Investigation into Measuring Blood Pressure with a Mobile Phone

Molly Moen
Computer Science & Engineering
University of Washington
Seattle, WA
moenm@uw.edu

ABSTRACT
Hypertension, or high blood pressure, is a symptom-less disease which can lead to heart attack, stroke, and many other health problems. Currently, getting a blood pressure measurement involves either going to the doctor or using a relatively expensive piece of equipment. We investigated the feasibility of measuring blood pressure with an Android phone, with minimal external hardware. The application currently collects data using the phone, a computer, a force sensor and a gyroscope, and has the potential to only need an external force sensor to get all the data it needs. The application has the potential to be a good substitute for a blood pressure reading by a trained professional.

Keywords
Mobile; health; blood pressure; gyroscope; force; camera; pressure

INTRODUCTION
Measuring blood pressure is an essential component of health care. Blood pressure is an indicator of a person’s cardiac and circulatory health [6]. An accurate blood pressure reading is essential for diagnosing hypertension, or high blood pressure. Hypertension rarely has symptoms, but is very dangerous for a person’s health, especially when left untreated. High blood pressure puts a person at higher risk for heart attack, stroke, arteriosclerosis (hardening of the arteries), heart disease, and kidney damage [20]. Blood pressure can almost always be lowered with a combination of medication and lifestyle changes, but a diagnosis must be made first.

A diagnosis for hypertension is made by measuring blood pressure with a sphygmomanometer (blood pressure cuff). The cuff is placed around the arm, temporarily stops the flow of blood through an artery in the arm. Then air is let out of the cuff while a stethoscope is placed over the artery. The first sound that is heard occurs at the systolic pressure, or the pressure at which the heart beats. The last sound heard occurs at the diastolic pressure, or the pressure between heart beats [6]. A blood pressure reading is reported in units of millimeters of mercury, or mmHg.

Hypertension is not the first disease that comes to mind when thinking about developing countries. However, it has become a major health concern in these regions. Data has shown hypertension is common in developing countries, and is on the rise. However, rates of awareness, treatment, and control are low in these regions [11]. One of the reasons for the lack of diagnosis and treatment is that there are not enough trained health professionals in these countries [13]. If there were a way for blood pressure to be measured by anyone, with a readily available device such as a smartphone, it could have a huge impact on these countries.

This paper discusses an Android phone application with the potential to measure blood pressure with minimal external hardware. The application currently can collect data using a computer, and an external force sensor and gyroscope.

RELATED WORK
Heart Rate
Photoplethysmography, or PPG, is a simple optical method to measure heart rate by detecting blood volume changes in the microvascular bed of tissue [1]. The method works as follows: when the heart beats, the capillary at the tip of the finger fills with blood. When it is full of blood, it will block the amount of light that can pass through the tissue. As the blood retracts, more light can pass through the tissue [12]. The PPG waveform is made up of a pulsatile waveform which shows the blood volume changes from each heart beat and a slowly varying waveform which has components attributed to various measurements such as respiration [1]. This technology can be used to measure heart rate by simply taking measurements at the skin’s surface.
In order to use PPG, all that is required is a light source to illuminate the tissue and a detector to measure the variations in light intensity [1]. This method is used in many medical devices currently, such as pulse oximeters (sensors placed on the fingertip that measure oxygen saturation). Beat-to-beat blood pressure monitors, that measure blood pressure on a continuous basis, also use PPG [1].

PPG can now be performed using a cellular phone. Applications exist that use either the front-facing or back-facing camera on the phone. The front-facing camera measures the color change in the face caused by each heartbeat, and therefore require good lighting in order to work. Other applications allow the user to use the back-facing camera by putting their fingertip over the camera and using the flash as the light source to illuminate the tissue [12].

Using a phone for PPG, the heart rate can be found from the camera data as follows. Each frame is split into its red, green, and blue components (RGB). By plotting the RGB signals, the heart rate can be seen as the peaks and valleys in the lines. Previous studies have shown that this method is effective for measuring heart rate [12].

