CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Supporting Mobile Interaction Using Smartphone Barometer Sensing

A thesis submitted in partial fulfillment of the requirements

For the degree of Master of Science in Computer Science

By

Andrew Miner

December 2024

The thesis of Andrew Miner is approved:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dr. Jeffrey A. Wiegley Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dr. John J. Noga   Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dr. Ani Nahapetian, Chair Date

California State University Northridge

List of Figures

Figure 3.1.1 Initial sound pressure experimental setup. A mobile device is positioned exactly 12 inches away from a speaker. 5

Figure 3.1.2 The mobile device is placed 12 inches from the speaker as recordings are taken. 6

Figure 3.1.3 The mobile device is placed 9 inches from the speaker as recordings are taken. 7

Figure 3.1.4 The mobile device is placed 6 inches from the speaker as recordings are taken. 7

Figure 3.1.5 The mobile device is placed 3 inches from the speaker as recordings are taken. 8

Figure 3.1.6 The mobile device is placed ~0-1 inches from the speaker as recordings are taken. 8

Figure 3.1.7 Results for Sound Pressure Experiment Trail 1 9

Figure 3.1.8 Results for Sound Pressure Experiment Trial 2 10

Figure 3.1.9 Results for Sound Pressure Experiment Trial 3 10

Figure 3.1.10 Results for Sound Pressure Experiment Trial 4 11

Figure 3.1.11 Results for Sound Pressure Experiment Trial 5 11

Figure 3.1.12 Results for Sound Pressure Experiment Trial 6 12

Figure 3.1.13 Results for Sound Pressure Experiment Trial 7 12

Figure 3.1.14 12 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 12 inches from the speaker, from Trials 1 through 7. 13

Figure 3.1.15 9 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 9 inches from the speaker, from Trials 1 through 7. 13

Figure 3.1.16 6 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 6 inches from the speaker, from Trials 1 through 7. 14

Figure 3.1.17 3 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 3 inches from the speaker, from Trials 1 through 7. 14

Figure 3.1.18 ~0-1 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of ~0-1 inch from the speaker, from Trials 1 through 7. 15

Figure 3.2.1 Ambient Pressure Trial 1. This graph shows the ambient pressure recorded from a room closed off to the outside over a 5-minute time interval from 12:30 pm to 12:35 pm. 16

Figure 3.2.2 Ambient Pressure Trial 2. This graph shows the ambient pressure recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm. 17

Figure 3.2.3 Ambient Pressure Trial 3. This graph shows the ambient pressure recorded from a room closed off to the outside over a 1:15 time interval from 2:30 pm to 3:45 pm. 17

Figure 3.2.4 Ambient Pressure Trial 4. This graph shows the ambient pressure recorded from a room closed off to the outside over a 3-hour time interval from 5:30 pm to 8:30 pm. 18

Figure 3.3.1 The bottom center of the front face of the mobile device being tested. 19

Figure 3.3.2 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the bottom center of the front face of the mobile device. 19

Figure 3.3.3 The bottom center of the back face of the mobile device being tested. 20

Figure 3.3.4 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the bottom center of the back face of the mobile device. 21

Figure 3.3.5 The top center of the front face of the mobile device being tested. 21

Figure 3.3.6 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the top center of the front face of the mobile device. 22

Figure 3.3.7 The top center of the back face of the mobile device being tested. 23

Figure 3.3.8 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the top center of the back face of the mobile device. 23

Figure 4.0.1 Android application that replicates the front page of Reddit and provides user interactions through the barometer sensor via blowing on the device. 24

Figure 4.1.1 Code Snippet: Two callback functions that detect the number of blows and provide the blow interaction functionality and are housed in the MainActivity class. 26

Figure 4.1.2 Code Snippet: The second interaction if statement from the MainActivity class’s OnBlowEvent callback functions. This if statement provides the functionality for switching the screen scroll direction after a user blows twice on the device. 27

Figure 4.2.1 Code Snippet: The onSensorChange function from the BlowEventListener class that detects when a user blows on the device via a slope threshold detection method. 28

Figure 4.3.1 A confusion matrix containing the results of the first set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 93.75% accuracy. 29

Figure 4.3.2 A confusion matrix containing the results of the second set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 91.30% accuracy. 30

Figure 4.3.3 A confusion matrix containing the results of the third set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 94.12% accuracy. 31

Figure 4.3.4 A confusion matrix containing the results of all three sets of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 93.15% accuracy. 32

Figure 4.4.1 Graph of the ambient pressure recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm. 33

Figure 4.4.2 Graph of ambient pressure data with low-frequency information filtered out, recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm. 34

Figure 4.4.3 Code Snippet: The apply and calculateAlpha functions within the SensorFilter class that implements the first-order infinite impulse response filter. 35

Figure 4.4.4 Code Snippet: The onSensorChange function from the FilteredBlowListener class that detects when a user blows on the device via a pressure threshold method. 36

Figure 4.5.1 A confusion matrix containing the results of the first set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 93.75% accuracy. 37

Figure 4.5.2 A confusion matrix containing the results of the second set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 95.00% accuracy. 38

Figure 4.5.3 A confusion matrix containing the results of the third set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 91.67% accuracy. 39

Figure 4.5.4 A confusion matrix containing the results of all three sets of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 95.65% accuracy. 40

Table of Contents

[Signature Page ii](#SignaturePage)

[List of Figures iii](#ListOfFigures)

[Abstract vii](#Abstract)

[1. Introduction 1](#_Introduction)

[2. Related Work 3](#_Related_Work)

[3. Experimentation with Barometer Sensor 5](#_Experimentation_with_Barometer)

[3.1 Sound Pressure Experiment 5](#_Sound_Pressure_Experiment)

[3.2 Ambient Pressure Experiment 16](#_Ambient_Pressure_Experiment)

[3.3 Breathalyzer Experiment 19](#_Breathalyzer_Experiment)

[4. Barometer Interface Application 24](#_Barometer_Interface_Application)

[4.1 Blowing Language 25](#_Blowing_Language)

[4.2 Detecting Blows Using a Slope Threshold 27](#_Detecting_Blows_Using)

[4.3 Slope Threshold Evaluation 29](#_Slope_Threshold_Evaluation)

[4.4 Detecting Blows Using a Pressure Threshold 33](#_Detecting_Blows_Using_1)

[4.5 Pressure Threshold Evaluation 37](#_Pressure_Threshold_Evaluation)

[Conclusion 41](#_Conclusion)

[References 43](#_References)

Abstract

Supporting Mobile Interaction Using Smartphone Barometer Sensing

By

Andrew Miner

Master of Science in Computer Science

Smartphone devices have become pervasive over the last decade and are now ubiquitous in everyday life. Modern smartphones support a wide variety of sensors to monitor a device’s motion, position, and various environmental parameters. This thesis explores novel applications of one of these sensors, namely the barometer sensor. A range of experiments were conducted using the built-in barometer sensor of a smartphone. While the barometer sensor is not sensitive enough to detect pressure changes caused by loud audio sources placed within inches of the device, it is sensitive enough to detect atmospheric pressure changes caused by varying temperature and weather conditions. The experiments revealed that the sensor was able to detect when a user blows air over the device using the pressure built up in their lungs.

These experimental results lead to the idea that a practical use case for the barometer sensor may be using its ability to detect gusts of air blown over the device as a form of user input. To test the feasibility of this idea a smartphone application was developed that can detect when a user blows on the device. The application passes pressure data through a high-pass infinite impulse filter and then uses a pressure threshold to detect outliers that signal a gust of air. A blowing language can be created by assigning different numbers of sequential blows to different user actions. This blowing language enables a user to fully control the application using just the air pressure created by their lungs. Testing shows that this user input scheme is reliable and accurate enough to be used practically.

# Introduction

Smartphones have seen a substantial rise in popularity over the last decade. As of a 2023 poll conducted by Pew Research, 90% of Americans and 97% of American adults ages 18 to 49 own a smartphone [10]. These devices have become an invaluable tool for everyday life. As they have grown in popularity, their ability to sense the surrounding environment has become increasingly advanced. It is not uncommon for a modern smartphone to have ten or more sensors to monitor the device’s motion, position, and various environmental parameters. Most smartphones can measure light level, atmospheric pressure, orientation, rotation, and acceleration. Those are just some of the possible measurable parameters. There are likely many novel ways to take advantage of these sensors that have yet to be explored. This thesis aims to demonstrate the extent to which the built-in smartphone barometer sensor can be used in everyday applications.

The digital barometer sensors found in smartphones are micro-electromechanical (MEMs) devices, between 1 and 100 micrometers in size, that measure the local atmospheric pressure. There has been some research about possible applications for smartphone barometer sensors beyond just measuring altitude. Research shows that the sensor can aid GPS navigation, detect which floor a user is on in a building, accurately predict the location of a subway train, aid in screen pressure detection, and even detect tongue gestures using a barometer worn in the ear canal. Barometer sensors have a surprisingly wide range of unique applications. This is especially true when the sensor stays local to a person, like in the case of a smartphone.

For this thesis, a series of experiments were conducted using the built-in barometer sensor of a smartphone to develop an understanding of the barometer sensor’s capabilities and limitations. The first experiment, the Sound Pressure Experiment, is designed to gauge how sensitive the sensor is to sound waves. For this experiment, a Bluetooth speaker is placed at varying distances from the sensor while pressure measurements are taken. The goal of the next experiment, the Ambient Pressure Experiment, is to understand how the ambient pressure fluctuates throughout the day. In this experiment, a smartphone is placed in an isolated room and left on to record pressure data for multiple hours. The basis for the third and final experiment, the Breathalyzer Experiment, is the idea that a barometer in a smartphone might be sensitive enough to detect when a user blows over it using the pressure built up in their lungs. Users are tasked with blowing air over the device in specific areas while pressure measurements are taken.

The results of the last Breathalyzer experiment reveal an opportunity to use a barometer sensor as an input device, enabling users to manipulate an application’s user interface with just their lungs. To explore this opportunity a smartphone application and a blowing language are designed, implemented, and tested. The application mimics the social media platform Reddit, where users can interact with content posted by others. Users can upvote or downvote content. The blowing language provides users with a way to interact with the application using the barometer sensor within their smartphone. By assigning different actions to different numbers of blows in a sequence, users can access the full functionality of the application using just the barometer for input. To test the practicality of this barometer interface three instruction sets are used and tested repeatedly, the results of which are organized into confusion matrices.

