Lab 5 – Blood Pressure Measurement System

GOAL

Demonstrate a system that measures mean arterial pressure (MAP).

OBJECTIVES

1) Build the electronics to measure the air pressure from a pressure cuff.
2) Develop the Arduino data acquisition code.
3) Develop the MATLAB interface that processes data and displays mean arterial pressure (MAP).

GENERAL GUIDELINES

1) Each student must build his/her own circuit.
2) Students are allowed (even encouraged) to help each other. Of course, Buma will be around to provide assistance as well.

REQUIRED PARTS AND MATERIALS

Fig. 1: Pressure cuff (left) and MPX2050GP sensor (right).

- Lab kit, benchtop power supply, test probe kit
- Scope, Arduino + USB cable
- Pressure cuff + short length of 1/8” inner diameter rubber hose + female Luer-to-Barb hose connector + 3” diameter PVC tube
- MPX2050GP piezoresistive pressure sensor (one)
- AD620 instrumentation amplifier (one)
- Resistors: 270 ohm resistor (red/purple/brown) (one)
  1.0 kohm resistor (brown/black/red) (one)
  5.6 kohm resistor (green/blue/red) (one)

PART 0: INTRODUCTION

The blood pressure measurement system will use a pressure cuff, the MPX2050GP sensor, one AD620 instrumentation amplifier, the Arduino for data acquisition, and MATLAB for data processing. Here's how it works:

1) The pressure cuff:
    a. Inflate until blood flow is cut-off from the arm.
    b. Then, the cuff pressure slowly decreases over roughly 40 seconds.
2) The sensor monitors the cuff pressure.
    a. The first AD620 amplifier increases the sensor signal.
3) The amplifier output is recorded by the Arduino and sent to a computer.
4) The computer processes the data (using MATLAB).
    a. The first program imports data from the Arduino.
    b. The second program analyzes the voltage waveform to determine the MAP.

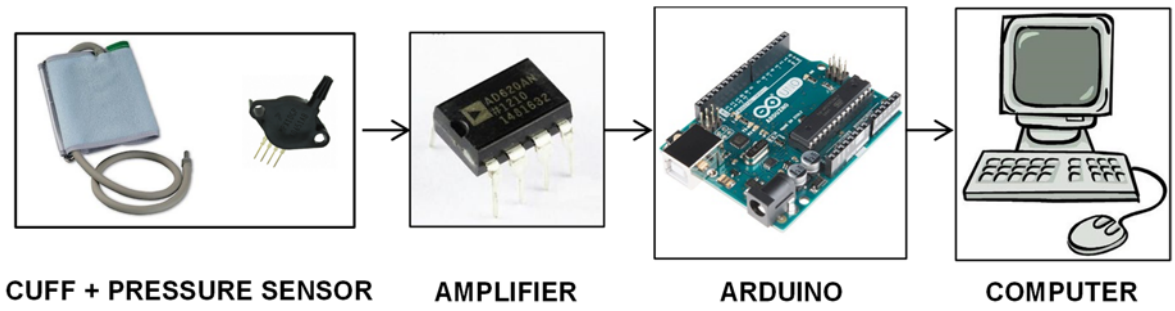


Fig. 2: Simplified block diagram of the blood pressure measurement system.

PART 1: PRESSURE SENSOR AND ELECTRONICS

The first part of the circuit is shown below. The pin diagrams for the sensor and AD620 are shown in Fig. 4.

