2 value of 1.1 is associated with a cumulative probability that indicates a good fit and a good estimation

of the error ($P = 65\%$).

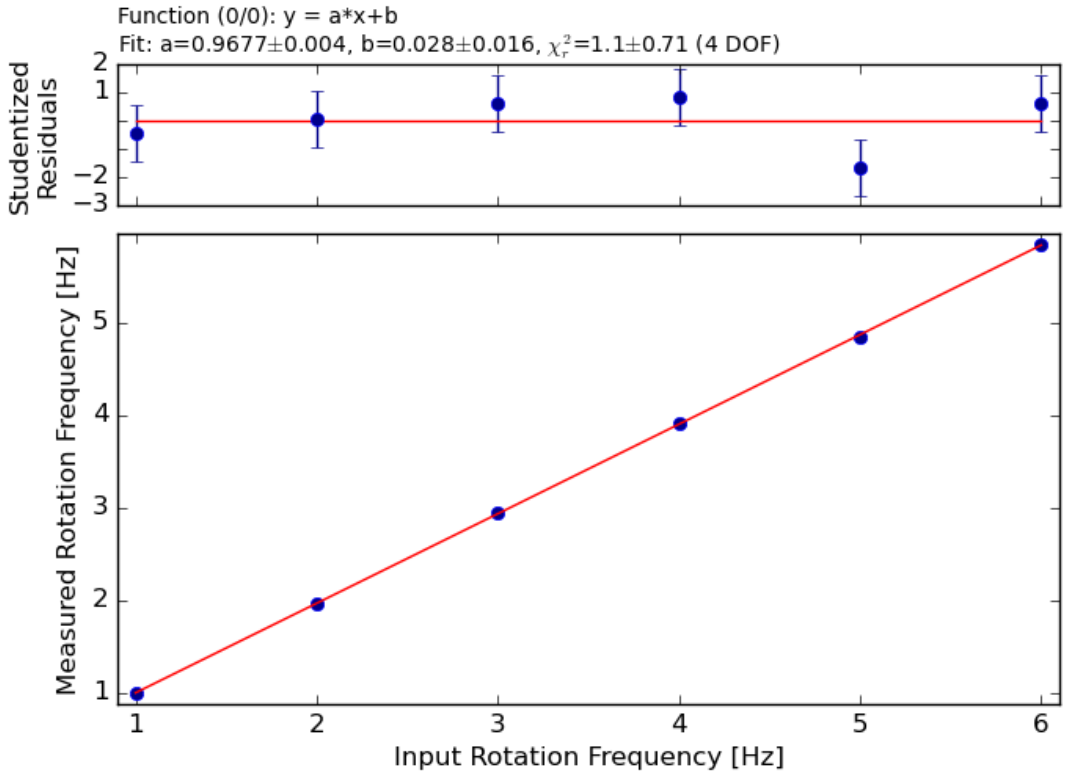

Figure 4: Calibration of the motor. We input different known RPS (rotations per second) through our Arduino code. We then recorded the frequency of rotation of the motor by recording a video of the rotating gear. The recorded video was performed at a known framerate (60fps) which allows us to infer the time it took for the motor to turn for a set number of turns (15) by counting the frames between the beginning and the end of the motor's motion.

3.3 Measuring Frequency Shifts

Once these calibrations were completed, the actual data was taken. To do this, the microphone was placed at the closest point to the speaker. While the speaker was emitting a known frequency, the gear would rotate and move the whole rack and microphone backwards and then forwards multiple times. An example of the data for an individual run can be seen in Figure 5. The frequency values on these datasets correspond to the raw frequencies measured by the microphone and passed through the reciprocal of the calibration function

of Equation 5.