Chapter 2 of the thesis dives into the related research about barometer sensors and their unique applications. Chapter 3 covers each experiment, how they are set up, and their resulting data. Chapter 4 details the barometer-controlled smartphone application and discusses its effectiveness as a barometer-based user interface.

Most people spend their daily lives carrying around a smartphone full of different micro-electromechanical sensors. The power of these sensors is often disregarded by software developers. This thesis shows that the barometer sensor alone has several unconventional use cases and can provide an entirely new type of interface. These sensors, including the barometer, offer a plethora of untapped potential and unique applications that have yet to be explored.

# Related Work

As smartphone sensor technology has advanced, there has been an increasing interest in using barometer sensors to supplement traditional localization techniques such as those that use GPS and Wi-Fi. Researchers developed a localization algorithm called B-Loc that builds a barometer fingerprint map of each floor in a building using crowdsourced data from smartphone users. This map enables the researchers to identify which floor a user is on using only the barometer on their smartphone [4]. Similarly, researchers can predict the location of a subway train using only a user’s smartphone barometer with an accuracy of 86%. They achieve this by comparing the elevation changes of a user’s smartphone with the known elevation of each train stop and a series of previously recorded elevation changes [12]. A separate study presents a novel way of improving the vertical positioning accuracy of GPS using the barometer on a smartphone [11]. Dr. Ye et al. created a way to identify a user’s geolocation solely using the barometer sensor in their smartphones. They use a curve fitting-based solution to smooth out the barometer sensor noise created by weather changes, they detect user moving activities with a deep learning restricted Boltzmann machine, and they create signatures of different locations with a clustering-based extraction algorithm. By combining these three components researchers were able to detect user daily locations with an accuracy of 85% while only using 22% of the energy needed for GPS [3].

Similarly to barometer-aided localization, researchers study ways to improve navigation systems using barometers. Researchers Che and Dong propose improving visual-inertial navigation systems by incorporating data from their novel sensor which integrates measurements from a global navigation satellite system and a barometer. They use a non-linear optimization method to couple measurements from their GNSS-Barometer sensor with visual and inertial information to improve real-time navigation. Their results demonstrate a competitive performance increase over existing methods [13]. Researchers Nakanishi, Kanata, and Sawaragi propose a similar method of improving GPS-INS (inertial navigation system) hybrid navigation systems in autonomous helicopters using a barometer sensor. By creating a model that understands the relationship between altitude and the air pressure in ground effect of a helicopter they can effectively estimate the terrain clearance of a helicopter [5].

Researchers also study the potential security applications of a smartphone’s barometer. They can identify when a door or window in a building is opened with 99% accuracy using only the barometer in a smartphone [7]. Researcher Yao et al. show that graphic-pattern-based implicit authentication in a smartphone can be improved with their novel implicit barometer-based authentication system. They argue the APIs provided by smartphone manufacturers to obtain on-screen pressure are not accurate enough to determine users’ touch patterns for authentication purposes. Instead, they use the barometer sensor along with other ambient sensors to more accurately measure on-screen pressure. They demonstrate that by using this method they can achieve a false acceptance rate of 0.45% and a false rejection rate of 0.49%, which greatly exceeds existing solutions [8].

There has also been research into human activity recognition applications of barometer sensors. Researchers use a wrist-worn barometer sensor and subsequent data processing algorithm to measure the hand-vertical travel distance of workers during manual material handling. They aim to create a solution for predicting load vertical locations to reduce the risk of low-back pain [9]. Researcher Chen et al., with the same goal of harm reduction, can detect a user falling to the floor in real-time with a sensitivity of 95.71% using a barometer in combination with an accelerometer [2]. In a much broader study researchers demonstrate the ability to classify eight human activities using inertial sensors and a barometer. First, they classify activities into dynamic or static classifications. Then they further classify these two categories into specific activities. They use three classifiers, including Random Forest and Support Vector Machine. Through experiments, they show that the proposed method is effective and has good performance [6].

Using a low power and low sampling rate barometer worn inside the ear cavity researchers can identify important tongue gestures, such as left, right, and forward. The goal of this study is to demonstrate a low-cost alternative when compared to in-ear microphones for detecting tongue gestures. To save energy, the researchers avoided using frequency domain features for classification, because they involved operations like the Fast Fourier Transform which require a lot of power. Instead, they focus on classifying temporal features which requires using a barometer sensor in both ear canals for better accuracy. Tongue movements typically produce stronger peaks of pressure than head movements, so researchers focus on classifying minimum and maximum samples within a sample window and the covariance of samples between the two ear canals within the same sample window. To classify these samples Error Correcting Codes based on binary Support Vector Machines (SVM), K-Nearest Neighbor (KNN), and Decision Trees (DT) were used. Using these methods tongue gestures could be classified with an accuracy of 94% for the SVM, 91% for KNN, and 89% for the Decision Tree classifier [1].

# Experimentation with Barometer Sensor

To develop an understanding of the capabilities and limitations of a smartphone’s built-in barometer sensor three experiments were conducted, namely the Sound Pressure Experiment, the Ambient Pressure Experiment, and the Breathalyzer Experiment.

## Sound Pressure Experiment

A sound pressure experiment was conducted to test the hypothesis that the barometer in a mobile device is sensitive enough to detect how far away a sound source is. The hypothesis is that as the mobile device gets closer to the sound source the pressure measurements taken by the barometer increase.

To begin the experiment a mobile phone is placed 12 inches away from a speaker, as seen in Figure 3.1.1. While the speaker is playing a 1 kHz tone the mobile phone records pressure values taken by the barometer. After recording 30 seconds of pressure data, the recording is stopped and the phone is moved up to 9 inches from the speaker. This process is repeated, moving the mobile device 3 inches closer to the speaker after each recording until the device and the speaker are almost touching. Figures 3.1.2 - 3.1.6 show how recordings are taken at 12 inches, 9 inches, 6 inches, 3 inches, and ~0-1 inch.

Figure 3.1.1 Initial sound pressure experimental setup. A mobile device is positioned exactly 12 inches away from a speaker.

Steps were taken to minimize the ambient pressure variance within the room where recordings were taken. All doors and windows to the recording room were closed and remained closed throughout the experiment. All fans and heating/air conditioning units were turned off for the duration of the experiment. After the phone was moved to a new location the observer waited approximately one minute before starting the recording to allow time for the pressure changes caused by the move to normalize.

This experiment was repeated in seven different trials, the results of which are shown in Figures 3.1.2 - 3.1.6. When comparing each trial there is no observable pattern shared between them. Trials 1 (Figure 3.1.7), 4 (Figure 3.1.10), 6 (Figure 3.1.12), and 7 (Figure 3.1.13) all show a pressure decrease as the mobile device gets closer to the phone. Trial 2, shown in Figure 3.1.8, shows almost no change in the measured pressure. Trial 3, shown in Figure 3.1.9, shows chaotic changes in pressure with multiple increases and decreases throughout the trial. Trail 5, shown in Figure 3.1.11, shows a large dip in pressure which then returns to the initial values recorded at the start of the trial.



Figure 3.1.2 The mobile device is placed 12 inches from the speaker as recordings are taken.



Figure 3.1.3 The mobile device is placed 9 inches from the speaker as recordings are taken.



Figure 3.1.4 The mobile device is placed 6 inches from the speaker as recordings are taken.



Figure 3.1.5 The mobile device is placed 3 inches from the speaker as recordings are taken.



Figure 3.1.6 The mobile device is placed ~0-1 inches from the speaker as recordings are taken.

The wide variety of results and the lack of a significant pattern suggest that the initial hypothesis is incorrect, and the barometer sensor in the tested mobile device is not sensitive enough to determine the distance between it and a sound source.

The data of these seven trials was further analyzed to determine why the results are so inconsistent. Each of the seven trials was recorded consecutively one after the other on the same day, starting with Trial 1 and ending with Trial 7. This fact makes it possible to extract data from each trial and build a history of the pressure in the recording room throughout all seven trials.  To do this the recordings are grouped by the distances they were recorded at and their average values are graphed. Figure 3.1.14 shows one of these graphs which consists of all recordings taken at a distance of 12 inches. All of these distance-specific graphs, shown in Figures 3.1.14 - 3.1.18, show that throughout the experiment the ambient pressure of the recording room was slowly dropping. Steps were taken throughout the experiment to minimize any possible change in pressure caused by the observer, which suggests that this constant pressure drop is caused by the weather outside of the recording room which cannot be controlled.

This analysis shows that, without a way of mitigating changes to the ambient pressure caused by the weather, it is not feasible to measure the distance between a sound source and a mobile device using only the barometer in the mobile device.