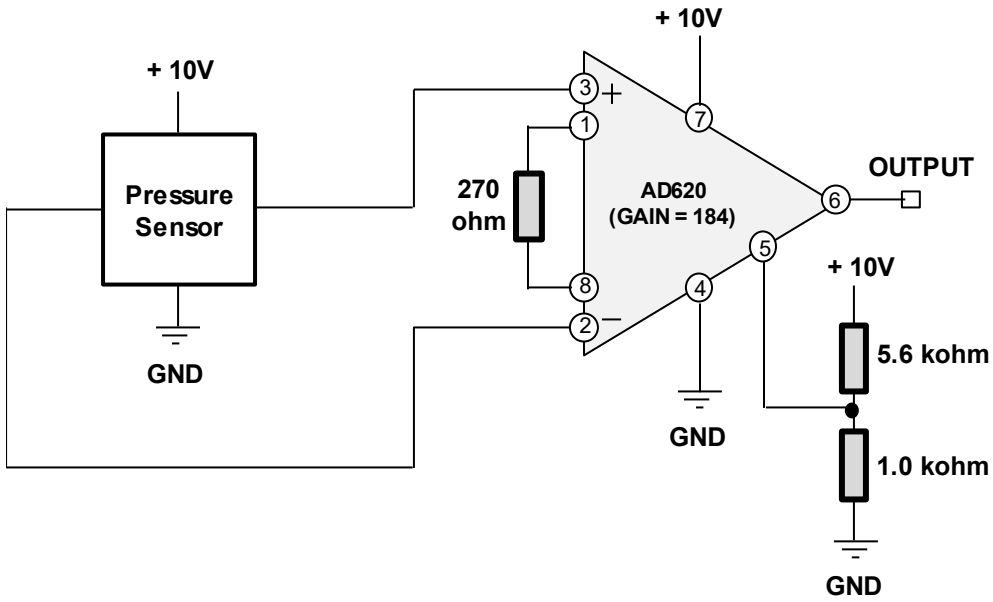


Fig. 3: First part of the signal conditioning electronics.

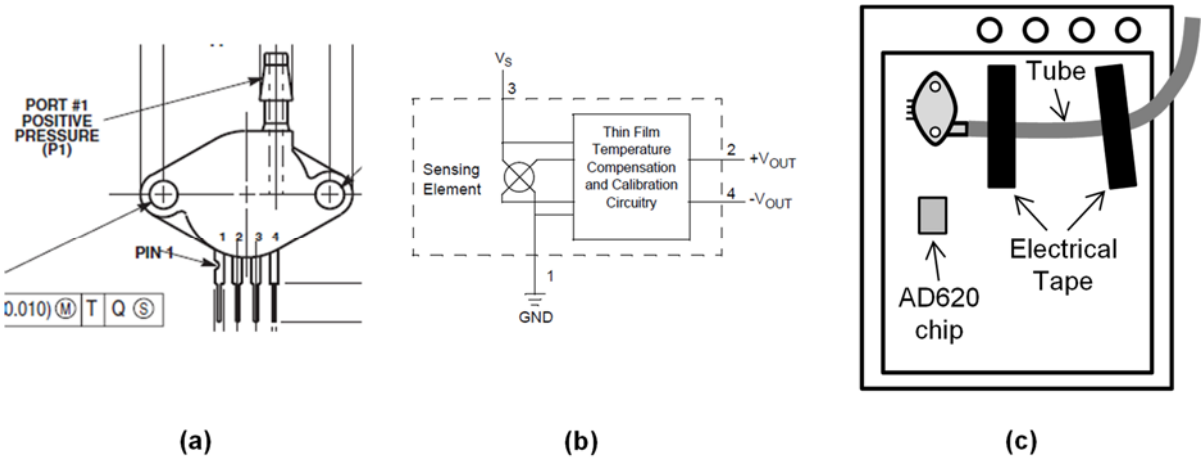
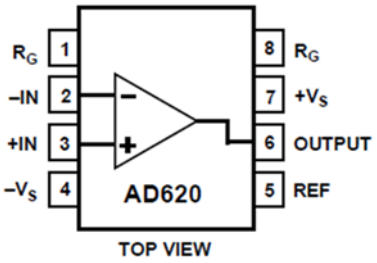


Fig. 4: (a) External appearance of the MPX2050GP pressure sensor. Pin 1 is FARTHEST away from the nozzle. (b) Simplified block diagram of the internal electronics of the sensor. (c) Suggested location of the sensor and AD620 chip on the breadboard. Later on you will attach a rubber tube to the sensor and secure it to the breadboard with electrical tape.