**Blood Pressure**

There are currently phone applications that can measure blood pressure, but they require an external blood pressure cuff. One such application for iPhone uses the iHealth Blood Pressure Dock, an external kit consisting of a blood pressure cuff and charging station for the phone, which costs $100. The cuff sends the blood pressure reading to the phone, and the application keeps track of readings [25].

Studies have also been successfully completed using photoplethysmography to measure blood pressure, along with a cuff to apply pressure. One such study used a cuff for the wrist which was equipped with a sensor for receiving the PPG signals. The wrist cuff was inflated (similar to a standard blood pressure cuff) to a high pressure. Then the cuff was deflated and when the PPG signal appeared, that pressure was recorded as the systolic pressure, and when the PPG signal disappeared again, that pressure was recorded as the diastolic pressure. This method was found to be within the recommended error bound for blood pressure as determined by the Association for the Advancement of Medical Instruments (5 mmHg). [22]

**Pressure on the Phone**

One goal for our application is measuring the force on the phone without an external force sensor. Research done by Goel et al. [10] was able to roughly determine the force on a phone with their GripSense system. This system was able to distinguish between three levels of pressure with 95.1% accuracy [10]. GripSense used the vibration motor on the phone and the gyroscope to infer the pressure on the phone. The system found that when pressure was applied to the screen, the finger of the person touching the screen would absorb some of the vibration, and therefore decrease the gyroscope reading. This system could only distinguish between three levels of pressure (light, medium, and heavy), so it only provided a coarse reading of the pressure.

**APPLICATION DEVELOPMENT**

The data collection application was developed for the Android platform, using the Eclipse integrated development environment. The application was tested on a Samsung Galaxy Nexus running Android version 4.3 as well as a Samsung Nexus S running Android version 4.0.4. The first basic goal for the application was to record audio, accelerometer, camera and gyroscope data. When the data is being recorded, the application turns on the LED on the back of the phone, and vibrates at a pattern specified by the user (where the pattern is a certain number of seconds on and off).

Audio data recording was included on the application because the initial idea for the application was to have the vibration on for enough time to infer the force on the phone, and then to turn it off. The audio data would show when the vibration motor was on and when it was off, as the vibration makes a distinctive sound. The camera data was included so that the PPG signal could be found. The gyroscope data was included in order to infer the force on the phone. The accelerometer data was also recorded in case it helped with inferring the force, but it was not used in the final implementation. The audio data was also not used in the end, as it was decided to keep the vibration motor on the entire time.

The first iteration of the application used the Android IntentService to store the accelerometer and gyroscope data. The IntentService is a base class which handles asynchronous requests. Each request is handled on a worker thread that automatically stops itself when its task is finished. This service is ideal for receiving small bits of data at a time, as the data can be handled quickly and the thread does not need to be managed manually. The sensors were set up with SensorEventListener that dispatch an intent whenever a new sensor reading is taken. The reading is recorded in a file stored on the phone, along with a corresponding timestamp.

The IntentService worked well for the accelerometer data. However, the gyroscope data rate we needed was 1 kHz. The Samsung Galaxy Nexus did not collect gyroscope information at that high a rate, and the Samsung Nexus S was not fast enough to store the gyroscope at this rate without slowing down and potentially losing some data.

One attempt at a solution for the problem of too fast a data rate for the gyroscope was to use a thread that would handle both listening for the data from the sensor and saving it to a...
file. This thread was implemented using the Java Runnable class. The thread is started when the application starts collecting data, and it is passed in the Sensor from which the data was obtained, as well as the file to write to. Whenever a new reading is generated by the sensor, it is written to the file. All of this is done on a separate thread from the regular application thread, so that the application is not be slowed down by the fast rate of the gyroscope. However, the application was still too slow using this method for gyroscope data collection. The accelerometer data collection was changed to this method because it was slightly slower rate.