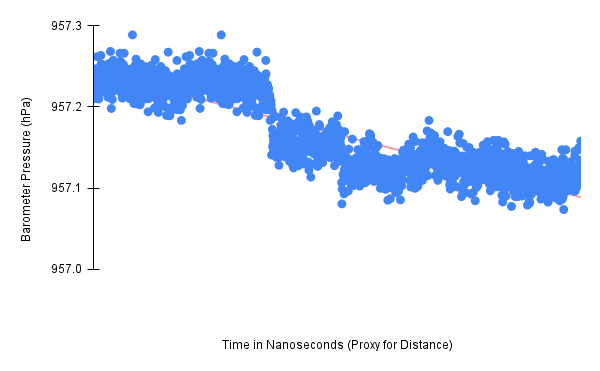


Figure 3.1.7 Results for Sound Pressure Experiment Trail 1

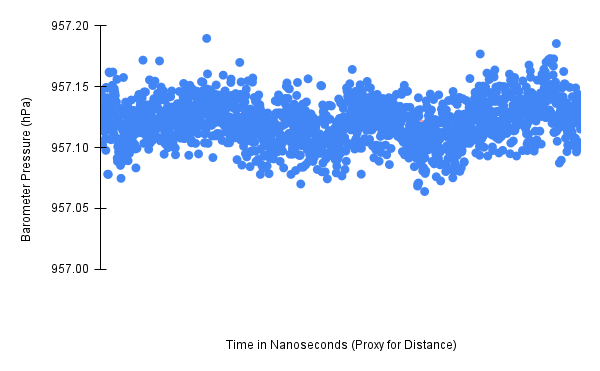


Figure 3.1.8 Results for Sound Pressure Experiment Trial 2

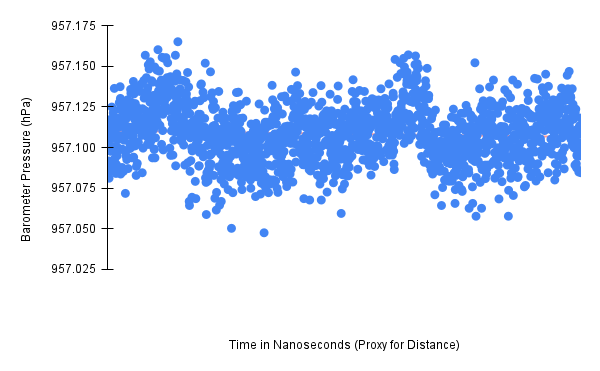


Figure 3.1.9 Results for Sound Pressure Experiment Trial 3

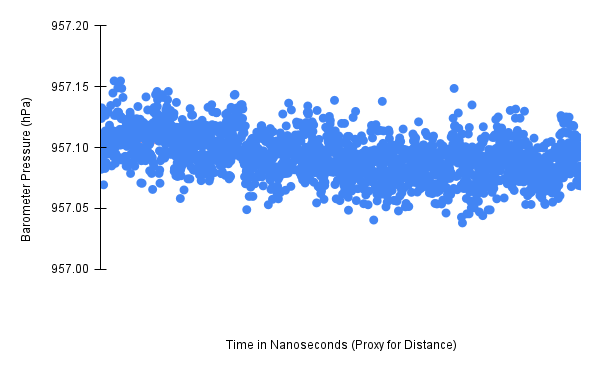


Figure 3.1.10 Results for Sound Pressure Experiment Trial 4

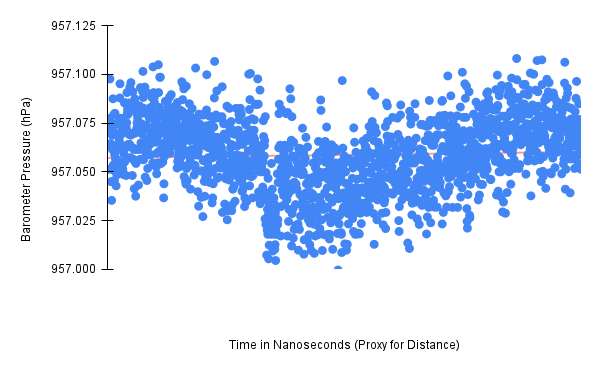


Figure 3.1.11 Results for Sound Pressure Experiment Trial 5

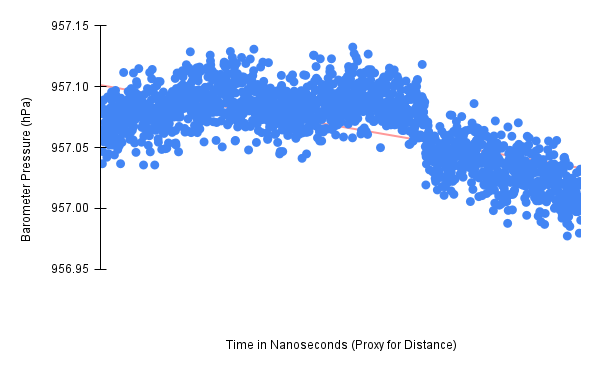


Figure 3.1.12 Results for Sound Pressure Experiment Trial 6

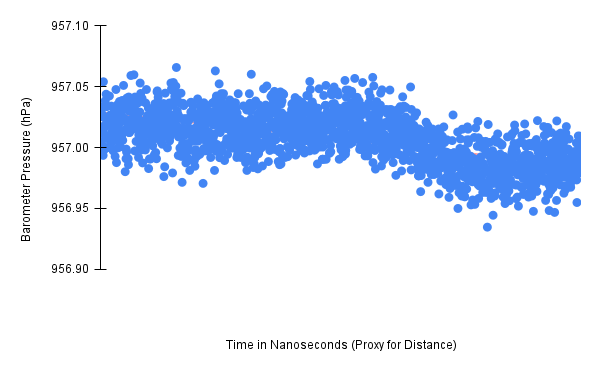


Figure 3.1.13 Results for Sound Pressure Experiment Trial 7

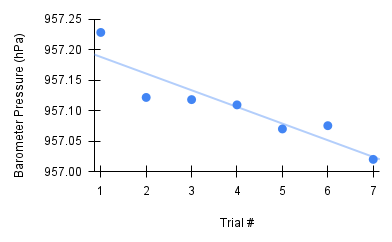


Figure 3.1.14 12 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 12 inches from the speaker, from Trials 1 through 7.

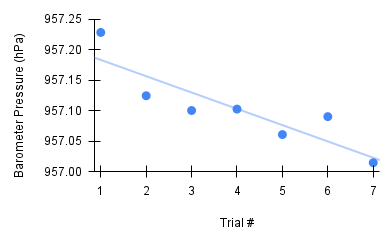


Figure 3.1.15 9 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 9 inches from the speaker, from Trials 1 through 7.

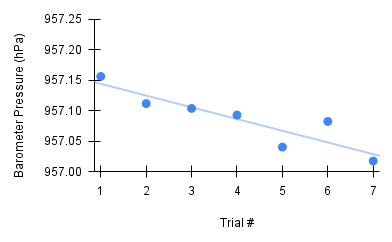


Figure 3.1.16 6 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 6 inches from the speaker, from Trials 1 through 7.

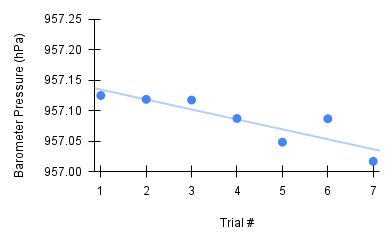


Figure 3.1.17 3 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of 3 inches from the speaker, from Trials 1 through 7.

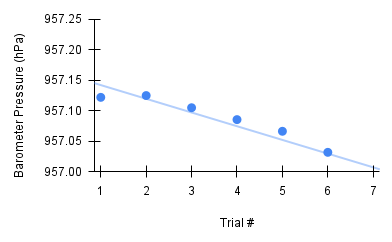


Figure 3.1.18 ~0-1 Inch Barometer Pressure Values. This graph shows the average pressure of all recordings taken at a distance of ~0-1 inch from the speaker, from Trials 1 through 7.

## Ambient Pressure Experiment

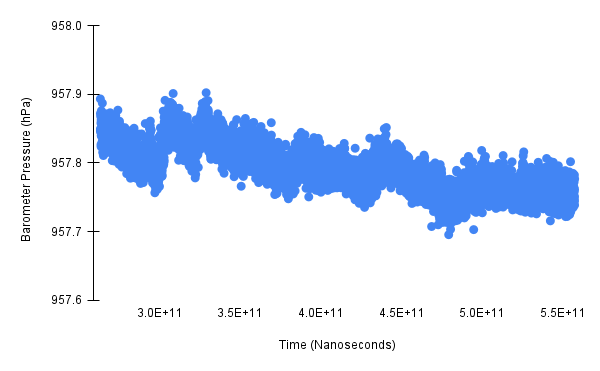
 To verify that the ambient pressure is changing throughout the day at a rate that interferes with possible barometer applications. Multiple pressure recordings were made in intervals ranging from 5 minutes to 3 hours.

Figure 3.2.1 Ambient Pressure Trial 1. This graph shows the ambient pressure recorded from a room closed off to the outside over a 5-minute time interval from 12:30 pm to 12:35 pm.

For the first trial, the mobile device was left in a closed-off room with no windows for 5 minutes from 12:30 pm to 12:35 pm. The mobile device was placed onto a flat surface and set to record the barometer pressure throughout the 5-minute interval. Figure 3.2.1 shows the results of this trial. Over a very short time interval, the barometer pressure measurements showed a downward trend, dropping by approximately 0.1 hPa.

For the second trial, the mobile device was left in a closed-off room with no windows for 30 minutes from 1:00 pm to 1:30 pm. The mobile device was placed onto a flat surface and set to record the barometer pressure throughout the 30-minute interval. As seen in Figure 3.2.2, over the 30-minute interval the barometer pressure measurements trended down, dropping by a total of approximately 0.5 hPa.

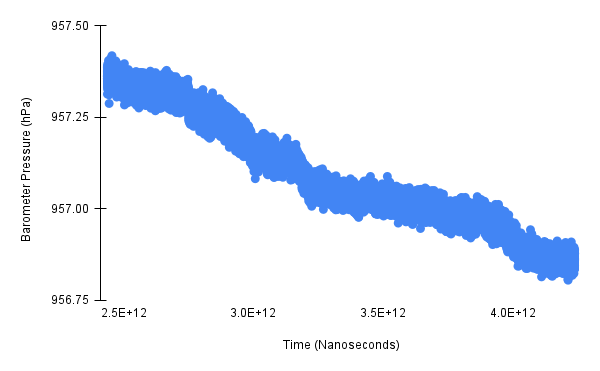
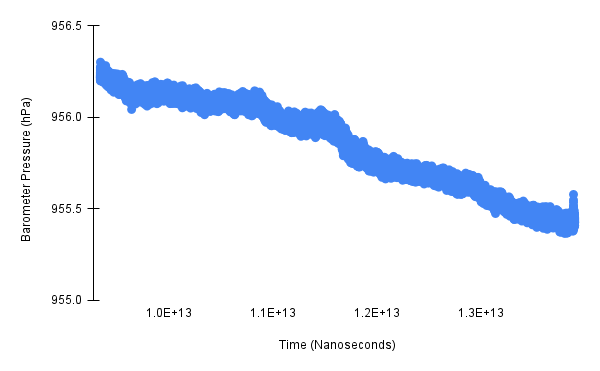
For the third trial, the mobile device was left in a closed-off room with no windows for 1 hour and 15 minutes from 2:30 pm to 3:45 pm. The mobile device was placed onto a flat surface and set to record the barometer pressure throughout the 75-minute interval. The results shown in Figure 3.2.3 match the results from the previous two trials. The barometer measurements continued to trend down, losing approximately 0.8 hPa over the 75-minute time interval.

Figure 3.2.2 Ambient Pressure Trial 2. This graph shows the ambient pressure recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm.

Figure 3.2.3 Ambient Pressure Trial 3. This graph shows the ambient pressure recorded from a room closed off to the outside over a 1:15 time interval from 2:30 pm to 3:45 pm.