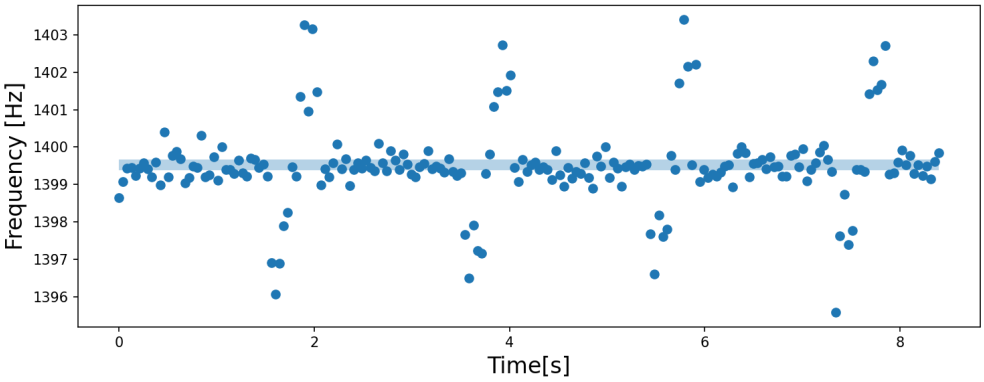

Figure 5: An example of the frequency recorded by the microphone when it is moved on the track. We observe that each time the microphone moves it is characterized on the plot by a dip and then a rise compared to the set frequency of 1400 Hz. This corresponds to a backward and forward velocity of the microphone. This particular run was performed at a velocity of $77.7 \pm 0.3\ cm/s$ and 4 complete back and forth motions were performed. The solid blue band represents the standard deviation of frequencies when the microphone is static.

We repeat this procedure for an array of velocities (see Figure A3). However, because the sampling frequency of the microphone is finite, the raw data does not guarantee that we did in fact measure the maximum possible shift in frequency for a set velocity. The microphone only moves for a short amount of time, so the number of datapoints taken during the time that the microphone is moving isn't large, particularly at higher velocities. We then select the datapoint that is the farthest away from the base frequency as the frequency shift for that velocity. Hence, the error on the maximum shift is higher for increasing velocity, in a manner that is proportional to the velocity. Repeating this procedure for different velocities yields the expected linear model as can be seen in Figure 6.

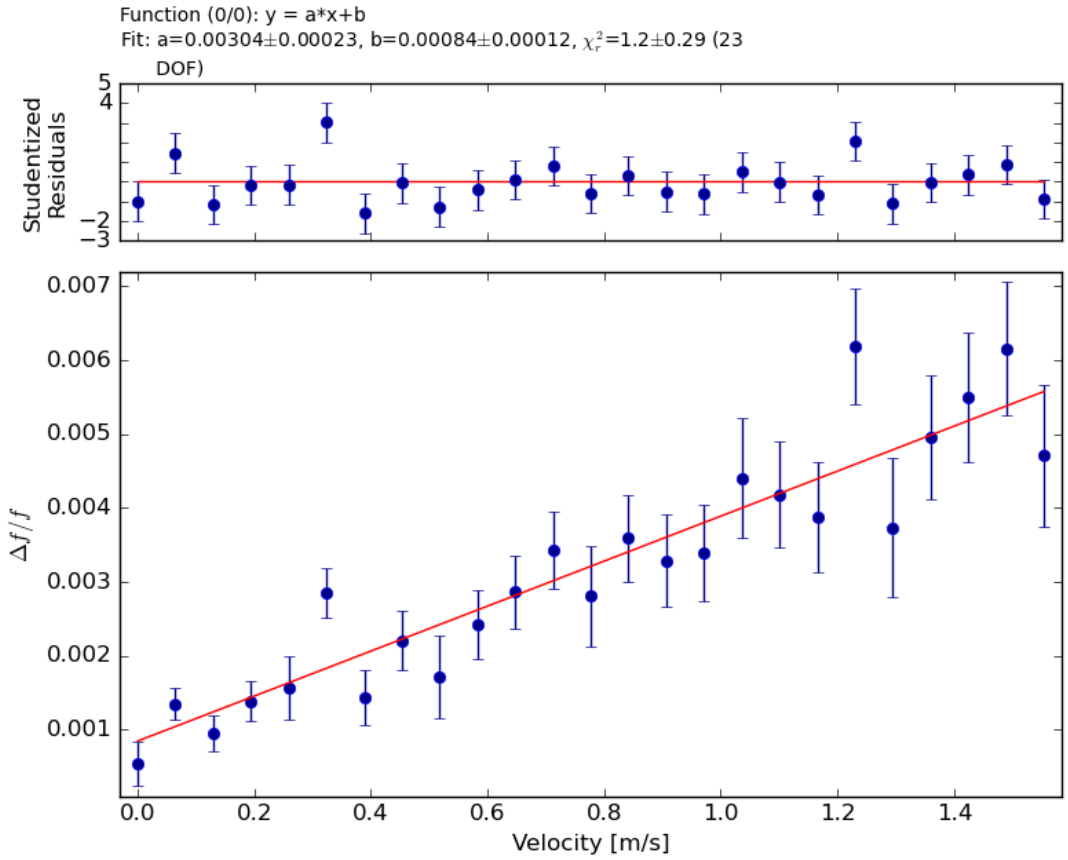

Figure 6: The maximal frequency shift for the different microphone velocities. We observe the expected linear relationship between those quantities. All of the frequency and velocity values correspond to the raw values passed through the reciprocal calibration functions to re-map them to their physical values. We do not observe structure in the residuals.

