Quals Project - Finger Based Pre-Hypertension Detection

ALEXANDER CHING, University of Washington, USA

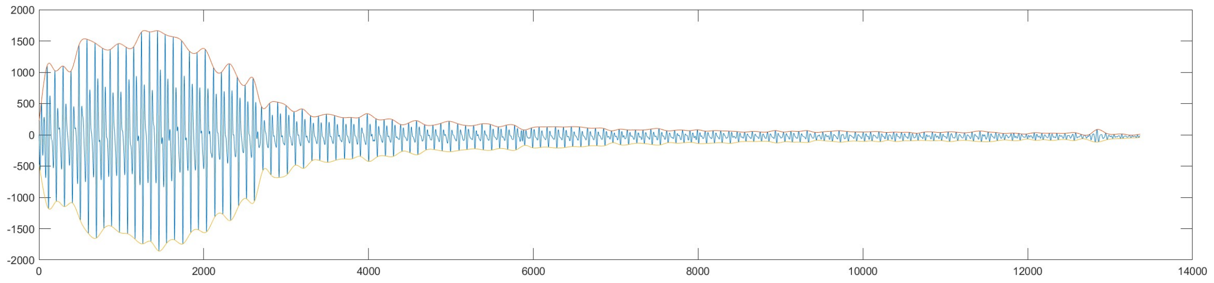

Fig. 1. A filtered sample of a raw ppg reading.

1 INTRODUCTION

Blood pressure (bp) is an essential metric used in healthcare to inform the diagnosis of a myriad of health issues. Elevated blood pressure can be an indicator of cardiovascular disease and early detection can be vital to avoiding unwanted medical issues.

Off the shelf bp monitors are generally relatively expensive cumbersome devices as they require a pump in order to inflate a cuff as well as a battery large enough to power such a pump. They can be easily damaged and can be hard to use well over clothing. This project attempts to design a finger based form factor that can help to detect pre-hypertensive blood pressures.

2 BACKGROUND

Hypertension, or high blood pressure, is the root cause of stroke and heart disease which are both leading causes of death in the US, contributing to half a million deaths in 2019 [7].

One of the most common devices used to monitor bp that is available to consumers measure using an oscillometric method and a cuff that inflates around a user's arm. While these have become ubiquitous, they still carry many disadvantages such as being expensive, large, and more easily damaged due to the need for a pump. Because oscillometric methods essentially look at changes in blood volume flow over a range of pressures, it can be applied to several parts of the body. Previous work such as by Chandrasekhar et al. [3] and Bui et al. [2] attempt to calculate blood pressure from the finger and earlobe respectively using oscillometric methods. This project is similar to the work of the former and tries to determine relative blood pressures to detect early hypertension. There are also projects such as Glabella, [9] and Seismo [10] that also utilize a ppg signal to calculate blood pressure in a novel form factor, but use pulse transit time instead of oscillometric techniques for the calculation.

Elevated blood pressure or pre-hypertension is an important indicator that cardiovascular disease can develop if you do not address it. A blood pressure reading consists of a systolic, and diastolic portion in units of mmHg. The systolic blood pressure represents the pressure on your arteries when your heart is pumping blood, whereas the diastolic is the pressure in between pumps. [6]

Photoplethysmography (ppg) has become a common method to detect the amount of blood flow. This method works by using a light source, typically a red, green, or infrared LED, and shining it at or through a blood vessel. Using a photodetector, we can see how much light gets reflected or transmitted through the blood. As the volume

Author's address: Alexander Ching, atching@cs.washington.edu, University of Washington, P.O. Box 1212, Seattle, Washington, USA, 98105.

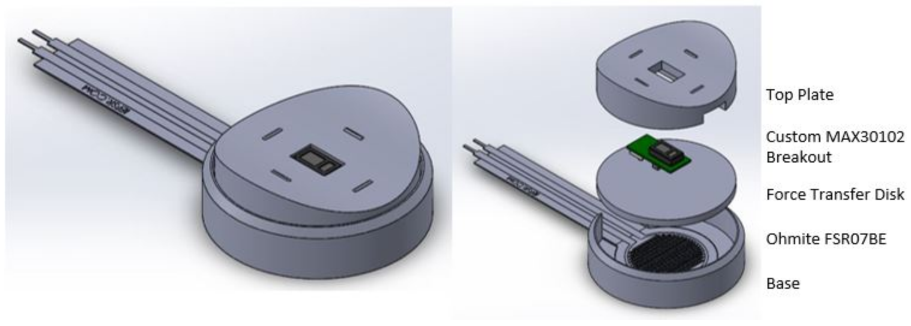

Fig. 2. Mechanical design of the sensor disk.

of blood cells containing hemoglobin changes in an area, the amount of light detected will change accordingly. We take advantage of this optical way of collecting the oscillogram needed to calculate blood pressure.

3 DESIGN

3.1 Mechanical Design

The mechanical design consists of a stack of parts that can be seen in Figure 2. This is the third major revision of the mechanical design. See the Appendix section for a description of past designs.

The top plate provides and ergonomic cover for the user's finger to press against and has an opening for the ppg sensor to sit just under the finger. That cover pushes on the force transfer disk, which is just a disk that tapers down to a 13mm diameter base, and transfers the thumb's force to the Ohmite force sensitive resistor (FSR). The signals are routed back to the breadboard that contains the Teensy with a flat flex cable and some soldered wires.

3.2 Electrical Design

The electrical design consists of the sensors and a microcontroller to process and display the information.

A Teensy 3.6 was used as the primary microcontroller using Arduino as the programming language to make use of the existing firmware libraries. The Teensy is attached to a breadboard that breaks out the pins for the sensors. The sensor used is for measuring ppg is a MAX30102. This is an integrated pulse oximetry and heart rate monitor module that includes the red and infrared LEDs, photodetectors, and circuitry needed to see the ppg signal. The signal data is digitized on the module itself and is read out over the I2C bus. Both a MAX30101 and MAX30102 were used in the data capture, but since the green LED is not used, the sensors are otherwise identical. A custom breakout printed circuit board was designed (see figure) as it allowed for the improved mechanical design described above. The board connected to a flat flex cable that was then soldered to a breakout board.

The sensor used for measuring force was a Ohmite FSR07BE, a force sensitive resistor. These sensors are essentially potentiometers that vary resistance based on the force applied to an active zone. This particular sensor was chosen as the force range was within the range that a finger may exhibit. A resistor divider was used and the voltage read using the built in ADC of the microcontroller. Because the FSR is a non-linear device, the readings had to be calibrated to associate an analog reading with a force measurement.

4 DATA COLLECTION

Data was collected from participants using the following procedure for each data set. A total of 60 data sets were collected across 6 participants. The number of data sets collected from each participant varied as some had difficulties following the procedure well.

(1) Using a commercially available off the shelf blood pressure cuff (Omron Evolv Wireless BP Monitor) a reference blood pressure reading was taken and recorded.
(2) The participant then places their thumb above the ppg sensor on the sensing disk and slowly compresses the sensor by squeezing.
(3) A live stream of the force sensor reading is displayed on the computer monitor and the participant tries to increase the force slowly to allow for the best oscillometric data.

Each participant repeated this multiple times as many times as they could. Each data set was stored as a comma separated variable file (CSV) with the following data points: A timestamp, FSR ADC Reading, Red PPG Reading, Infrared PPG Reading. This data was streamed to the PC over a serial COM port and displayed on the Arduino Serial Terminal. At the end of each collection, the streamed data would be copied and labelled with the reference blood pressure reading from the Omron monitor.

Separately, the FSR was calibrated using the following method. The resultant calibration curve can be see in Figure 7 in the Appendix.

(1) Tape sensor disk to a kitchen scale and set the reading to grams.
(2) Run Arduino Code to display the raw ADC reading from the FSR in real time on a serial monitor.
(3) Push on the sensor disk with your finger to simulate a compression.
(4) Take a photo that includes the kitchen scale reading and the serial monitor.
(5) Repeat as you slowly increase the force of your finger compression.

Using a photo to collect the data allowed for the most accurate simultaneous reading of the kitchen scale and the raw FSR reading. Otherwise it would be nearly impossible to hold the force constant while reading the scale and computer screen at the same time.

4.1 Data Processing

The data was processed in MATLAB first by cleaning the data in Algorithm 1, then using that processed data set to calculate the blood pressure and compare it to the reference in Algorithm 2 and 3. Please see actual MATLAB code in Appendix. The raw data can be seen in Figure 8 and an example of the algorithm's outputs can be seen in Figure 3.

The data is first processed, filtered, fitted and plotted as described in Algorithm 1. Figure 9 in the Appendix shows the result of the first for loop show in the algorithm above. Visually inspecting the output plots allows us to see if the digital Butterworth filter parameters are accurately removing the DC bias from the data caused by ppg sensor drift and participant breathing. A bandpass of 1.8Hz - 4.3Hz mimicks the analog filter range from Chandrasekhar et al[3]. A bandpass of 0.5Hz - 10Hz was also tried which corresponds to the general freq range of ppg signals though the results were not as good. The window for peak enveloping function was determined by manually checking the number of data points between peaks. Once the enveloping was acceptable, the amplitude could be calculated. A polynomial fit of the force vs amplitude was done to smooth the data points and then used as the oscillogram to apply the fixed ratio and derivative methods to. Parameters, such as the bandpass frequencies and polynomial fit order, were tuned until the general algorithm resulted in acceptable envelope and amplitude plots.

The fixed ratio algorithm, Algorithm 2, for determining the systolic and diastolic blood pressures was used by first finding the peak amplitude and then finding the index of the 0.55 and 0.85 of the peak amplitude respectively. The plot of this data can be see in Figure 10 in the Appendix and an example of one data set in Figure 3. Examining the plots me check that the peak was accurately found since the Butterworth filter would often cause the start and trailing amplitudes of the data set to be artificially high causing an inaccurate peak to be used.