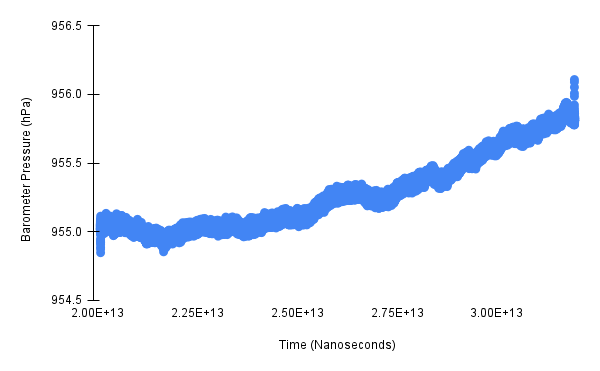
For the fourth trial, the mobile device was left in a closed-off room with no windows for three hours from 5:30 pm to 8:30 pm. The mobile device was placed onto a flat surface and set to record the barometer pressure throughout the 3-hour interval. The results shown in Figure 3.2.4 differ slightly from the last three trials. Instead of showing a downward trend, this trial shows an upward trend, with the barometer measurements increasing by approximately 0.8 hPa over the 3-hour time interval.

Figure 3.2.4 Ambient Pressure Trial 4. This graph shows the ambient pressure recorded from a room closed off to the outside over a 3-hour time interval from 5:30 pm to 8:30 pm.

Trails one through three were taken during the early afternoon as the ambient temperature outside was increasing. The downward trend of these three trials suggests that as the outside temperature increases it causes the newly warmed air to rise into the atmosphere, reducing the pressure close to the ground. Trial 4 was taken in the early evening as the ambient temperature outside was decreasing. The upward trend of this trial supports the conclusions made about the previous three trials. This trial suggests that as the outside temperature decreases the newly cooled air falls towards the ground, increasing the ambient pressure measure on the ground. Despite each trial being isolated to a closed-off room with no windows, the changes in temperature and pressure outside significantly impact the pressure measurements of each trial. This analysis confirms that changes in the ambient pressure caused by the weather must be taken into consideration when designing possible use cases for a barometer sensor.

## Breathalyzer Experiment

The bottom center of the front face of the mobile device being tested.
In this section a breathalyzer experiment is conducted to test the barometer's sensitivity and determine if it could function as an input device. The mobile device was held out in front of the user's face and the user was instructed to blow with as much force as possible onto specific areas of the device.

Figure 3.3.1 The bottom center of the front face of the mobile device being tested.

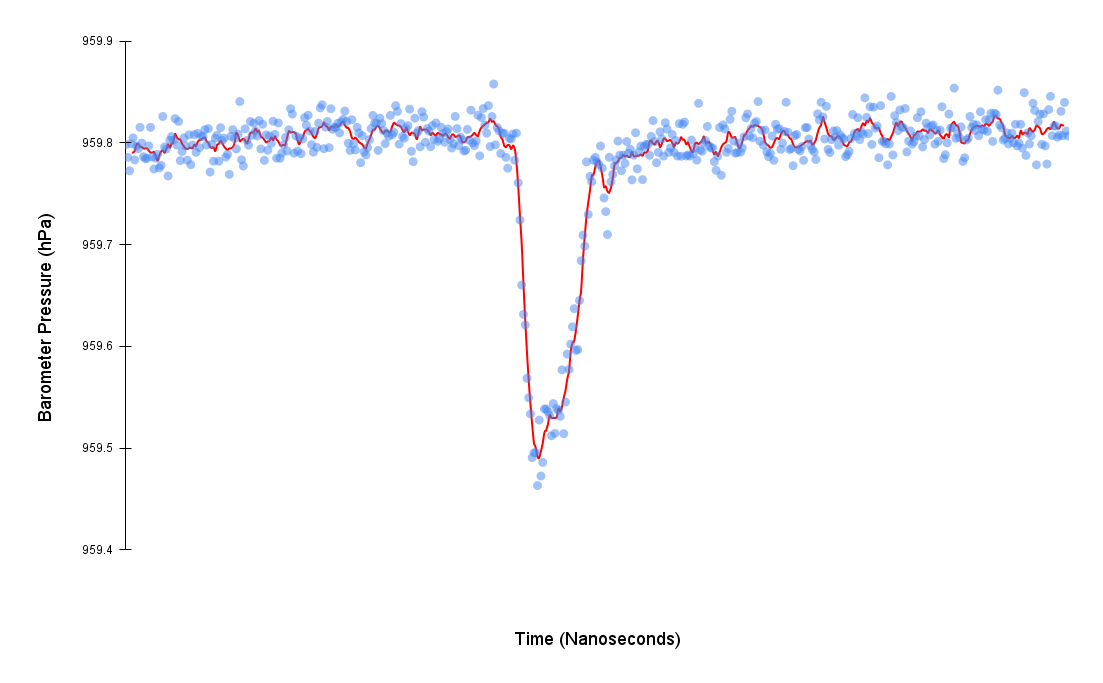


Figure 3.3.2 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the bottom center of the front face of the mobile device.

The experimental setup for the first trial of these experiments had the user hold the phone approximately four inches in front of their mouth. The user was then instructed to blow on the bottom center of the front face of the mobile device, as seen in Figure 3.3.1, with as much force as possible for as long as possible.

As shown in Figure 3.3.2, when the user blew on the device the average barometer value dropped significantly from approximately 959.8 hPa to below 959.5 hPa.

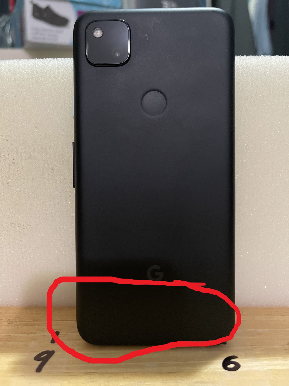
The drop observed in this trial is far beyond the regular variance of the barometer sensor’s measurements and can be attributed to the user’s blowing action. Although the hypothesis was that the pressure measured by the barometer would show an increase from the user blowing air over it, the opposite result was observed. Following this first trial, it is believed that these results are caused by the venturi effect dropping the pressure near the sensor as air blows rapidly around the device.

Figure 3.3.3 The bottom center of the back face of the mobile device being tested.

The experimental setup for the second trial had the user hold the phone approximately four inches in front of their mouth. The user was then instructed to blow on the bottom center of the back face of the mobile device, as seen in Figure 3.3.3, with as much force as possible for as long as possible.

As shown in Figure 3.3.4, when the user blew on the device the results observed were very similar to the first trial. Like the first trial, the pressure fell from an average of 959.8 hPa to below 959.5 hPa, a significant decrease.

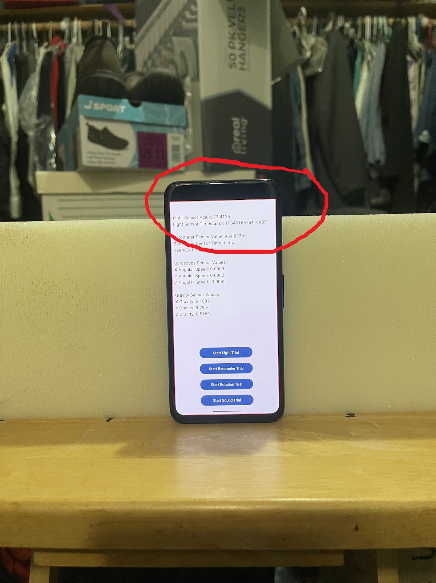
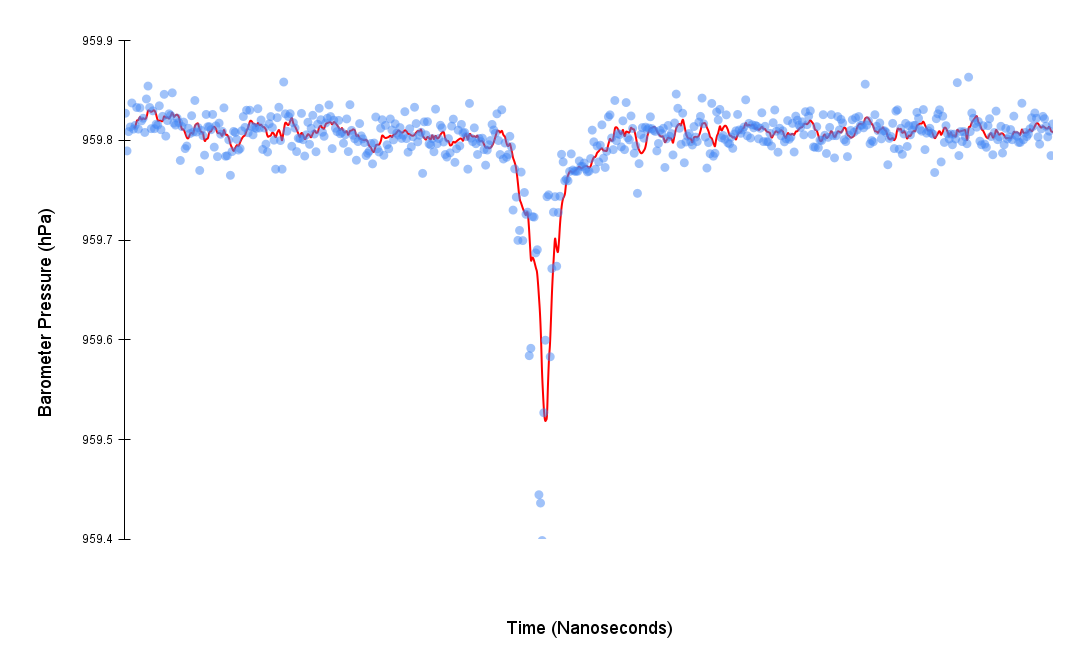
The drop observed in this second trial is also beyond the regular variance of the barometer sensor’s measurements and can be attributed to the user’s blowing action. Keeping in line with the first trial, the pressure measured during this second trial dropped instead of increasing. These results reinforce the hypothesis that the venturi effect is causing the pressure measured to drop. Having the user blow on the front versus the back of the phone appears to make little to no difference in the magnitude of the pressure dropped.

Figure 3.3.4 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the bottom center of the back face of the mobile device.

Figure 3.3.5 The top center of the front face of the mobile device being tested.

