Tubular Structures

Compliance Testing Device

Device Protocol

THE BIOFOUNDRY

A MEINIG SCHOOL LAB

Prepared by: Ave Kludze (akk86@cornell.edu)

Mentor: Anthony R. D'Amato (ard223@cornell.edu)

Research Lab: Biofoundry (Yadong Wang Lab)

DATE: 5/22/23

Table of Contents

Hardware:

Wiring Diagram

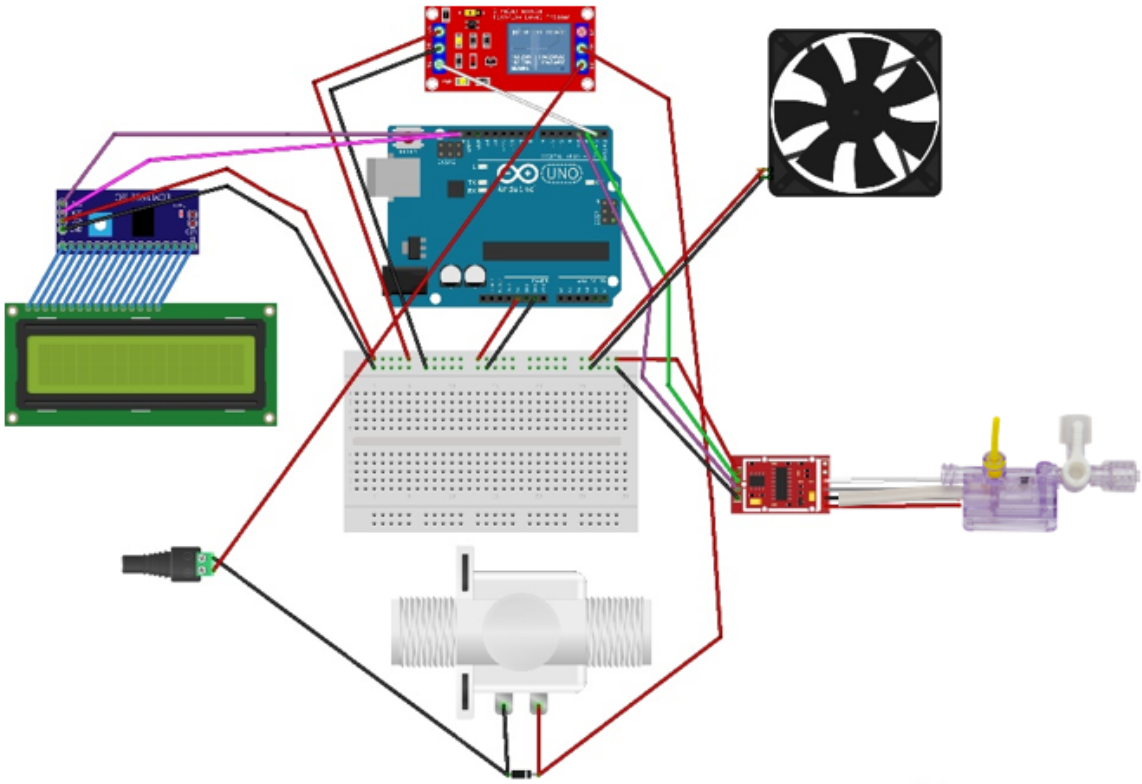


Figure 1. Wiring diagram of the pressure measurement component

Wiring Components (Explained)

- **Arduino Uno:** programmable open-source microcontroller board that functions as a mini-computer
    - | Purpose for Project: control circuit components according to computer code
- **Ball Valve:** flow control device which uses a pivoting ball to control fluid flowing through it
    - | Purpose for Project: control water flow and build pressure within the device
- **Edwards Truwave Pressure Transducer:** clinical pressure transducer that relays blood pressure information
    - | Purpose for Project: records pressure information in mmHg and transmits data to Arduino
- **HX711 Weighing Sensor Module:** 24-bit analog-to-digital converter (ADC) that is designed for weighing scales and industrial control applications to interface (differential amplifier)
    - | Purpose for Project: to increase the strength of the signal from the pressure transducer (e.g., the pressure transducer's signal is in microvolts and Arduino only reads signals in volts)

- **5V Relay Module:** The 5V relay module can be used to control a load such as a lighting system, motor, or solenoid. It can also be used to switch AC or DC voltages
    - | Purpose for Project: acts as a switch that turns off and on the power supply to the ball valve
- **LCD screen:** a type of flat panel display that uses liquid crystals in its primary form of operation
    - | Purpose for Project: functions as an external display system for water pressure in mmHg
- **LCM1602 IIC:** an adapter IIC Serial Interface for the LCD screen
    - | Purpose for Project: acts as an adaptor that simplifies LCD wiring to the Arduino
- **2.1mm Barrel Jack with Terminal Block:**
    - | Purpose for Project: acts as a plug for 12V power supply
- **Fan: 5V DC fan**
    - | Purpose for Project: functions as a ventilator to prevent potential overheating of the device
- **Diode:** a semiconductor device that essentially acts as a one-way switch for current
    - | Purpose for Project: functions to prevent frying of relay-circuitry, voltage-spikes, and back emf

Note: For a full table of all items used with vendor name, part number, and visuals see Part List

Reduced Bill of Materials with Key Components/Totals Only

Component	Qty	Total Cost (USD)
Arduino Uno	1	28.5
Motorized ball Valve ¼”	1	34.9
Truwave Pressure Transducer	1	34.95
HX711 Load Cell Amplifier	1	3.7
HiLetgo 1pc 5V One Channel Relay Module Relay Switch	1	3.7
5V DC Cooling Fan	1	3.99
GeeekPi I2C 1602 LCD Display Module	1	5.99
12V DC Power Connector (Barrel Jack)	1	2.99
VIVOSUN 480GPH Submersible Pump (1800L/H, 25W)	1	16.99
Solderless Prototype PCB Breadboard	1	0.99
Total Cost Key Components		136.7

*Laser micrometer is not included in this calculation

Flow Diagram (PI&D Symbols)

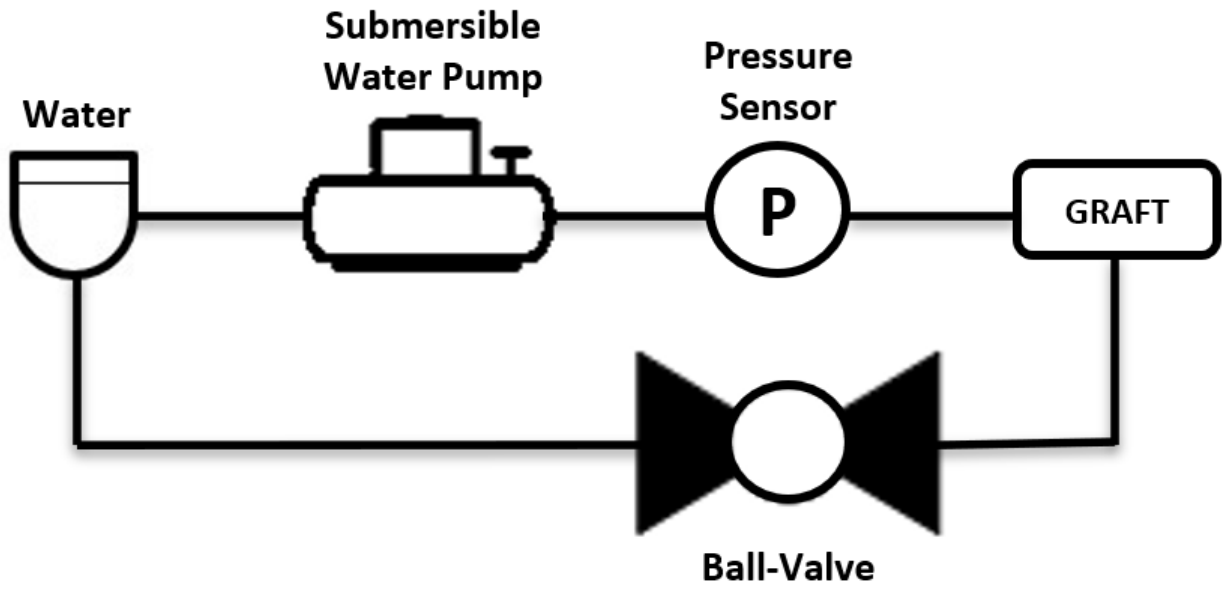


Figure 2. PI&D diagram of water flow

This is a simplified diagram only depicting the path of water flow through the compliance testing device. The water first flows from the submersible water pump to the pressure sensor, graft, and lastly to the ball valve. The water eventually flows back into the reservoir once the ball valve is open. The process is repeated as water exits from the submersible water pump and it is repeated for cycles.

Components All (Explained)

- Heat Shrink Tubing: increase mechanical support and provide heat insulation
- 6 mm vinyl tubing and 0.5” vinyl tubing: connect tubing components
- Arduino USB Cable: connect Arduino to the computer
- 12 V DC Power Supply: supply 12 V DC power to the ball-valve
- Grip Cable Glands: prevent tugging/pulling and damaging of circuit components
- XT60 Plug Female and Male Connector: connect ball-valve to circuit and diode

HX711 (Differential Amplifier) & Edwards-Truwave

The HX711’s primary purpose is to amplify signals from pressure transducers or load cells and transmit these signals to a microcontroller such as the Arduino Uno. To obtain a pressure signal, the HX711 functions as a differential amplifier to get the difference between the two voltage pins and amplify the signal. This is necessary since the pressure transducer reads in the microvolt range (sensitivity 5.0μ V/V/mm Hg ± 1%) whereas the Arduino uno reads in the volt range.

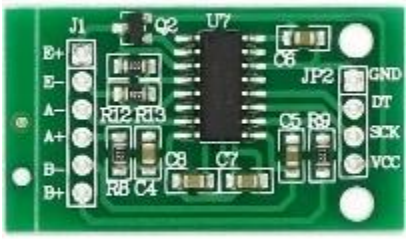


Figure 3. HX711 - Load Cell Amplifier 24-Bit ADC Converter

To acquire a signal, 4 of the 5 output pins are needed on the Edwards Truwave pressure transducer. These outputs are VCC, GND, SIGNAL+, and SIGNAL-, where pressure is proportional to (SIGNAL+ minus SIGNAL-). The pressure transducer appeared to be a strain gauge-type bridge with four connections. The fifth connection was likely a screen for the cable. The output locations are proprietary information. However, the exact output locations were identified using resistance measurements between pairs of wires (e.g, V+ and V- or Signal+ and Signal-). The electrical connections between the HX711, Edwards-Pressure Transducer, and Arduino Uno are displayed in the diagram and table below.

Figure 4. Pin map for Edwards-Pressure Transducer

Table 1. Electrical connections of the Edwards-Pressure Transducer to HX711 module

Wires	Pressure Transducer Electrical Connections	HX711 Electrical Connections
Wire 1	**V+ (5V)**	**E+**
Wire 2	**SIGNAL+**	**A+**
Wire 3	**SIGNAL-**	**A-**
Wire 4	**V- (GND)**	**E-**
Wire 5	—--	

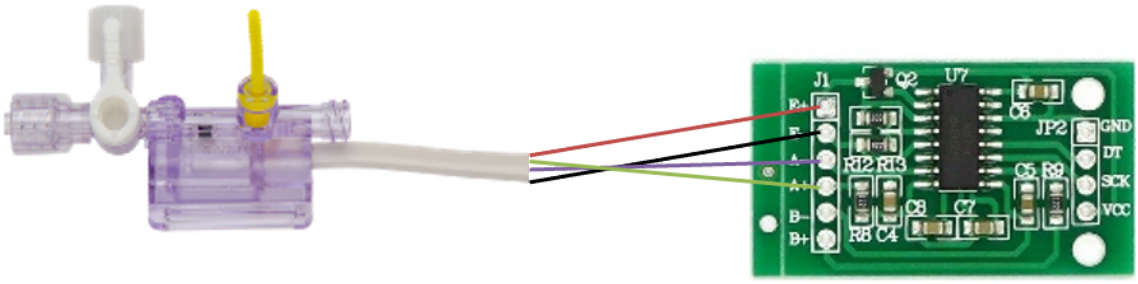


Figure 5. Pin map for Edwards-Pressure Transducer to HX711

The HX711 pins GND, DT, SCK, and VCC have standard connections to the Arduino Uno. For completeness, GND was connected to the ground on the breadboard, VCC was connected to the 5 volts on the breadboard, and SCK was connected to digital pin 2, and DT was connected to digital pin 3.