The next attempt at a solution for the gyroscope problem was to use an external gyroscope sensor. This way, a faster phone could be used even if it did not have a fast gyroscope. The external gyroscope used is connected to a Phidget, a simple adapter that can be connect a sensor to a computer via USB. These sensors work with the Android platform, as well as with many different programming languages. Since we were already connecting a gyroscope Phidget, we also added a force Phidget, which was intended as a temporary helper. The intent was that later we could use the gyroscope readings to infer the force.

The Phidgets need to be plugged into a device with a USB port, which Android phones do not have. The initial plan to deal with this issue was to have the Phidgets transmit their data from a computer to which they are plugged into the phone via the Phidget WebService. The Phidget WebService transmits the Phidget’s sensor readings between devices that are connected to the same network. However, it was soon discovered that Phidgets can only transmit at 62.5 Hz over the WebService, so we had to switch to simply collecting the data on the computer.

A Python program was written that took the gyroscope and force data from the sensors and wrote it to a file. This program used the Phidget Python library, which makes it easy to get the data coming in from Phidget sensor. As in the previous discussion, the data is written to a file along with the timestamp of the when the data is received.

The next issue was storing the camera data. Raw android camera data is stored in a byte array which is very large in size (18,432 elements). This array was too big to be used even if it did not have a fast gyroscope. The raw camera data is in YUV420 (NV21) format. YUV format consists of a brightness component (Y) and two color components (U, V). For the photoplethysmography signal, RGB (Red, Green Blue) values were needed. Since the RGB was needed eventually, and the YUV420 format was too much data to store, it was decided to convert the raw data to RGB format, and then only store the average red, green and blue values for each frame.

In YUV420 format, the Y, U, and V components are grouped together in the byte array, instead of alternating between them. This makes it a non-trivial calculation to convert to RGB format. Each red, green, and blue pixel is found by first finding the YUV values for that pixel, and then performing a calculation on the YUV values to find the RGB values. Even the RGB averages were too much for the phone to handle (like the gyroscope, the camera recorded at 1 kHz). Since the phone already had to be connected to a computer to store the Phidget information, the solution we found to this problem was to output the camera data to the Android log. The computer then parsed the output and stored it into a file.

The parsing of the camera data was done using the same Python program that was taking in the force and gyroscope data. This program is set up to read from the Android log. When the application starts recording, it outputs a starting message to the log: this starts the recording of the force and gyroscope data as well as starts parsing the camera data. When the application is stopped, it will output an ending message, so the program will stop recording force and gyroscope data and stop looking for camera data to parse.

The resulting application collects accelerometer, audio, and camera data. The accelerometer data is stored into a file on the phone in the format (timestamp, x, y, z). The audio data is stored into a .wav file, a lossless representation. When the app starts, the user can select whether they want audio, camera, accelerometer, or any combination of the three recorded. They can also choose the vibration pattern (i.e., how many seconds on and off the vibration should be). The default is for the vibration to always be continuous.

The proper setup to run the application is to have the phone and both the force and gyroscope Phidgets plugged into a computer. The force and gyroscope sensors are attached to the phone (the gyroscope on the back of the phone, the force sensor on the front of the phone). The computer needs to be running the Python script which records the Phidget data and takes the camera data from the log and outputs it to a file. There is a button on the application to start the recording, which outputs a starting message to indicate to the Python program to start recording force and gyroscope data. When the recording is started, whichever of the combination of camera, accelerometer, and audio are selected to record will start recording (or, in the case of the camera, start outputting data to the log). While the application is in recording mode, the LED on the back of the phone is on, and if audio and/or accelerometer are recording, those files will be created on the phone.

**DATA COLLECTION**

**Participants**

The user study consisted of 23 users (12 male, 11 female) of ages 20-67 (mean=36.5, median=28, standard
There were 11 users with normal blood pressure (systolic below 120 mm Hg, diastolic below 80 mmHg), 8 with prehypertension (systolic between 120-139 mmHg, diastolic between 80-89 mmHg), 3 with stage 1 hypertension (systolic between 140-159 mmHg, diastolic between 90-99 mmHg), and 1 with stage 2 hypertension (systolic 160 or more mmHg, diastolic 100 or more mmHg).