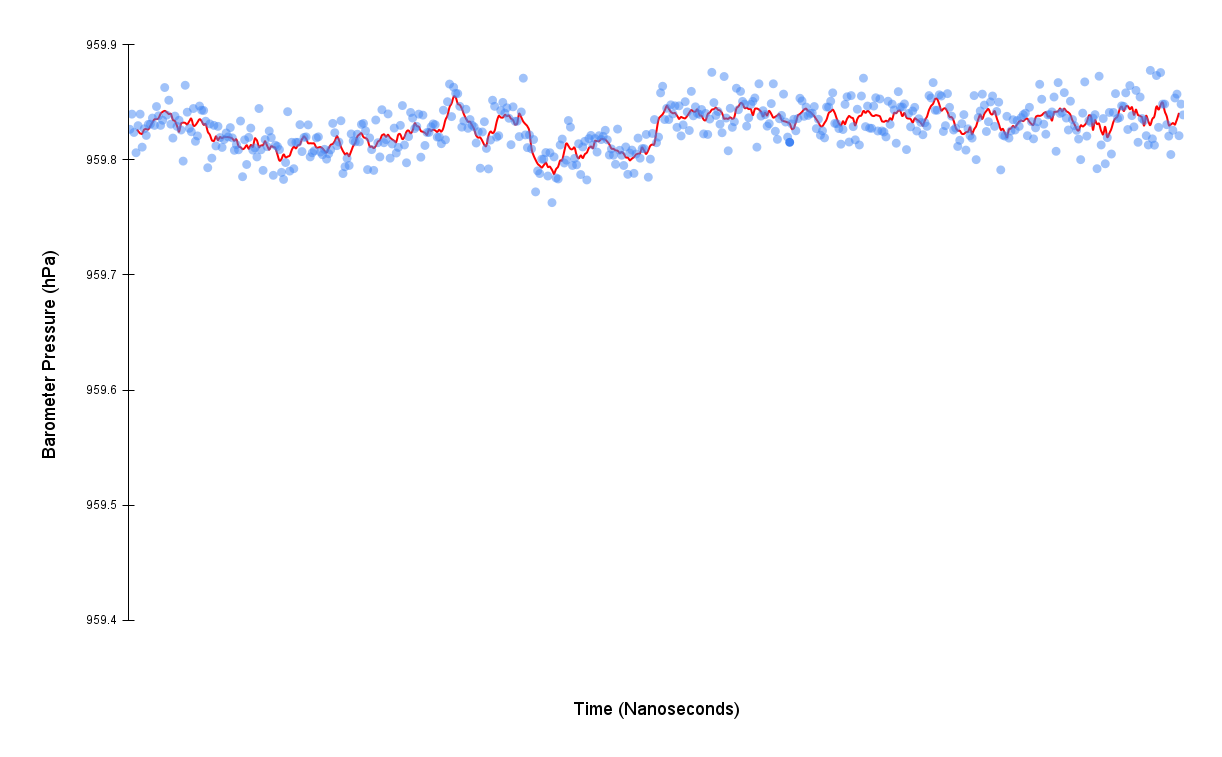
The experimental setup for the third trial had the user hold the phone approximately four inches in front of their mouth. The user was then instructed to blow on the top center of the front face of the mobile device, as seen in Figure 3.3.5, with as much force as possible for as long as possible.

Figure 3.3.6 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the top center of the front face of the mobile device.

Figure 3.3.6 shows that, unlike the first two trials, the barometer measured a much less significant drop in pressure while the user was blowing. Graphing the trend line of the pressure values shows that there was a decrease in the average measurement, but it was less than a 0.1 hPa difference.

The drop observed in this third trial appears to be beyond the regular variance of the barometer sensor’s measurements but not by a significant margin. This suggests that the barometer sensor is located in the bottom half of the mobile device.

The experimental setup for the fourth trial had the user hold the phone approximately four inches in front of their mouth. The user was then instructed to blow on the top center of the back face of the mobile device, as seen in Figure 3.3.7, with as much force as possible for as long as possible.

Figure 3.3.8 shows that, just like the third trial, the barometer measured a small drop in pressure while the user was blowing. This trial matched the results seen in the third trial. The barometer values did drop below the average ambient pressure but by a very small amount, less than 0.1 hPa.

The results of these four trials show that the barometer sensor is sensitive enough to be used as an input device. The difference between the results in the first two trials and the last two trials shows that blowing toward the bottom of the device gives more significant values. These results suggest the barometer sensor is located in the bottom half of the mobile device. These results also show that the initial hypothesis that a pressure increase would be observed is wrong. This is likely due to the venturi effect causing the pressure near the barometer sensor to drop.

Figure 3.3.7 The top center of the back face of the mobile device being tested.

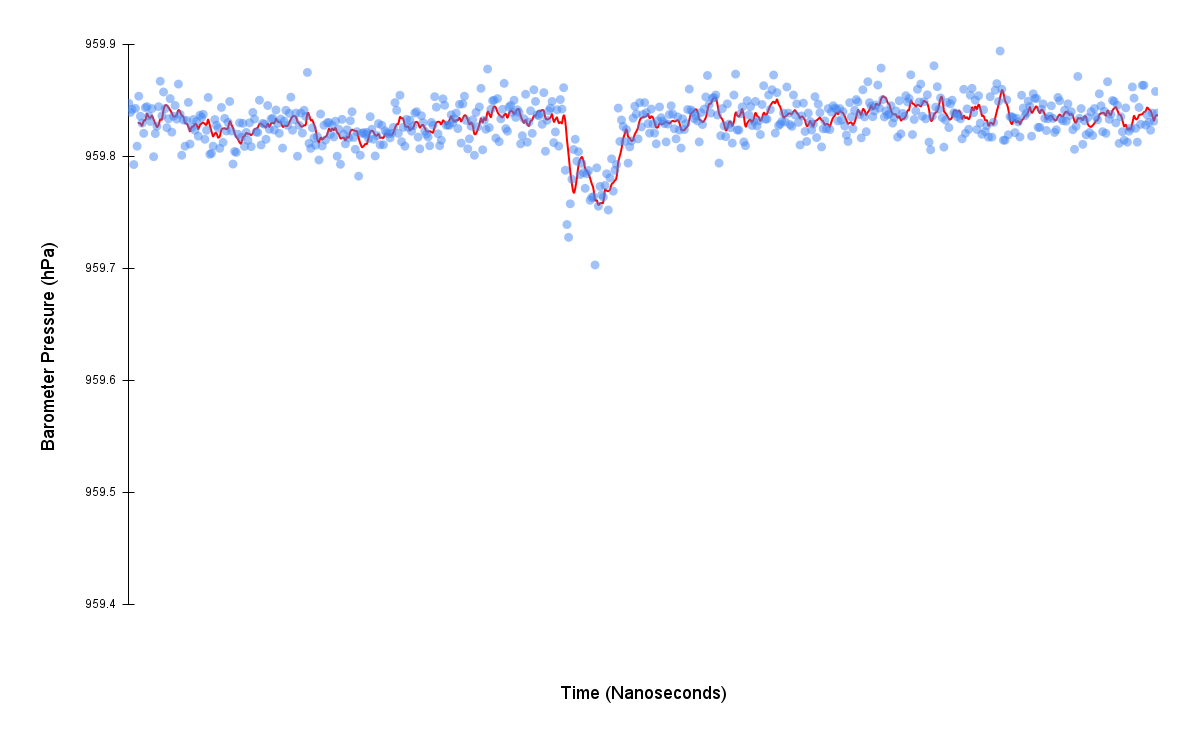


Figure 3.3.8 Barometer Pressure vs. Time. This chart shows a drop in the pressure recorded while blowing on the top center of the back face of the mobile device.

# Barometer Interface Application

The experimental results demonstrate that the most practical approach to support mobile interaction using the barometer is to use it as a button-style input device. The breathalyzer experiment in Section 3.3 shows that the barometer can sense when a user is blowing on a mobile device. Treating a single blow as a simulated button press allows users to interact with mobile devices using just the pressure built up in their lungs.

A small blowing language can be constructed to facilitate this kind of mobile interaction. Different numbers of consecutive blows within a short time frame, analogous to button taps, can be assigned to different functionality within a mobile application.

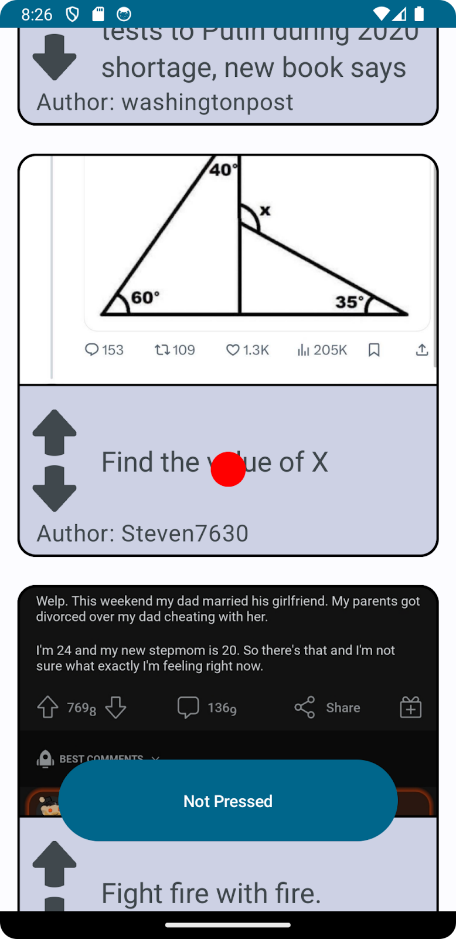
To test the practicality of this approach an Android application was written to simulate the front page of the website Reddit.com. The app uses Reddit.com’s API to request the first 100 posts and displays them in a scrollable list. Each post consists of a title, an upvote button, a downvote button, and an image if one is included in the post. In a fixed position in the center of the screen, there is a red cursor that is used to determine the post with which to interact. The app includes a button at the bottom of the screen for testing purposes, which can be used to simulate blowing on the barometer.

Figure 4.0.1 Android application that replicates the front page of Reddit and provides user interactions through the barometer sensor via blowing on the device.

## Blowing Language

A blowing language was created for input. The blowing language consists of four distinct on-screen actions, using one to four blows in a sequence. The four input options provided by this blowing language allow for full control of the developed Android app. The on-screen actions map to the blowing actions in the following way:

1. One long continuous blow scrolls the screen for the duration of the blow.
2. Two blows within 500 ms of each other switches the direction the screen scrolls. By combining these first two input options the user can scroll up and down on the screen with just blowing.
3. Three blows, with each blow being within 500 ms of the previous blow, toggles the upvote button of the post underneath the red cursor. If the upvote button is unselected and grayed out, three blows will select it, turning it red. If the upvote button is red, three blows will deselect it, turning it gray. Regardless of the state of the upvote button, if the downvote button is blue, three blows will deselect it turning it gray.
4. Four blows, with each blow being within 500 ms of the previous blow, toggles the downvote button of the post underneath the red cursor. If the downvote button is unselected and grayed out, four blows will select it, turning it blue. If the downvote button is blue, four blows will deselect it, turning it gray. Regardless of the state of the downvote button, if the upvote button is red, four blows will deselect it turning it gray.