HX711 (Sampling Rate)

The default sample rate of the HX711 is 10Hz. To increase the sampling rate to 80Hz, a hardware change on the HX711 must be made. It should be noted that different HX711 versions (e.g, SparksFun, Stemedu, AiTrip) may require different hardware changes such as a solder pad. The HX711 used in this project (DIYmall) requires lifting or cutting the pad of pin 15, and then soldering pin 15 to pin 16. The diagram below shows the location of the hardware change, and the change is indicated in red. For a visual aid, see this video.

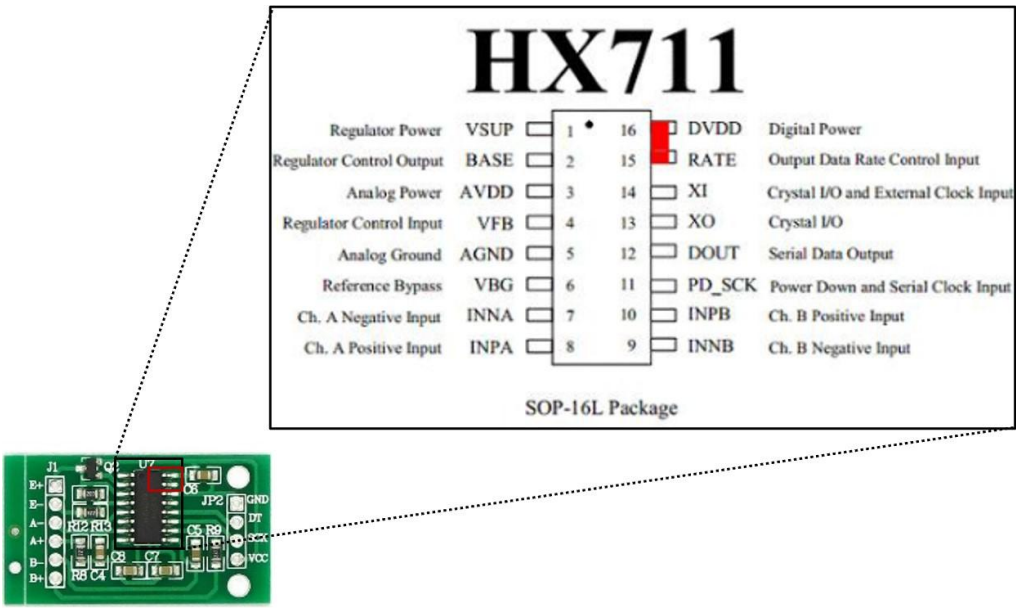


Figure 6. Location of hardware modification to HX711

Pressure Calibration (Edwards Truwave Pressure Sensor)

The pressure transducer was calibrated using a baumanometer that reads from 0 mmHg to 300 mmHg. The calibration curve was obtained from the pressure gauge and load signal (ADC value) readings. The ADC value is a digital signal read from Arduino which corresponds to a pressure value. The pressure values were recorded in increments of 20 mmHg with the corresponding ADC value. The operational range for pressure acquisition is 0–300 mmHg.

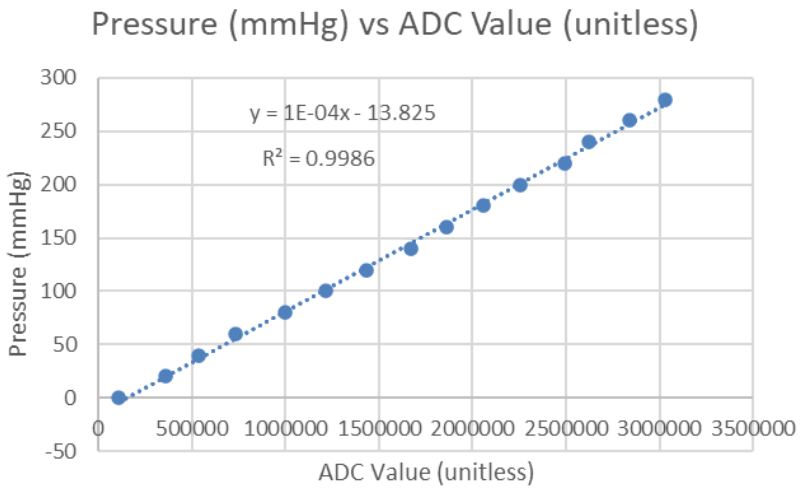


Figure 7. The ADC value recorded by the Arduino Uno and the corresponding pressure values in mmHg. The R2 coefficient shows a strong linear correlation and the regression line is used to calibrate the device.

The coefficient of determination of the linear least-squares fit the data, $R2 > 0.99$, indicating very strong linearity and a lack of noise.

3D-Printing (Fusion Autodesk)

An enclosure was created with Autodesk Fusion to enclose the electrical components and the Arduino Uno. Gates were made for USB connector, 12 V DC Power Supply, pressure sensor wires, ball-valve wires, and the LCD. Additionally, a screwable lid was created to fix the enclosure. The cable grip glands were super-glued in the enclosure to prevent tugging of the pressure transducer. The 12 V DC barrel jack was super-glued in order to avoid the moving of the power supply components. Lastly, both the LCD and relay switch were screwed onto the enclosure.

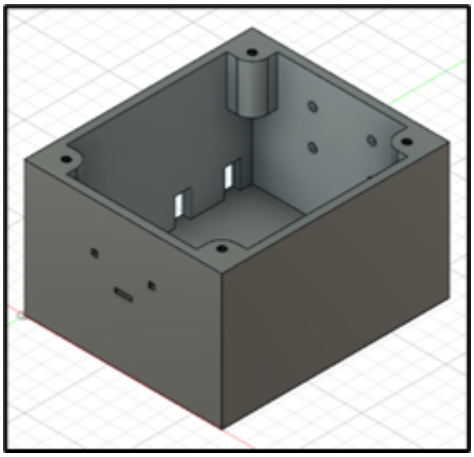


Figure 8. Print orientation for the 3D-printed enclosure. The enclosure is 117 mm in length, 135 mm in width, and 79.763 mm in height.

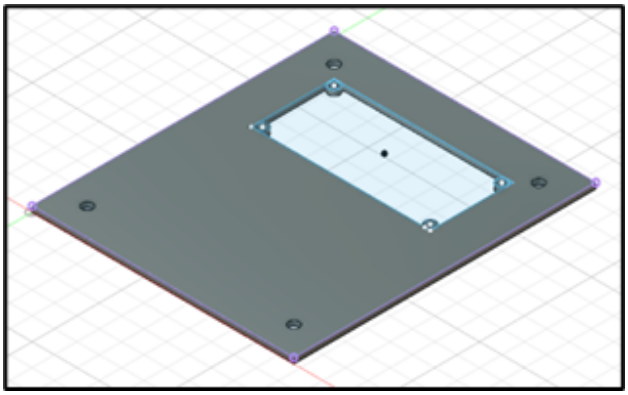


Figure 9. Print orientation for the 3D printed cover. The cover is 117 mm in length, 135 mm in width, and 2.381 mm in height.

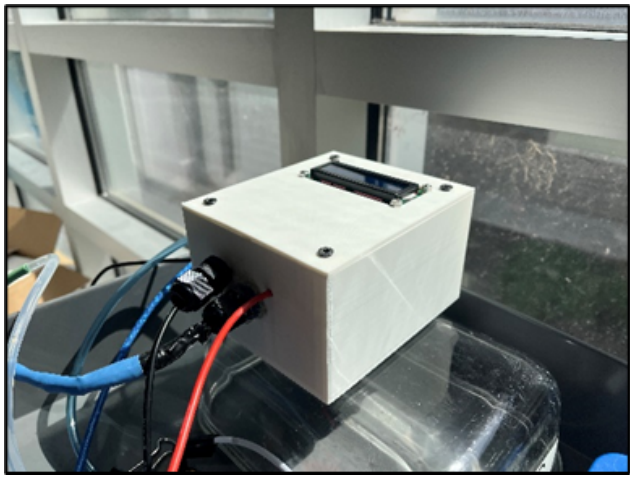


Figure 10. Final 3D-printed enclosure and lid configuration

Build Instructions Summarized

The main steps for the construction of the device are:

1. Order an Edwards-Truwave Pressure Transducer and the wiring components listed.
2. Print the enclosure and cover lid using a 3D printer and the associated Autodesk Fusion files.
3. Isolate the top four wires on the Edwards-Truwave Pressure Transducer. Cut the plastic between each wire so that each wire pin is isolated from the other.
4. Perform a hardware change as indicated in the section of HX711 (Sampling Rate) which will increase the sampling rate to 80 Hz.
5. Solder the wires of each pin to the HX711 using the configuration in Figure 5 and then connect the HX711 to the Arduino Uno.
6. Calibrate the Edwards-Truwave Pressure Transducer using a baumanometer as indicated the Pressure Calibration section.
7. Connect the hardware components as indicated in the Wiring Diagram section.
8. Glue the breadboard, Arduino Uno, and 12 V barrel jack to the bottom of the 3D-printed enclosure.
9. Screw the LCD to the cover lid and the 5V relay module to the enclosure.
10. Glue the four wires of the Edwards-Truwave Pressure Transducer while leaving a gap between each wire.
11. Electrical tape the four wires to prevent potential tugging and damage of the wires.
12. Glue the grip cable glands into the 3D-printed enclosure. Connect the Edwards-Truwave Pressure Transducer wires to the grip cable gland and then tighten the grip cable gland.
13. Assemble the cover lid and enclosure by screwing both parts together.

Figure 11. LCD screwed onto cover lid and connected to electrical circuit

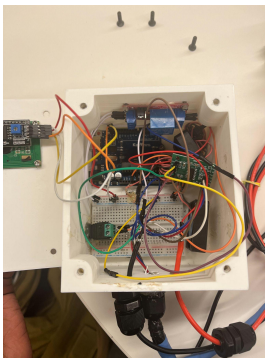


Figure 12. Inside the 3D-printed enclosure (top-view)

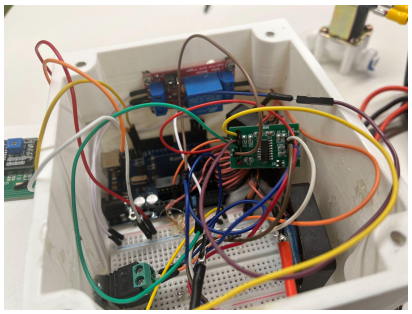


Figure 13. Inside the 3D-printed enclosure (side-view)

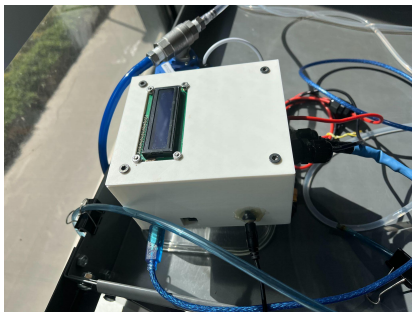


Figure 14. 3D-printed enclosure and cover lid connected together showing connections to 12V power supply and Arduino Uno

Software:

Arduino:

The Arduino Uno is an open-source microcontroller that processes analog and digital inputs and output pins. It is programmed in C++ within the Arduino Integrated Development Environmental (IDE) to program inputs and outputs from electronic components. It is powered through the Arduino USB cable which supplies 5V power to the board once it is connected to both the board and the user's computer.

The Arduino communicates through a serial board/monitor or serial plot. The Arduino code will be read and then loaded onto the Arduino Uno once the USB is connected to both the computer and the board. In this project, the Arduino Uno functions to process signals from the Edwards-Truwave Pressure Transducer and display these reads through the serial monitor/plot (e.g, pressure versus time). It also controls the LCD and ball-valve (valve-function versus time) as well as other electrical components listed in the Wiring Diagram. The programming libraries used in this project include HX711.h, Wire.h, and LiquidCrystal_I2C.h. The program will first calculate the water pressure, display the pressure on the LCD, open/close the ball-valve based on water-pressure, and then repeat the process.

Figure 15. Photograph of Arduino IDE with the loaded project program file.

PUTTY:

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers (putty.org). In this project, PUTTY is used to read data from the Arduino Serial monitor and save them onto a .csv file.