**Apparatus Used**

The data was collected using an Android phone (Samsung Nexus S running Android 4.0.4) connected to a computer running a python script. The external sensors used were Phidgets, which are simple sensors that can be connected to a computer via USB, and work with many different platforms. The phone has two Phidgets connected to it. One Phidget had an Interlink Electronics Force Sensor (model FSR 402 Short) attached to it. The other Phidget used two-axis gyroscope.

**Data Collection Procedure**

The data was collected from users who each participated in a 20-30 minute study. First, the user’s thumb width and diameter were measured in millimeters using a digital caliper. The user was asked to sit upright in a chair, legs uncrossed, with the left arm at heart level. The user’s blood pressure was then measured using the conventional method (“gold standard”) of a sphygmomanometer (blood pressure cuff). This method was performed twice to ensure accuracy.

The user was next taught to hold the phone with their thumb pad covering the camera and LED light on the back of the phone, and their forefinger pressing the pressure sensor. The user was also instructed to hold the phone with very light pressure, and the Python program and phone application were started. This meant the phone would start vibrating and collecting camera data, and the Python program would collect and store gyroscope and force data, as well as store the camera data coming from the phone. The user would hold the phone with light pressure for ten seconds or longer. Then they were told to press very hard on the phone, then very slowly release the pressure. This process of press hard and release was then repeated. The whole process of two conventional blood pressure measurements, and two presses on the phone was repeated another two times. Finally, the user’s blood pressure was measured two more times. The users were compensated with ten dollars for their time.

**Challenges with Data Collection**

The conventional blood pressure measurement was done by students who had only learned how to measure blood pressure for this project, and were therefore very inexperienced. Errors could have occurred in these measurements due to this lack of experience. In addition, only one blood pressure cuff was available with which to do the measurements. The American Heart Association recommends the blood pressure cuff be 80 percent of the patient’s arm circumference, in order to properly compress the brachial artery [21]. For at least one user in the user study, this was not the case, and this user’s “gold standard” blood pressure reading was most likely inaccurate.

In addition, the data collection process with the user holding the phone had many potential issues. First was getting the user to release the pressure slowly enough. It required a lot of thumb and forefinger control to release the pressure as slowly as we wanted. Some users did not have this kind of control. In addition, if the user twitched while holding the phone, the gyroscope reading would respond accordingly, and therefore not align with the force reading, which made later analysis difficult.

**DATA ANALYSIS**

We wanted to be able to eventually remove the force sensor in favor of using the gyroscope to infer force. Our hypothesis was that when the force reading would increase the gyroscope reading along the y-axis would decrease, as was observed in GripSense. This is because the increase in pressure would absorb some of the vibration of the phone, so the gyroscope reading would go down. However, this effect was only observed in eight of the sixty-nine trials (see Figure 1). A much stronger effect was seen along the z-axis. When the force on the phone increased, so did the gyroscope reading along the z-axis for fifty-nine of the sixty-nine trials (see Figure 2). This was because as the user pressed on the phone, it would start vibrating more along the z axis to compensate for the pressure.

![Figure 1. One of the eight trials in which the y-axis of the gyroscope had the expected correlation (y-axis is middle graph, force is top graph)
Figure 2. One of the fifty-nine trials where the z-axis of the gyroscope correlates with the force reading. This trial also shows the behavior seen in sixty-one trials where the y-axis did not correlate with the force.

Despite the setback of not observing the correlation we expected, the z-axis correlation looked very promising. Most of the time, when the force had clear peaks from the user pressing hard on the phone, the gyroscope reading also increased, and vice versa. In order to map the gyroscope data to the force data the following steps were taken. A high pass filter was applied first to the gyroscope data. Then all data points from the force data where the reading was greater than 600 or less than 150 were removed, and the corresponding data points from the gyroscope data were also removed. This got rid of very low force readings, where the gyroscope reading would not tell us much about the force, and readings that were much higher than we cared about. Next the force reading was converted, as the data received from the sensor was in volts, so it was converted to grams/cm², which have a linear relationship with mm Hg (1 gram/cm² = 0.7356 mm Hg). The force data was then smoothed out by taking the mean over a window of 100 data points. The gyroscope data was also smoothed out, but instead of taking the mean, the 95th percentile over every 100 points was taken. Next, the force data and gyroscope data were smoothed out again, this time taking the mean over every 3 data points, with a repeat of 2. Finally the Matlab regress function was applied to the force and gyroscope data to obtain a regression coefficient.