The application is structured around two classes, the MainActivity class, which contains the core logic, and the BlowEventListener class, which detects when a user blows on the device. To create the blowing language functionality two functions were created in the MainActivity class and registered to the BlowEvenListener class’s list of event callbacks. Figure 4.1.1 shows a truncated version of these two functions.

When an OnBlowEvent is triggered the MainActivity class checks if a previously detected blow exists. If it does exist then its associated ending timestamp is compared with the starting timestamp of the current blow to determine if they fall within a specified time window, the blowGapMS variable in Figure 4.1.1. If they do fall within that time window they are considered part of the same blow sequence. The current blow is then stored to an array of associated blows, the variable blows in Figure 4.1.1.

The variable blows is an array containing instances of the BlowEvent data class. The BlowEvent data class serves as a pair containing the starting timestamp of the blow and the ending timestamp of the blow. When the blow is initially stored in the blows array during the OnBlowEvent callback the ending timestamp is not known, so a value of -1 is stored in its place. When the OnReleaseEvent callback is triggered the ending timestamp of the last recorded blow is updated from -1 to the current timestamp.

A screen shot of a computer screen

Description automatically generated

Figure 4.1.1 Code Snippet: Two callback functions that detect the number of blows and provide the blow interaction functionality and are housed in the MainActivity class.

In Figure 4.1.1, the collapsed if statements provide the functionality of the blowing language. There is an if statement for each of the four possible blow interactions. One collapsed if statement is in the OnBlowCallback function and the other three collapsed if statements are in the OnReleaseCallback function. The reason one if statement is separated from the rest is that it implements the scrolling functionality, which takes place during the duration of a user’s blow, not after it like the three other interface interactions.

Figure 4.1.2 expands one of the collapsed if statements from Figure 4.1.1. When the number of blows is reached to begin a blowing interaction an associated coroutine is launched. This coroutine first waits for the blowGapMS delay allowing the user to input more blows and change which interaction they want. After the blowGapMS delay is waited, the coroutine checks to ensure that no more blows were inputted and that it is still the interaction intended by the user. It does this by first checking that the size of the blows array still matches the number of blows associated with the interaction, two in the case of Figure 4.1.2. Then it checks if enough time has passed, blowGapMS in Figure 4.1.2, since the end of the last blow to ensure that the blow data held in the blows array is the same data that was there when the coroutine launched. The other if statements shown in Figure 4.1.1 all have their own coroutines that perform the same steps as Figure 4.1.2. However, the coroutine for scrolling the screen is a special case. It does an additional check to ensure that the end timestamp of the last blow in the blows array is still equal to -1. It does this because the screen should only scroll while the user is actively scrolling. In addition to this, instead of comparing the time of the end of the last blow to the blow gap, it compares the time of the start of the last blow. Each of these coroutines implements their associated interaction, giving A computer screen with text and numbers

Description automatically generatedthe user full access to every interaction in the application through just blowing.

Figure 4.1.2 Code Snippet: The second interaction if statement from the MainActivity class’s OnBlowEvent callback functions. This if statement provides the functionality for switching the screen scroll direction after a user blows twice on the device.

## Detecting Blows Using a Slope Threshold

The BlowEventListener class uses the device’s barometer to sense one or more blows. This class implements the SensorEventListener interface provided by the Android API. Every time the value read by the barometer sensor changes, the function onSensorChanged within the BlowEventListener class gets called. The BlowEventListener does not provide the blowing language functionality. Instead, it offers two callbacks based on the state of the barometer. When a blow is sensed the BlowEventListener will call all the callbacks registered to its OnBlowEvent, which is done via the registerOnBlowCallback function. When a previously sensed blow is finished the BlowEventListener will call all the callbacks registered to its OnReleaseEvent, which is done via the registerOnReleaseCallback function.

To detect when a blow happens the onSensorChanged function needs to be able to detect data outliers within the stream of barometer pressure data. As seen in previous experiments, the barometer sensor has a wide variance in the pressure it measures, and the ambient pressure changes throughout the day. To account for these two facts, a slope threshold method is used to detect outliers. In this method, the slope between the current pressure measurement and the previous pressure measurement is calculated. If that calculated slope is larger than a predetermined threshold it is considered an outlier and part of a blow. This method accounts for the variance in the sensor by using a threshold larger than the average slope caused by the variance in the measurements. Additionally, this method is not sensitive to gradual ambient pressure changes.

A single blow consists of multiple outlier data points. To be considered an outlier the absolute value of the slope between the current data point and the previous data point must be greater than a predetermined threshold, the variable anomalyThreshold in Figure 4.2.1. Through trial and error, a value of 1.5x10^-9 \* barScalar was chosen for the anomaly threshold. The variable barScalar is used to scale all the recorded data to prevent precision loss. Outliers detected within a specified time window, the variable blowWindowNS in Figure 4.2.1, are considered part of the same blow. Through trial and error, a blow window of 450 ms was chosen. To reduce false positives an OnBlowEvent is only triggered after three outlier data points have been collected. If three or more data points have been collected, and the 450 ms time frame has elapsed since the last outlier was detected, the OnReleaseEvent is triggered.

Figure 4.2.1 Code Snippet: The onSensorChange function from the BlowEventListener class that detects when a user blows on the device via a slope threshold detection method.

## Slope Threshold Evaluation

The slope threshold outlier detection implementation of the application was tested using three sets of repeatable instructions, the results of which are tabulated into multiple confusion matrices. Each set of instructions was performed three separate times.

The first set of instructions consists of the following steps:

1. Scroll the screen down three posts.
2. Switch the scroll direction.
3. Scroll the screen up three posts.
4. Upvote the post under the cursor.
5. Downvote the post under the cursor.

A screenshot of a graph

Description automatically generated

Figure 4.3.1 A confusion matrix containing the results of the first set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 93.75% accuracy.

Figure 4.3.1 shows the results of this first set of instructions. These results show a 100% accuracy for the screen scrolling, scrolling direction, and downvote post features, and a 75% accuracy for the upvote post feature. The application was able to accurately predict 15 out of 16 of the user’s actions, resulting in an accuracy of 93.75%.

The second set of instructions consists of the following steps:

1. Upvote the post under the cursor.
2. Scroll the screen down six posts.
3. Switch the scroll direction.
4. Upvote the post under the cursor.
5. Scroll the screen up three posts.
6. Downvote the post under the cursor.

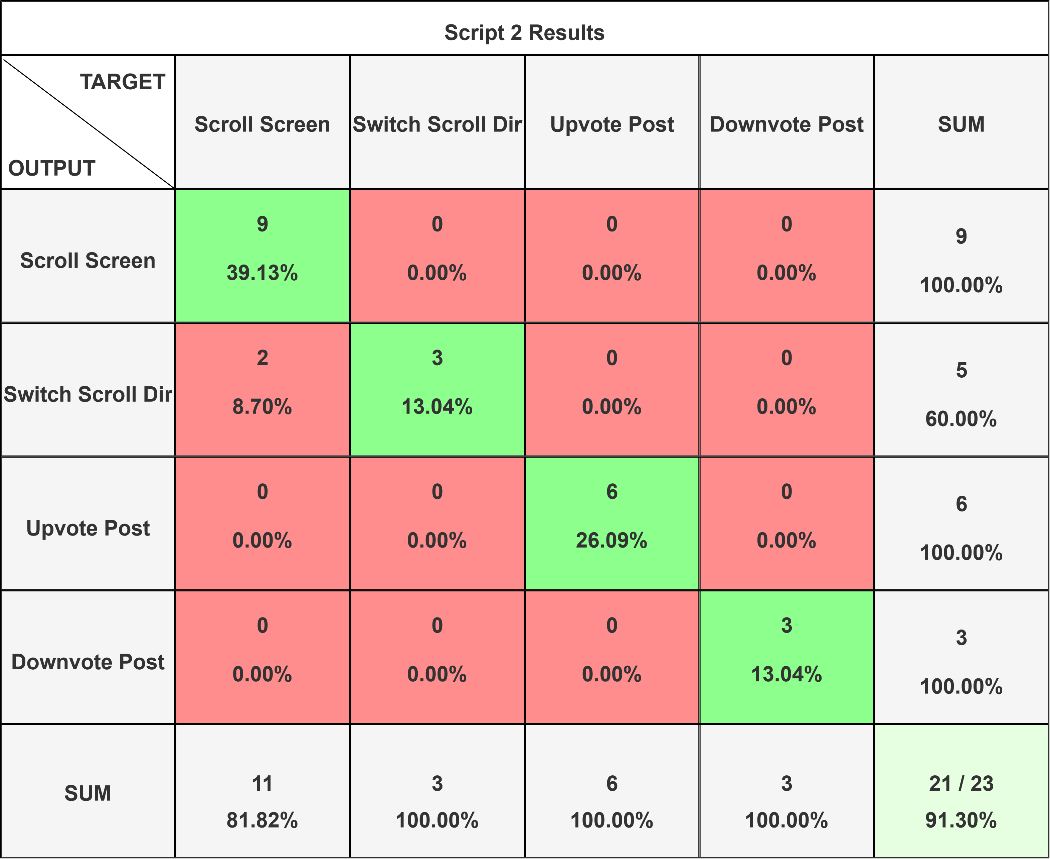


Figure 4.3.2 A confusion matrix containing the results of the second set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 91.30% accuracy.

Figure 4.3.2 shows the results of this second set of instructions. These results show a 100% accuracy for the scrolling direction, upvote post, and downvote post features, and an 81.82% accuracy for the screen scrolling feature. The application was able to accurately predict 21 out of 23 of the user’s actions, resulting in an accuracy of 91.30%.

The third set of instructions consists of the following steps:

1. Scroll the screen down two posts.
2. Downvote the post under the cursor.
3. Scroll the screen down two posts.
4. Downvote the post under the cursor.
5. Switch the scroll direction.
6. Scroll the screen up three posts.
7. Upvote the post under the cursor.
8. Switch the scroll direction.
9. Scroll the screen down one post.
10. Upvote the post under the cursor.