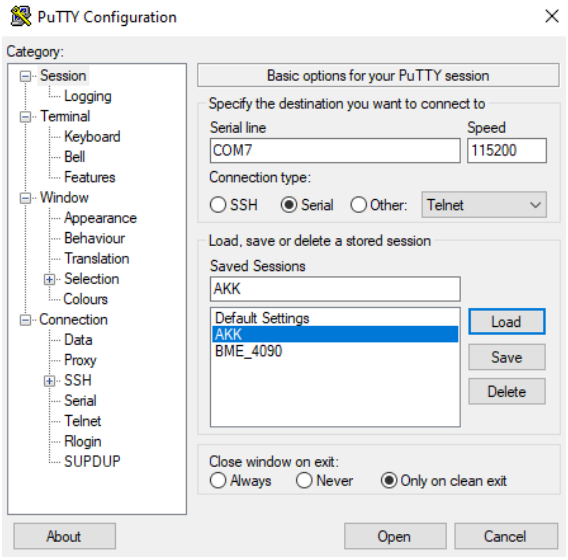


Figure 16. PuTTY Configuration

LabChart Pro (AD-INSTRUMENTS):

LabChart physiological data analysis software records data from devices to acquire biological signals from multiple sources. In this project, LabChart Pro was used to record diameter versus time and save them onto a .xlsx file.

MATLAB:

MATLAB is a programming language that works with numeric data. In this project, MATLAB was used to sync data from the Arduino UNO (Pressure, Valve-Step Function vs. Time) and LabChart Pro (Diameter vs. Time). The program will first load .csv data from the Arduino and .xlsx data from the laser micrometer. The program will sort data into a usable form by removing unnecessary rows and then assign headers to each column. The program will find peaks in both datasets, sync data, and lastly adjust for a time delay based on peak data. The program will find peaks for data once it's been synced and then calculate compliance. Lastly, the program will plot the sync data.

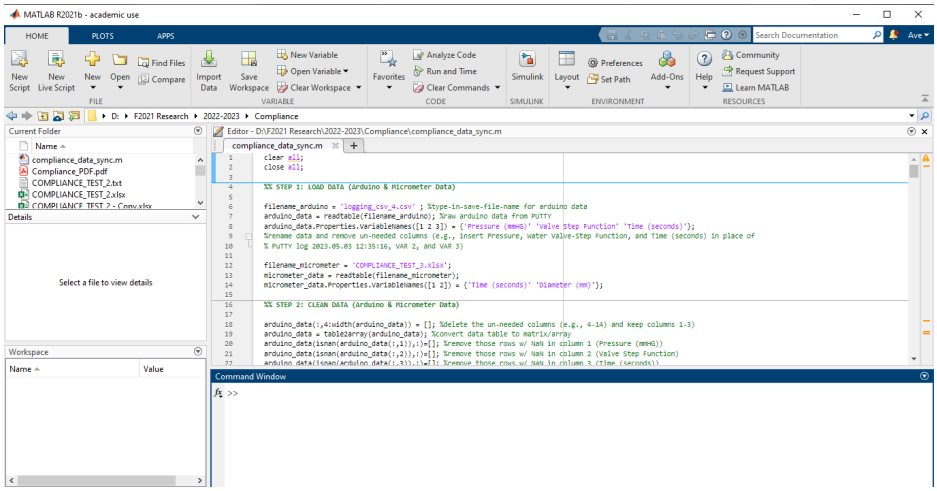


Figure 17. MATLAB Configuration

Design Files Summary:

Table 2. List of design files

Design filename	File type	Open source license	Link
TEBV+Cover+-+final+-+new.stl	3D-printable (.stl)	MIT License	3D-Printed-Cover
TEBV+Encasing.stl	3D-printable (.stl)	MIT License	3D-Printed-Encasing
serial_plott_water_valve_and_TruWave_pressure_sensor_HX711_lcd_.ino	Arduino project (.ino)	MIT License	Arduino-Code-Sensor
water_valve_relay_test_1.ino	Arduino project (.ino)	MIT License	Arduino-Code-Relay-Test
compliance_data_sync.m	MATLAB project (.m)	MIT License	MATLAB-Code
20180627_CETS_Compliance.adiset	LabChart Pro project (.adiset)	MIT License	Compliance-Code
TEBV_Circuit_Diagram.fzz	Fritzing file (.fzz)	MIT License	Fritzing-File

Folder Locations (Software Files): https://drive.google.com/drive/folders/1alEJ_aO36uzibLPPDvTpYj_1aO54k-lP?usp=share_link

Data Acquisition/Analysis:

The pressure and valve-step function versus time is processed from the Arduino Uno and then to PuTTY. The diameter versus time data is processed from LabChart. Both datasets are read and then processed into MATLAB.

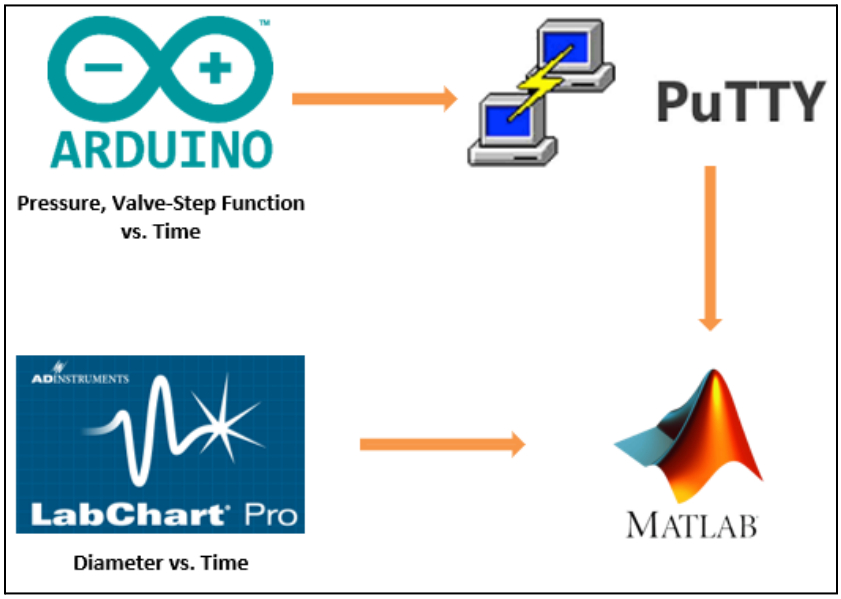


Figure 18. Data acquisition workflow

Operation Protocol

A. Arduino:

1. Open the Arduino file ‘serial_plott_water_valve_and_TruWave_pressure_sensor_HX711_lcd_.ino’. If this is not available, download the file from the ‘Software-Files’ folder on google drive
2. Plug the Arduino USB cable into the computer and then into the Arduino Uno board.
3. Check if the correct board and port are selected. To do this step, first click tools then click “Board” and select the correct board (e.g., Arduino Uno). Secondly, click port and select any port available (typically COM7 or COM9). Make a note of the port being used from the very top of the window (e.g. COM7).

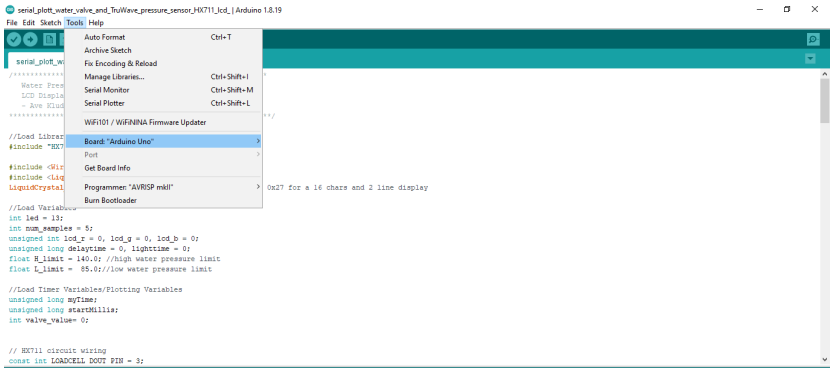


Figure 19. Diagram showing the location of ‘Tools’ Command

4. Load the program file onto the Arduino Uno. To do this step, there are two options. The first option is to click the blue right arrow  in the top left corner of the screen. The second option is to click “Sketch” and then “Upload” (or Ctrl+U) in the top left corner of the screen.
5. Open the serial plot/board. Click Tools and then ‘Serial Monitor’ (or Ctrl+Shift+M). To open the plot, click Tools and then ‘Serial Plot’ (or Ctrl+Shift+L).
    a. Note: In older versions of the IDE, either the board or monitor can be opened at the same time. In newer versions, both the board and monitor can be opened at the same time.
        i. In newer versions, open the serial monitor and then the serial plot.
    b. The serial monitor is the most important to be functional as these data points will be read by PuTTY and later downloaded onto a .csv file.

B. PUTTY:

1. Concentrate first on getting the Arduino program before running and finding the data being written into your serial monitor window through PUTTY.
    a. PUTTY can be downloaded at https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
2. Close the Serial window in the Arduino as we want to open this same port in PuTTY. These two cannot be open at the same time.
    a. Arduino will send data to the port (e.g., COM9) in PUTTY

3. Open the PUTTY application. If the COMPLIANCE session is available click it, press load, and move to step 6, and ignore step 7. If not download the saved session from Final_Compliance_PUTTY_setting.reg, move to step 6, and ignore step 7.
    a. Note: The following steps below are a manual setup from PUTTY if a saved session is not available through downloading files or is not already on a computer
4. Click on 'Session' at the left-hand side of the window. Enter the COM port (e.g. COM12) in the "Serial line" field, then click the "Serial" button. Select the speed to be "115200".
5. On the left-hand side, select 'Logging', then select 'Printable output', and maybe select 'Always'. Overwrite it if that is what you want.
6. Click 'Browse' and look for a place and file name where you can save the file you logged. It's a good idea to put the file name extension .csv (such as 'logging.csv') so that Excel can immediately recognize it.
7. Go back to the Session window and in 'Saved Sessions', enter your name or initials then click 'Save'. By doing this, every time you open PuTTY, just select your Saved session and click 'Load' and you won't have to jump through all those hoops again.
8. To start logging in, just click on 'Open'.
9. Reference:
    a. https://www.circuitbasics.com/logging-arduino-data-to-files-on-a-computer/#:~:text=Enter%20the%20COM%20port%20(e.g.,save%20the%20file%20you%20logged.

C. LabChart:

1. **DAQ (steps 1 -4):** Obtain an AD Instruments PowerLab DAQ with at least two input channels, capable of recording voltages up to +/- 10 V (e.g. PowerLab 8/30). Connect it via USB to a PC with the associated LabChart software installed (e.g. LabChart 7).
2. Open the settings file 20180627_CETS_Compliance. If this file is not available, recreate it as such using the steps below. Most of these settings can be accessed under Setup > Channel Settings. If the file is available, skip to step 5.
3. Turn off and hide the display of input channels 3-8.
4. Under "Channel Settings" for channel 1, apply the following:
    a. Title: Pressure
    b. Sampling Rate: 10/s
    c. Range: 2V
    d. Input Settings:
        i. Range: 2V
        ii. Lowpass: Off
        iii. Single-ended
        iv. AC coupled: not checked
        v. Mains filter: not checked
        vi. Invert: not checked
    e. Units:
        i. Point 1: 0 V, 0 mmHg
        ii. Point 2: 2 V, 200 mmHg

iii. Units: mmHg
iv. Decimal places: 0
v. Unit conversion: On

f. Computed input:
i. Function: raw data
ii. Raw data input: 1
iii. Range: 2 V

g. Calculation:
i. Smoothing
1. Source channel: Ch1
2. Triangular (Bartlett) window
3. Samples: 9

h. - Under "Channel Settings" for channel 2, apply the following:
i. Title: Diameter
ii. Sampling Rate: 10/s
iii. Range:10 V
iv. Input Settings:
1. Range:10 V
2. Lowpass: Off
3. Single-ended
4. AC coupled: not checked
5. Mains filter: not checked
6. Invert: not checked
v. Units:
1. Point 1:0 V,0 mm
2. Point 2:10 V, 10 mm
3. Units; mm
4. Decimal places:3
5. Unit conversion: On
vi. Computed input:
1. Function: raw data
2. Raw data input: 2
3. Range: 10 V
vii. Calculation:
1. Smoothing
a. Source channel: Ch2
b. Triangular (Bartlett) window
c. Samples: 9
d. Note: The smoothing and sampling rate of either channel may be adjusted if preferred. Set up the display as desired.

5. **Laser micrometer (step 2-12):** Obtain an LS-7601 laser micrometer. Two coaxial cables exit the back of the monitor. Furthermore, there are 23 screw terminals just above the cable exits. Use a

multimeter to validate that one coaxial cable's shield is tied to the screw of input 21 (0 V) and that its pin is tied to the screw of input 22 (OUT1).