The regression coefficients found ranged from 64.4968 to 144.9583. When these coefficients are applied to the data, the force and gyroscope data do not align well. The mean square error varies from a minimum of 532 and a maximum of 6.146e+03 with an average of 1.7599e+03 (median =1.2413e+03, variance = 1.9538e+06). Although there were a few trials with very high mean square error, most had a much lower mean square error than with just a regression coefficient.

However, with the decreased mean square error came an increased deviation in coefficients. The equation produced is $y = ax + b$, where $y$ = force, $x$ = gyroscope, and $a$ and $b$ are constants. The ‘a’ coefficient had a variance of 3.2375e+04 and the ‘b’ intercept had a variance of 3.0027e+04.

One theory for why the coefficients varied so much is that the force to gyroscope relationship is based on factors specific to each user. One possible factor that could have changed the relationship between force and gyroscope reading was thumb size. However, this theory turned out to be incorrect. The following graphs plot the ‘a’ coefficient from the linear model versus thumb width (Figure 4) and thumb diameter (Figure 5).
These graphs do not have a strong enough correlation between thumb size and coefficient. There is a slight upward trend, but the data is too scattered for the upward trend to be significant.

The next step was to see if a good equation could be found for a single user. However, even within a user there was a large amount of variability in the equation from force to gyroscope reading. For example, the equations for the three trials from user sixteen produced were as follows (where $x =$ gyroscope reading): trial 1: $240.62x – 166.8$, trial 2: $81.966x – 50.938$, trial 3: $117.77x – 65.171$. The mean square errors using these equations were: trial 1: 381.9405, trial 2: 91.0457, trial 3: 125.7593 (this was one of the best correlations among the users). The first trial had a much worse mean square error than the second two trials because the gyroscope data was shakier. The plots for these trials are Figures 6, 7, and 8 (the gyroscope and force data are modified in these graphs using the same method as before).

Even though all three trials came from the same user, they had very different equations mapping the gyroscope reading to the force reading. Using an equation from a different trial yielded a very poor mean square error (for example, if the equation from trial two is used to transform trial one and trial three, the mean square error for trial one jumps from 326.9437 to 7560.7 and the mean square error for trial three jumps from 86.4580 to 2260.2). This same problem was seen for many users.

The camera data analysis was done by another person working on the project. Preliminary analysis shows a difference between users with high blood pressure and those with low blood pressure. Figure 9 is a plot of the red camera data versus time for a user with stage 1 hypertension, and Figure 10 is same plot for a user with
normal blood pressure. The length of time when the user with normal blood pressure’s pulse cannot be seen (the flat line in Figure 10) is much longer than the same line in Figure 9. This means it took longer for the second user’s pulse to appear after applying pressure, or that they have a lower blood pressure than the user from Figure 9.

![Figure 9. Plot of red camera data versus time for a user with high blood pressure](image)

![Figure 10. Plot of red camera data versus time for a user with normal blood pressure.](image)

**FUTURE STEPS**

The force and gyroscope data provided an unfortunate blockade for the project. The goal was to not need external sensors, but as of yet there is not an accurate way to determine the force reading from the gyroscope reading. In order to get an accurate blood pressure reading, an accurate force reading is necessary, as the force is what determines the blood pressure reading.