- **Step 1a**: Install the MPX2050GP pressure sensor.
  - The pin diagram for the sensor is shown in Fig. 4(a).
    - The sensor leads are bent to allow the sensor to be placed flat against the breadboard.
  - Place the sensor in the upper left corner of the breadboard (see Fig. 4c).

- Use NEAT and COLOR-CODED wiring!
  - Red = +10V Black = GND Yellow = other
  - Feel free to consult Buma's board as an example.

- **Step 1b**: Install the instrumentation amplifier.
  - Place this chip somewhere below the sensor (see Fig. 4c).
  - The pin diagram for the amplifier is shown to the right.
    - The gain resistor R_G is 270 ohm, which should produce a differential gain $A_d = 184$.
    - The AD620 is powered by +10V and GND.
    - Remember to connect Pin5 to the 5.6 kohm and 1 kohm resistors.



- **Step 1c**: Set up the pressure cuff.
  - Lightly wrap the pressure cuff around the 3-inch diameter PVC tube.
    - Do not wrap the cuff too tightly (slight wiggle room is fine).
    - This is only for initial testing of your system.
      - **Press the gray valve button near the bulb and squeeze out all the air from the cuff**.
  - Attach the pressure cuff to the sensor chip.
    - Attach the exposed end of the short rubber hose to the sensor's input barb.
    - Attach the short rubber hose's female luer connector to the pressure cuff's male luer connector.
    - Secure the rubber tubing to the breadboard with electrical tape (see Fig. 4c).
      - This prevents the tubing from pulling the sensor off the breadboard!

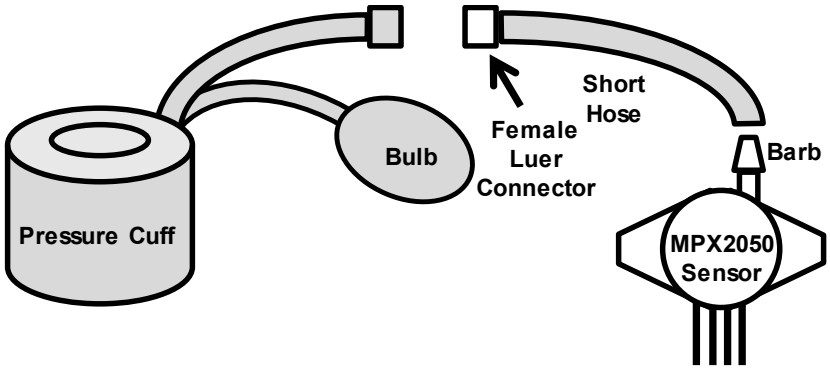


Fig. 5: Hose connections between pressure cuff and sensor.

- **Step 1d**: Hook up the benchtop power supply to provide +10V and GND to the breadboard.
  - You should (and better) know how to do this by now.

- **Step 1e**: Configure the scope.
  - Attach a scope probe to the amplifier output (Pin 6 of AD620 chip).

- Turn on the scope and press "Default Setup" to reset the scope. If necessary, press the "**Force**" button to enable local (e.g. Front Panel) control.
- Dial in the following scope settings:
  - Make sure both the scope and the probe are set to "1X"
  - Vertical settings:
    - Vertical scale (large knob) = 1 V/div.
    - Offset (small knob) = -3V to make 0V in the lower half of the screen (see Fig. 6).
  - Horizontal settings:
    - Scale (large knob) = 5 sec/div
    - Adjust the offset (small knob) to move the orange "T" marker near the left edge of the screen (see Fig. 6).
  - Press the CURSORS button:
    - Select Mode >> Manual
    - Select Type >> Amplitude
    - Adjust Cursor A to be at +4.7V (see Fig. 6).
    - This cursor makes it MUCH easier to know when to stop inflating the cuff!
  - Push the "TRIG MENU" button, go to "SOURCE" and select "AC Line".
  - Press the "Run/Stop" button so that it appears RED.