6. If the connections are bad, uninstall the monitor from its base and check the wiring on the monitor underside. Be especially careful that the inputs are not shorted. When finished, connect this coaxial cable to input channel 2 on the PowerLab.
7. Power on the laser micrometer. Ensure that the LCD display is measuring the vertical diameter of a black object onscreen, that it is in "Run" mode, that "OUT1" is displayed, and that the "Standard" value displayed is 0.00000 .
8. If it is not, consult the owner's manual for how to correct these settings. Whether the display shows "GO", "LO", "HI", or any other message in that field is irrelevant to the measurement accuracy. The micrometer function may be tested using an object of known width.
9. If the micrometer LCD display is showing a black field, check the connections on the cables running to the measuring heads. Usually, this is a poor signal transmission resulting from cable displacement.
10. Verify that the pressure channel is turned OFF and ONLY the diameter channel is on. To access this setting, click Setup>Channel Settings>

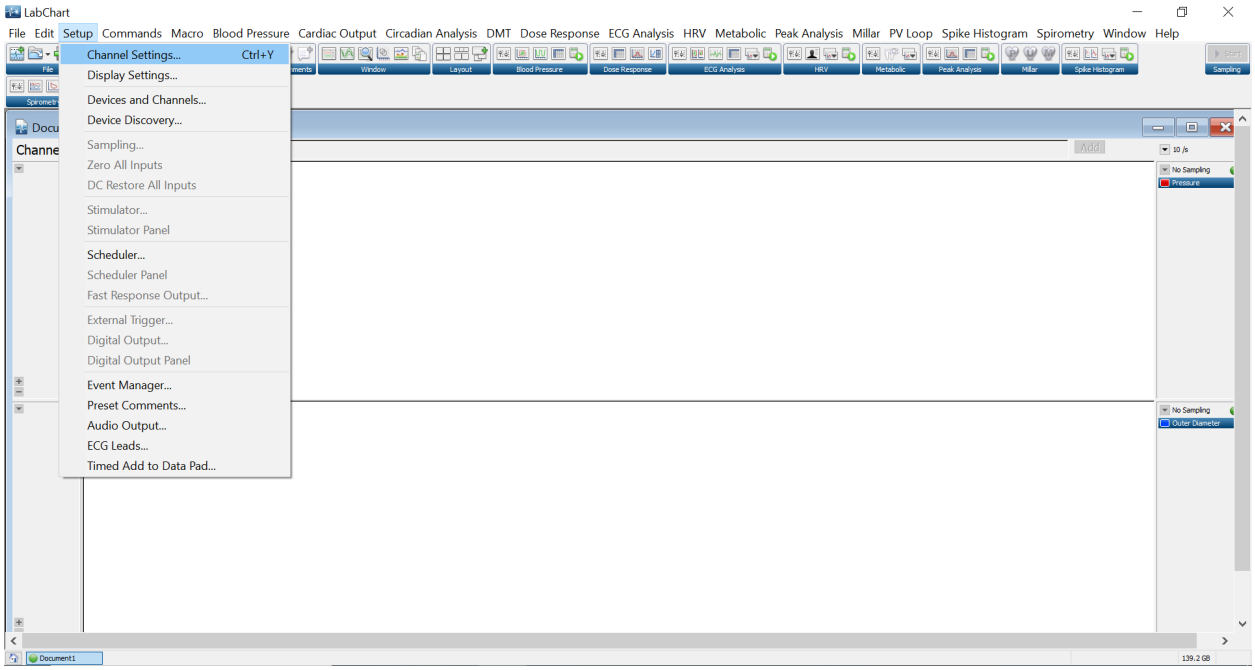


Figure 20. Channel selection in PUTTY

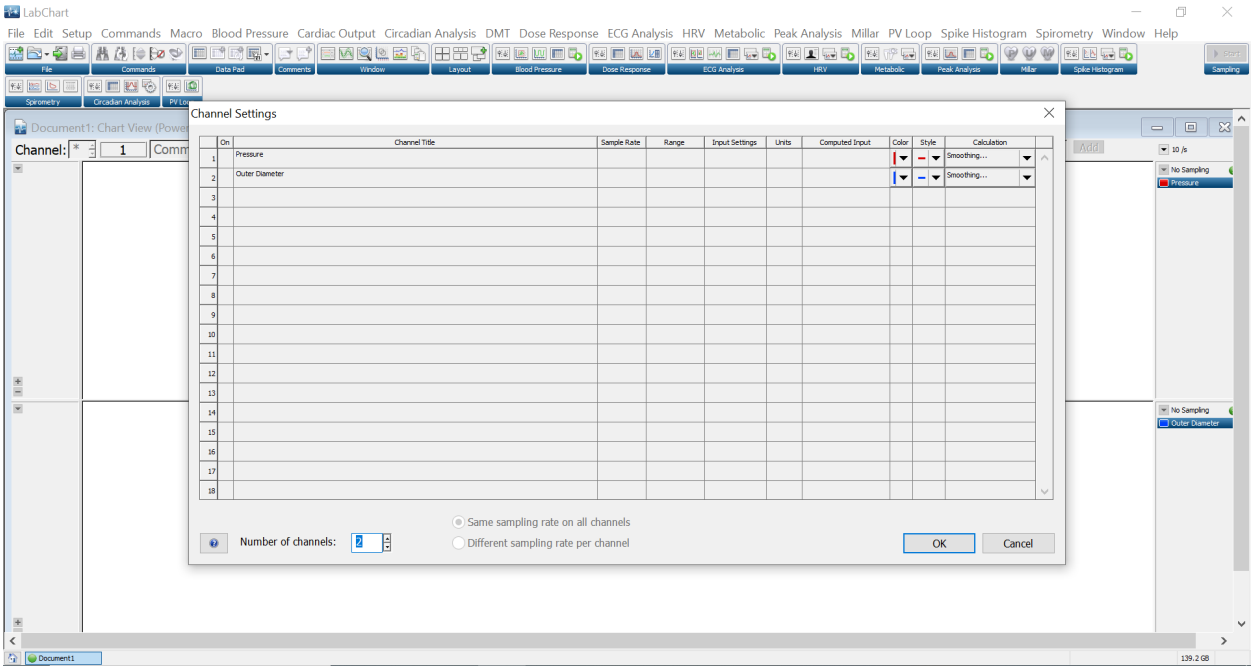


Figure 21. Channel Selection in PUTTY

11. To start recording, click the micrometer record towards the right side of the screen or Commands>Record
12. To save, click File>Save As> . The file will save as a .txt file. To convert to an .xlsx file, open the .txt file. Left-click 'Select All' and copy the file onto a .xlsx file
13. Reference: Compliance_PDF.pdf

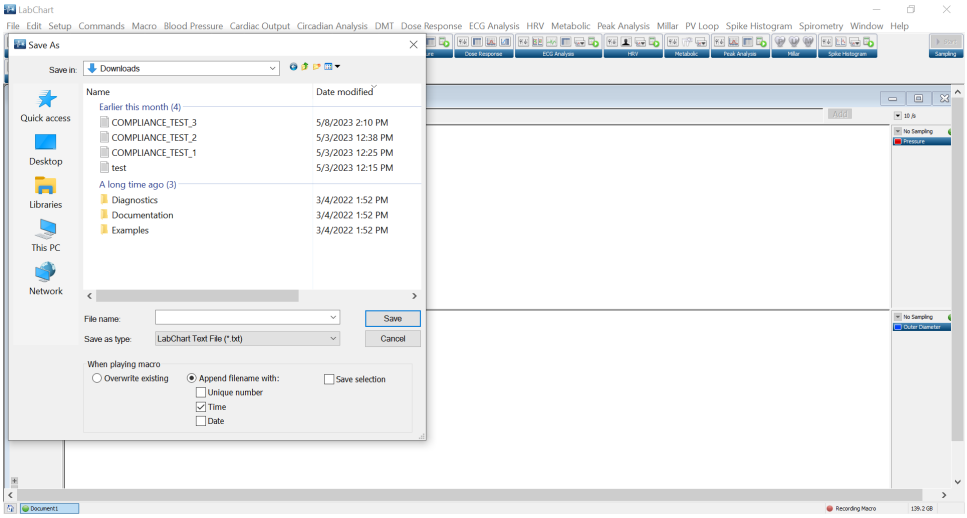


Figure 22. Exporting Data File in PUTTY

D. Getting All Together (Collect Data):

1. Verify that PuTTY, Arduino, and LabChart are fully functioning as indicated in the steps above (A, B, and C).
2. Verify that the Arduino USB Cable is plugged into the computer and the microcontroller board.
3. Plug in the 12 V DC power supply to the barrel jack.
4. Place the submersible water pump inside the water reservoir and connect the plug to a power supply.
5. Start logging data by clicking on 'Open' on the PUTTY program and start recording data by clicking 'Record' on the LabChart program.
6. For PUTTY, the setup provides you the option to append or overwrite your file. Exit the program to stop data collection and view the data file.
7. For LabChart, to save, click File>Save As> . The file will save as a .txt file. To convert to an .xlsx file, open the .txt file. Left-click 'Select All' and copy the file onto a .xlsx file

E. MATLAB (Analyze Data):

1. Rename file list images and run program, save program files
2. Verify that all saved data files are within the same folder.
3. Verify that the Arduino Data is saved as a .csv file. In the program, change the filename_arduino to your saved file name. For example, filename_arduino = 'file_name.csv' where file_name_arduino.csv is the saved file name.
    a. Note: The program will remove unnecessary columns and rows that do not contain the data. It will re-assign the column name within the program variables.

Editor - D:\F2021 Research\2022-2023\Compliance\compliance_data_sync.m

compliance_data_sync.m

```
clear all;
close all;

%% STEP 1: LOAD DATA (Arduino & Micrometer Data)

filename_arduino = 'logging_csv_4.csv' ; %type-in-save-file-name for arduino data
arduino_data = readtable(filename_arduino); %raw arduino data from PUTTY
arduino_data.Properties.VariableNames([1 2 3]) = {'Pressure (mmHG)' 'Valve Step Function' 'Time (seconds)'};
%rename data and remove un-needed columns (e.g., insert Pressure, Water Valve-Step Function, and Time (seconds) in place of
% PUTTY log 2023.05.03 12:35:16, VAR 2, and VAR 3)

filename_micrometer = 'COMPLIANCE_TEST_3.xlsx';
micrometer_data = readtable(filename_micrometer);
micrometer_data.Properties.VariableNames([1 2]) = {'Time (seconds)' 'Diameter (mm)'};

%% STEP 2: CLEAN DATA (Arduino & Micrometer Data)

arduino_data(:,4:width(arduino_data)) = []; %delete the un-needed columns (e.g., 4-14) and keep columns 1-3)
arduino_data = table2array(arduino_data); %convert data table to matrix/array
arduino_data(isnan(arduino_data(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Pressure (mmHG))
arduino_data(isnan(arduino_data(:,2)),:)=[]; %remove those rows w/ NaN in column 2 (Valve Step Function)
arduino_data(isnan(arduino_data(:,3)),:)=[]; %remove those rows w/ NaN in column 3 (Time (seconds))
```

Figure 23. MATLAB changing csv/xlsx file

4. Verify that the laser micrometer data is saved as a .xlsx file. In the program, change the filename_micrometer to your saved file name. For example, filename_micrometer = 'file_name_micrometer.xlsx';
    a. Note: The program will remove unnecessary columns and rows that do not contain the data. It will re-assign the column name within the program variables.

5. Click ‘Run’ 

and wait for the program to run.
6. Three figures should appear that provide synchronized data. Click File>Save As> and change the file type to JPEG and save the file onto your computer.

Troubleshooting

Error in Uploading Arduino Code: Check that you have the correct port selected. If it is correct, try pressing the board's reset button after initiating the upload. This error can happen if the board on the selected port has been disconnected from the computer, or reassigned to a different port. Make sure the board is connected to your computer.

Link:
https://support.arduino.cc/hc/en-us/articles/4403365313810-Errors-when-uploading-a-sketch#:~:text=Check%20that%20you%20have%20the,is%20connected%20to%20your%20computer.

Error in Relay/Ball-Valve: Relay Test/Repair Relay: Load the program file “water_valve_relay_test_1.ino”. Verify that the water valve is opening and closing. Verify that there is a clicking sound of the relay. Verify that the power indicator is green on the relay and that the relay status indicator is red. If there is no clicking sound of the relay and the lights are not powered, then the relay must be replaced. If there is a clicking sound but no opening or closing of the ball valve, then the ball valve must be replaced.

To replace the relay, unscrew the relay from the enclosure. Connect DC+ to the 5V power rail on the breadboard. Connect DC- to the ground power rail on the breadboard. Connect IN to digital pin 5. Connect NC to the (+) DC barrel jack. Connect COM to the positive side of the diode-vall connector. See the diagrams below for a better visual and understanding.