The derivative algorithm, Algorithm 3, uses the same fitted polynomial as the fixed ratio algorithm. We find the points on the curve that corresponds to the max and min slope of the curve next to the peak amplitude

Algorithm 1 Process Raw Data

```
$dataRaw \leftarrow rawdata$                                   ▷ Read in raw data sets into indexed cell array
$dataRaw \leftarrow dataRaw(n : m)$                            ▷ Select relevant range that contains data
rawForceToGrams = exponentialFit(calibration data)
for i = 1 : length(data) do                                    ▷ Loop over all data sets in data cell array
    $data \leftarrow dataRaw(i)$
    $timestamps \leftarrow data(1)$
    $adcForce \leftarrow data(2)$
    $redPPG \leftarrow data(3)$
    $irPPG \leftarrow data(4)$
    force = rawForceToGrams(adcForce)
    butterIR = 8th Order Butterworth Bandpass [1.8Hz - 4.3Hz] Filter (irPPG)      ▷ Digital filter
    [up lo] = envelope (butterIR)
    amplitude = up - lo
    [sortedForce(i), sortedAmplitude(i)] = sort([force amplitude], 'descend')     ▷ Sort by descending force
    Y(i) = polyfit(sortedForce(i), sortedAmplitude(i), 12)                      ▷ 12th order polynomial fit
    plot                                                       ▷ Plot relevant data for visual accuracy check
end for
```

Algorithm 2 Calculate Blood Pressure Using Fixed Ratio Method

```
for i = 1 : length(data) do
    maxAmplitude = argmax(Y(i)(sortedForce(i)))                ▷ Find the peak of the fitted polynomial
    systolicAmplitude = 0.55*peak                              ▷ Fixed ratio method for finding systolic
    diastolicAmplitude = 0.85*peak                             ▷ Fixed ratio method for finding diastolic
    find sysForce < maxAmplitude where Y(sysForce) = systolicAmplitude
    find diaForce > maxAmplitude where Y(diaForce) = diastolicAmplitude
    plot                                                       ▷ Plot relevant data for visual accuracy check
end for
remove data sets where visual accuracy check of analysis results are clearly wrong
scale calculated data by arbitrary constant k to align bp closer to ref bp range
sort([referenceSystolic sysForce])
sort([referenceDiastolic diaForce])
plot                                   ▷ Plot comparisons of reference and calculated blood pressures
plot                                   ▷ Plot Bland-Altman
```

that corresponded to the diastolic and systolic points respectively. This point was found by taking the second derivative of the fitted polynomial and finding the roots which would correspond to the inflection points of the curve. The plot of this data can be see in Figure 11 in the Appendix and an example of one data set in Figure 4.

Due to the nature of the data, manual tuning of the data sets and filtering algorithms were necessary to clean up anomalies or bad data fitting. The data sets that had obviously bad data or too much noise to yield a accurate polynomial fit were removed from the final results. See Figure 12 in the Appendix for removed data.

The systolic and diastolic data is separated from each other and paired with the reference blood pressure taken from the Omron BP Monitor. Because the calculated mmHg is outside the range of possible pressures, arbitrary constants were applied to the calculated systolic and diastolic values to get them more in line with the

Algorithm 3 Calculate Blood Pressure Using Derivative Method

```
for i = 1 : length(data) do
    maxAmplitude = argmax(Y(i)(sortedForce(i)))        ▷ Find the peak of the fitted polynomial
    Z = $f''(Y)$                                        ▷ Take the Second-Order derivative of the fitted polynomial
    roots = x where Z(x) = 0                            ▷ Find the roots of Z, i.e. inflection points of Y
    sysForce = first root above x where Y(x) = maxAmplitude
    diaForce = first root below x where Y(x) = maxAmplitude
    plot                                                ▷ Plot relevant data for visual accuracy check
end for
remove data sets where visual accuracy check of analysis results are clearly wrong
scale calculated data by arbitrary constant k to align bp closer to ref bp range
sort([referenceSystolic sysForce])
sort([referenceDiastolic diaForce])
plot                                                    ▷ Plot comparisons of reference and calculated blood pressures
plot                                                    ▷ Plot Bland-Altman
```

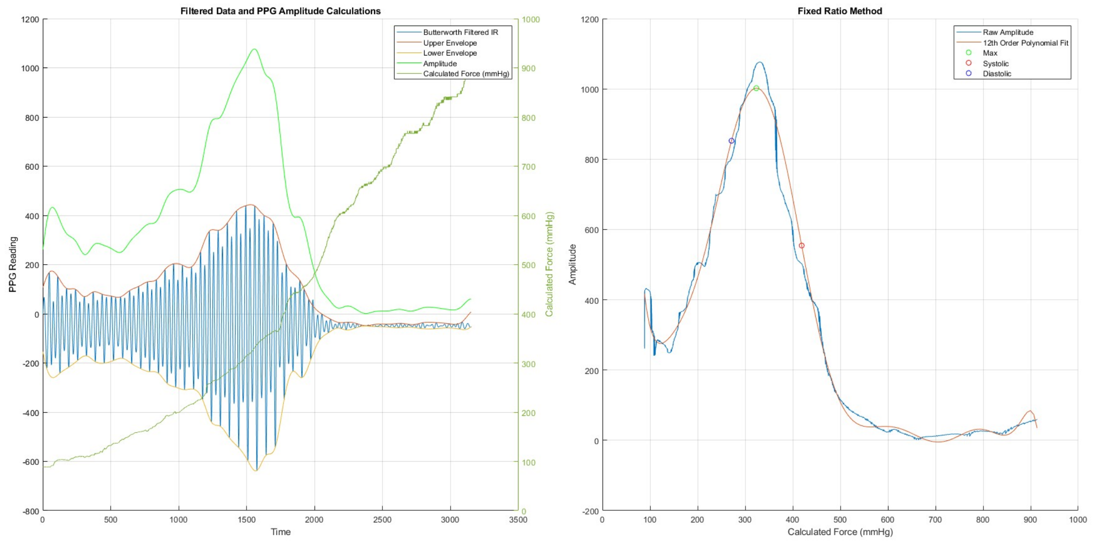

Fig. 3. An example of how one of the data sets were processed using the fixed ratio method.

reference ones. This constant encapsulates the difference in pressure readings between a actual pressure sensor and inferring the pressure from a force reading as well as any other physical phenomenon. As we are mainly interested in the relative blood pressures and not a necessarily accurate reading, the scalar was arbitrarily chosen to best fit the calculated values with the set of reference readings. Discussion of why the calculated bp is so far off without this scalar is in the discussion section.

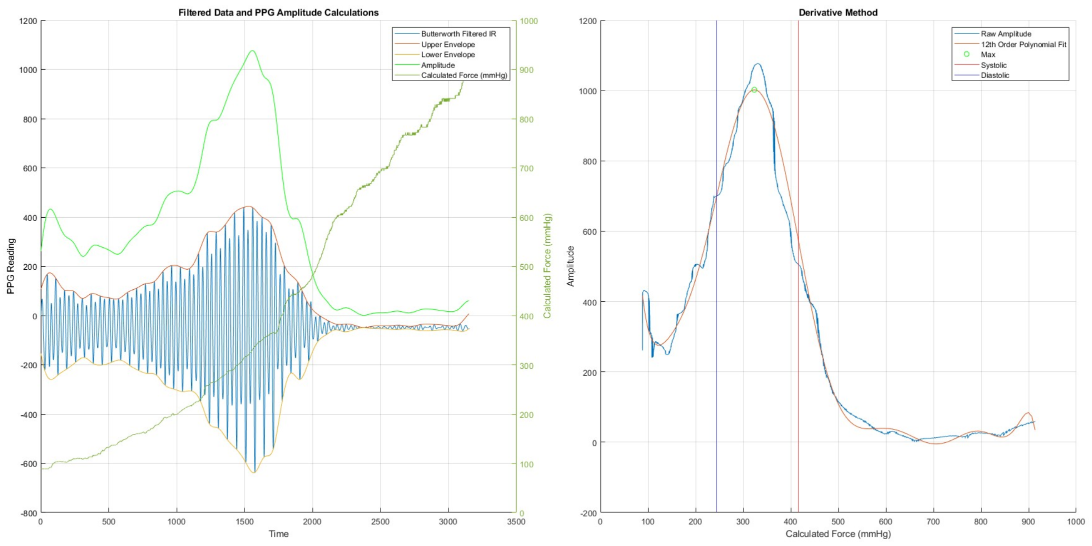

Fig. 4. An example of how one of the data sets were processed using the derivative method.

5 RESULTS

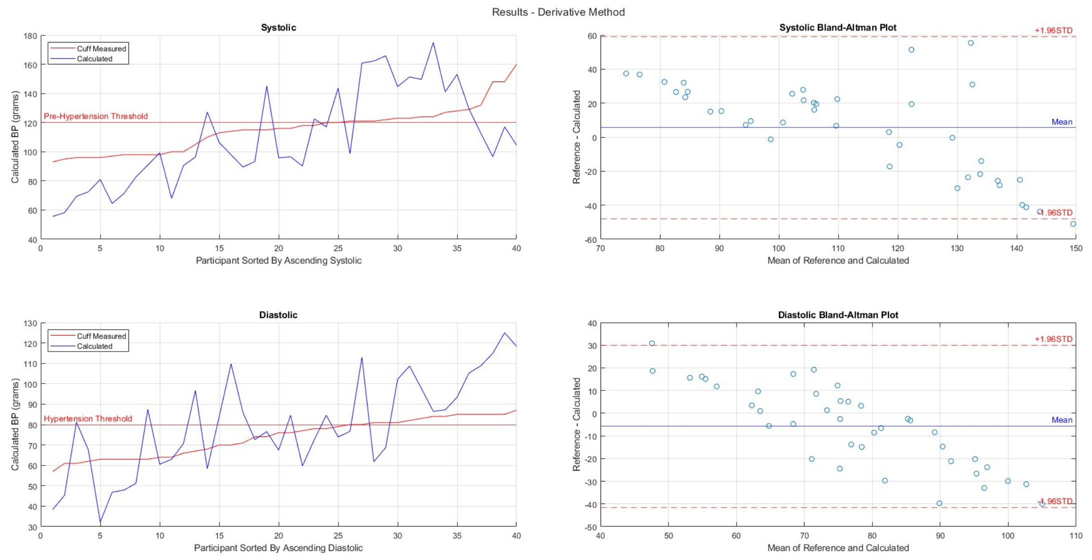

Fig. 6. Reference vs calculated comparison of the systolic and diastolic blood pressures along with the Bland-Altman plots analyzed using the derivative method.