A screenshot of a graph

Description automatically generated

Figure 4.3.3 A confusion matrix containing the results of the third set of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 94.12% accuracy.

Figure 4.3.3 shows the results of this second set of instructions. These results show a 100% accuracy for the scrolling direction and upvote post features. The scrolling screen feature had an accuracy of 93.33%. The accuracy for the downvote post feature was 85.71%. The application was able to accurately predict 32 out of 34 of the user’s actions, resulting in an accuracy of 94.12%.

A screenshot of a graph

Description automatically generated

Figure 4.3.4 A confusion matrix containing the results of all three sets of instructions tested on the slope threshold outlier detection implementation. The matrix shows a 93.15% accuracy.

Figure 4.3.4 shows the results of all three sets of instructions combined. With an accuracy of 90.63%, the scrolling screen feature has the lowest accuracy of all the features. This is to be expected because the slope threshold outlier detection method used by the BlowEventListener class, described in section 4.2, is not good at detecting long continuous blows. A long-duration blow will have a steep slope at the start and end of the blow, but during the middle of the blow the slope will be approximately 0. These 0-value slopes fail to trigger the threshold, causing a long-duration blow to be treated as multiple different shorter blows. The switch scrolling direction feature had the highest accuracy of all the features, 100%. This is also an expected result. The switch direction feature is the easiest feature to detect. It consists of the fewest number of blows in a sequence and does not rely on long-duration blows like the screen scrolling feature. The upvote and downvote features have similar accuracy ratings. The upvote feature has a rating of 93.75%, and the downvote feature has a rating of 92.31%. User error caused these ratings to fall below 100%. Each feature only missed predicting one instance, which was caused by the user failing to chain blows in a sequence within the correct time interval. The application was able to accurately predict 68 out of 73 of the user’s actions, resulting in an accuracy of 93.15%. These results show that this user input scheme is reliable and accurate enough to be used practically.

## Detecting Blows Using a Pressure Threshold

Using a slope threshold method to detect outliers in the barometer data is effective enough to detect when a user blows air across a smartphone. However, the technique suffers from an inability to accurately detect blows that persist over a long duration. In the middle of a long duration blow the slope between data points is near 0 and fails to trigger the slope threshold. This causes the BlowEventListener outlier detection code in Figure 4.1 to sometimes treat a long duration blow as multiple distinct blows. In addition to this issue, the slope threshold detection method requires the user to blow with a large amount of force to trigger the detection, making it inconvenient to use.

To fix these issues and improve upon the application an alternative pressure threshold method is used. The core idea of this method is to use a threshold based on the current pressure value measured by the barometer instead of the slope between the last two measurements. This method was initially passed over in favor of the slope threshold method because it is sensitive to ambient pressure changes caused by external variables like the weather. Unlike the slope threshold method, this method does not inherently scale as the ambient pressure changes. To make this method practical, the ambient pressure changes must be filtered out, allowing for the use of a constant pressure threshold.

A blue line with white text

Description automatically generated

Figure 4.4.1 Graph of the ambient pressure recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm.

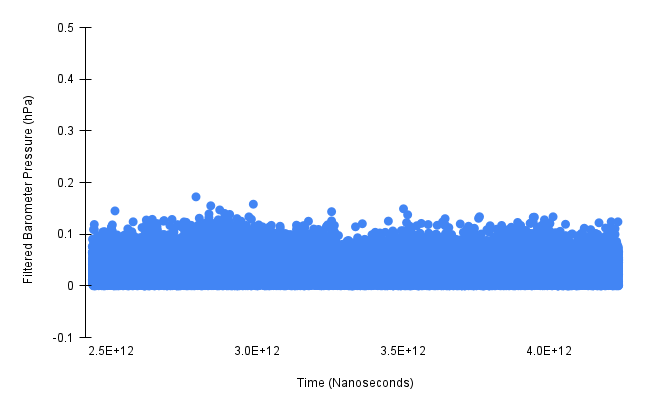


Figure 4.4.2 Graph of ambient pressure data with low-frequency information filtered out, recorded from a room closed off to the outside over a 30-minute time interval from 1:00 pm to 1:30 pm.

A high-pass infinite impulse response filter can filter out low-frequency changes in the ambient pressure caused by temperature and weather while preserving high-frequency changes like those caused by a blow. This type of filter uses a cutoff frequency to filter out low-frequency changes. Figure 4.4.1 shows 30 minutes of ambient pressure data recorded from a room closed off to the outside. This data shows the ambient pressure slowly decreasing throughout the 30 minutes, likely due to the temperature changing outside. Figure 4.4.2 shows the absolute value of the same data after a high-pass infinite impulse response filter is applied. Figure 4.4.2 only contains variations in the pressure data that occur at a rate greater than the cutoff frequency used, once every 5 minutes in this case. The filter removes the slow continuous changes caused by the temperature and weather and produces a graph with an average slope of 0.

To implement a high-pass infinite impulse response filter a SensorFilter class was created. This class stores the previous pressure measurement, its associated timestamp, and the cutoff frequency. It uses these three variables to calculate the filtered output of the next pressure measurement it receives. Figure 4.4.3 shows two functions within the SensorFilter class, the apply function and the calculateAlpha function. These are the two functions that apply the filter and perform all the related calculations. A cutoff frequency of 1 / (5 \* 60) Hz is used to filter out changes that occur at a rate of once every 5 minutes or slower.

A computer screen with text

Description automatically generated

Figure 4.4.3 Code Snippet: The apply and calculateAlpha functions within the SensorFilter class that implements the first-order infinite impulse response filter.

To apply this filter to the barometer data a FilteredBlowListener class was created. This class inherits the BlowEventListener class discussed in section 4.2. The FilteredBlowListener class functions identically to the BlowEventListener class. The only exception is its implementation of the onSensorChanged function, shown in Figure 4.4.4. Lines 25 – 32 in Figure 4.4.4 initialize the filter with the first data point and reset the filter after the number of data points recorded reaches the resetInterval variable’s value. The resetInterval variable is a constant value set to 100,000. The filter is reset after every 100,000 data points to prevent drift in its output caused by precision loss compounding over time. The recorded pressure values are also multiplied by a constant scalar value, barScalar, to scale them up and prevent precision loss. Line 36 in Figure 4.4.4 calls the SensorFilter’s apply function and gets the absolute value of what it returns, effectively filtering out low-frequency information from the current measurement. If the filtered value is greater than or equal to the anomolyThreshold variable it is considered an outlier data point and saved into the blowData list. Once three outlier data points are detected in a sequence a blow event is triggered, which calls all the registered callback functions. If the filtered value is less than the anomolyThreshold variable then the data point is not considered an outlier and is used to update the SensorFilter’s internal variables. Through trial and error, a value of 0.20 \* barScalar was selected for the anomolyThreshold variable.

A screenshot of a computer program

Description automatically generated The FilteredBlowListener class, in conjunction with the SensorFilter class, can detect when a user blows air over their smartphone using a pressure threshold outlier detection scheme by first filtering out changes to the ambient pressure caused by weather and temperature.

Figure 4.4.4 Code Snippet: The onSensorChange function from the FilteredBlowListener class that detects when a user blows on the device via a pressure threshold method.

## Pressure Threshold Evaluation

The filtered threshold outlier detection implementation of the application was tested using three sets of repeatable instructions, the results of which are tabulated into multiple confusion matrices. Each set of instructions was performed three separate times.

The first set of instructions consists of the following steps:

1. Scroll the screen down three posts.
2. Switch the scroll direction.
3. Scroll the screen up three posts.
4. Upvote the post under the cursor.
5. Downvote the post under the cursor.

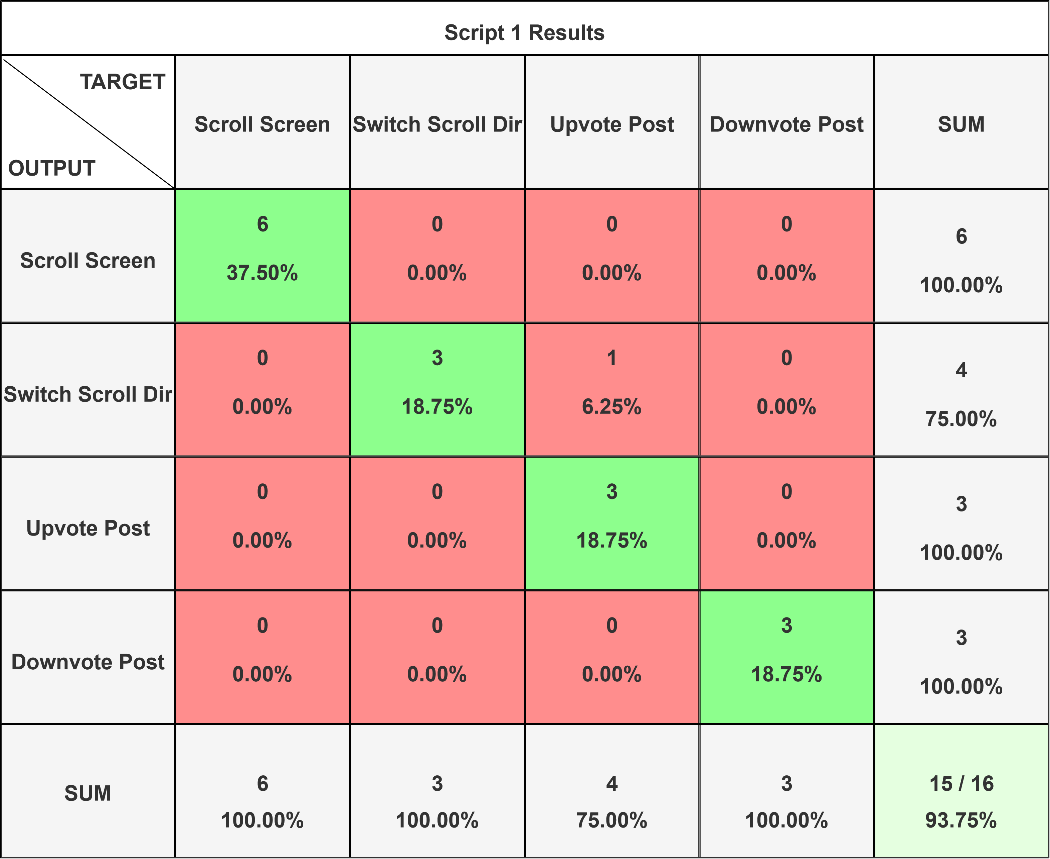


Figure 4.5.1 A confusion matrix containing the results of the first set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 93.75% accuracy.