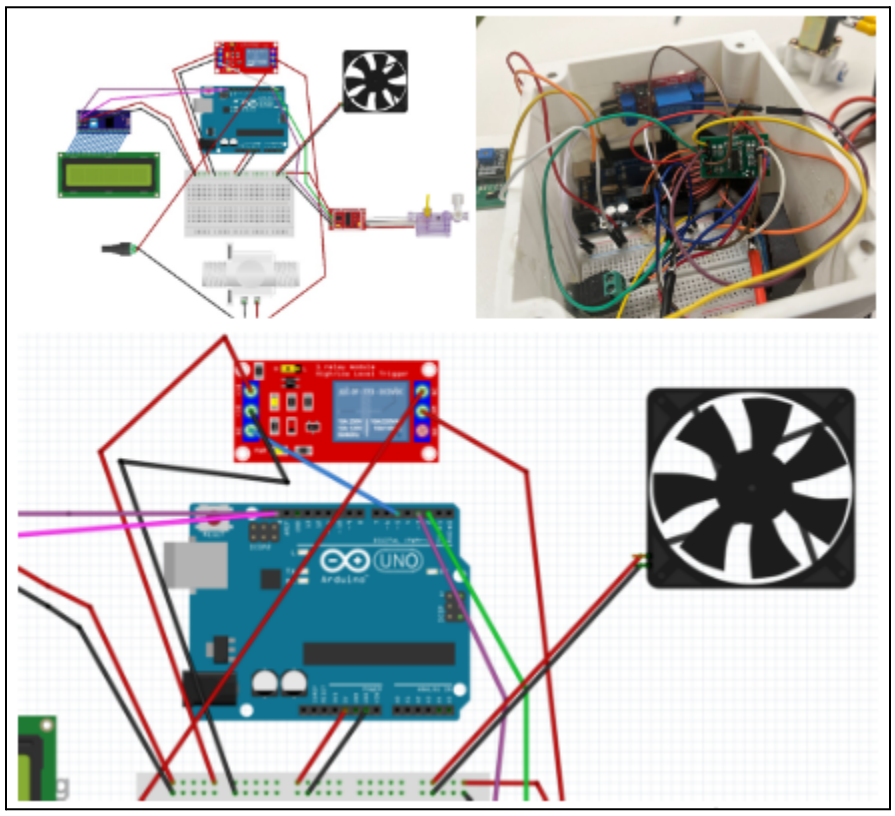


Figure 24. Circuit Diagram with sketch and photo-image versions

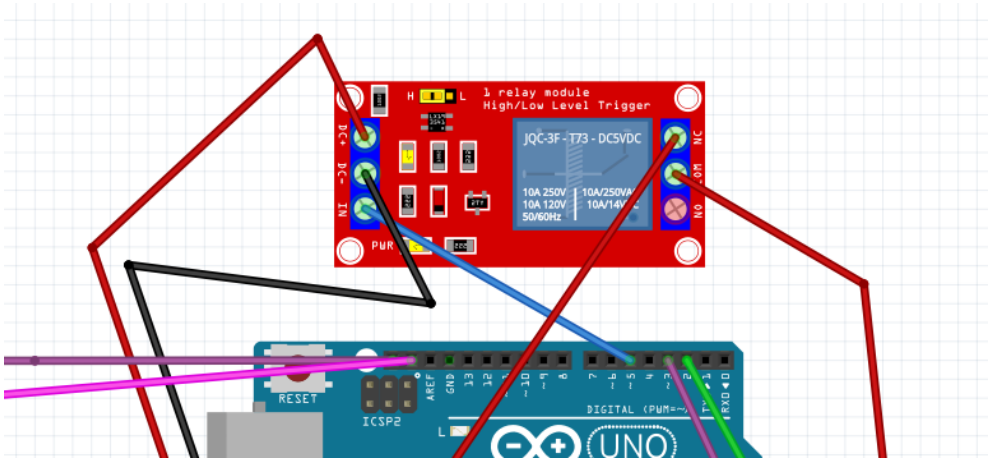
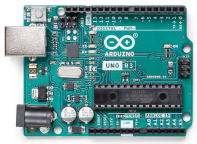
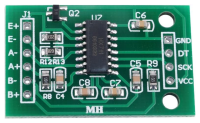
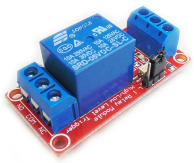
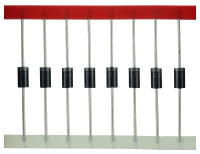


Figure 25. Close-up of Relay Module

Part List (Full)

Full Name	Image	Description	Link
Arduino Uno		Arduino is an open-source hardware, software, and content platform with a worldwide community of over 30 million active users.	https://www.amazon.com/Arduino-A000066-ARDUINO-UNO-R3/dp/B008GRTSV6
Motorized Ball Valve-1/4" Stainless Steel Ball Valve		This valve comes in stainless steel. It is 1/4” in diameter, and is full port. This means the diameter of the opening in the ball is a full 1/4”.	https://www.amazon.com/dp/B06XXPZHVB?ref=ppx_yo2ov_dt_b_product_details&th=1
Edwards Truwave Pressure Transducer		TruWave pressure monitoring transducer kits are sterile, single-use kits that relay blood pressure information from a pressure monitoring catheter to a patient monitoring system.	https://www.edwards.com/gb/devices/Pressure-Monitoring/Transducer
DIYmall Hx711 Weight Weighing Load Cell Conversion Module Sensors		This HX711 amplifier module uses 24 high precision A/D converter chip hx711, 2 selectable differential input channels. This weighing sensor amplifier module can use to be weight scales, kitchen scale, industrial process control etc.	https://www.amazon.com/DIYmall-Weighing-Conversion-Sensors-Microcontroller/dp/B010FG9RXO
HiLetgo 2pcs 5V One Channel Relay Module Relay Switch with OPTO Isolation High Low Level Trigger		Module description: 1.The module Operating voltage 5V; 2.Fault-tolerant design, even if the control line is broken, the relay will not operate; 3.The power indicator (green), the relay status indicator (red); 4.Module size: 50mm*26mm*18.5mm (L*W*H).	https://www.amazon.com/HiLetgo-Channel-optocoupler-Support-Trigger/dp/B00LW15A4W/ref=asc_df_B00LW15A4W/?tag=hyprod-20&linkCode=df0&hvadid=198090929431&hvpos=&hvnetw=g&hvrand=4397058200521600931&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9005779&hvtargid=pla-348614466893&psc=1

GeeekPi 2-Pack I2C 1602 LCD Display Module 16X2 Character Serial Blue Backlight		Use the i2c protocol to reduce the occupation of I/O ports, making it easier to add to the project, and less wiring is more beautiful. Commonly used in: Internet of things, DIY project, home animation, smartbuilding, maker's DIY project.	https://www.amazon.com/GeeekPi-Character-Backlight-Raspberry-Electrical/dp/B07S7PJYM6/ref=sr_1_3?crid=3K4VCPH83MVM0&keywords=LCD+1602&qid=1684974831&sprefix=lcd+1602%2Caps%2C154&sr=8-3
Female 12v DC Power Jack Adapter Connector		Simply & Professional appearance for power cabling. Easier for camera installation, save time and more securer cable connection. No electrical tap, No splicing, no crimping, but only a small screw driver. Power Plug features one power connector end DC female to another end terminal for CCTV camera connected. Support power for CCTV camera accessories, such as video balun.	https://www.amazon.com/Ksmile%C2%AE-Female-2-1x5-5mm-Adapter-Connector/dp/B015OCV5Y8/ref=sr_1_15?crid=26KFVD3H2RWWW&keywords=2.1mm+Barrel+Jack+with+Terminal+Block+female&qid=1684975064&sprefix=2.1mm+barrel+jack+with+terminal+block+female%2Caps%2C140&sr=8-15
Fan DC 5V/3.3V Brushless CPU Cooling Fan		Cool Fan Size (L x W x H): Approximately 30 x 30 x 7 mm. Easy to install the fan for pi by screws, screw length: 15 mm. Fan Connector: 2 pins, Rated Voltage: DC 5V, Working Voltage: DC 3.0-5.8 V.	https://www.amazon.com/Raspberry-Aokin-Brushless-Cooling-RetroFlag/dp/B07KPLBDRN/ref=sr_1_3?crid=2O5U1M2W3UXTA&keywords=5V+DC+fan&qid=1684975325&sprefix=5v+dc+fan%2Caps%2C282&sr=8-3
Diode (3A)		N5822 Data: Forward rectified current:3A,Maximum recurrent peak reverse voltage:40V Feature:Low forward voltage/High current capability/Low leakage current/High surge capability	https://www.amazon.com/BOJACK-Schottky-DO-201AD-Electronic-Silicon/dp/B07Q4HYL1P/ref=sr_1_6?crid=3MVRI7KIMQHD2&keywords=Diode+3A+%28amp%29&qid=1684975675&sprefix=diode+3a+amp+%2Caps%2C489&sr=8-6
VIVOSUN 480GPH Submersible Pump (1800L/H, 25W) 0.5”		The 5-ft. long three-core power cord offers convenient installation wherever you need, and the adjustable flow regulating valve lets you control the water flow rate, so you can get just the amount of water pressure you want;	https://www.amazon.com/VIVOSUN-Submersible-Fountain-Aquarium-Hydroponics/dp/B082M7FNK6?th=1

12 V DC Power Supply		12V 2A Switching Power Supply Adapter 2 Pack Input Voltage: 100-240V AC; Output Voltage: 12V DC 2000 mA	https://www.amazon.com/100-240V-Transformers-Switching-Applications-Connectors/dp/B077PW5JC3/ref=sr_1_4?crid=U74DJQIKJSYH&keywords=12+V+DC+Power+Supply&qid=1684976371&sprefix=12+v+dc+power+supply%2Caps%2C614&sr=8-4
MGI SpeedWare 1/4" NPT Strain Relief Nylon Cord Grip Cable Glands		WIRE PROTECTION — Provides strain relief to cables and wires passing through plates or connecting to electrical equipment. Outer Drill Diameter: 0.54" EASY INSTALLATION — Simply pass your cable through the gland and tighten the cap. Fits 0.12" to 0.25" (3 - 6mm).	https://www.amazon.com/MGI-SpeedWare-Strain-Relief-Cable/dp/B08MYM2TSJ/ref=sr_1_5?crid=1E5WLFBF36EW9&keywords=strain%2Brelief%2Bgrips&qid=1677518043&sprefix=strain%2Brelief%2Bgrips%2Caps%2C184&sr=8-5&th=1
DIYmall Arduino USB Cable		USB Cable to connect the microcontroller to a PC or Mac It used for arduino u no 2560 r3, printer Cable length:52cm What you will get:2pcs USB Cable	https://www.amazon.com/DIYmall-Cable-Arduino-2560-Pack/dp/B09JRXT1TY/ref=sr_1_3?crid=3COCDDAXW47QV&keywords=Arduino+USB+Cable&qid=1684976472&sprefix=arduino+usb+cable%2Caps%2C1485&sr=8-3
XT60 Plug Female and Male Connector		Include 3 PCS XT60 female cable and 3PCS XT60 male cable Good soldering quality. Copper contact. Low resistance Used for RC car, quadcopter and boat battery connection	https://www.amazon.com/Pairs-Female-Connector-Silicon-Battery/dp/B07QH249CR
Beduan Stainless Steel 1/4" Hose Barb to 1/4" Male NPT Home Brew Fitting Water Fuel Air		This Fitting is 1/4" barb to 1/4" NPT male, suitable for hose tubing with 1/4" ID. Material: Made of 304 stainless steel, strength and resistance to corrosion. Maximum working pressure of 2800 psi, Temperatures range from -868 to 1472 ℉/-500 to 800 °C	https://www.amazon.com/Beduan-Stainless-Steel-Fitting-Water/dp/B07NB6VZF7/ref=sr_1_1?keywords=Beduan%2BStainless%2BSteel%2B1%2F2%22%2BHose%2BBarb%2Bto%2B1%2F4%22%2BMale%2BNPT%2BHome%2BBrew%2BFitting%2BWater%2BFuel%2BAir%2

			B(Pack%2Bof%2B2&qid=1681158871&s=hi&sr=1-1-catcorr&th=1
D-NYX 10 Rolls Extra Long Teflon Tape 1/2 Inch(W) X 520 Inches(L)		Our sealant tape is great for a wide range of pipework and plumbing jobs. Use them for sealing off valves, dripping taps and leaky faucets.	https://www.amazon.com/dp/B09NY6V13M?psc=1&ref=ppx_yo2ov_dt_b_product_details