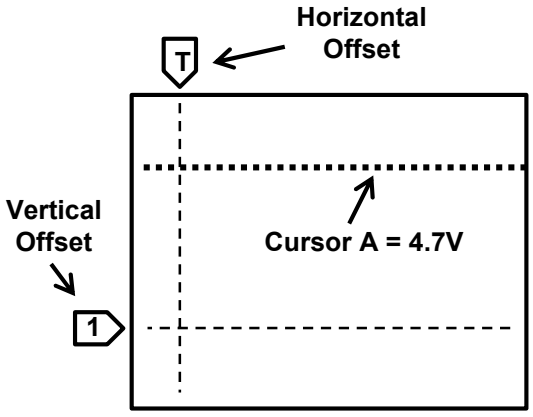


Fig. 6: The vertical and horizontal offsets on your scope should look something like this.

- **Step 1f**: Test the electronics.
  - Press the "SINGLE" button on the scope, wait 5 seconds, then start inflating the cuff (roughly two squeezes per second) until the voltage just crosses the 4.7V cursor line, **then stop and let the cuff slowly deflate**. Some comments:
    - Make sure the cuff has no air at the beginning (i.e. press the gray valve and squish the cuff).
    - After you stop inflating, you should hear a faint hissing sound as the pressure slowly drops.

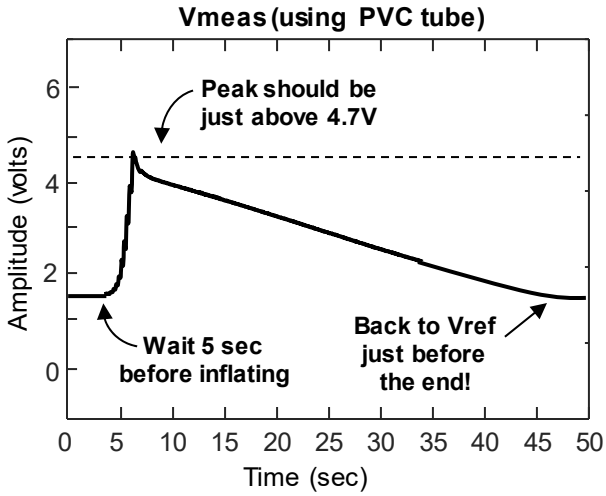


Fig. 7: The cuff inflation and deflation should resemble this. Stop inflating when the peak just crosses 4.7V. The end of the trace should return to baseline before the end.

- **Vmeas should return to baseline a few seconds before the end of the trace** (see Fig. 7).
  - If the cuff deflates much too fast or slow, let Buma know (he can make adjustments).
  - At the end of each test, make sure you get rid of all air from within the cuff.

- **Step 1g**: Connect the Arduino to your circuit with TWO wires.
  - Yellow wire = (PinA0 to AD620 output) Black wire = (Arduino GND to breadboard GND)
- **Step 1h**: Keep the scope probe connected to your circuit.
  - Watching the scope waveform will help you properly inflate the cuff on your arm (later in lab).

OK, now you're done with the hardware! Moving on to programming …

(End of Part 1)

PART 2: ARDUINO

For this lab, we want to record 50 seconds of data. We'll choose a sampling interval of dt = 50 ms and N = 1000 samples.

- **Step 2a**: Starting with last week's Arduino code, modify it to do the following:
  (1) Time between readings is T = 50 milliseconds.
  (2) Number of samples per trace is N = 1000.
- **Step 2b**: Upload your code and observe the Arduino output using the **Serial Monitor** on the computer.
  - Make sure the bottom right of the window is set to "57600 baud".
  - You should notice the letter "**a**" on the first line.
  - Type in the letter "**y**" in the command line and press the **Send** button.
    - You should see a rapid burst of voltage values when you send 'y'. This will last about 50 seconds, so it takes a while to finish.
    - **The Arduino's yellow Tx LED should be blinking rapidly during the data burst**.
  - Save your program under a different name (e.g. YourName_Lab5_Arduino).