Figure 4.5.1 shows the results of this first set of instructions. These results show a 100% accuracy for the screen scrolling, scrolling direction, and downvote post features, and a 75% accuracy for the upvote post feature. The application was able to accurately predict 15 out of 16 of the user’s actions, resulting in an accuracy of 93.75%.

The second set of instructions consists of the following steps:

1. Upvote the post under the cursor.
2. Scroll the screen down six posts.
3. Switch the scroll direction.
4. Upvote the post under the cursor.
5. Scroll the screen up three posts.
6. Downvote the post under the cursor.

A screenshot of a graph

Description automatically generated

Figure 4.5.2 A confusion matrix containing the results of the second set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 95.00% accuracy.

Figure 4.5.2 shows the results of this second set of instructions. These results show a 100% accuracy for the screen scrolling, scrolling direction, and upvote post features, and a 75% accuracy for the downvote post feature. The application was able to accurately predict 19 out of 20 of the user’s actions, resulting in an accuracy of 95.00%.

The third set of instructions consists of the following steps:

1. Scroll the screen down two posts.
2. Downvote the post under the cursor.
3. Scroll the screen down two posts.
4. Downvote the post under the cursor.
5. Switch the scroll direction.
6. Scroll the screen up three posts.
7. Upvote the post under the cursor.
8. Switch the scroll direction.
9. Scroll the screen down one post.
10. Upvote the post under the cursor.

A screenshot of a graph

Description automatically generated

Figure 4.5.3 A confusion matrix containing the results of the third set of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 91.67% accuracy.

Figure 4.5.3 shows the results of this third set of instructions. These results show a 100% accuracy for the scrolling direction, upvote post, and downvote post features. The scrolling screen feature had an accuracy of 83.33%. The application was able to accurately predict 11 out of 12 of the user’s actions, resulting in an accuracy of 91.67%.

A screenshot of a graph

Description automatically generated

Figure 4.5.4 A confusion matrix containing the results of all three sets of instructions tested on the filtered threshold outlier detection implementation. The matrix shows a 95.65% accuracy.

Figure 4.5.4 shows the results of all three sets of instructions tested on the filtered threshold outlier detection implementation. The screen scrolling feature has an accuracy of 96.43%. This is a 5.8% increase in accuracy over the previously tested slope threshold outlier detection implementation. As expected, the filtered threshold detection method used by the FilteredBlowListener class can detect long continuous blows more accurately than the slope threshold detection method. The switch scrolling direction feature had the highest accuracy of all the features, 100%. This is also an expected result. The switch direction feature is the easiest feature to detect. It consists of the fewest number of blows in a sequence and does not rely on long blows of air like the screen scrolling feature. The upvote and downvote features have similar accuracy ratings. The upvote feature has a rating of 93.75%, and the downvote feature has a rating of 92.31%. User error caused these ratings to fall below 100%. Each feature only missed predicting one instance, which was caused by the user failing to chain blows in a sequence within the correct time interval. The application was able to accurately predict 66 out of 69 of the user’s actions, resulting in an accuracy of 95.65%.

These results show that the filtered threshold detection implementation is 2.5% more accurate than the slope threshold detection implementation and is 5.8% better at detecting when the user wants to scroll the screen.

# Conclusion

The sensors within modern smartphones represent a large untapped potential of novel applications. Research has already shown that just one of these sensors, a barometer, can be used in unique and unsuspecting ways.

To explore the novel applications of a barometer sensor multiple experiments were conducted throughout this thesis. The first of these experiments was the Sound Pressure Experiment, which sought to determine if a smartphone’s barometer sensor is sensitive enough to detect the distance of an audio source. The experiment showed this was not the case. However, an interesting downward trend emerges from the data when graphing the measurements between trials that were taken at the same distance. This trend suggests that the ambient pressure in the room that the experiment was conducted in was not constant but was instead continuously changing.

To further investigate the downward trend observed in the first experiment, a second Ambient Pressure Experiment was conducted. This experiment was designed to be as simple as possible to isolate the variables that might cause the observed trend. A smartphone was left recording barometer data in an internal room with no windows and a single door that was left closed for the duration of the experiment. These recordings show that the pressure inside this isolated room continuously changes throughout the day. As the temperature outside got warmer the pressure inside the room dropped. As the temperature outside dropped, the pressure inside increased. These results suggest that the ambient pressure of a room, even one isolated from the outside, is heavily dependent on the temperature and weather conditions. Any possible application for a barometer sensor beyond reading the local atmospheric pressure must consider the continuous change in ambient pressure.

A third experiment was conducted to explore how sensitive the barometer sensor would be to a user blowing air over the device with their lungs. This experiment showed promising results. The barometer was able to easily detect a user blowing air so long as the user blew onto the right place on the device. The barometer was much more sensitive to blows when the user was blowing on the bottom of the device, suggesting that is where the barometer is located in the Pixel 4a. This third experiment suggests that the barometer may be able to be used as a user input device.

A smartphone application was developed to test whether using the barometer as a user input device is practical. The application mimics the website Reddit.com, a social media platform where users can post text and images that other users can vote on. A Reddit post consists of text, an image if one exists, and an upvote and downvote button. The first 100 posts from Reddit are requested by the application and then are displayed in a scrollable list.

To interact with the application using just the barometer a blowing language was created. This language works by assigning different user actions to a unique number of blows in a sequence. For example, blowing once scrolls the list for the duration of the blow, and blowing twice switches the scroll direction. The full blowing language implemented in the application can be found in Chapter 4. This blowing language covers all the functionality of the application and allows a user to fully interact with it using just their lungs.

Multiple tests were conducted to test how accurate a barometer user interface is using both a slope threshold outlier detection implementation and a filtered threshold outlier detection implementation. The results of this testing show that the filtered threshold outlier detection implementation is 2.5% more accurate and can detect the screen scrolling action 5.8% better than the slope threshold outlier detection implementation. Out of the 69 actions taken by the user, the filtered threshold outlier detection implementation accurately detected all but three of them. The overall accuracy of this implementation was 95.00%. The filtered threshold outlier detection implementation was found to be the more robust solution.

Barometer sensors are one of many micro-electromechanical sensors built into modern day smartphones that have unsuspecting use cases waiting to be realized. Using a barometer as a user input device by detecting when a user blows over a smartphone proves to be a novel practical use case for a smartphone’s barometer sensor.

# References

1. Balz Maag, Zimu Zhou, Olga Saukh, and Lothar Thiele. 2017. Barton: Low power tongue movement sensing with In-Ear Barometers. *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)* (December 2017). DOI:<http://dx.doi.org/10.1109/icpads.2017.00013>
2. Diansheng Chen, Yu Zhang, Wei Feng, and Xiyu Li. 2018. A wireless real-time fall detecting system based on barometer and Accelerometer. *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (July 2012). DOI:<http://dx.doi.org/10.1109/iciea.2012.6361022>
3. Haibo Ye, Li Sheng, Tao Gu, and Zhiqiu Huang. 2019. SELoc: Collect your location data using only a barometer sensor. *IEEE Access* 7 (June 2019), 88705–88717. DOI:<http://dx.doi.org/10.1109/access.2019.2925460>
4. Haibo Ye, Tao Gu, Xianping Tao, and Jian Lu. 2016. Scalable floor localization using barometer on smartphone. Wirel. Commun. Mob. Comput. 16, 16 (November 2016), 2557–2571. <https://doi.org/10.1002/wcm.2706>
5. Hiroaki Nakanishi, Sayaka Kanata, and Tetsuo Sawaragi. 2011. Measurement model of barometer in ground effect of unmanned helicopter and its application to estimate terrain clearance. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics* (November 2011). DOI:<http://dx.doi.org/10.1109/ssrr.2011.6106764>
6. Lili Xie, Jun Tian, Genming Ding, and Qian Zhao. 2018. Human activity recognition method based on inertial sensor and Barometer. *2018 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)* (March 2018). DOI:<http://dx.doi.org/10.1109/isiss.2018.8358140>
7. Muchen Wu, Parth H. Pathak, and Prasant Mohapatra. 2015. Monitoring building door events using barometer sensor in smartphones. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). Association for Computing Machinery, New York, NY, USA, 319–323. <https://doi.org/10.1145/2750858.2804257>
8. Muyan Yao, Dan Tao, Ruipeng Gao, Jiangtao Wang, Sumi Helal, and Shiwen Mao. 2022. PresSafe: Barometer-based on-screen pressure-assisted implicit authentication for smartphones. *IEEE Internet of Things Journal* 10, 1 (August 2022), 285–302. DOI:<http://dx.doi.org/10.1109/jiot.2022.3199657>
9. Nozhan Ghoreishi, Femi Olugbon, John LaCourse, Sajay Arthanat, Dain LaRoche, and Diliang Chen. 2023. The potential of wrist-worn barometer for measuring load vertical location in manual material lifting tasks: A pilot study. *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (October 2023). DOI:<http://dx.doi.org/10.1109/biocas58349.2023.10388547>
10. Olivia Sidoti, Risa Gelles-Watnick, Michelle Faverio, Sara Atske, Kaitlyn Radde, and Eugenie Park. 2024. Mobile Fact Sheet. (January 2024). Retrieved October 14, 2024 from <https://www.pewresearch.org/internet/fact-sheet/mobile/#mobile-phone-ownership-over-time>
11. Ping-Fan Ho, Chia-Che Hsu, Jyh-Cheng Chen, and Tao Zhang. 2018. Using Barometer on Smartphones to Improve GPS Navigation Altitude Accuracy. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18). Association for Computing Machinery, New York, NY, USA, 741–743. <https://doi.org/10.1145/3241539.3267735>
12. Satoshi Hyuga, Masaki Ito, Masayuki Iwai, and Kaoru Sezaki. 2015. Estimate a user's location using smartphone's barometer on a subway. In Proceedings of the 5th International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT '15). Association for Computing Machinery, New York, NY, USA, Article 2, 1–4. <https://doi.org/10.1145/2830571.2830576>
13. Yifan Che and Jiuxiang Dong. 2024. A real-time, robust visual-inertial navigation system tightly coupled with GNSS and Barometer. *IEEE Sensors Letters* (May 2024), 1–4. DOI:<http://dx.doi.org/10.1109/lsens.2024.3399552>