Note:
Screw Sizes:
○ Cap Screw: 4-40 x 3⁄8” SH4S038
■ Head Diameter: 0.223" (5.6642 mm)
○ Nut: N4S025
○ Hex: HK050 0.050”

Wires:
○ Sensor Wires: 10 mm (slit)
○ Valve Wires: 2 mm (slit)

Encasing Dimensions: 135 mm x 117 mm x ~ 75 mm (L x W x H)
■ ~height is extrusion

Commented Arduino Code

```
/************************************************************

  Water Pressure Sensor, Water/Solenoid Valve, and

  LCD Display System

  - Ave Kludze, Biofoundry.

************************************************************/


//Load Librariesd

#include "HX711.h"


#include <Wire.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display


//Load Variables

int led = 13;

int num_samples = 5;

unsigned int lcd_r = 0, lcd_g = 0, lcd_b = 0;

unsigned long delaytime = 0, lighttime = 0;

float H_limit = 140.0; //high water pressure limit

float L_limit =  85.0;//low water pressure limit


//Load Timer Variables/Plotting Variables
```

```
unsigned long myTime;

unsigned long startMillis;

int valve_value= 0;



// HX711 circuit wiring

const int LOADCELL_DOUT_PIN = 3;

const int LOADCELL_SCK_PIN = 2;

float ADC_val_avg;

float Pressure;

volatile float f;


//Relay Board

const int RELAY_ENABLE = 5;


// System Delays

int valve_delay = 1000;

int LCD_delay = 500;


HX711 scale;


void setup() {

  // put your setup code here, to run once:
```

```
startMillis = millis();


//SetUp for Pressure Sensor
Serial.begin(115200);
scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN); // initialize the pressure sensor


//SetUp for LCD
lcd.init();                     // initialize the lcd
lcd.init();
// Print a message to the LCD.
lcd.backlight();
lcd.setCursor(3,0);


Serial.begin(115200);
//Serial.println("hello start");
lighttime = millis();
lcd.setCursor(0, 0);


//SetUp for Water Valve
lcd.print("Water Pressure:");
/// pinMode(led, OUTPUT);
pinMode(RELAY_ENABLE,OUTPUT);
digitalWrite(RELAY_ENABLE,HIGH); //Close Valve
```

```
 Serial.begin(115200);



}

void loop() {

 if (valve_value ==0){
  digitalWrite(RELAY_ENABLE,HIGH);
 }

 // put your main code here, to run repeatedly:

 // Order: 1) Calculate Water Pressure 2) Display Water Pressure on LCD
 //        3) Open or Close Water Valve based on Water Pressure 4) Repeat Process

 // 1) Water Pressure Sensor Main Code
 //digitalWrite(RELAY_ENABLE,LOW);
 //digitalWrite(RELAY_ENABLE,HIGH); //Close VALVE

 myTime = millis();

 if (scale.is_ready()) {
```

```
        long reading = scale.read();


        //TEST:
        //f = scale.get_units(5);
//Serial.println(f);



        //INSTANT:
//Serial.print("HX711 reading: ");
//Serial.println(reading);
        //Pressure  =  reading *0.00009-10; //- 10; // // units: mmHg
//Serial.print("Pressure Reading (mmHg): ");
//Serial.println(Pressure);


        //AVERAGE:
//Serial.print("HX711 reading (average): ");
        ADC_val_avg = scale.read_average(num_samples);
//Serial.println(ADC_val_avg);    // print the average of 20 readings from the ADC
        Pressure  =  ADC_val_avg *0.00009-10; //- 10; // // units: mmHg
//Serial.print("Average Pressure Reading (mmHg): ");
//Serial.println(Pressure);


        // NOTE:
```

```
        // equation from calibration curve: y = x*1*10^{-4}-13.825 where x is the ADC value
        // 9E-05x

} else {
        //lcd.setCursor(3, 1);
        lcd.print(" "); //HX711_not_found.
        //lcd.clear();
        //delay(1000);
}

//delay(valve_delay);

// 2) Load LCD Display with Water Pressure Valve
lcd.setCursor(3, 1);
//delay(LCD_delay);
lcd.print(Pressure);
//delay(LCD_delay);

// 3) Open or Close Water/Solenoid Valve
//delay(valve_delay); //Wait 1 Second

if (Pressure > H_limit){ // Pressure Too HIGH. Open Valve.
  //lcd.print("Pressure High. Open Valve");
```

```
  //Serial.println("Pressure Too HIGH. Open Valve");  //CHANGE TO LCD
  //Serial.println("Relay On");
  digitalWrite(RELAY_ENABLE, LOW);
        valve_value = 300;
  //Serial.println(valve_value);
        //if (digitalRead(RELAY_ENABLE) == 1){
        // valve_value = 300;
        //}
  //delay(valve_delay); //Wait 1 Second
  //lcd.print(Pressure)

}
else if (Pressure < L_limit){ // Pressure Too LOW. Close Valve.
  //lcd.print("Pressure Low. Close Valve");
  //Serial.println("Pressure Too LOW. Close Valve");
  //Serial.println("Relay OFF");
        digitalWrite(RELAY_ENABLE,HIGH);
  //delay(valve_delay); //Wait 1 Second
        valve_value = 0;
  //Serial.println(valve_value);
        //if (digitalRead(RELAY_ENABLE) == 0){
        //valve_value = 0;
        //}
```

```
 }
 Serial.print(Pressure); Serial.print(", ");
 Serial.print(valve_value); Serial.  print(", ");
 Serial.print(myTime/1000.0, 3); Serial.print(", ");
 Serial.println();
 //delay(1000);  // wait a second so as not to send massive amounts of data


//  Serial.print("Sensor_Value:"); Serial.print(Pressure); Serial.print(", ");
//  Serial.print("Valve_Open/Close:"); Serial.print(valve_value); Serial.  print(", ");
//  Serial.print("Time"); Serial.print(myTime/1000); Serial.print(", ");
//  Serial.println();


  //delay(valve_delay); //Wait 1 Second


//float H_limit = 100.0; //high water pressure limit
//float L_limit =  50 //low water pressure limit
//LCD Prints: Pressure & Valve/Open/Cloze
}
```

Commented MATLAB Code

```
clear all;

close all;
```

STEP 1: LOAD DATA (Arduino & Micrometer Data)

```
filename_arduino = 'logging_csv_4.csv' ; %type-in-save-file-name for arduino data

arduino_data = readtable(filename_arduino); %raw arduino data from PUTTY

arduino_data.Properties.VariableNames([1 2 3]) = {'Pressure (mmHG)' 'Valve Step Function' 'Time
%rename data and remove un-needed columns (e.g., insert Pressure, Water Valve-Step Function, and %
PuTTY log 2023.05.03 12:35:16, VAR 2, and VAR 3)

filename_micrometer = 'COMPLIANCE_TEST_3.xlsx';

micrometer_data = readtable(filename_micrometer);

micrometer_data.Properties.VariableNames([1 2]) = {'Time (seconds)' 'Diameter (mm)'};
```

STEP 2: CLEAN DATA (Arduino & Micrometer Data)

```
arduino_data(:,4:width(arduino_data)) = []; %delete the un-needed columns (e.g., 4-14) and keep
arduino_data = table2array(arduino_data); %convert data table to matrix/array

arduino_data(isnan(arduino_data(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Pressure
(mmHG))

arduino_data(isnan(arduino_data(:,2)),:)=[]; %remove those rows w/ NaN in column 2 (Valve Step
arduino_data(isnan(arduino_data(:,3)),:)=[]; %remove those rows w/ NaN in column 3 (Time (seconds))

micrometer_data(:,3:width(micrometer_data)) = []; %delete the un-needed columns (e.g., 4-14) and
micrometer_data = table2array(micrometer_data); %convert data table to matrix/array

micrometer_data(isnan(micrometer_data(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Time
micrometer_data(isnan(micrometer_data(:,2)),:)=[]; %remove those rows w/ NaN in column 2 (Diameter
for n = 1:height(micrometer_data)

micrometer_data(n,2) = micrometer_data(n,2); % Copy original value

micrometer_data(n,2) = micrometer_data(n,2) + 10; %Add 10 because laser micrometer measures end

delay_index_1 = 105; %approximately 5 second delay from arduino data-collection
```

```
delay_index_2 = 50; %approximately 5 second delay from micrometer data-collection

arduino_data = arduino_data(delay_index_1:end,:);

micrometer_data = micrometer_data(delay_index_2:end,:);
```

STEP 3: FIND INITIAL PEAKS (Pressure, Diameter vs. Time)

```
%Not Necessary but could help: Valve Step Function vs. Time

arduino_time = arduino_data(:,3); %load arduino_data into separate components

arduino_valve = arduino_data(:,2);

arduino_pressure = arduino_data(:,1);

micrometer_time = micrometer_data(:,1);

micrometer_diameter = micrometer_data(:,2);

[pressure_max,time_max_1] = findpeaks(arduino_pressure,arduino_time,'MinPeakDistance',1.20); %find
[diameter_max,time_max_2] = findpeaks(micrometer_diameter,micrometer_time); %find max diameter 1

[pressure_min,time_min_1] = findpeaks(-arduino_pressure,arduino_time,'MinPeakDistance',1.20);
%[diameter_min,time_min_2] = findpeaks(-micrometer_diameter,micrometer_time); %find min diameter
diameter_min = abs(diameter_min); %get-real result

pressure_min = abs(pressure_min); %get-real result

pressure_max_n_min = cat(1,pressure_max,pressure_min); %combine data

time_max_n_min_1 = cat(1,time_max_1,time_min_1);

diameter_max_n_min = cat(1,diameter_max,diameter_min); %combine data

time_max_n_min_2 = cat(1,time_max_2,time_min_2);

pressure_time_max_min = cat(2,pressure_max_n_min,time_max_n_min_1); %combine pressure and time
diameter_time_max_min = cat(2,diameter_max_n_min,time_max_n_min_2); %combine pressure and
time pressure_time_max_min = sortrows(pressure_time_max_min,2); %sort data based on time

diameter_time_max_min = sortrows(diameter_time_max_min,2); %sort data based on time

for n = 1:length(pressure_time_max_min) %if duplicate datapoints within the same interval

if n == length(pressure_time_max_min)
```

```
break

end

if
abs((pressure_time_max_min(n,1)-pressure_time_max_min(n+1,1)))/(pressure_time_max_min(n,temp_a
vg_pressure = mean(pressure_time_max_min(n:n+1,1));

temp_avg_time = mean(pressure_time_max_min(n:n+1,2));

pressure_time_max_min(n,:) = cat(2,temp_avg_pressure,temp_avg_time);

pressure_time_max_min(n+1,:) = [];

end

end
```

STEP 4: SYNC DATA, ADJUST FOR TIME DELAY (Pressure, Diameter vs.

