Hardware documentation--Kelsey

Bill of materials

Circuit diagram

How to do epoxy

Better ways

Recommendations--Kelsey

Rod support structure for robotics

Try layering many chips in the same device at different levels

Suggestion of future tests

Add magnetometer to data logs

**Table of Contents**

[Software Documentation](#_lgudyf9qlvfp)

[Software overview](#_8go107evrjd0)

[Adapting code to new sensors](#_mr3av7pyqmvf)

[Software dependencies and installation](#_u9h7pcul0bjx)

[Other resources](#_lraf1p3467xe)

[Presentation of Data](#_qunpk7e47h38)

[Pressure sensor only](#_co18ojc21i12)

[Fully Encased 10-DOF and ESP8266](#_lp1fb6ebeadz)

[Ocean Test](#_dxwfw1i0oqwb)

[Pressure Chamber Test](#_7k2x2lr6dfdf)

# **Software Documentation**

## Software overview

The IMU device uses the ESP8266 microcontroller to execute code written in the Arduino programming environment, retrieve readings from 10-DOF sensor, save data to the SD card, and make data accessible over wifi. Most of the existing code is a combination of example scripts combined together into one cohesive script. As a broad overview, the software starts by looking for a specific wifi SSID for a specified duration. If found, the ESP8266 will switch to data serving mode and create a web server that allows people to browse and download the files on the SD card through a web browser on any computer connected to the same local area network (e.g. the same wifi) when they browse to the IP address of the ESP8266. If the wifi SSID is not found, the ESP8266 will switch to logging mode. First it will emit a long LED blink, followed by a number of short LED blinks, the number of which communicate which log is being saved. The ESP8266 will continue logging data from the IMU at the specified rate until it runs out of power.

The software that controls all functions of the IMU is split into three files for the sake of simplicity, shown below.

**imu\_log\_and\_serve.ino**

* Top level script, contains Arduino’s setup() and loop()
* Runs on startup
* Decides which mode (data logging or data serving) the ESP8266 should switch to
* Contains all user-defined parameters of interest.
  + Logging rate, wifi address, filenames, etc.

**SDWebServer.h**

* Provides all functions needed to browse and download files from the SD card.
* Nearly unchanged from original example code

**imu\_logger.h**

* Connects to SD card and IMU and allows for logging IMU sensor readings at specified intervals to that SD card.

## Adapting code to new sensors

One of the primary advantages of the structure described above is that `imu\_logger.h` and its functions can be easily replaced with code for different sensors. This would allow for the mode manager and web server to continue working unchanged.

Specifically, you will need to modify `imu\_logger.h` to work for your sensor. The function `setup\_sensor()` should be changed to initialize your sensors and `log\_sensor()` to pull your data from your sensors and send them to the SD card. You may also want to update `imu\_log\_and\_serve.ino` with new parameters specific to your system and change some file names.

In the future, it may be helpful to provide further abstraction by automatically handling the writing of data to the SD card, but for now this must be handled by the programmer or simply copied over.

## Software dependencies and installation

1. Set up the Adafruit Feather ESP8266
   1. Download the latest version of the Arduino IDE ([download page](https://www.arduino.cc/en/Main/Software)).
      1. If you already have it, it would be a good idea to update.
   2. Use the Adafruit Feather ESP8266 guide, [starting here](https://learn.adafruit.com/adafruit-feather-huzzah-esp8266?view=all#install-the-esp8266-board-package), to install the board package and configure the IDE for it. Stop once you reach the “Blink Test” section.
2. Set up the 10-DOF sensor
   1. In the Arduino IDE, open the Library Manager (Sketch>Include Library>Manage Libraries)
   2. Search “Adafruit Unified Sensor” and look for the library with that exact name and install it. For some reason it will probably be all the way at the bottom.
   3. Search “Adafruit LSM303DLHC” and install it.
   4. Search “Adafruit L3GD20 U” and install it.
   5. Search “Adafruit BMP085 Unified” and install it.
3. Set up server code on SD card
   1. Put the files contained in “SdRoot” on the top level directory of your SD card (the folder’s contents, not the folder itself).
      1. This folder should be provided in the Code folder alongside this documentation. This is a slight modification of what came with the original example code.
         1. For the original, look in the “/Code/Archive/” folder or `libraries/ESP8266WebServer/examples/SDWebServer/` where your Arduino libraries are stored.
      2. Note: One of our recommendations is to update the html on the SD card to include the file located at “<https://cdnjs.cloudflare.com/ajax/libs/ace/1.1.9/ace.js>” because the server needs this to be cached on your computer in order to work, and most often you will not have connection to the internet and just be running on a local network. Consider doing this.