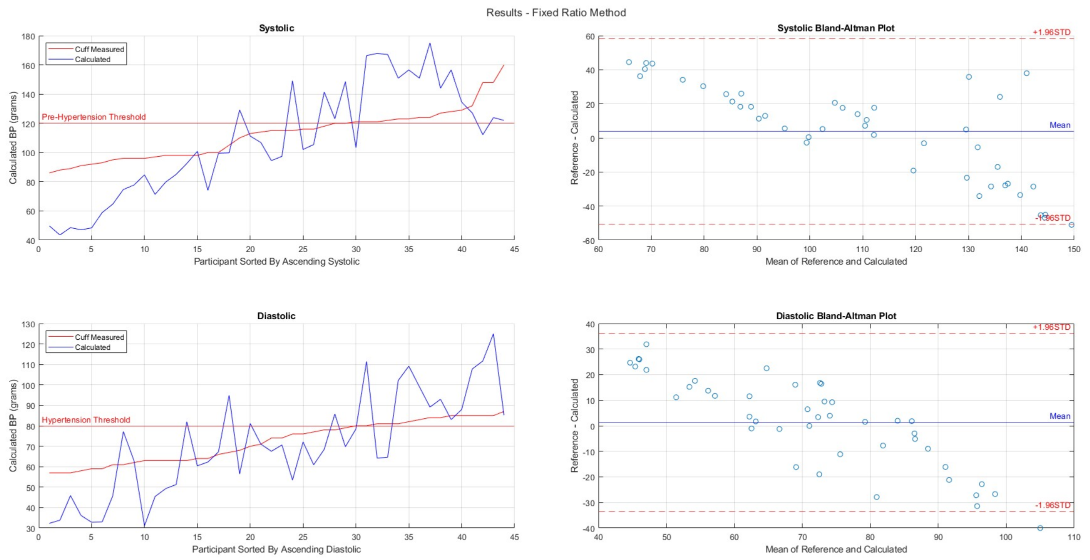

Fig. 5. Reference vs calculated comparison of the systolic and diastolic blood pressures along with the Bland-Altman plots analyzed using the fixed ratio method.

Figures 5 and Figure 6 plots the final results with the data sorted by increasing reference bp. The accompanying Bland-Altman plot [1] is also shown to visualize the differences between the collection of bp via the commercial cuff, and our device.

While the blood pressure calculated from our device does not necessarily yield a highly accurate output, the trend of the data does follow the reference blood pressures taken from the commercial bp monitor. The threshold for pre-hypertension of the systolic and hypertension of the diastolic is also relatively accurate. While there are outliers, you can see that the majority of the calculated blood pressures fell on the appropriate side of the threshold.

The Bland-Altman plots for both the fixed ratio and derivative methods would suggest that for both the systolic and diastolic there tends to be a bias in the lower blood pressures where the calculated is lower than the reference, with the reverse true in the higher blood pressures.

6 DISCUSSION

One flaw in this design is the fact that the pressure applied to the users' fingers in non consistent area which could account for variations in the calculated mmHg pressure. A traditional cuff based monitor encapsulates the entire arm or finger and the air pressure is used as a direct measurement. In our design we used a curved surface to allow for the best ergonomics and control of the pressure applied by the finger.

A previous design of this finger blood pressure device can be seen in the Appendix. In this previous iteration a raised fixed area is used so that a direct constant area is applied to all participants. However, it was difficult to design an enclosure with a usable raised area due to the fact that the MAX30102 sensor had to be mounted to a printed circuit board. The area used to convert the measured force to a pressure was the area of the devices contact to the FSR and not the area under the participant's finger. While the area differences in cm^2 between participants' finger areas is likely small it could account for some of the inaccuracies. For example the higher blood pressure data sets were taken from a participant with smaller fingers. The calculated systolic bp at the higher range did not follow the trend of the rest of the data, and a reduced finger area would have resulted in a higher calculated mmHg for that data set. This could also explain the lower blood pressure anomalies where the Bland-Altman plot indicated a bias of a lower calculated bp versus the reference since those data sets were also taken from participants with smaller fingers.

Use of the fixed ratio method of determining systolic and diastolic pressures while easy to compute, is not as accurate as other methods [4]. As shown in some of the early explorations in oscillometric methods [8], the ratio is actually range and dependant on the blood pressure. Our analysis used the typical ratio of 0.55 and 0.85 for systolic and diastolic respectively used by previous studies. The derivative method can be accurate, but relies on a clean data set[4].

Other designs such as that done by Chandrasekhar et al [3] did not use an integrated sensor and instead used discrete components to obtain a more favorable geometry. That project did note in a follow up paper [5] that the contact pressure of the finger to the sensor will also result in a variation in the amplitude of the ppg waveform. That factor likely plays some role in our numerical inaccuracies as well.

Another possible reason for the inaccuracies in the higher blood pressures could be from the non-linearity of the force sensor. Because the FSR follows a logarithmic curve, the dynamic range at higher forces decreases exponentially and the accuracy of the force measurements taken with the ppg signals will not be as accurate. Basically the higher the force, the higher the chance of a remapping error. This would essentially shift the oscillogram which would affect the final blood pressures calculated.

A much larger data set from more users would be required to improve the data correlation and validate this design. This project gathered data from mostly low to med-high blood pressure individuals so the upper blood pressures are not as accurately validated. However, the goal of detecting pre-hypertension does fall within the range of the majority of our data. We also only have data from n=6 participants while most of the other studies have at least a couple dozen. Our participant pool had a fairly large range from mid 20s to early 70s. It is likely that a higher participant pool across a larger range of demographics would help to normalize the data across different physical traits.

7 CONCLUSION

In conclusion, this project showed that this method of blood pressure measurement shows promise in helping to detect pre-hypertensive blood pressures. While the calculated values are not within acceptable error compared to the reference, the trend of the data is correlative and can be useful for detecting when a user reaches early pre-hypertensive blood pressure. Future work in refining the mechanical design or using a more robust algorithm on a larger data set could help with more accurate results.

REFERENCES

[1] Zach Bobbitt. 2021. *How to Create a Bland-Altman Plot in Excel.* Retrieved April 26, 2023 from https://www.statology.org/bland-altman-plot-excel/

[2] Nam Bui, Nhat Pham, Jessica Jacqueline Barnitz, Zhana Zou, Phuc Nguyen, Hoang Truong, Taeho Kim, Nicholas Farrow, Anh Nguyen, Jianliang Xiao, Robin Deterding, Thang Dinh, and Tam Vu. 2019. BP: A Wearable System For Frequent and Comfortable Blood Pressure Monitoring From User's Ear. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19).* ACM, New York, NY. https://doi.org/10.1145/3300061.3345454

[3] Anand Chandrasekhar, Chang-Sei Kim, Mohammed Naji, Keerthana Natarajan, Jin-Oh Hahn, and Ramakrishna Mukkamala. 2018. Smartphone-based blood pressure monitoring via the oscillometric finger-pressing method. In *Sci Transl Med.* American Association for the Advancement of Science. https://doi.org/doi:10.1126/scitranslmed.aap8674

[4] Anand Chandrasekhar, Mohammad Yavarimanesh, Jin-Oh Hahn, Shih-Hsien Sung, Chen-Huan Chen, Hao-Min Cheng, and Ramakrishna Mukkamala. 2019. Formulas to Explain Popular Oscillometric Blood Pressure Estimation Algorithms. *Frontiers in Physiology* 10 (2019).

https://doi.org/10.3389/fphys.2019.01415

[5] Anand Chandrasekhar, Mohammad Yavarimanesh, Keerthana Natarajan, Jin-Oh Hahn, and Ramakrishna Mukkamala. 2020. PPG Sensor Contact Pressure Should Be Taken Into Account for Cuff-Less Blood Pressure Measurement. *IEEE Transactions on Biomedical Engineering* 67, 11 (2020), 3134–3140. https://doi.org/10.1109/TBME.2020.2976989

[6] Mayo Clinic. 2022. *Blood pressure chart: What your reading means.* Retrieved April 26, 2023 from https://www.mayoclinic.org/diseases-conditions/high-blood-pressure/in-depth/blood-pressure/art-20050982

[7] Centers for Disease Control and Prevention. 2021. *Facts About Hypertension.* Retrieved May 16, 2022 from https://www.cdc.gov/bloodpressure/facts.htm

[8] L.A. Geddes, M Voelz, C Combs, D Reiner, and Charles F Babbs. 1982. Characterization of the Oscillometric Method for Measuring Indirect Blood Pressure. *Weldon School of Biomedical Engineering Faculty Publications* 66 (1982). http://docs.lib.purdue.edu/bmepubs/66

[9] Christian Holz and Edward Wang. 2017. Glabella: Continuously Sensing Blood Pressure Behavior using an Unobtrusive Wearable Device. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 1 (2017).

[10] Edward Jay Wang, Junyi Zhu, Mohit Jain, TienJui Lee, Elliot Saba, Lama Nachman, and Shwetak N. Patel. 2018. Seismo: Blood Pressure Monitoring using Built-in Smartphone Accelerometer and Camera. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018).