One possible direction the application could go in would be for it to only give coarse blood pressure readings (i.e., it outputs “normal”, “prehypertension”, “stage 1 hypertension” or “stage 2 hypertension”). However, this type of application has very limited use. It would only be useful for someone who did not know their blood pressure at all, and just wanted to see if they were normal. Beyond making sure that you do not have high blood pressure, the application would not be very helpful. Those who need to check their blood pressure frequently are those with hypertension, and they need to know the exact reading so they can see if their blood pressure is increasing, decreasing, or staying the same [17]. In addition, the distinction between normal and hypertension is only a difference of 20 mmHg for systolic and 10 mm Hg for diastolic, which is a relatively small difference in pressure. The gyroscope readings cannot make this distinction either; they can only tell if the force is increasing or decreasing, but not what the actual reading is, even with an error bound of 10 mm Hg.

The goal for the application is to get real time results for the user’s blood pressure. One way to do this that would build on previous work would be to have the user perform the same method of data collection as in the user study. The user would hold the phone’s thumb covering the camera and flash, with their finger on the force sensor. The phone and external sensors would still be hooked up to the computer, which would be running the Python application. The user would start data collection, and press lightly on the phone for ten seconds or less. Then they would press very hard and slowly release, and repeat, then stop collection. The data that came in from the camera would be analyzed and the time when the heartbeat started and stopped would be found. Then the force at those times would be found, and converted to mmHg. These two force readings would be the blood pressure.

The implementation described previously would still have the issue of needing to be connected to a computer to work. There are currently two main reasons the phone needs to be connected to the computer; it is not fast enough to handle 1 kHz of data, and it needs an external force sensor. The issue of the phone not being fast enough could be solved with a faster phone, or with a new method of saving the data. One possible solution is to store all the camera data in memory while the application is running, and store it in a file after all the data has been recorded. The force sensor presents more of a problem. It was initially intended as a temporary helper, with the gyroscope eventually taking its place. However, after a lot of data analysis, it was found that the gyroscope data cannot predict the force data well.

One possible solution to the force sensor problem would be to have a force sensor that connects to the phone, which would still be external hardware but at least it would not require a computer. If a cheap force sensor could be developed that would plug into the phone and transmit its data to the phone, it would make the application very portable and easy to use.
Another issue with getting the application eventually encapsulated on the phone is the camera data. The current method of having the camera data output to the computer and then having the computer parse it is also problematic. It means the computer is an integral part of the application, which takes away from the goal of having a self-contained phone application.

The 1 kHz sampling rate slowed down the application on both the Samsung Galaxy Nexus and the Samsung Nexus S. A faster phone would be necessary in order to record camera data without needing to resort to a computer to store the data.

APPLICATIONS

The application would be most useful if it did not need to be connected to a computer, but it could still have many uses if a computer connection is necessary. If the phone did not need to be connected to a computer, the possible applications are immense, especially for the developing world.

In the developing world, mobile phones are widely used. As of 2012, there were nearly 5 billion mobile subscribers in the developing world [8]. The best focus of technology for this area of the world is through mobile phones, as it will reach the most people.

As mentioned previously, diseases such as hypertension are on the rise in the developing world, especially in urban areas [13]. As late as 1940, hypertension was practically non-existent in the developing world; one reported rate in Ethiopia was 1.8 percent [11]. However, hypertension rates are now very high. The highest rate is seen in South Africa, where 78 percent of adults age 50 or older have hypertension. In Mexico the rate is 58 percent, in Ghana 57 percent, in China 53 percent, and in India 32 percent [3].

Some possible reasons for this large increase in hypertension in these countries are lifestyle factors such as a diet high in salt, obesity, alcohol consumption, tobacco use, and low physical activity [3][11]. These factors are seen most often in urban areas, where foods high in salt and fat are more common. In addition, the movement from rural to urban areas often correlates with a decrease in physical activity and an increase in stress, both of which can cause an increase in blood pressure [11].

Not only is hypertension on the rise in developing countries, but it is often left untreated in this region. Many in the developing world are not aware of the risks of hypertension. Developing countries often do not have national policy for hypertension and other non-communicable diseases, because they are focused on the treatment of communicable diseases. [11] Health care workers in these countries are therefore more focused on diagnosing other diseases, and often do not even check blood pressure. In some areas, especially in sub-Saharan Africa, health units do not have a functioning sphygmomanometer [11].