3.3.1 The Speed of Sound

This fit then allows for an experimental determination of the speed of sound in air. Taking the inverse of the slope yields a speed of sound $c = 328 \pm 24\ m/s$. This corresponds to a 4% error on the theoretical speed of sound in $20°C$ air of 343.2 m/s.

4 Discussion

As previously stated in subsection 3.3, the fact that the sampling frequency of the microphone is finite affects the accuracy of our measurements. Each data point we gathered is the FFT of 128 sample frequencies from the microphone, meaning we can only get a frequency value every 128 samplings. Since our rack is small, we only move it 1.5 turns of the gear, which means that our microphone is only in motion for a small amount of time, and inversely proportional to the velocity. This isn't an issue at slower speeds, but since we have to wait 128 samples from the microphone to get a frequency reading, it starts to become an issue as we increase the velocity of the microphone. At these higher speeds, the microphone isn't in motion for long enough to gather enough data points to conclusively determine the maximum Doppler shift, which is why the error on the maximum Doppler shift increases as the speed increases. To fix this issue, we would need to use a longer rack, but that would come with more mechanical issues, as we would then need to use multiple gears and motors to move this longer, heavier rack. We would also need to significantly increase the amplitude of the sound emitted by the speaker such that the microphone still picks up its frequency when at the furthest distance from the speaker.

Our determined value for the speed of sound, $c = 328 \pm 24\ m/s$, is within 1 standard deviation of the actual value of 343.2 m/s, which leads us to believe that our setup for recording data is sound and that our error was properly estimated. The experimental value is then 4% off from the theoretical expectation. This is on-par with other experimental determinations of the speed of sound, such as using Kundt's method [7], [8]. However, the uncertainty on the experimental value is 7%. Bringing the abovementioned modifications to the setup would effectively reduce the uncertainty on the experimental value, and yield a more precise determination of the speed of sound.

Regarding the linear fits of our data, it is found that no residuals show any structure, and that all fits are good. The respective χ_r^2 values for the fits of the microphone calibration, the motor calibration and the Doppler shift (Figure 3, Figure 4, and Figure 6) are $1.63, 1.1$ and 1.2 respectively, which indicates that the departures of the data from the fit are indeed due to statistical fluctuations, and that there does not appear to be unforeseen systematic errors

in the experiment. This agrees with our expectations that the response from the microphone and motor were linear, and agrees with the theoretical model explaining the Doppler effect.

5 Conclusions

To measure Doppler shift, a microphone was placed at the end of a rack and pinion system. The gear was secured to a motor which made it turn at computer controlled velocities. A speaker was then oriented towards the microphone while emitting $1400Hz$ pure-sine frequency, as the gear and rack system made the microphone move back and forth. By measuring the shift in frequencies as the microphone moved at different speeds, the speed of sound in air was determined to be $c = 328 \pm 24\ m/s$, which corresponds to a 4% error compared to the literature value of 343.2 m/s.

References

[1]R. Kleppner Daniel Kolenkow, *An introduction to mechanics* (Cambridge University Press, 2014).

[2]A. Rienstra S.W. Hirschberg, *An introduction to acoustics* (Eindhoven University of Technology, 2021).

[3]E. Oran Brigham, *The fast fourier transform and its applications* (Prentice-Hall, 1988).

[4]R. Oshana, "4 - overview of digital signal processing algorithms", in *Dsp software development techniques for embedded and real-time systems* (Elsevier/Newnes, 2006), pp. 59–121.

[5]C. Lettsome, *My weekend project: audio frequency detector using an arduino*, https://www.youtube.com/watch?v=wbeV0J30LGQ&t=752s&ab_channel=ClydeLettsome%5C%2CPhD%5C%2CPE, (accessed: 04/22/2022).

[6]Vellman, *Arduino® compatible microphone sound sensor module*, (2017) https://www.robotshop.com/media/files/pdf/sound-sensor-module-arduino-datasheet.pdf.

[7]S. Hellesund, "Measuring the speed of sound in air using a smartphone and a cardboard tube", Physics Education **54**, 035015 (2019).