8 APPENDIX

A ADDITIONAL FIGURES

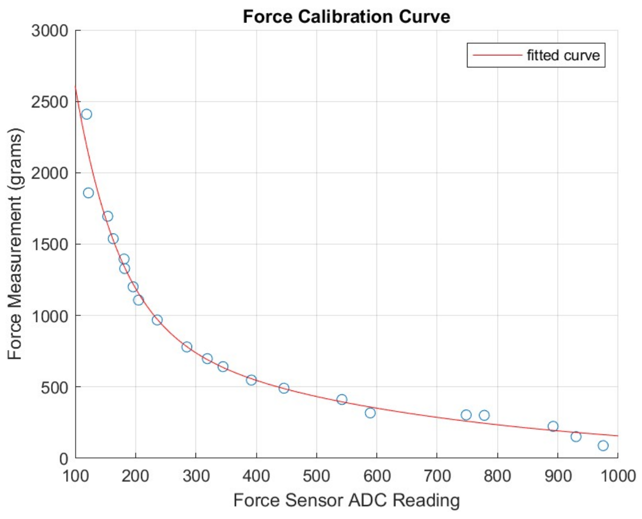

Fig. 7. The force v ADC reading exponential calibration curve.

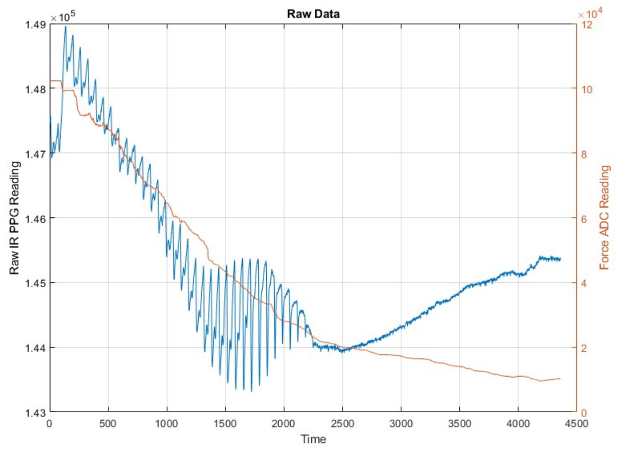

Fig. 8. The raw data collected from the infrared ppg sensor and force sensitive resistor.

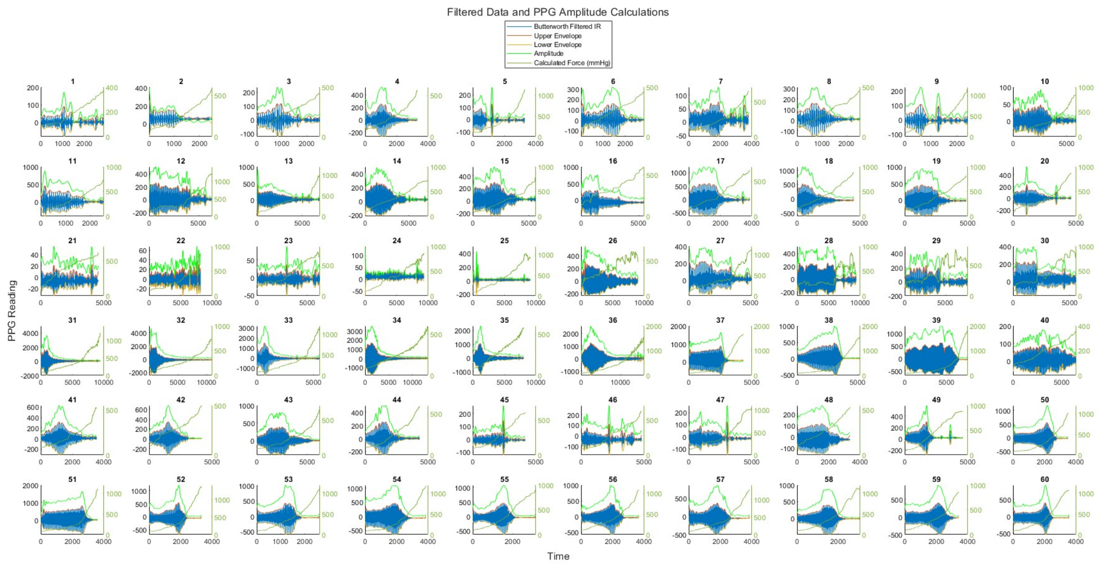

Fig. 9. The Butterworth band pass filtered data sets along with the upper and lower envelopes for ppg amplitude (and by proxy blood volume) calculations.

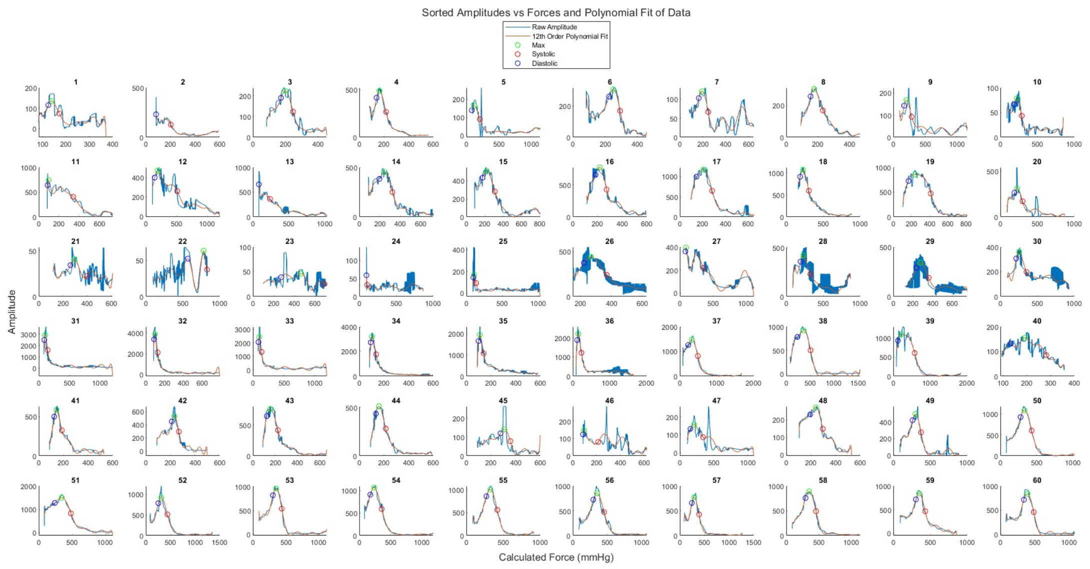

Fig. 10. Fixed Ratio Method. After the amplitude envelope is calculated, the forces are sorted with their amplitudes and a polynomial fit of the resultant data set is used to smooth the data. We then find the amplitudes and corresponding pressures that are the systolic and diastolic blood pressures based on the fixed ratio algorithm.

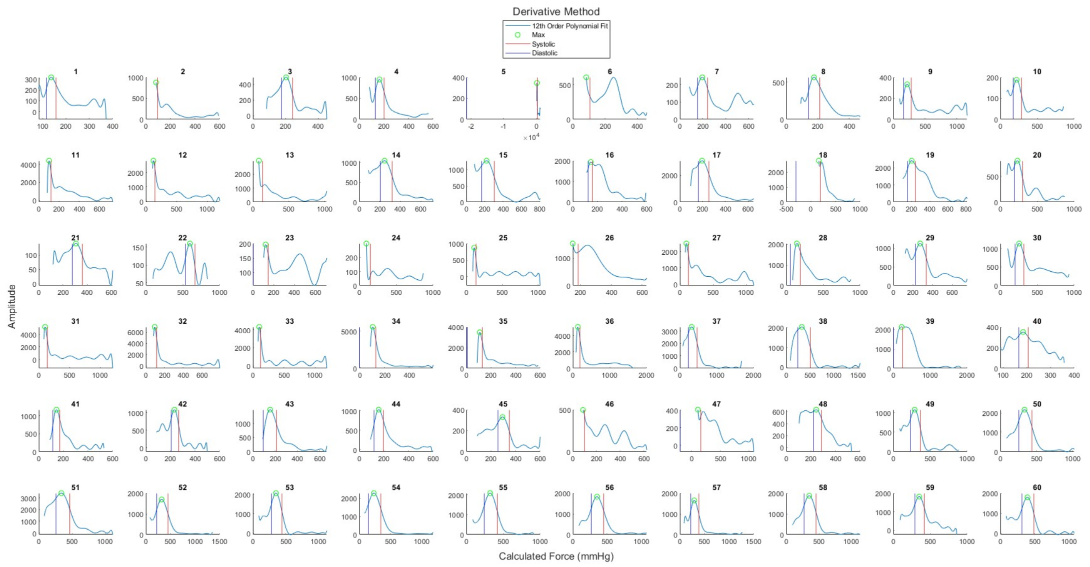

Fig. 11. Derivative Method. After the amplitude envelope is calculated, the forces are sorted with their amplitudes and a polynomial fit of the resultant data set is used to smooth the data. We then find the max and min slope leading up to the peak amplitude and corresponding pressures that are the systolic and diastolic blood pressures based on the derivative method.

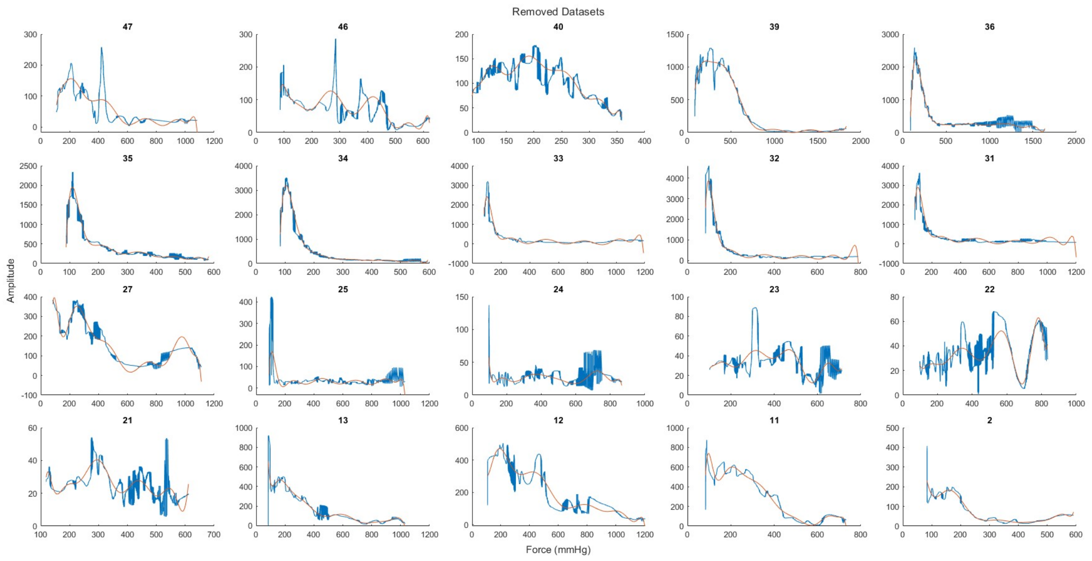

Fig. 12. These are the data sets that were removed due to bad data and bad polynomial fit

B DESIGN ITERATIONS

This project included a few different iterations of the hardware before settling on the configuration described in the write up above.