This technology could be very helpful for measuring blood pressure without a trained health professional. In the developing world, there are large shortages of health care workers. It is estimated that in Africa there is a shortage of 800,000 health care workers [13]. This leaves many without the most basic of health care, and therefore they are without the knowledge of their own health.

Although an easy way to measure blood pressure would not solve these problems completely, it would help. If a phone could reliably measure blood pressure, it would mean that no training was necessary in order to start measuring blood pressure and checking for hypertension. All measurements would be done using the same system, so there wouldn’t be an issue with variations among different people reading the blood pressure.

Since the application runs on the Android platform, an Android phone is needed to make the measurements. Although Android phones are less common that basic mobile phones in the developing world, the number of Android smartphones in the developing world is increasing as they become cheaper and more ubiquitous. An 80 dollar smartphone running the Android operating system recently sold 350,000 units in Kenya, and as prices are dropping, more and more Android phones are seen in these regions [7].

If health clinics in urban areas (where hypertension is most common) had access to an Android smartphone, they would not need to invest in a sphygmomanometer, or in the training to learn to use one properly. Instead they could simply install the application (and possibly buy a cheap force sensor) and begin checking blood pressure quickly and reliably.

The Android phone could also become a multi-purpose health tool. There are many phone applications already that assist with health-related tasks. Researchers have developed an application that can, with a $2 attachment, conduct an eye exam to get a prescription for glasses [14]. A portable ultrasound has also been approved by the FDA that can plug into a smartphone [14]. Devices that plug into smartphones and can measure blood glucose have also been developed [23].

The application could also be used for monitoring of blood pressure, in addition to diagnosis of high blood pressure. Those with hypertension need to check their blood pressure every couple of weeks at a minimum, every day if they have recently changed their medication [17]. At home monitoring would be easy if the patient only needed to press on a phone for about 30 seconds. This would be useful both in the developing and the developed world. In the developing world, if the patient has access to a smartphone, they can monitor their blood pressure with little additional equipment. If they are very far from a
health care facility, they can keep track of their own health, which is especially important if they have been diagnosed with hypertension. In the developed world, patients can also keep track of their blood pressure in between visits, without needing to buy expensive equipment or deal with the uncomfortable arm cuff.

At home blood pressure monitoring also gives a better reading than the single reading one gets at the doctor’s office. From 25-40 percent of people whose blood pressure reads as hypertensive in the doctor’s office have blood pressure that reads as normal outside the doctor’s office [18]. This effect is called ‘white coat hypertension’, and is caused by the increased anxiety of visiting the doctor [5]. The different in reading can be greater than 30 mmHg [18]. In addition, studies have shown that the opposite effect can also occur; patients can have normal blood pressure readings in the office, but have higher readings outside the office. This effect is called ‘masked hypertension’, and it occurs in about 10% of people whose blood pressure appears normal in the office [5].

These incorrect readings made in the doctor’s office can lead to unnecessary treatment, or no treatment at all when treatment would be necessary. The solution to these effects is to measure the blood pressure multiple times a day, outside of the doctor’s office, where the effects are not present. This is usually done in the developed world with ambulatory blood-pressure monitoring (ABPM).

Ambulatory blood-pressure monitoring is where the patient will wear a blood pressure cuff on their arm which is connected to a pouch they wear around their waist for a day or more. The cuff will automatically take readings every 15-30 minutes throughout the day and night, then the readings will be downloaded to a computer [19]. The cuff usually is an oscillatory blood pressure monitor, which works by detecting vibration in the artery. When the artery is compressed so no blood can flow through, there is no vibration through the artery. When the pressure is decreased and the blood starts to flow through the artery, it exerts a vibration on the arterial wall, which the cuff detects (this occurs at the systolic blood pressure). When the artery is no longer compressed at all and blood is flowing normally, there is no more vibration, and this point occurs at the diastolic blood pressure [2].