(End of Part 2)

PART 3: MATLAB (2 programs)

This week, we will have twice the fun and write TWO MATLAB programs! ☺

1) The first program will acquire data from the Arduino.
   - This program is run ONCE during an experiment to obtain raw voltage data.
2) The second program analyzes the voltage waveform to obtain the mean arterial pressure (MAP).
   - This program can be re-run multiple times to tweak various parameters (e.g. Vref) in the signal processing.
   - This two-step approach avoids the need to re-do the entire cuff inflation/deflation experiment multiple times – your arm will thank you for this.

- **Step 3.1**: Starting with your MATLAB code from last week, make the following changes:
  - The number of samples per trace is **N** = 1000.
  - The sampling interval is **dt** = 0.050 seconds.
  - Change **Vdata** to **Vmeas**.
  - Since we are performing a SINGLE data acquisition, we can make a plot AFTER data has been acquired.
    - Remove the lines of code involving **h=figure, plot, drawnow**, **q=0**, and **set(h,'ButtonDownFcn')**.
    - Remove the **while (q = = 0)** loop (but keep the code inside the loop).
    - Remove the **drawnow** command (since we are not looping the data acquisition and display).

- **Step 3.2**: Perform a trial data acquisition:
  - Press the "SINGLE" button in the top right of the scope.
  - Next, start your MATLAB code -- you should see the Arduino start to blink rapidly, which means it is happily recording data!
  - **WAIT AT LEAST 5 SECONDS**, then start inflating the pressure cuff while watching the scope. **Stop inflating when the scope voltage crosses 4.7V**. Then stop and leave everything alone for about a minute.
    - Hopefully you will see a figure like Fig. 8a.
    - **Demo your code to Buma**.
    - **Save this program under a different name (e.g. YourName_Lab5_DataAcquisition).**

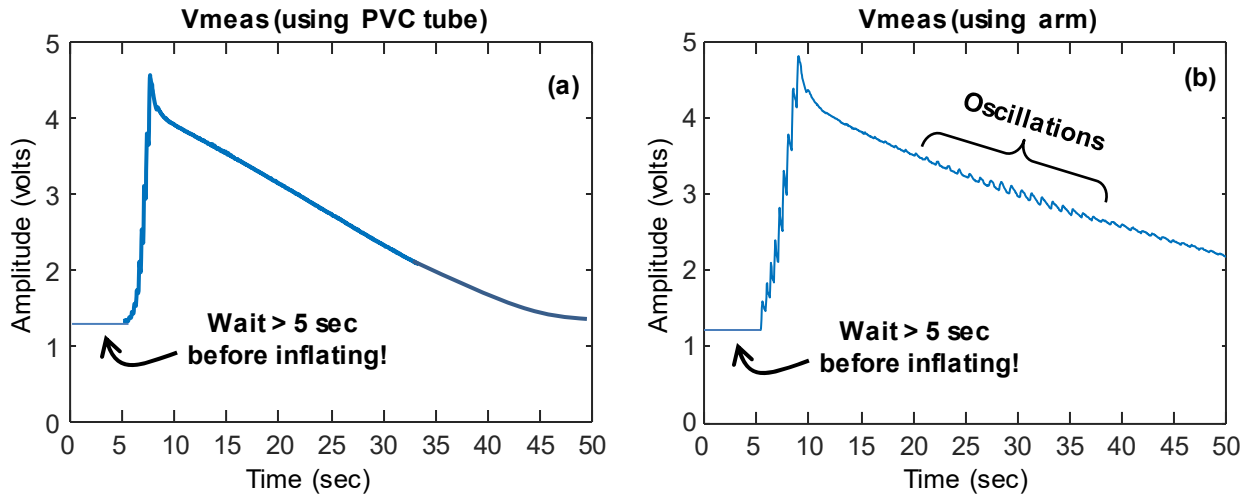


Fig. 8: (a) Your Arduino and MATLAB data acquisition programs work if you get a plot like this using the PVC tube. Make sure you wait at least 5 seconds before inflating the cuff! (b) You should get a plot like this when you use your own arm!