The finger blood pressure device, used a spring to act as the source of pressure on the finger for an oscillometric blood pressure measurement. Off the shelf blood pressure cuffs use a pump to inflate a band around a user's upper arm to a pressure above the upper pressure of an expected blood pressure reading. The cuff deflates and the oscillation amplitude back pressure caused by the user's blood pressure is what is measured and used to calculate the blood pressure. We mimic this procedure by using a spring's force against the user's finger.

$$F_s = -kx \tag{1}$$

Hooke's Law, as shown in the equation above, states the force exerted by a spring, F_s, is linearly proportional to the compressed distance, x, by the spring constant, k. This concept allows us to measure the force by measuring the distance instead of the force directly.

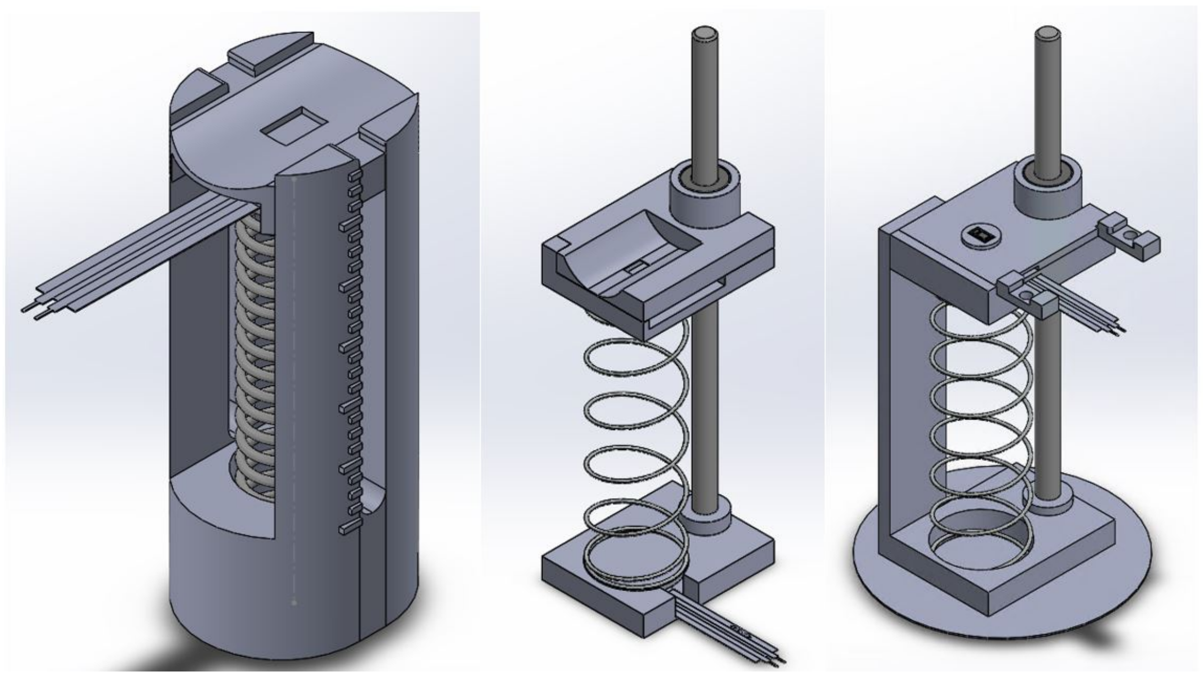

Fig. 13. Iterations of the mechanical design. Starting from a purely 3D printed device to using a bearing and shaft, to the addition of a distance sensor.

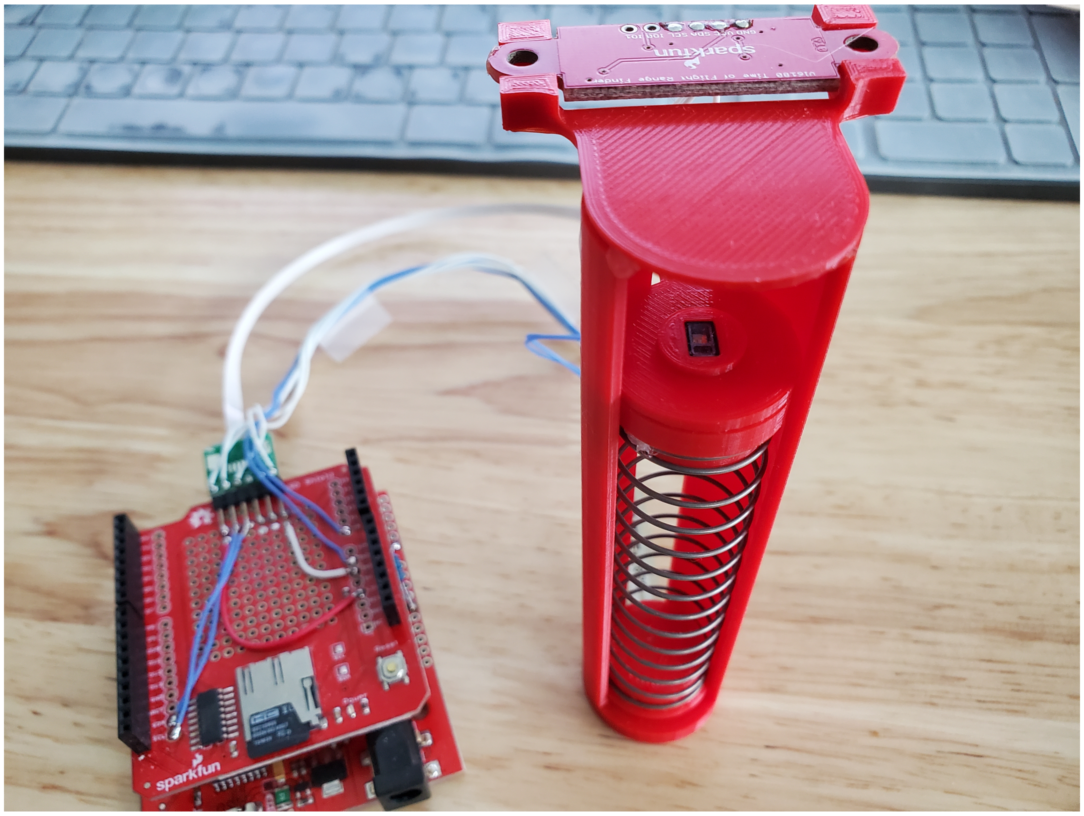

Fig. 14. Final iteration of the spring based device.

The mechanical design of this device are primarily composed of a base that the spring sat on and a finger module that housed the sensing electronics and where the finger rested. The design of the mechanical setup went through several iterations before a final design was decided on as can be seen in the figure above.

The first design used a 3D printed base that the module slide down on. While this was an easy to build design, consisting of only needing a 3d printer, there were several mechanical disadvantages. Because the vertical motion is guided just by the grove on base there was a lot of tilt which meant the spring wasn't always compressed parallel to it's axis.

The second design moved to using a linear shaft and a ball bearing to achieve smooth linear motion in line with the spring. However this was very unstable and while the motion was parallel to the spring axis, the device would very often rotate around the shaft axis.

The third design mounted the distance sensor to the side of the device and added a parallel guide so that the motion was more in line with the spring. This design proved mostly successful but was still difficult to use.

The final spring design did not use a shaft and instead pivoted to using a round external shell to guide the internal round depressor disk. This allows the system to be much cheaper and yielded pretty good mechanical actuation with minimal tilt. This also allowed for easier control as the user would grasp the device in one hand and squeeze instead of just depressing with the device on the table.

C PREVIOUS DATA COLLECTION

The procedure used to test this device is similar to a procedure from a consensus document by the AAMI, ESH, and ISO for validating blood pressure measuring devices. Because the goal of this study is to show the data collected by this device can be correlated to blood pressure and not to accurately measure blood pressure, some liberties were taken with the full procedure described below.

There was one observer who guided the study participant throughout the study with the following procedures.

(1) Use a commercially available off the shelf blood pressure cuff to take a reading off the participant's left arm. The majority of these measurements were done with an Omron Evolv wireless blood pressure monitor, with a Medline one used with one participant.
(2) Use our device with the participant's right index finger, placing the approximate middle of the distal phalanx onto the sensor. Because the waveform was plotted in real time, it allowed for some adjustment by the participant until a good ppg signal was seen. The participant would compress the spring to the most compressed state, then slowly lift their finger which mimics the inflation and deflation of a blood pressure monitor.
(3) Repeat step 2 for a total of 4 measurements. This is done mainly to help with bad readings due to the user contributing the change in pressure rather than an external force as done by the monitor cuff.
(4) Repeat steps 1-3 a total of 4 times for a total of 4 measurements with the Omron, and 16 measurements on Spring Pressure.

The data was then averaged among each participants' data set and a similar process described in Algorithm 2 was used. However because of the incomplete data in certain regions due to the poor data collection the correlation was very poor. This was in part due to the Arduino library used that applied a Butterworth filter that could not compensate for the large DC shifts.