You should now be all good to go to upload the code to the ESP8266 and run the code (assuming the hardware is set up too).

## Other resources

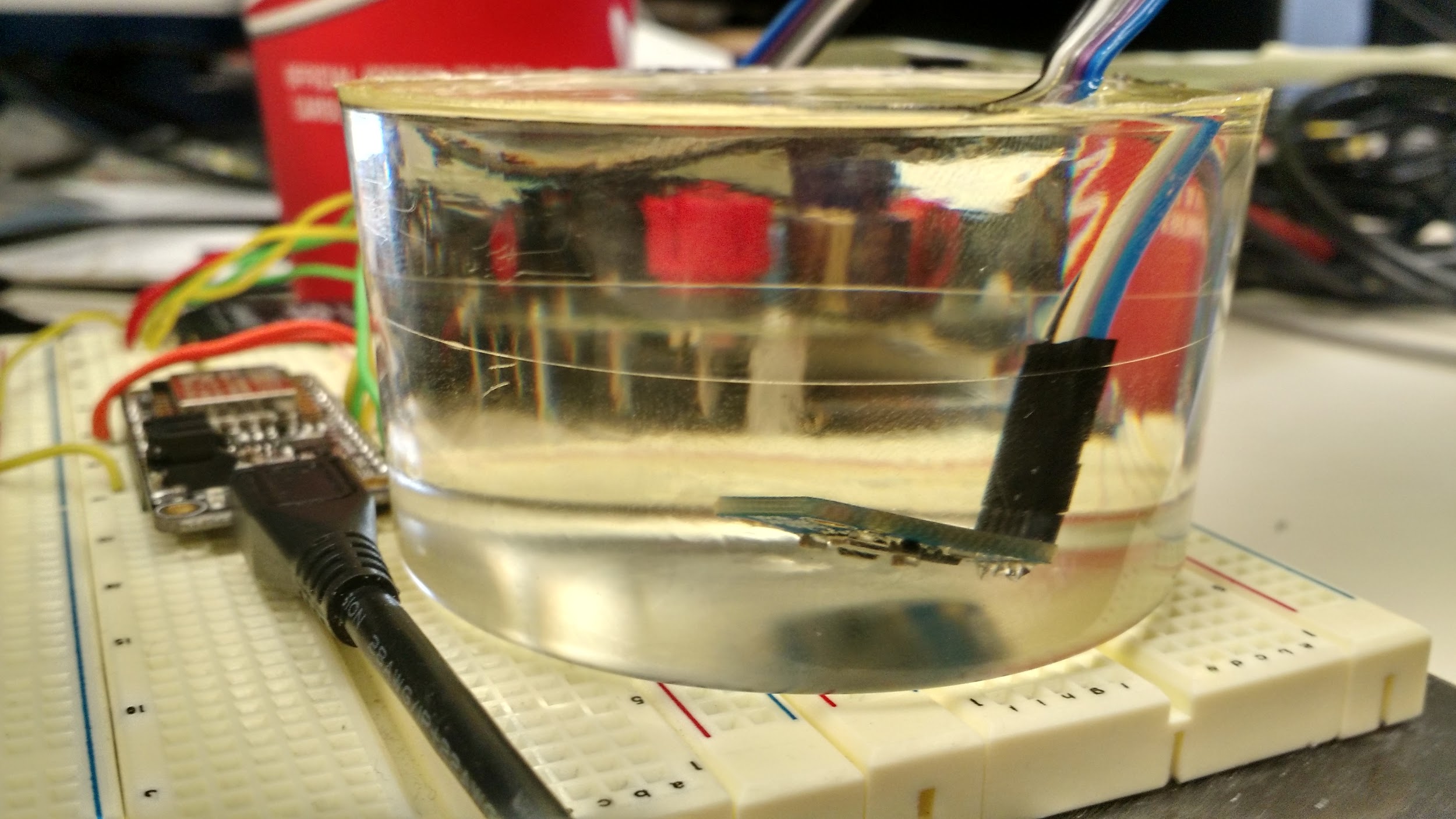
Below are links to the original guides used to create this code and links to the product pages.