Although ABPM provides an accurate reading of a patient’s blood pressure, it is uncomfortable for a patient to wear the cuff all day and have it constantly inflating and putting pressure on their arm. In addition, the cost of doing an ABPM measurement can range from 50-900 dollars (the reason for this large range is the difference between a hospital buying the equipment and using it for many patients versus an outside company running the test) [5]. It would be much easier and cheaper to just install an application on the patient’s phone, lend them a cheap force sensor, and have them measure their blood pressure throughout the day.

Using a phone for an easier, cheaper reading throughout the day would work in both developed and developing countries. In developing countries, the phone could be borrowed for a day from the health clinic with the necessary add-ons (at least a force sensor), and the patient could measure their blood pressure throughout the day. In order to ensure the patient did not forget to measure their blood pressure frequently, the phone could be set up to have reminder alarms throughout the day.

This use for the application would also be feasible even if a computer was necessary for the application to work. If the patient already had a computer and a mobile phone (a common occurrence in the developed world), all their health care provider would need to provide would be a force sensor add-on and the appropriate downloads. The patient would then be able to measure their blood pressure many times throughout the day, without the increased anxiety of being in a doctor’s office or the discomfort of the blood pressure cuff.

**DISCUSSION**

This application could have far-reaching implications once it can find blood pressure readings instantly, and if the issue of needing to be connected to a computer can be solved. These are not small tasks, but they are by no means impossible. The biggest issue is the high amount of computation needed by the phone in order to complete those tasks. A fairly advanced phone is necessary in order to simply record the camera data as it comes into the phone, let alone do any advanced processing to find the peaks of the heartbeat.

The application would be most useful in the developing world, where trained health care professionals and advanced equipment are harder to find. However, the phones (and currently computers) that are necessary in order for the application to work are going to be more easily accessible in the first world.

However, the application could make a difference in the first world too. One out of every three adults in the United States has hypertension [9], while 58 percent of adults in the United States have a smartphone [15], and (as of 2012), 78.9% of households in the United States had a computer [24]. If these people did not need to buy extra equipment to keep track of their blood pressure, and only needed an extra force sensor, and potentially a computer, blood pressure monitoring would be easier and cheaper for them.

The ease of at home monitoring with a smartphone does have drawbacks, especially in the developing world where there are fewer health care workers. Always being able to check your blood pressure may mean always worrying about your blood pressure. In a questionnaire done by patients with high blood pressure about at home monitoring, about a quarter of those surveyed expressed concern that constantly checking their blood pressure would increase their anxiety over their condition, which would
have an adverse effect on their life [4]. In addition, if their blood pressure readings are surprising or upsetting, the phone will do little to comfort them as a health care worker would. An important aspect of medicine is the explanations that come with potentially bad news, and a reading on a phone can seem impersonal and scary. This is especially a problem if the patient is in a developing country and there are no trained health care workers nearby they can talk to after getting an upsetting result [16].

Despite these drawbacks, this application could have very positive results. By not needing to wait for a trained professional to take a patient’s blood pressure, they could find out that their blood pressure needs attention before it becomes a bigger problem. Untrained volunteers could measure people’s blood pressures in developing countries, and give out information to those with high blood pressure about how to control it. It could save the lives of those who never would have known they had a health problem.

CONCLUSION
A phone application that can accurately measure blood pressure with little external hardware would be very useful. It would give those with hypertension an easy way to keep track of their health in between doctor’s visits. Also, it would provide a reading that is unaffected by ‘white coat syndrome’. Finally, it would allow those in developing countries to measure their blood pressure accurately, if they do not have access to a health care professional who is trained to do so. This application could prevent heart attacks and other health problems, especially in regions where blood pressure monitoring is not a normal occurrence in health care.

The application has the potential to give an accurate blood pressure reading, once the ability to do data analysis is built into the application. It will either need an external force sensor that can easily connect to the phone, or a computer will be needed to extract the force data. In addition, the application will need to either be run on a very fast phone, or have computational assistance from a computer, in order for all the data to be recorded. Once these issues are resolved, the application will be incredibly helpful.

REFERENCES