D ARDUINO CODE

```
    /*
  MAX30105 Breakout: Output all the raw Red/IR/Green readings
  By: Nathan Seidle @ SparkFun Electronics
  Date: October 2nd, 2016
  https://github.com/sparkfun/MAX30105_Breakout

  Outputs all Red/IR/Green values.

  Hardware Connections (Breakoutboard to Arduino):
  -5V = 5V (3.3V is allowed)
  -GND = GND
  -SDA = A4 (or SDA)
  -SCL = A5 (or SCL)
  -INT = Not connected

 The MAX30105 Breakout can handle 5V or 3.3V I2C logic. We recommend powering the board with 5V
  but it will also run at 3.3V.

  This code is released under the [MIT License](http://opensource.org/licenses/MIT).
*/
```

```
/******************************************************************************
 * SparkFun_VL6180X_demo.ino
 * Example Sketch for VL6180x time of flight range finder.
 * Casey Kuhns @ SparkFun Electronics
 * 10/29/2014
 * https://github.com/sparkfun/SparkFun_ToF_Range_Finder-VL6180_Arduino_Library
 *
 * The VL6180x by ST micro is a time of flight range finder that
 * uses pulsed IR light to determine distances from object at close
 * range.  The average range of a sensor is between 0-200mm
 *
 * Resources:
 * This library uses the Arduino Wire.h to complete I2C transactions.
 *
 * Development environment specifics:
 *  IDE: Arduino 1.0.5
 *  Hardware Platform: Arduino Pro 3.3V/8MHz
 *  VL6180x Breakout Version: 1.0
 *  **Updated for Arduino 1.6.4 5/2015**

 *
 * This code is beerware. If you see me (or any other SparkFun employee) at the
 * local pub, and you've found our code helpful, please buy us a round!
 *
 * Distributed as-is; no warranty is given.
 ******************************************************************************/

#include <Wire.h>
#include "MAX30105.h"
#include <SPI.h>
#include <SD.h>
#include <SparkFun_VL6180X.h>

MAX30105 particleSensor;

#define debug Serial //Uncomment this line if you're using an Uno or ESP
//#define debug SerialUSB //Uncomment this line if you're using a SAMD21

#define VL6180X_ADDRESS 0x29
VL6180xIdentification identification;
VL6180x distanceSensor(VL6180X_ADDRESS);

const int chipSelect = 8; // Sparkfun MicroSD shield CS to D8
int pulseSensorPort = 0;
int forceData = 0;
int cardiogramData = 0;
int distanceData = 0;
uint32_t redPPG = 0;
uint32_t greenPPG = 0;
uint32_t irPPG = 0;

File dataFile;

void setup()
{
  debug.begin(115200);

//  debug.println("MAX30105 Basic Readings Example");

  // Initialize PPG Sensor /////////////////////////////////////////////////
  if (particleSensor.begin() == false)
  {
    debug.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }

//  particleSensor.setup(); //Configure sensor. Use 6.4mA for LED drive
//  particleSensor.setLEDMode(2);
//  particleSensor.setFIFOAverage(1);
//  particleSensor.setADCRange(8192);

  //Let's configure the sensor to run fast so we can over-run the buffer and cause an interrupt
   byte ledBrightness = 0x90; //Options: 0=0mA to 255=50mA
   byte sampleAverage = 1; //Options: 1, 2, 4, 8, 16, 32
   byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
   byte sampleRate = 200; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
   int pulseWidth = 69; //Options: 69, 118, 215, 411
   int adcRange = 16384; //Options: 2048, 4096, 8192, 16384

  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Configure


  // Initialize Distance Sensor /////////////////////////////////////////////////

  distanceSensor.getIdentification(&identification); // Retrieve manufacture info from device memory
  //printIdentification(&identification); // Helper function to print all the Module information

    if(distanceSensor.VL6180xInit() != 0){
    Serial.println("FAILED TO INITALIZE"); //Initialize device and check for errors
  };

  distanceSensor.VL6180xDefautSettings(); //Load default settings to get started.
```

```
//  Serial.println("Distance sensor initialized");

  // SD CARD INITIALIZATION /////////////////////////////////////////////////
  // SD code adapted from Datalogger Arduino example by Tom Igoe

  // see if the card is present and can be initialized:
  if (!SD.begin(BUILTIN_SDCARD)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while (1);
  }

  dataFile = SD.open("data.txt", FILE_WRITE);
  // Bug, can't seem to create new file. Change printed statement below to differentiate
  // between data collections.
  dataFile.println("Data Collection Started------------------------");
  dataFile.println("test");
  dataFile.close();

//  Serial.println("SD initialized");

  dataFile = SD.open("data.txt", FILE_WRITE);

  Wire.setClock(400000);

  delay(500);

}

long test = 0;
long test2 = 0;
int counter = 0;
long test3 = 0;

void loop()
{
//  test3 = millis();
//  test = micros();

//  distanceData = distanceSensor.getDistance();
  forceData = analogRead(A0);
//  debug.print("force distance: ");
//  test2 = millis();
//  debug.println(test2 - test);
//  test = millis();
```

```
  redPPG = particleSensor.getRed();
//  greenPPG = particleSensor.getGreen();
  irPPG = particleSensor.getIR();
//  test2 = millis();
//  debug.print("ppg: ");
//  debug.println(test2 - test);
//  test = micros();

  // Print Data to Serial Monitor
//  debug.print(greenPPG);
//  debug.print(",");
  debug.print(micros());
  debug.print(", ");
  debug.print(forceData*100);
  debug.print(", ");
  debug.print(redPPG);
  debug.print(", ");
  debug.print(irPPG);
//  debug.print(",");
//  debug.print(distanceData);
//  debug.print(",");
//  debug.print(forceData);
  debug.println();

//   Save to SD card
//  if(dataFile){
//    dataFile.print(redPPG);
//    dataFile.print(",");
//    dataFile.print(greenPPG);
//    dataFile.print(",");
//    dataFile.print(irPPG);
//    dataFile.print(",");
//    dataFile.print(distanceData);
//    dataFile.print(",");
//    dataFile.print(forceData);
//    dataFile.println();
//  } else{
//    debug.println("error sd file");
//  }
//  test2 = micros();
//  debug.print("sd: ");
//  debug.println(test2 - test);

//  counter = counter + 1;
//  if (counter > 1000){
```

```
//    dataFile.flush();
//    counter = 0;
//  }
//  debug.println(counter);

//  debug.print("time:");
//  debug.println(millis() - test3);
}
```