* Adafruit Feather HUZZAH ESP8266
  + [Learning Guide / Quick Start](https://learn.adafruit.com/adafruit-feather-huzzah-esp8266?view=all), Adafruit Learn.
  + [Product page](https://www.adafruit.com/product/2821)
* Adafruit 10-DOF IMU Breakout
  + [Learning Guide / Quick Start](https://learn.adafruit.com/adafruit-10-dof-imu-breakout-lsm303-l3gd20-bmp180?view=all), Adafruit Learn.
    - LSM303DLHC
      * [Learning Guide](http://learn.adafruit.com/lsm303-accelerometer-slash-compass-breakout/overview)
      * [Library files](https://github.com/adafruit/Adafruit_LSM303DLHC), Github.
    - L3GD20
      * [Learning Guide](http://learn.adafruit.com/adafruit-triple-axis-gyro-breakout)
      * [Library files](https://github.com/adafruit/Adafruit_L3GD20_U), Github.
    - BMP180
      * [Learning Guide](http://learn.adafruit.com/bmp085/)
      * [Library files](https://github.com/adafruit/Adafruit_BMP085_Unified), Github.
  + [Product page](https://www.adafruit.com/product/1604), Adafruit.

# **Presentation of Data**

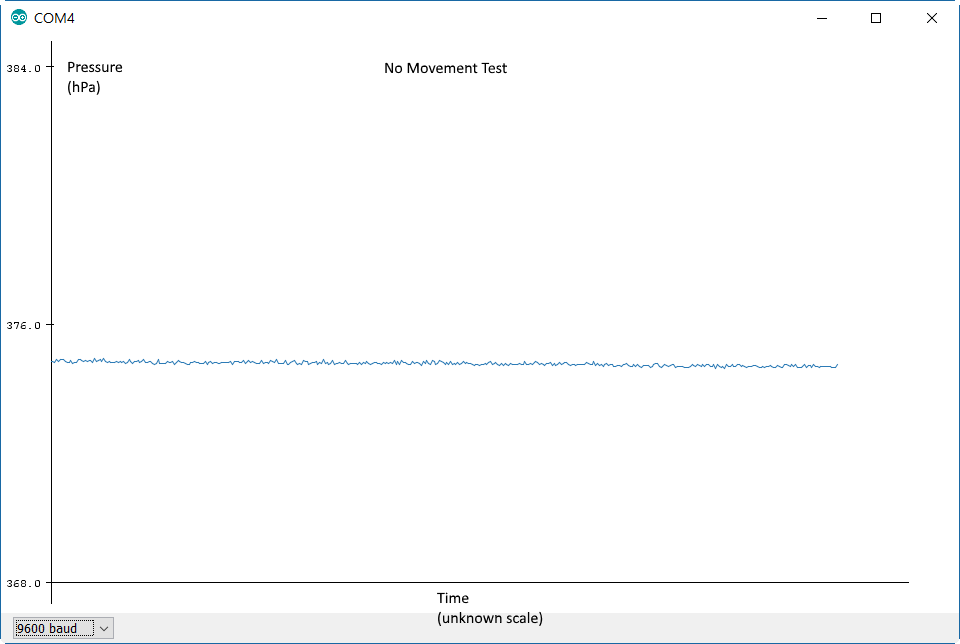
## Pressure sensor only

These tests were done on the device below. This is the Adafruit BMP180 encased in epoxy connected to an external ESP8266. The graphs shown below are only saved in visual form, the data points were not saved (because they are from the Arduino IDE data plotter) but were slightly annotated. The graphs can also be found in the “Data/BMP Only/” folder. Graphs are also presented in the order they were recorded.