[8]S. O. Parolin and G. Pezzi, "Smartphone-aided measurements of the speed of sound in different gaseous mixtures", The Physics Teacher **51**, 508–509 (2013).

6 Appendix

6.1 Setup

We went through many iterations of our setup before finally settling on one that worked. We originally started of with an 8 spoke gear with much larger teeth, thinking that the larger teeth would reduce slipping between the gear and the rack. The problem was that with larger teeth, the rack can't be moved as fast. A smaller test gear with smaller teeth was then created as well as a new rack system with smaller teeth. This system worked well with our original stepper motor, but we couldn't manage to reach fast enough speeds. After attempting to push the original stepper motor to its limits without success, we decided to switch to a different stepper motor, for which a new gear was made, which was as big as the original gear, but with the smaller teeth of the prototype. This new system worked very well at lower speeds, but as we tried to push the motor to higher speeds, we ran into a problem. The axle of the motor would begin to deform the gear attachment to the point that the gear would just spin loosely while attached to the axle. To solve this problem, we decided to add a screw and nut to our gear such that we could tighten the end of the screw onto the flat part of the axle to hold the gear steady on the motor. We also increased the infill of the 3d printed gear to 100% for the inner cylinder (increased from 15% for the rest of the gear and all previous models). With this fix, our gear did not suffer any wear, even when being run at high speeds. All the gears mentioned can be seen below in Figure A1.

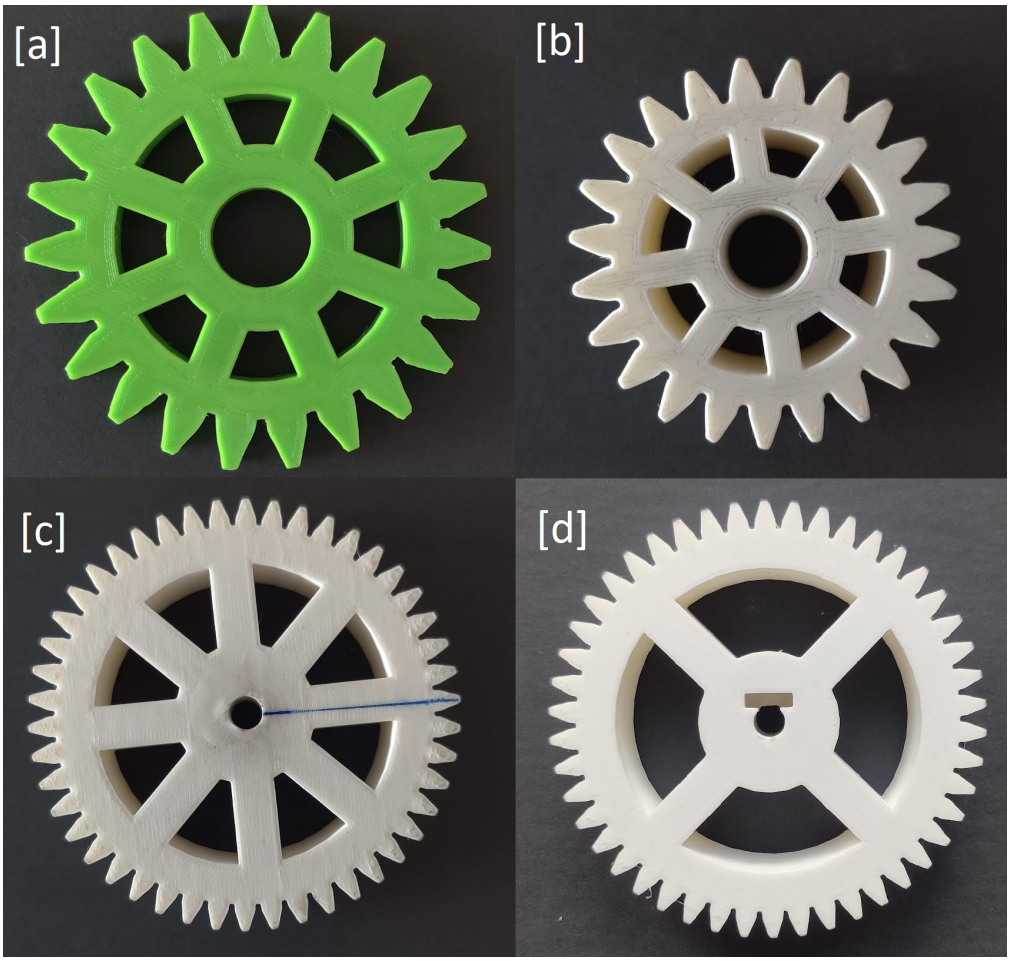

Figure A1: Each different iteration of the gear. From **[a]**, the first model, to **[b]**, the smaller teeth prototype, to **[c]** the large gear with small teeth and finally ending up on **[d]**, the adjustable attachment gear.