E MATLAB ANALYSIS CODE

```
clear all;
close all;
clc;

rawData{1} = csvread('1-93_63.csv');
rawData{2} = csvread('1-96_60.csv');
rawData{3} = csvread('1-96_62.csv');
rawData{4} = csvread('1-96_63_2.csv');
rawData{5} = csvread('1-97_63.csv');
rawData{6} = csvread('1-98_63.csv');
rawData{7} = csvread('1-98_64.csv');
rawData{8} = csvread('1-98_68.csv');
rawData{9} = csvread('1-100_64.csv');
rawData{10} = csvread('1-105_66.csv');
rawData{11} = csvread('2-100_69.csv');
rawData{12} = csvread('2-108_68.csv');
rawData{13} = csvread('2-109_76.csv');
rawData{14} = csvread('2-113_76.csv');
rawData{15} = csvread('2-114_81.csv');
rawData{16} = csvread('2-115_74.csv');
rawData{17} = csvread('2-115_76.csv');
rawData{18} = csvread('2-116_78.csv');
rawData{19} = csvread('2-118_77.csv');
rawData{20} = csvread('2-121_79.csv');
rawData{21} = csvread('3-131_92.csv');
rawData{22} = csvread('3-137_89.csv');
rawData{23} = csvread('3-137_94.csv');
rawData{24} = csvread('3-144_89.csv');
rawData{25} = csvread('3-144_92.csv');
rawData{26} = csvread('4-132_70.csv');
rawData{27} = csvread('4-146_72.csv');
rawData{28} = csvread('4-148_74.csv');
rawData{29} = csvread('4-148_78.csv');
rawData{30} = csvread('4-160_80.csv');
rawData{31} = csvread('5-86_59.csv');
```

```
rawData{32} = csvread('5-88_57.csv');
rawData{33} = csvread('5-89_59.csv');
rawData{34} = csvread('5-91_57.csv');
rawData{35} = csvread('5-92_58.csv');
rawData{36} = csvread('5-95_54.csv');
rawData{37} = csvread('6-121-85.csv');
rawData{38} = csvread('6-124_87.csv');
rawData{39} = csvread('6-132_92.csv');
rawData{40} = csvread('1-99_57.csv');
rawData{41} = csvread('1-95_61.csv');
rawData{42} = csvread('1-98_61.csv');
rawData{43} = csvread('1-96_57.csv');
rawData{44} = csvread('1-100_63.csv');
rawData{45} = csvread('2-118_70.csv');
rawData{46} = csvread('2-113_66.csv');
rawData{47} = csvread('2-111_69.csv');
rawData{48} = csvread('2-116_71.csv');
rawData{49} = csvread('2-110_67.csv');
rawData{50} = csvread('6-115_81.csv');
rawData{51} = csvread('6-121-85.csv');
rawData{52} = csvread('6-123_84.csv');
rawData{53} = csvread('6-124_80.csv');
rawData{54} = csvread('6-120_81.csv');
rawData{55} = csvread('6-127_83.csv');
rawData{56} = csvread('6-123_82.csv');
rawData{57} = csvread('6-129_84.csv');
rawData{58} = csvread('6-128_85.csv');
rawData{59} = csvread('6-120_85.csv');
rawData{60} = csvread('6-122_85.csv');

% rawData{1} = csvread('');
% rawData{2} = csvread('');
% rawData{3} = csvread('');
% rawData{4} = csvread('');
% rawData{5} = csvread('');
% rawData{6} = csvread('');
% rawData{7} = csvread('');
% rawData{8} = csvread('');
% rawData{9} = csvread('');
% rawData{0} = csvread('');

systolic = [93 96 96 96 97 98 98 98 100 105 100 108 109 113 114 ...
            115 115 116 118 121 131 137 137 144 144 132 146 148 148 160 ...
            86 88 89 91 92 95 121 124 132 99 95 98 96 100 118 ...
            113 111 116 110 115 121 123 124 120 127 123 129 128 120 122];
```

```
diastolic = [63 60 62 63 63 63 64 68 64 66 69 68 76 76 81 ...
            74 76 78 77 79 92 89 94 89 92 70 72 74 78 80 ...
            59 57 59 57 58 54 85 87 92 57 61 61 57 63 70 ...
            66 69 71 67 81 85 84 80 81 83 82 84 85 85 85];


processedData{1} = rawData{1}(434:3300,:);
processedData{2} = rawData{2}(495:3000,:);
processedData{3} = rawData{3}(84:2600,:);
processedData{4} = rawData{4}(291:3600,:);
processedData{5} = rawData{5}(220:3500,:);
processedData{6} = rawData{6}(150:3000,:);
processedData{7} = rawData{7}(133:3923,:);
processedData{8} = rawData{8}(450:3000,:);
processedData{9} = rawData{9}(591:3000,:);
processedData{10} = rawData{10}(1322:7200,:);
processedData{11} = rawData{11}(251:2800,:);
processedData{12} = rawData{12}(122:6500,:);
processedData{13} = rawData{13}(48:7000,:);
processedData{14} = rawData{14}(185:7400,:);
processedData{15} = rawData{15}(186:5800,:);
processedData{16} = rawData{16}(227:5700,:);
processedData{17} = rawData{17}(133:4000,:);
processedData{18} = rawData{18}(94:4600,:);
processedData{19} = rawData{19}(138:5000,:);
processedData{20} = rawData{20}(05:4600,:);
processedData{21} = rawData{21}(357:4900,:);
processedData{22} = rawData{22}(224:8350,:);
processedData{23} = rawData{23}(283:6540,:);
processedData{24} = rawData{24}(195:9500,:);
processedData{25} = rawData{25}(305:9400,:);
processedData{26} = rawData{26}(917:9920,:);
processedData{27} = rawData{27}(1342:6437,:);
processedData{28} = rawData{28}(137:9511,:);
processedData{29} = rawData{29}(447:8200,:);
processedData{30} = rawData{30}(541:6115,:);
processedData{31} = rawData{31}(135:9550,:);
processedData{32} = rawData{32}(50:11000,:);
processedData{33} = rawData{33}(64:5630,:);
processedData{34} = rawData{34}(287:13650,:);
processedData{35} = rawData{35}(743:8800,:);
processedData{36} = rawData{36}(5009:21000,:);
processedData{37} = rawData{37}(2888:7200,:);
processedData{38} = rawData{38}(300:5480,:);
processedData{39} = rawData{39}(7350:14900,:);
processedData{40} = rawData{40}(1700:8200,:);
```

```
processedData{41} = rawData{41}(1045:4570,:);
processedData{42} = rawData{42}(650:4800,:);
processedData{43} = rawData{43}(660:6900,:);
processedData{44} = rawData{44}(395:4700,:);
processedData{45} = rawData{45}(850:5000,:);
processedData{46} = rawData{46}(306:5100,:);
processedData{47} = rawData{47}(490:5500,:);
processedData{48} = rawData{48}(354:3700,:);
processedData{49} = rawData{49}(716:5300,:);
processedData{50} = rawData{50}(124:3600,:);
processedData{51} = rawData{51}(2800:6400,:);
processedData{52} = rawData{52}(182:3500,:);
processedData{53} = rawData{53}(176:2800,:);
processedData{54} = rawData{54}(152:4000,:);
processedData{55} = rawData{55}(150:3300,:);
processedData{56} = rawData{56}(134:3800,:);
processedData{57} = rawData{57}(157:4000,:);
processedData{58} = rawData{58}(222:3300,:);
processedData{59} = rawData{59}(180:3600,:);
processedData{60} = rawData{60}(149:3700,:);

% processedData{1} = rawData{1}(:,:);
% processedData{2} = rawData{2}(:,:);
% processedData{3} = rawData{3}(:,:);
% processedData{4} = rawData{4}(:,:);
% processedData{5} = rawData{5}(:,:);
% processedData{6} = rawData{6}(:,:);
% processedData{7} = rawData{7}(:,:);
% processedData{8} = rawData{8}(:,:);
% processedData{9} = rawData{9}(:,:);
% processedData{0} = rawData{0}(:,:);

peakDistance = [82 91 82 88 83 79 86 87 86 80 81 93 84 85 89 ...
                94 85 93 92 98 85 84 99 101 88 106 113 113 103 108 ...
                102 99 101 100 97 118 75 78 64 90 85 92 77 89 89 ...
                84 87 86 82 64 71 64 69 70 69 67 66 67 65 67];

% forceReading = [1023 1000 860 650 410 390 170];
% forceGram = [0 100 150 320 450 670 1075];
% forceRawToGram = fit(forceReading', forceGram', 'exp1');

forceReading = [975 930 892 778 748 589 542 446 392 345 319 285 236 205 196 182 181 163 154 122 119];
forceGram = [89 152 224 301 304 318 412 491 548 642 698 780 969 1108 1201 1329 1396 1538 1695 1859 2410];
% forceReading = [892 778 748 589 542 446 392 345 319 285 236 205 196 182 181 163 154 122 119];
% forceGram = [224 301 304 318 412 491 548 642 698 780 969 1108 1201 1329 1396 1538 1695 1859 2410];
forceRawToGram = fit(forceReading', forceGram', 'exp2');
```

```
figure;
hold on;
plot(forceReading, forceGram, 'o');
plot(forceRawToGram);
xlabel('Force Sensor ADC Reading');
ylabel('Force Measurement (grams)')
title('Force Calibration Curve');
grid on;

time = [0:pi/60:2*pi];
bufferSine = 10*sin(10*time);

fig = figure;
t = tiledlayout('flow');

for i = 1:length(processedData)
%     if(i == length(processedData)/2)
%         figure;
%         tiledlayout('flow');
%     end

    data = processedData{i};
    force = data(:,2);
    force = forceRawToGram(force/100); % Convert force reading to grams
    force = force / (pi*(1.3/2).^2) * 0.7355591352766806; % Convert g/cm2 to mmHg
    ppgR = data(:,3) - mean(data(:,3));
    ppgIR = data(:,4) - mean(data(:,4));

    fcL = 1.8;
    fcH = 4.3;
    % fcL = 0.5;
    % fcH = 10;
    fs = 1/(mean(diff(data(:,1)))/1000000);
    order = 8;

    [b a] = butter(order, fcL/(fs/2), 'high');
    butterHIR = filtfilt(b, a, ppgIR);
    [b a] = butter(order, fcH/(fs/2), 'low');
    butterIR = filtfilt(b, a, butterHIR);
    butterIR = butterIR - butterIR(1);
    filteredData{i} = butterIR;
    filteredForce{i} = force;
    butterIR = [bufferSine'; butterIR; bufferSine'];
%
%      highIR = highpass(ppgIR, 0.5, fs);
%      butterIR = lowpass(highIR, 10, fs);

    nexttile
%     hold on;
%     plot(highIR);

%     figure;
%     plot(butterR);
%     hold on;
%     plot(lowR);

    [up{i} lo{i}] = envelope(butterIR, peakDistance(i)-15, 'peak');
    butterIR = butterIR(length(bufferSine):end-length(bufferSine));
    up{i} = up{i}(length(bufferSine)+1:end-length(bufferSine));
    lo{i} = lo{i}(length(bufferSine)+1:end-length(bufferSine));
    hold on;
    plot(butterIR);
    plot(up{i});
    plot(lo{i});

    amplitude{i} = up{i} - lo{i};
%     amplitude{i}(amplitude{i} > (max(butterIR) - min(butterIR))) = 0;
    plot(amplitude{i},'g');

    [sortedForce{i} sortedIndex{i}] = sort(force, 'descend');