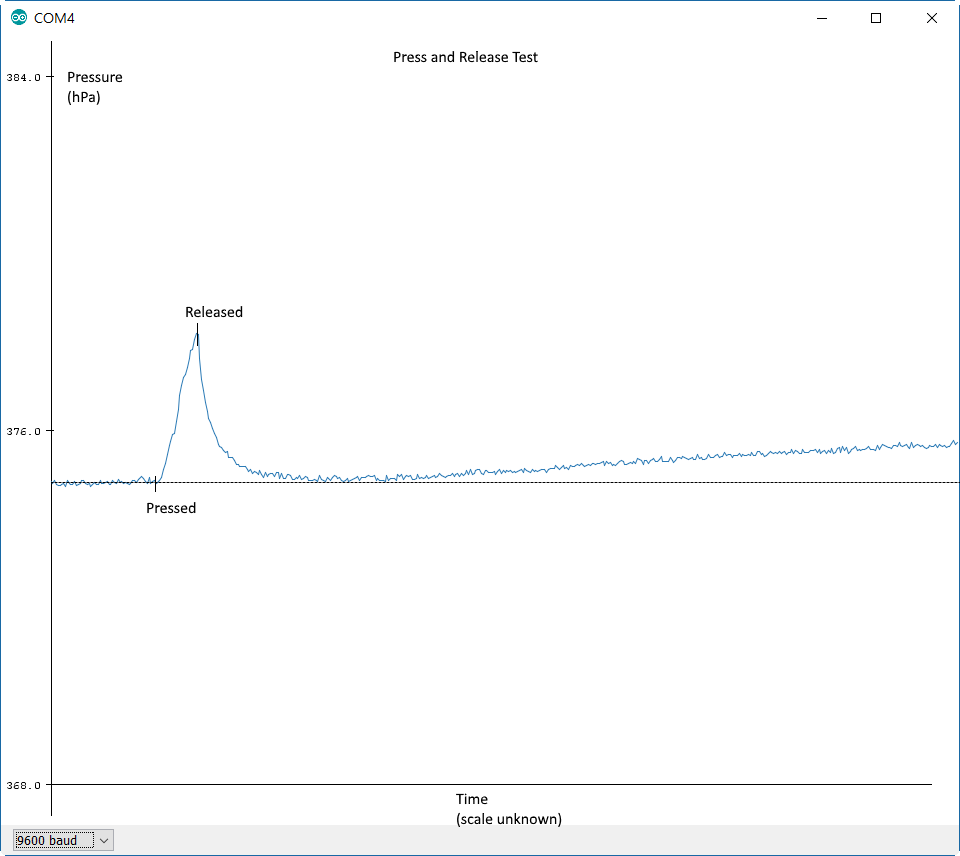
**Test 1: Leaving the sensor sitting on a table.**

Results: Pressure stays constant, not noticeable drift.



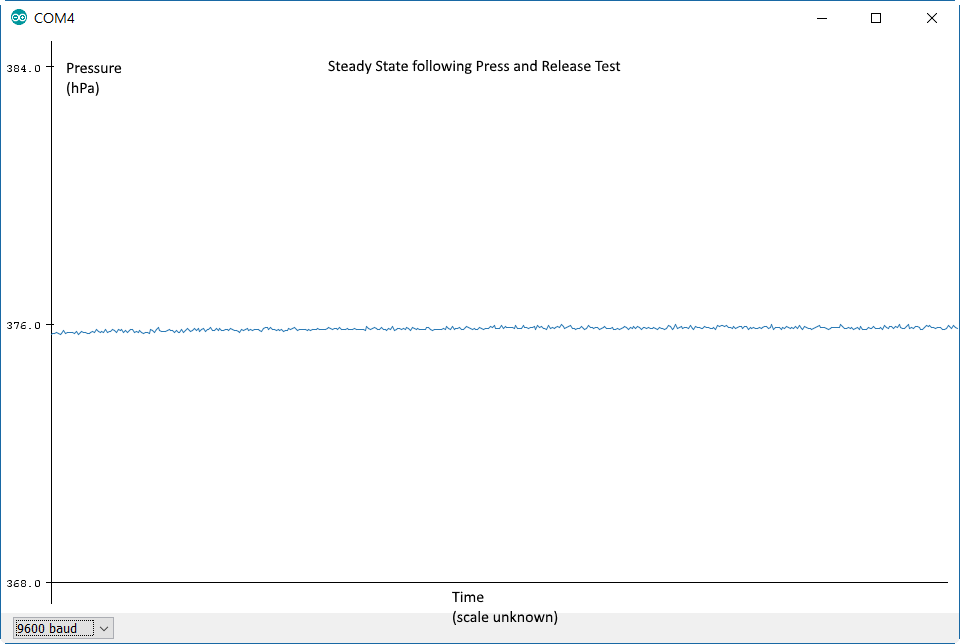
**Test 2: Pressing on epoxy on the side closest to the pressure sensor (ie the bottom).**

Results: Pressing is quickly recognised as a pressure increase but zeros out slower and the steady state value grows slightly.