The limitation to how fast we could make the rack go then became the torque of the motor. When trying to push the motor to very fast speeds, it simply did not have enough torque to push the rack and stop it at those speeds. Our gear however, was unharmed even when rotating at high speeds.

The gear and rack system was all 3d printed using the 3d printers in The Gearbox. The gear and rack system was all designed by A. Desrochers using AutoCad.

Figure A2: Attachment of the 3D printed gear on to the axle of the motor. The axle has a cylindrical shape with a flat section, on which the screw applies pressure when tightened. There is a hole in the teeth of the gear on the left side of the picture to allow to insert a screwdriver to tighten the screw. This allowed for the gear to be secured in place without getting the plastic damaged from the rotation of the motor at high velocities.

6.2 Figures

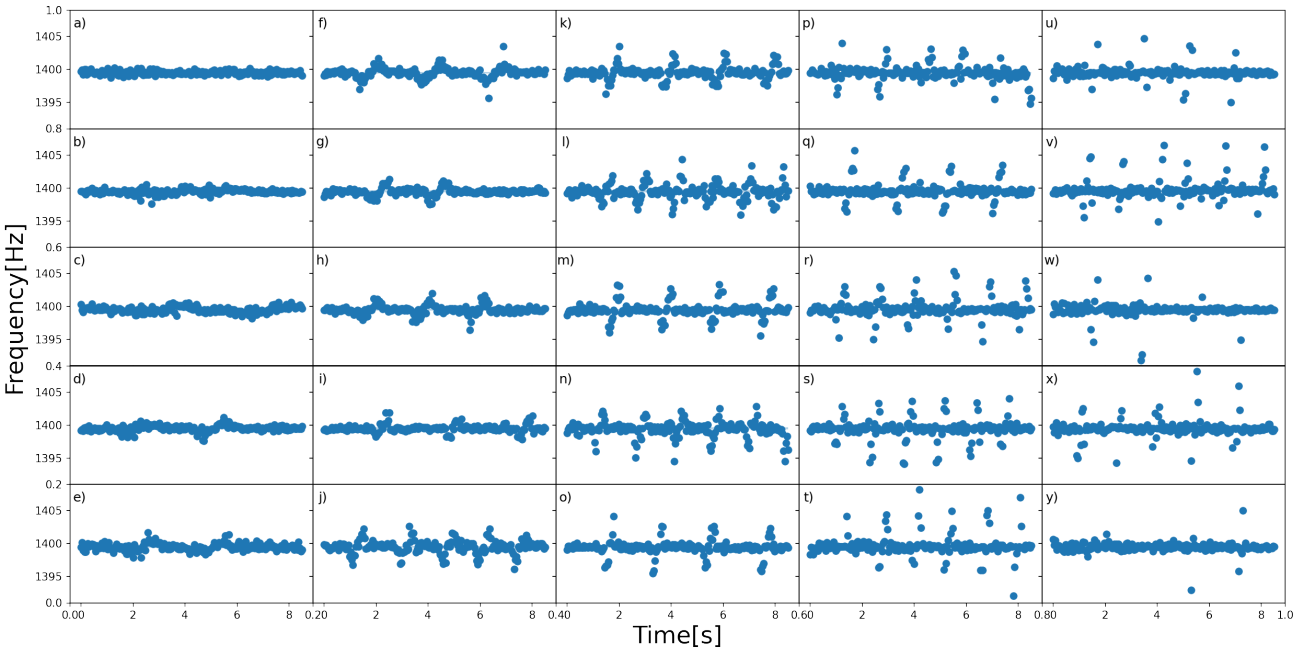

Figure A3: The Frequency recorded over time for varying speeds. We started at a speed (a) 0 turns/second and incremented the speed by 0.25 turns/s until we reached (y) 6 turns/s. The frequency emitted for each case was 1400 Hz. The dips show the corresponding backwards and forwards velocities of the microphone.

6.3 Code

The code for the Python/Arduino pair can all be seen here