- **Step 3.3**: Make a voltage measurement with the pressure cuff on your arm.
  - Snugly wrap the cuff around your upper arm -- make sure no air is in the cuff!
  - Run your MATLAB data acquisition code.
    - **WAIT AT LEAST 5 SECONDS** before inflating the cuff – stop inflating when the scope voltage crosses 4.7V!
      - NOTE: Vmeas = 4.7V corresponds to a cuff pressure of about 150 mmHg.
    - Squeeze the bulb a couple times per second -- inflation should be done in less than 10 sec.
    - **REMAIN VERY STILL AND RELAXED** for the next 40 seconds (until the code is done). You should get something like Fig. 8b.
    - Now that you have your data, remove the cuff from your arm.

- **Step 3.4**: Create a new MATLAB program (Editor >> New >> Script) to obtain the MAP.
  - The basic signal processing steps to obtain the mean arterial pressure (MAP) are the following:
    (1) Convert **Vmeas** to cuff pressure **Pcuff**.
    (2) Apply a digital high-pass filter to **Pcuff** to isolate the oscillations.
    (3) Find the location of the peak oscillation in the filtered **Pcuff** signal.
    (4) The **MAP** is the value of **Pcuff** corresponding to this location!
  - First, we need to convert **Vmeas** to cuff pressure **Pcuff**:
    - Define the values for sensor sensitivity **S** (V/mmHg) and amplifier gain **Ad**.

- Define the reference voltage **Vref** as 1.7 for now (tweak this later).
  - Convert to **Vmeas** to pressure **Pcuff** using your formula from PreLab5.
- Make a new figure showing a plot of **Pcuff** versus time.
- Run your new program (NOT the first MATLAB program).
  - You should get a plot that looks similar to Fig. 9.
  - Adjust the value of **Vref** and re-run the code until the initial cuff pressure is ZERO.

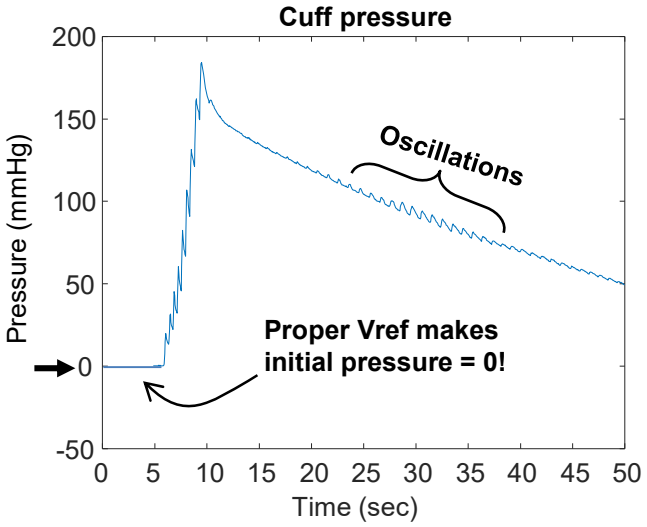


Fig. 9: Your plot of Pcuff vs time should look something like this after the proper value of Vref has been determined.

- Second, we need to isolate the cuff oscillations by sending the **Pcuff** waveform through a 4th order Butterworth digital high-pass filter. This removes the slow pressure decline and highlights pulses.
  - Use the **[b, a] = butter(order, Wn, ftype)** command (see MATLAB help documentation) to design a digital high-pass filter with the following parameters:
    - Filter order is 4.
    - The digital frequency is defined by **Wn = fc/(0.5*fs)**.
      - We want the corner frequency to be **fc** = 0.5 Hz.
      - NOTE: The sampling frequency **fs** is related to the sampling interval **dt**.
    - Filter type is **'high'**.
  - Now that we've designed our filter, apply it to the **Pcuff** signal by using the **filtfilt** command.
    - Do something like: **Pcuff_HPF = filtfilt(b, a, Pcuff);**
  - Make a new plot that shows **Pcuff_HPF**.
    - Do something like: plot(Pcuff_HPF);
    - This plots the waveform versus sample index (rather than time). This makes it easier to select certain samples for peak analysis.
    - You should end up with a waveform that resembles Fig. 10.