On the next page, note that the steady state stays higher after pressing and showing the increase in pressure shown above.

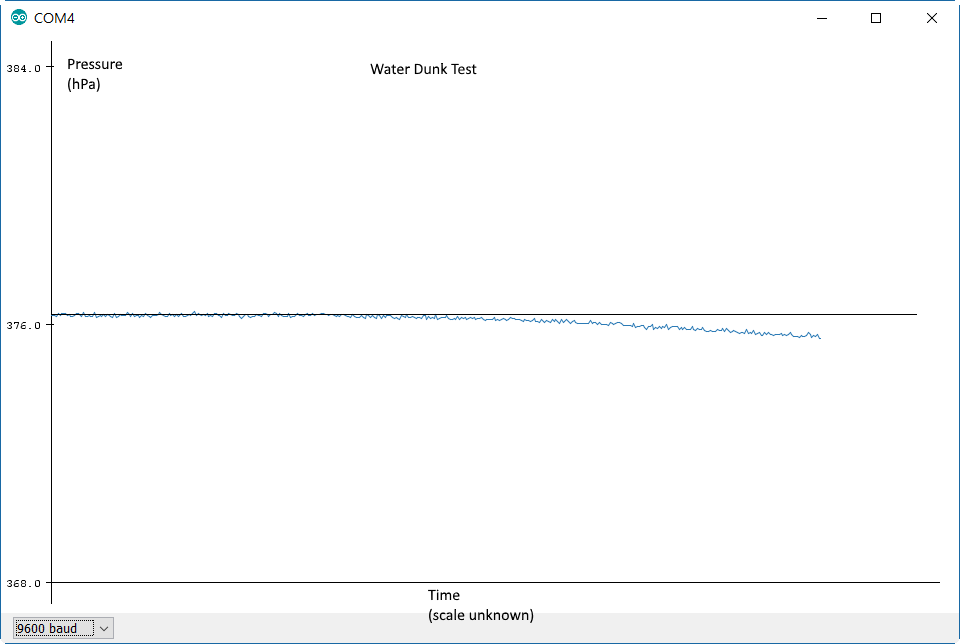
Steady state after pressing (immediately follows last graph):

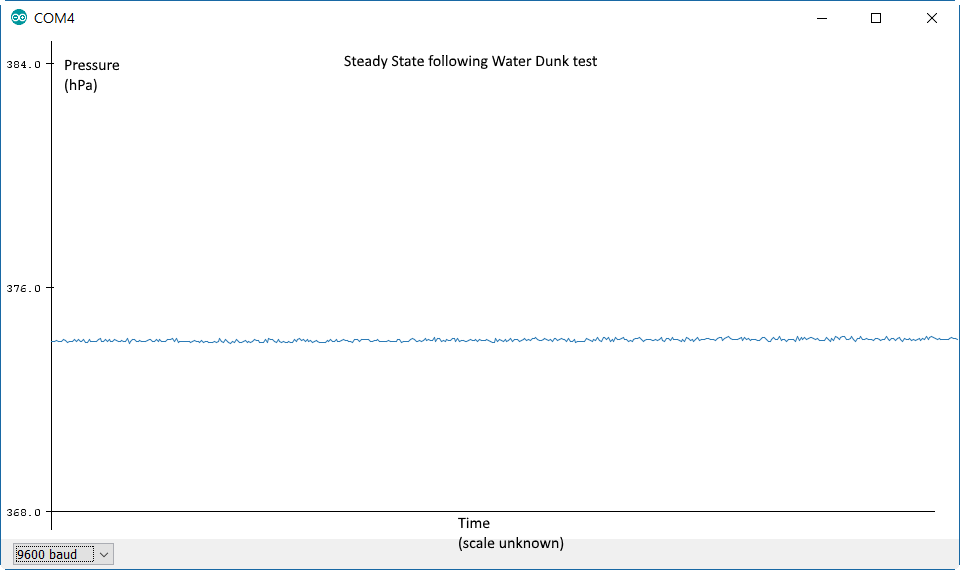


Subsequent tests showed that if we press and release again, and allow a steady state to be reached, the steady state value increases even further..