Time) BASED ON PEAKS

```
% ADJUST FOR TIME DELAY IN Pressure vs. Time

for m = 1:length(arduino_data)

% for each data point, if the time-point is less than the first peak-time

% point, then delete the data point

if arduino_time(m) < pressure_time_max_min(1,2)

arduino_time(m) = NaN;

arduino_pressure(m) = NaN;

%if the time-point is greater than the first peak-time correct by

%subtracting the first-peak time (time correct)

elseif arduino_time(m) == pressure_time_max_min(1,2)

arduino_time(m) = arduino_time(m) - pressure_time_max_min(1,2);
```

```
%if the time-point is greater than the first peak-time correct by

%subtracting the first-peak time (time correct)

elseif arduino_time(m) > pressure_time_max_min(1,2)

arduino_time(m) = arduino_time(m) - pressure_time_max_min(1,2);

end

end

arduino_time(isnan(arduino_time(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Time (seconds))

arduino_pressure(isnan(arduino_pressure(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Time for
j = 1:length(pressure_time_max_min)

pressure_time_max_min(j,2) = pressure_time_max_min(j,2) - pressure_time_max_min(1,2);

end

% ADJUST FOR TIME DELAY IN Diameter vs. Time

for m = 1:length(micrometer_data)

% for each data point, if the time-point is less than the first peak-time

% point, then delete the data point

if micrometer_time(m) < diameter_time_max_min(1,2)

micrometer_time(m) = NaN;

micrometer_diameter(m) = NaN;

%if the time-point is greater than the first peak-time correct by

%subtracting the first-peak time (time correct)

elseif micrometer_time(m) == diameter_time_max_min(1,2)

micrometer_time(m) = micrometer_time(m) - diameter_time_max_min(1,2);

%if the time-point is greater than the first peak-time correct by

%subtracting the first-peak time (time correct)
```

```
elseif micrometer_time(m) > diameter_time_max_min(1,2)

micrometer_time(m) = micrometer_time(m) - diameter_time_max_min(1,2);

end

end

micrometer_time(isnan(micrometer_time(:,1)),:)=[]; %remove those rows w/ NaN in column 1 (Time
micrometer_diameter(isnan(micrometer_diameter(:,1)),:)=[]; %remove those rows w/ NaN in column for
j = 1:length(diameter_time_max_min)

diameter_time_max_min(j,2) = diameter_time_max_min(j,2) - diameter_time_max_min(1,2);

end
```

STEP 5: FIND PEAKS FOR SYNCED DATA (Pressure, Diameter vs. Time)

```
%%re-find peaks based on adjusted-time

[pressure_max,time_max_1] = findpeaks(arduino_pressure,arduino_time,'MinPeakDistance',1.20); %find
[diameter_max,time_max_2] = findpeaks(micrometer_diameter,micrometer_time); %find max diameter
[pressure_min,time_min_1] = findpeaks(-arduino_pressure,arduino_time,'MinPeakDistance',1.20);
%[diameter_min,time_min_2] = findpeaks(-micrometer_diameter,micrometer_time); %find min diameter
diameter_min = abs(diameter_min); %get-real result

pressure_min = abs(pressure_min); %get-real result

pressure_max_n_min = cat(1,pressure_max,pressure_min); %combine data

time_max_n_min_1 = cat(1,time_max_1,time_min_1);

diameter_max_n_min = cat(1,diameter_max,diameter_min); %combine data

time_max_n_min_2 = cat(1,time_max_2,time_min_2);

pressure_time_max_min = cat(2,pressure_max_n_min,time_max_n_min_1); %combine pressure and time
diameter_time_max_min = cat(2,diameter_max_n_min,time_max_n_min_2); %combine pressure and
time pressure_time_max_min = sortrows(pressure_time_max_min,2); %sort data based on time

diameter_time_max_min = sortrows(diameter_time_max_min,2); %sort data based on time

%%re-find peaks based on adjusted-time
```

```
for n = 1:length(pressure_time_max_min) %if duplicate datapoints within the same interval (e.g.,

if n == length(pressure_time_max_min)

break

end

if
abs((pressure_time_max_min(n,1)-pressure_time_max_min(n+1,1)))/(pressure_time_max_min(n,temp_a
vg_pressure = mean(pressure_time_max_min(n:n+1,1));

temp_avg_time = mean(pressure_time_max_min(n:n+1,2));

pressure_time_max_min(n,:) = cat(2,temp_avg_pressure,temp_avg_time);

pressure_time_max_min(n+1,:) = [];

end

end

%Organize Peak-Data to START with D_high and P_high

if diameter_time_max_min(1,1) < diameter_time_max_min(2,1)

diameter_time_max_min(1,1) = NaN;

diameter_time_max_min(1,2)= NaN;

end

diameter_time_max_min(isnan(diameter_time_max_min(:,1)),:)=[];

if pressure_time_max_min(1,1) < pressure_time_max_min(2,1)

pressure_time_max_min(1,1) = NaN;

pressure_time_max_min(1,2)= NaN;

end

pressure_time_max_min(isnan(pressure_time_max_min(:,1)),:)=[];

%Organize Peak-Data to END with D_low and P_low
```

```
if diameter_time_max_min(end,1) > diameter_time_max_min(end-1,1)

diameter_time_max_min(end,1) = NaN;

diameter_time_max_min(end,2)= NaN;

end

diameter_time_max_min(isnan(diameter_time_max_min(:,1)),:)=[];

if pressure_time_max_min(end,1) > pressure_time_max_min(end-1,1)

pressure_time_max_min(end,1) = NaN;

pressure_time_max_min(end,2)= NaN;

end

pressure_time_max_min(isnan(pressure_time_max_min(:,1)),:)=[];

if pressure_time_max_min(1,2) < diameter_time_max_min(1,2) %pressure start peak time is smaller
time_shift = diameter_time_max_min(1,2) - pressure_time_max_min(1,2); %calculate the time shift/for n
= 1:length(diameter_time_max_min)

diameter_time_max_min(n,2) = diameter_time_max_min(n,2) - time_shift;

end

for m = 1:length(micrometer_time)

micrometer_time(m) = micrometer_time(m) - time_shift;

end

%delete time and pressure data that have negative time

temp_diameter_time_merge = cat(2,micrometer_time,micrometer_diameter);

temp_diameter_time_merge(temp_diameter_time_merge(:,1)<0,:)=[];

micrometer_time = temp_diameter_time_merge(:,1);

micrometer_diameter = temp_diameter_time_merge(:,2);

end
```

```
if pressure_time_max_min(1,2) > diameter_time_max_min(1,2) %pressure start peak time is greater
time_shift = pressure_time_max_min(1,2) - diameter_time_max_min(1,2); %calculate the time shift/for n
= 1:length(pressure_time_max_min)

pressure_time_max_min(n,2) = pressure_time_max_min(n,2) - time_shift;

end

for m = 1:length(arduino_time)

arduino_time(m) = arduino_time(m) - time_shift;

end

%delete time and pressure data that have negative time

temp_pressure_time_merge = cat(2,arduino_time,arduino_pressure);

temp_pressure_time_merge(temp_pressure_time_merge(:,1)<0,:)= [];

arduino_time = temp_pressure_time_merge(:,1);

arduino_pressure = temp_pressure_time_merge(:,2);

end
```

STEP 6: DATA ANALYSIS: COMPLIANCE CALCULATION (Wu. W. et al (2012)

Nature Medicine)

```
%Ensure vectors are same size

if length(diameter_time_max_min) ~= length(pressure_time_max_min)

if length(diameter_time_max_min) > length(pressure_time_max_min)

size_diff = length(diameter_time_max_min) - length(pressure_time_max_min);

diameter_time_max_min = diameter_time_max_min(1:end-size_diff,:);

end

else if length(diameter_time_max_min) < length(pressure_time_max_min)
```

```
size_diff = length(pressure_time_max_min) - length(diameter_time_max_min);

pressure_time_max_min = pressure_time_max_min(1:end-size_diff,:);

end

end

Compliance_Value = [];

Compliance_Time_Intervals = [];

for n = 1:2:length(diameter_time_max_min)

if n == length(diameter_time_max_min) && rem(length(diameter_time_max_min),2) == 1

break %if the last index is ODD, quit the loop

end

D_high = diameter_time_max_min(n,1);

D_low = diameter_time_max_min(n+1,1);

P_high = pressure_time_max_min(n,1);

P_low = pressure_time_max_min(n+1,1);

t_i = diameter_time_max_min(n,2);

t_f = diameter_time_max_min(n+1,2);

C = (10000* (D_high-D_low))/((D_low)*(P_high-P_low));

Compliance_Value(end+1) = C;

Compliance_Time_Intervals(end+1) = t_f;

end
```

STEP 7: PLOT SYNC-ED DATA WITH PEAKS

```
figure(1);

plot(arduino_time,arduino_pressure, pressure_time_max_min(:,2), pressure_time_max_min(:,1), 'pg')
```

```
xlabel('Time (seconds)')

ylabel('Pressure (mmHG)')

title('Pressure (mmHG) vs. Time (seconds)')

figure(2);

plot(micrometer_time,micrometer_diameter,diameter_time_max_min(:,2),
diameter_time_max_min(:,1),'xlabel('Time (seconds)')

ylabel('Diameter (mm)')

title('Diameter (mm) vs. Time (seconds)')

figure(3);

subplot(3,1,1,'align');

plot(arduino_time,arduino_pressure, pressure_time_max_min(:,2), pressure_time_max_min(:,1), 'pg')

ylim([min(pressure_time_max_min(:,1))-5 max(pressure_time_max_min(:,1))+5])

xlabel('Time (seconds)')

ylabel('Pressure (mmHG)')

%title('Pressure (mmHG) vs. Time (seconds)')

subplot(3,1,2, 'align');

plot(micrometer_time,micrometer_diameter,diameter_time_max_min(:,2),
diameter_time_max_min(:,1),'ylim([min(diameter_time_max_min(:,1))-0.01
max(diameter_time_max_min(:,1))+0.01])

xlabel('Time (seconds)')

ylabel('Diameter (mm)')

%title('Diameter (mm) vs. Time (seconds)')

subplot(3,1,3, 'align');

%plot(Compliance_Time_Intervals,Compliance_Value)

plot(Compliance_Time_Intervals,Compliance_Value,'o')
```

```
ylim([min(Compliance_Value)-0.1 max(Compliance_Value)+0.1])

xlabel('Time (seconds)')

ylabel('Compliance (%/100 mmHg)')

%title('Compliance (%/100 mmHg) vs. Time (seconds)')
```

Further Improvements/Future

- **Non-porous grafts:** device currently supports only non-porous grafts. It could be improved by supporting porous grafts possibly by looking into pumps that can pump blood clotting-material
- **Electrical components:** the electrical components work and operate as intended but could be improved by looking into cheaper options. For example, the ball-valve could be replaced with a cheaper functioning ball-ball
- **3D-printing:** the 3D-printed box could be improved by fixing dimensions on the .stl files so that it no longer requires drilling of holes (e.g, drilling of holes for cable grips or Arduino uno power-supply)
- **Quartz Crystal Oscillator pins:** the current sampling rate is fast with 80 Hz, but quartz crystal oscillator pins operate to increase the sampling rate. Thus, one may use such an electronic item to increase the sampling rate by connecting it to the HX711
    - https://www.youtube.com/watch?v=Rp_M0NbDSpo
- **Relay:** the current relay is an electronic-hobbyist 5V relay and could be improved by using an industrial relay that has greater longevity. The connections have not been verified throughout this project nor has its utility.
    - https://www.amazon.com/dp/B07NWD8W26?psc=1&ref=ppx_yo2ov_dt_b_product_details
- **Laser micrometer:** the current laser micrometer is approximately 10,000 USD. If one could find an alternative to measure diameter accurately, it would significantly reduce the cost of the project.
- **Flow Sensor:** the current device does not quantify flow and does not produce flow versus time plots. Thus, one may use an electronic part to quantify the water flow versus time.
    - https://www.amazon.com/DIGITEN-Sensor-Flowmeter-Counter-Connect/dp/B07QS17S6Q/ref=sr_1_3?keywords=arduino+flow+meter&qid=1686978390&sr=8-3

References

1. https://www.nature.com/articles/nm.2821
2. https://www.biorxiv.org/content/10.1101/2020.09.13.295097v1.full
3. https://www.sciencedirect.com/science/article/pii/S2468067222000633
4. https://ir.canterbury.ac.nz/handle/10092/104342
5. https://onlinelibrary.wiley.com/doi/10.1002/elsc.201600138
6. https://link.springer.com/article/10.1007/s10439-006-9099-3
7. https://link.springer.com/content/pdf/10.1007/s10439-006-9099-3.pdf
8. https://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=4784&context=all_theses [arduino model ~ custom built model]
9. https://bc-robotics.com/tutorials/controlling-a-solenoid-valve-with-arduino/
10. https://wiki.dfrobot.com/Gravity__Water_Pressure_Sensor_SKU__SEN0257
11. https://www.edwards.com/gb/devices/Pressure-Monitoring/Transducer
12. https://livingsys.com/product/replacement-pressure-transducer-2/
13. https://www.youtube.com/watch?v=RfrDtAEQ95c&t=315s
14. https://www.youtube.com/watch?v=Rp_M0NbDSpo&feature=youtu.be
15. https://www.youtube.com/watch?v=0cxS-a837bY&feature=youtu.be
16. https://www.instructables.com/Make-Beautiful-Plots-From-Live-Arduino-Data-and-Sa/
17. https://forum.arduino.cc/t/load-cell-no-analog-output-with-hx711/395150
18. https://forum.arduino.cc/t/hx711-with-higher-load-cell-voltage/323615
19. https://forum.arduino.cc/t/read-mv-from-hx711/556345/3
20. https://forum.arduino.cc/t/connecting-edwards-truwave-sensor-to-arduino/1060873/8
21. https://www.adafruit.com/product/5066?gclid=CjwKCAjwp9qZBhBkEiwAsYFsb5F
22. https://makersportal.com/blog/2020/6/4/mps20n0040d-pressure-sensor-calibratio
23. https://www.circuitbasics.com/logging-arduino-data-to-files-on-a-computer/#:~:text=Enter%20the%20COM%20port%20(e.g.,save%20the%20file%20you%20logged.
24. https://mcw.marquette.edu/biomedical-engineering/cardiovascular-regenerative-engineering-lab/testing-vascular-grafts.php
25. https://patents.google.com/patent/US7604602B2/en

Appendix A: Data Analysis (LabChart)

Data may be exported from LabChart 7 for further processing. Most analyses require the graft inner diameter. Assuming incompressibility, this can be found from the unloaded dimensions, the axial stretch, and the measured outer diameter:

$$D_i = \sqrt{D_o^2 - \frac{1}{\lambda_x}\left(D_{o,\ominus}^2 - D_{i,\varnothing}^2\right)}$$

According to the supplementary methods of Wu, W. et al. (2012). Nature Medicine 18:1148-53, compliance % 100 mmHg is defined as:

$$C = \frac{D_{\text{high}} - D_{\text{low}}}{D_{\text{low}} \ (P_{\text{high}} - P_{\text{low}})} * 10,000$$

D, the inner diameter, may be in any units, but P should be in mmHg. The 10,000 converts to the % 100 mmHg units.