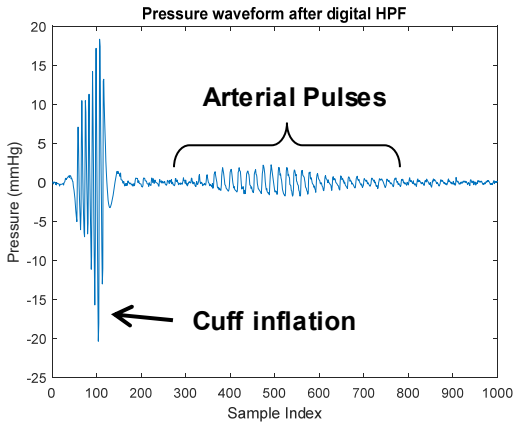


Fig. 10: Pressure waveform after digital high-pass filter.

- Third, find the array index of the maximum oscillation:
  - Use the **max** command to find the INDEX of the max oscillation in the DESIRED REGION of the **Pcuff_HPF** waveform. For example:
    - [val, ind] = max(vector) puts the max value of "vector" into "val" and the index of the max value into "ind".
    - NOTE: This is not the best way to determine MAP, but it will suffice for this lab.
- Fourth, determine the MAP!
  - Now that you have the index within the Pcuff_HPF waveform, select the appropriate value of the original **Pcuff** waveform!
  - Make a new figure showing a plot of **Pcuff and Pcuff_HPF versus TIME (see Fig. 11).**
    - For plotting purposes, you need to define a new time vector.
    - NOTE: Make sure you use the EXACT SAME sample index values (e.g. 200:end) for the time vector, Pcuff, and Pcuff_HPF waveforms!
  - If you want the MAP value to appear in your plot, do something like:
    - **title(sprintf('MAP = %3.1f mmHg', MAP));**

- **Step 3.5**: Run your **second** MATLAB code (NOT the first data acquisition code).
  - Two figure windows should appear.
    - The first figure shows a plot of cuff pressure vs time (like Fig. 9)
      - Make sure pressure = 0 at the beginning!
    - The second figure shows both descending cuff pressure and oscillations (Fig. 11).
      - Hopefully you get a MAP of roughly 100 mmHg.
      - If your MAP seems disturbingly high, do NOT worry! Our MAP algorithm is not very accurate, and many factors can influence your MAP (e.g. caffeine, stress).

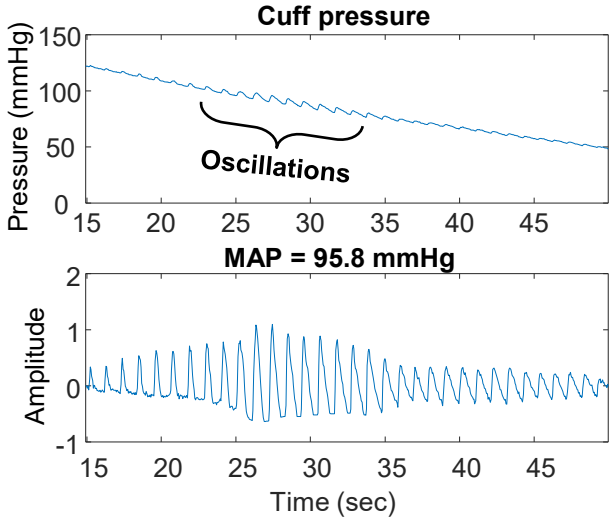


Fig. 11: Your MAP processing code should produce a plot showing the deflating portions of each waveform. Buma's MAP value doesn't look that great -- seems like he needs to do some exercise!

  - **Save the two figures (your version of Fig. 9 and Fig. 11) for your lab report and show to Buma**.
    - Buma will tabulate the MAP values and declare the class winner (lowest MAP).

(End of Lab5)