    sortedAmplitude{i} = amplitude{i}(sortedIndex{i});
    % plot(sortedAmplitude{i}, 'm');

    [p{i}, S{i}] = polyfit(sortedForce{i}, sortedAmplitude{i}, 12);
    Y{i} = polyval(p{i}, sortedForce{i});

    k{i} = polyder(polyder(p{i}));
    r{i} = roots(k{i});
    Z{i} = polyval(k{i}, sortedForce{i});


   % [curve{i}, goodness{i}, output{i}] = fit(sortedForce{i}, sortedAmplitude{i},'smoothingspline');
    % plot(curve{i});

    hold on;
    yyaxis right;
    plot(force);
    title(i);
end

leg = legend('Butterworth Filtered IR', 'Upper Envelope', 'Lower Envelope', 'Amplitude', 'Calculated Force (mmH
leg.Layout.Tile = 'north';

title(t, 'Filtered Data and PPG Amplitude Calculations')
xlabel(t, 'Time');
ylabel(t, 'PPG Reading');
yyaxis('right');

figure;
t = tiledlayout('flow')

systolicRatio = 0.55;
diastolicRatio = 0.85;
% systolicRatio = 0.61;
% diastolicRatio = 0.74;

for i = 1:length(processedData)
    nexttile;
    hold on;
    % plot(curve{i}, sortedForce{i}, sortedAmplitude{i});
    plot(sortedForce{i}, sortedAmplitude{i});
    plot(sortedForce{i}, Y{i});

    data = [sortedForce{i} Y{i}];

    [aMax(i) aMaxIndex(i)] = max(Y{i});
    plot(sortedForce{i}(aMaxIndex(i)), Y{i}(aMaxIndex(i)), 'go');

    aSystolic(i) = systolicRatio * aMax(i);
    [M, systolicIndex(i)] = min(abs(data(1:aMaxIndex(i),2) - aSystolic(i)));
    plot(sortedForce{i}(systolicIndex(i)), Y{i}(systolicIndex(i)), 'ro');
    systolicForce(i) = sortedForce{i}(systolicIndex(i));

    aDiastolic(i) = diastolicRatio * aMax(i);
    [M, diastolicIndex(i)] = min(abs(data(aMaxIndex(i):end,2) - aDiastolic(i)));
    diastolicIndex(i) = diastolicIndex(i) + aMaxIndex(i) - 1;
    plot(sortedForce{i}(diastolicIndex(i)), Y{i}(diastolicIndex(i)), 'bo');
    diastolicForce(i) = sortedForce{i}(diastolicIndex(i));

    title(i);
end

leg = legend('Raw Amplitude', '12th Order Polynomial Fit', 'Max', 'Systolic', 'Diastolic');
leg.Layout.Tile = 'north';
title(t, 'Fixed Ratio Method')
xlabel(t, 'Calculated Force (mmHg)');
ylabel(t, 'Amplitude');
```

```
% Derivative Method
figure;
t = tiledlayout('flow');
for i = 1:length(processedData)
    nexttile;
    hold on;
    plot(sortedForce{i}, Y{i});
    plot(sortedForce{i}(aMaxIndex(i)), Y{i}(aMaxIndex(i)), 'go');
    [aMax(i) aMaxIndex(i)] = max(Y{i});
    roots = sort(real(r{i}));
    maxForceIndex = sortedForce{i}(aMaxIndex(i));
    r1 = find(roots > maxForceIndex, 1, 'first');
    r2 = find(roots < maxForceIndex, 1, 'last');

    if r1
        systolicDerivative(i) = roots(r1);
        xline(systolicDerivative(i), 'r');
    end
    if r2
        diastolicDerivative(i) = roots(r2);
        xline(diastolicDerivative(i), 'b');
    end

    title(i);
end

leg = legend('12th Order Polynomial Fit', 'Max', 'Systolic', 'Diastolic');
leg.Layout.Tile = 'north';
title(t, 'Derivative Method')
xlabel(t, 'Calculated Force (mmHg)');
ylabel(t, 'Amplitude');


% Remove the datasets that contained what seems to be bad data.

figure;
t = tiledlayout('flow');
for i = [47, 46, 40, 39, 36, 35, 34, 33, 32, 31, 27, 25, 24, 23, 22, 21, 13, 12, 11, 2]
    nexttile;
    hold on;
    plot(sortedForce{i}, sortedAmplitude{i});
    plot(sortedForce{i}, Y{i});
    systolicForce(i) = [];
    systolicDerivative(i) = [];
    systolic(i) = [];
    diastolicForce(i) = [];
    diastolicDerivative(i) = [];
    diastolic(i) = [];
    title(i);
end
title(t,'Removed Datasets');
xlabel(t, 'Force (mmHg)');
ylabel(t, 'Amplitude');


% Sort Data by Reference BP
sortedSystolic = sortrows([systolic' systolicForce']);
sortedDiastolic = sortrows([diastolic' diastolicForce']);

sortedSystolicDerivative = sortrows([systolic' systolicDerivative']);
sortedDiastolicDerivative = sortrows([diastolic' diastolicDerivative']);

% Determine adjustment constant
% kSystolic = max(sortedSystolic(:,1)) / max(sortedSystolic(:,2));
% kDiastolic = max(sortedDiastolic(:,1)) / max(sortedDiastolic(:,2));

kSystolic = 175 / max(sortedSystolic(:,2));
kDiastolic = 125 / max(sortedDiastolic(:,2));

% Plot Fixed Ratio Method
figure;
t = tiledlayout('flow');

nexttile;
hold on;
plot(sortedSystolic(:,1),'r');
ylabel("BP Cuff Systolic");
% yyaxis right;
plot(sortedSystolic(:,2)*kSystolic,'b');
yline(120, '-r', 'Pre-Hypertension Threshold', 'LabelHorizontalAlignment', 'left');
xlabel("Participant Sorted By Ascending Systolic");
ylabel("Calculated BP (grams)");
title("Systolic");
legend('Cuff Measured', 'Calculated', 'Location', 'northwest');
grid on;

% nexttile;
% hold on;
% % plot(normalize(sortedSystolic(:,1),'range'), normalize(sortedSystolic(:,2),'range'), 'o');
% % plot(normalize(sortedDiastolic(:,1),'range'), normalize(sortedDiastolic(:,2),'range'), 'o');
%
```

```
% plot(sortedSystolic(:,1), sortedSystolic(:,2) * kSystolic, 'o');
% plot(sortedDiastolic(:,1), sortedDiastolic(:,2) * kDiastolic, 'o');
%
% plot([0:0.1:max(sortedSystolic(:,1))], [0:0.1:max(sortedSystolic(:,1))]);
% xlabel("BP Cuff Measured");
% ylabel("Calculated BP");
% title("Normalized Measured vs Calculated BP");
% legend('Systolic', 'Diastolic', 'Location', 'northwest');
% grid on;

nexttile;
hold on;
avgSystolic = (sortedSystolic(:,1) + sortedSystolic(:,2)*kSystolic)/2;
diffSystolic = sortedSystolic(:,1) - sortedSystolic(:,2)*kSystolic;
avgSystolicDiff = mean(diffSystolic);
lowerSystolic = avgSystolicDiff - 1.96*std(diffSystolic);
upperSystolic = avgSystolicDiff + 1.96*std(diffSystolic);

plot(avgSystolic, diffSystolic, 'o');
yline([avgSystolicDiff ], '-b', {'Mean'});
yline([lowerSystolic upperSystolic], '--r', {'-1.96STD', '+1.96STD'});
title("Systolic Bland-Altman Plot");
ylabel("Reference - Calculated");
xlabel("Mean of Reference and Calculated");
grid on

nexttile;
hold on;
plot(sortedDiastolic(:,1),'r');
ylabel("BP Cuff Diastolic (mmHg)");
% yyaxis right;
plot(sortedDiastolic(:,2)*kDiastolic,'b');
yline(80, '-r', 'Hypertension Threshold', 'LabelHorizontalAlignment', 'left');
xlabel("Participant Sorted By Ascending Diastolic");
ylabel("Calculated BP (grams)");
title("Diastolic");
legend('Cuff Measured', 'Calculated', 'Location', 'northwest');
grid on;

nexttile;
avgDiastolic = (sortedDiastolic(:,1) + sortedDiastolic(:,2)*kDiastolic)/2;
diffDiastolic = sortedDiastolic(:,1) - sortedDiastolic(:,2)*kDiastolic;
avgDiastolicDiff = mean(diffDiastolic);
lowerDiastolic = avgDiastolicDiff - 1.96*std(diffDiastolic);
upperDiastolic = avgDiastolicDiff + 1.96*std(diffDiastolic);
```

```
plot(avgDiastolic, diffDiastolic, 'o');
yline([avgDiastolicDiff ], '-b', {'Mean'});
yline([lowerDiastolic upperDiastolic], '--r', {'-1.96STD', '+1.96STD'});
title("Diastolic Bland-Altman Plot")
ylabel("Reference - Calculated");
xlabel("Mean of Reference and Calculated");
grid on;

title(t, "Results - Fixed Ratio Method");


% Plot Derivative
kSystolic = 175 / max(sortedSystolicDerivative(:,2));
kDiastolic = 125 / max(sortedDiastolicDerivative(:,2));

figure;
t = tiledlayout('flow');

nexttile;
hold on;
plot(sortedSystolicDerivative(:,1),'r');
ylabel("BP Cuff Systolic");
% yyaxis right;
plot(sortedSystolicDerivative(:,2)*kSystolic,'b');
yline(120, '-r', 'Pre-Hypertension Threshold', 'LabelHorizontalAlignment', 'left');
xlabel("Participant Sorted By Ascending Systolic");
ylabel("Calculated BP (grams)");
title("Systolic");
legend('Cuff Measured', 'Calculated', 'Location', 'northwest');
grid on;

nexttile;
hold on;
avgSystolic = (sortedSystolicDerivative(:,1) + sortedSystolicDerivative(:,2)*kSystolic)/2;
diffSystolic = sortedSystolicDerivative(:,1) - sortedSystolicDerivative(:,2)*kSystolic;
avgSystolicDiff = mean(diffSystolic);
lowerSystolic = avgSystolicDiff - 1.96*std(diffSystolic);
upperSystolic = avgSystolicDiff + 1.96*std(diffSystolic);

plot(avgSystolic, diffSystolic, 'o');
yline([avgSystolicDiff ], '-b', {'Mean'});
yline([lowerSystolic upperSystolic], '--r', {'-1.96STD', '+1.96STD'});
title("Systolic Bland-Altman Plot");
ylabel("Reference - Calculated");
xlabel("Mean of Reference and Calculated");
grid on
```

```
nexttile;
hold on;
plot(sortedDiastolicDerivative(:,1),'r');
ylabel("BP Cuff Diastolic (mmHg)");
% yyaxis right;
plot(sortedDiastolicDerivative(:,2)*kDiastolic,'b');
yline(80, '-r', 'Hypertension Threshold', 'LabelHorizontalAlignment', 'left');
xlabel("Participant Sorted By Ascending Diastolic");
ylabel("Calculated BP (grams)");
title("Diastolic");
legend('Cuff Measured', 'Calculated', 'Location', 'northwest');
grid on;

nexttile;
avgDiastolic = (sortedDiastolicDerivative(:,1) + sortedDiastolicDerivative(:,2)*kDiastolic)/2;
diffDiastolic = sortedDiastolicDerivative(:,1) - sortedDiastolicDerivative(:,2)*kDiastolic;
avgDiastolicDiff = mean(diffDiastolic);
lowerDiastolic = avgDiastolicDiff - 1.96*std(diffDiastolic);
upperDiastolic = avgDiastolicDiff + 1.96*std(diffDiastolic);

plot(avgDiastolic, diffDiastolic, 'o');
yline([avgDiastolicDiff ], '-b', {'Mean'});
yline([lowerDiastolic upperDiastolic], '--r', {'-1.96STD', '+1.96STD'});
title("Diastolic Bland-Altman Plot")
ylabel("Reference - Calculated");
xlabel("Mean of Reference and Calculated");
grid on;

title(t, "Results - Derivative Method");
```