Appendix B: Final Design (Images)

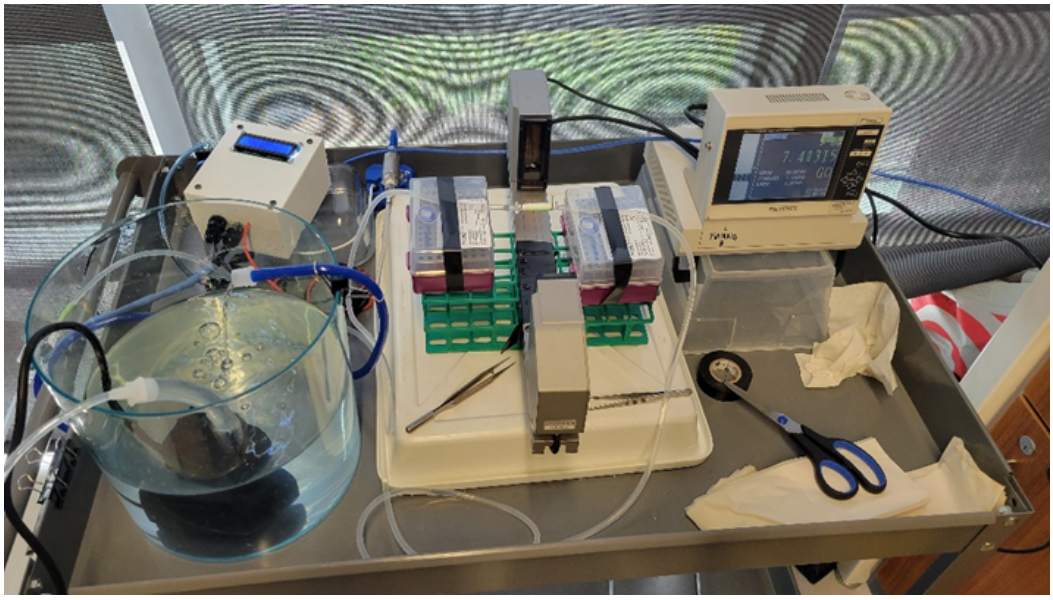


Figure 26. Final Lab Setup

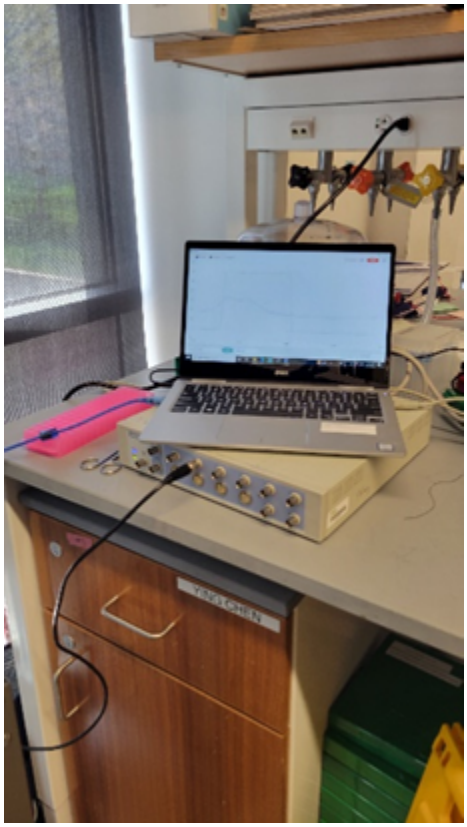


Figure 27. Final Lab Set with Computer and Acquisition Board

Appendix C: Serial Monitor (Arduino)

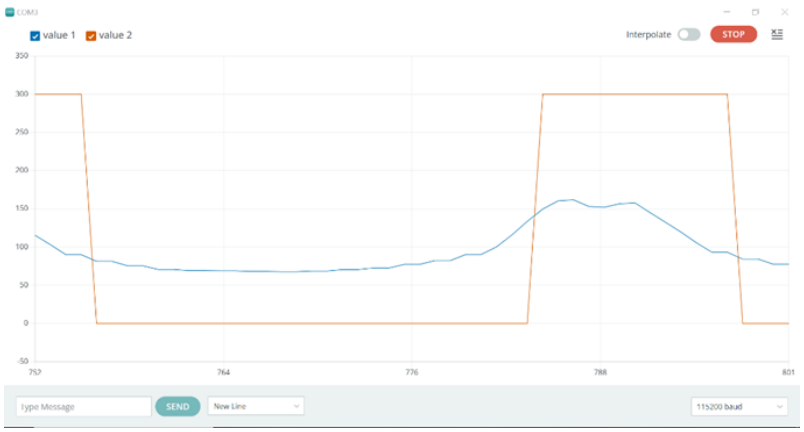
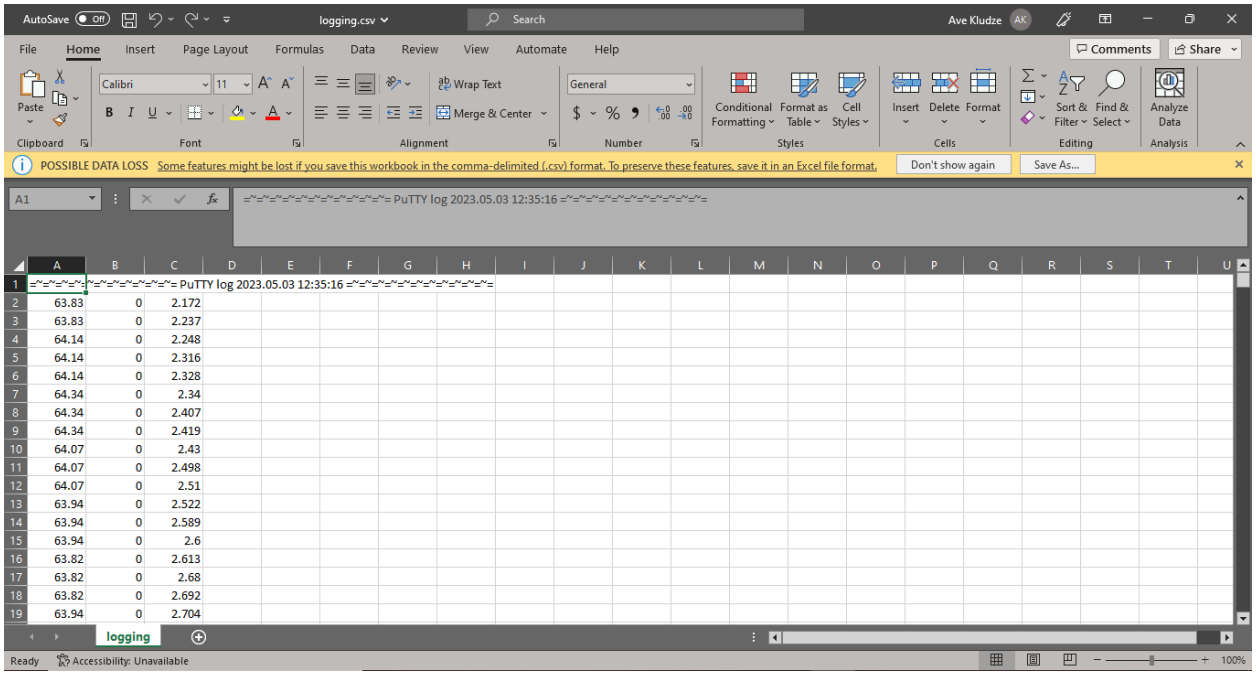


Figure 28. Serial Monitor with Pressure versus Time

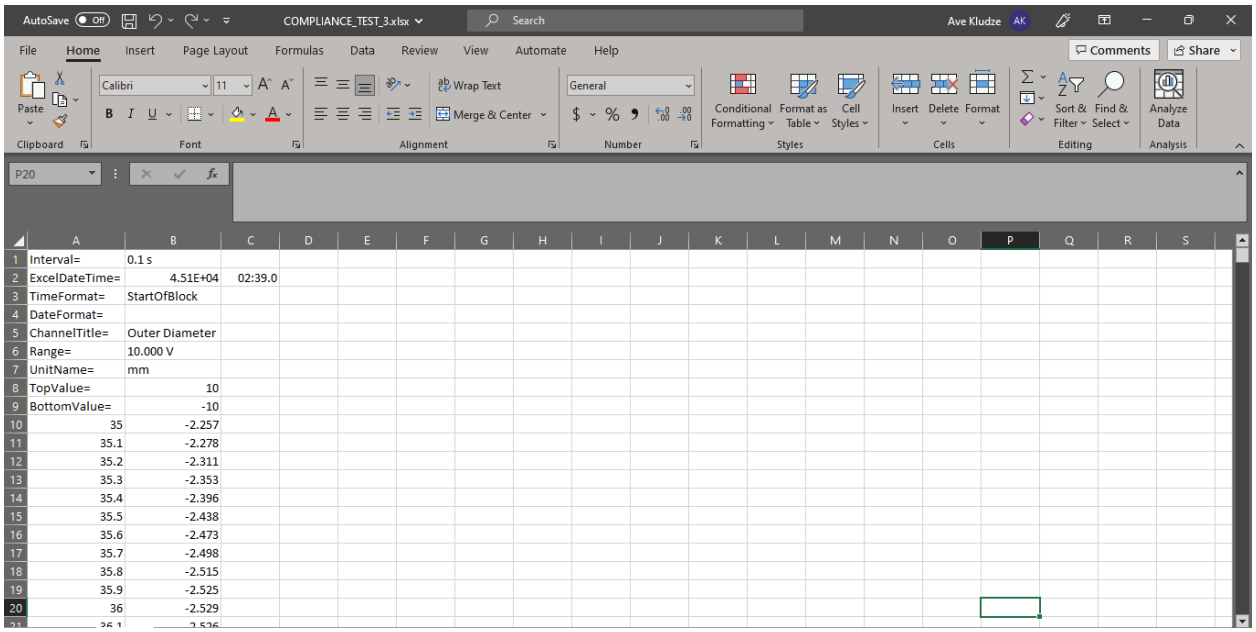
Appendix D: PUTTY Sample Output



	A	B	C
1	=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2023.05.03 12:35:16 =~=~=~=~=~=~=~=~=~=~=~=		
2	63.83	0	2.172
3	63.83	0	2.237
4	64.14	0	2.248
5	64.14	0	2.316
6	64.14	0	2.328
7	64.34	0	2.34
8	64.34	0	2.407
9	64.34	0	2.419
10	64.07	0	2.43
11	64.07	0	2.498
12	64.07	0	2.51
13	63.94	0	2.522
14	63.94	0	2.589
15	63.94	0	2.6
16	63.82	0	2.613
17	63.82	0	2.68
18	63.82	0	2.692
19	63.94	0	2.704

Figure 29. PUTTY Sample Output

Appendix E: LabChart Sample Output



	A	B	C
1	Interval=	0.1 s	
2	ExcelDateTime=	4.51E+04	02:39.0
3	TimeFormat=	StartOfBlock	
4	DateFormat=		
5	ChannelTitle=	Outer Diameter	
6	Range=	10.000 V	
7	UnitName=	mm	
8	TopValue=	10	
9	BottomValue=	-10	
10	35	-2.257	
11	35.1	-2.278	
12	35.2	-2.311	
13	35.3	-2.353	
14	35.4	-2.396	
15	35.5	-2.438	
16	35.6	-2.473	
17	35.7	-2.498	
18	35.8	-2.515	
19	35.9	-2.525	
20	36	-2.529	

Figure 30. LabChart Sample Output