**Test 3: Putting the epoxied BMP in bucket of water.**

Result: Pressure decreased (graph 1) and then returned to steady state in test 1 (graph 2).





**Test 4: Raise and lower epoxied BMP by about 1m in air.**

No changes.

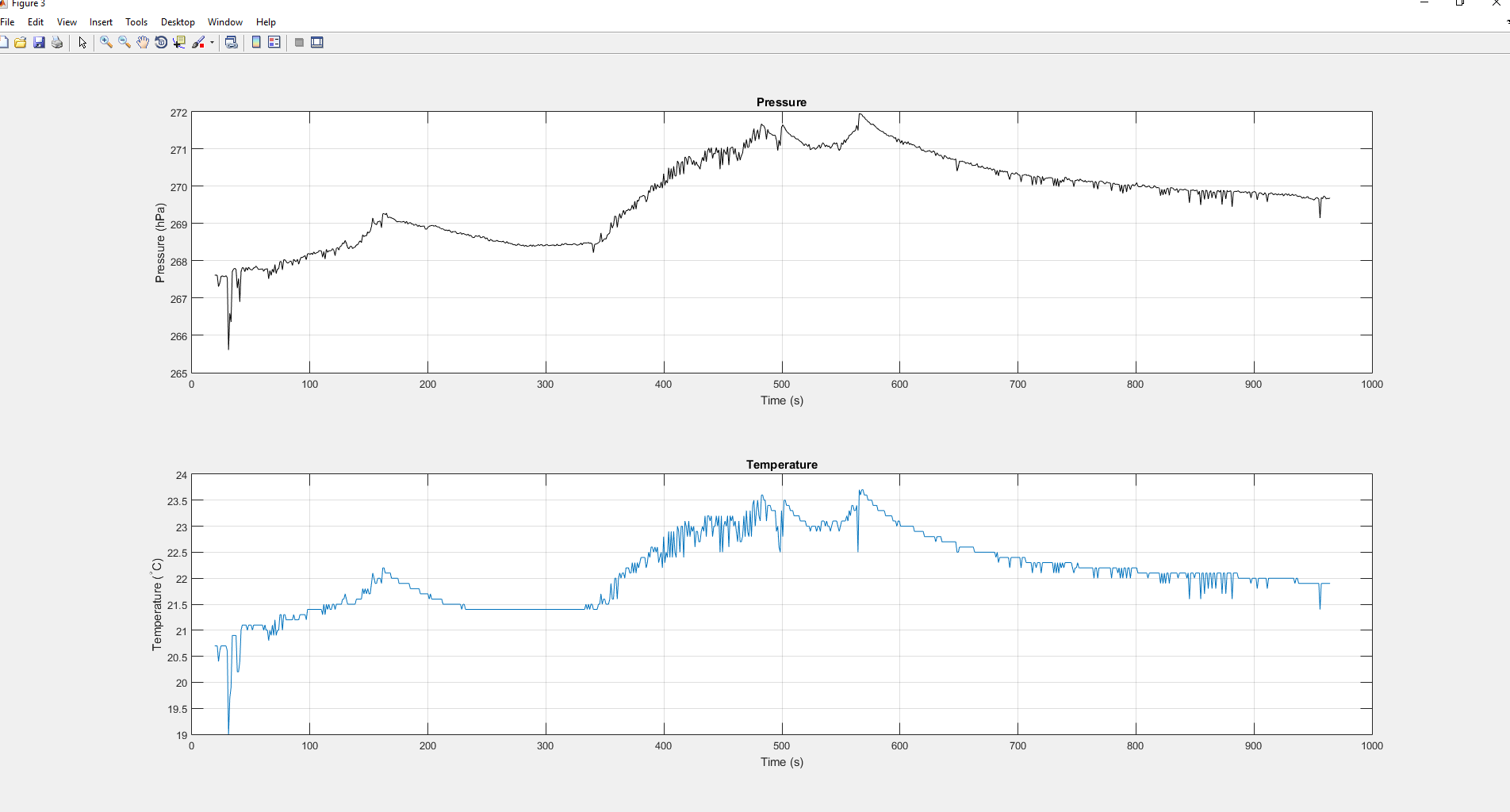
**Conclusions**

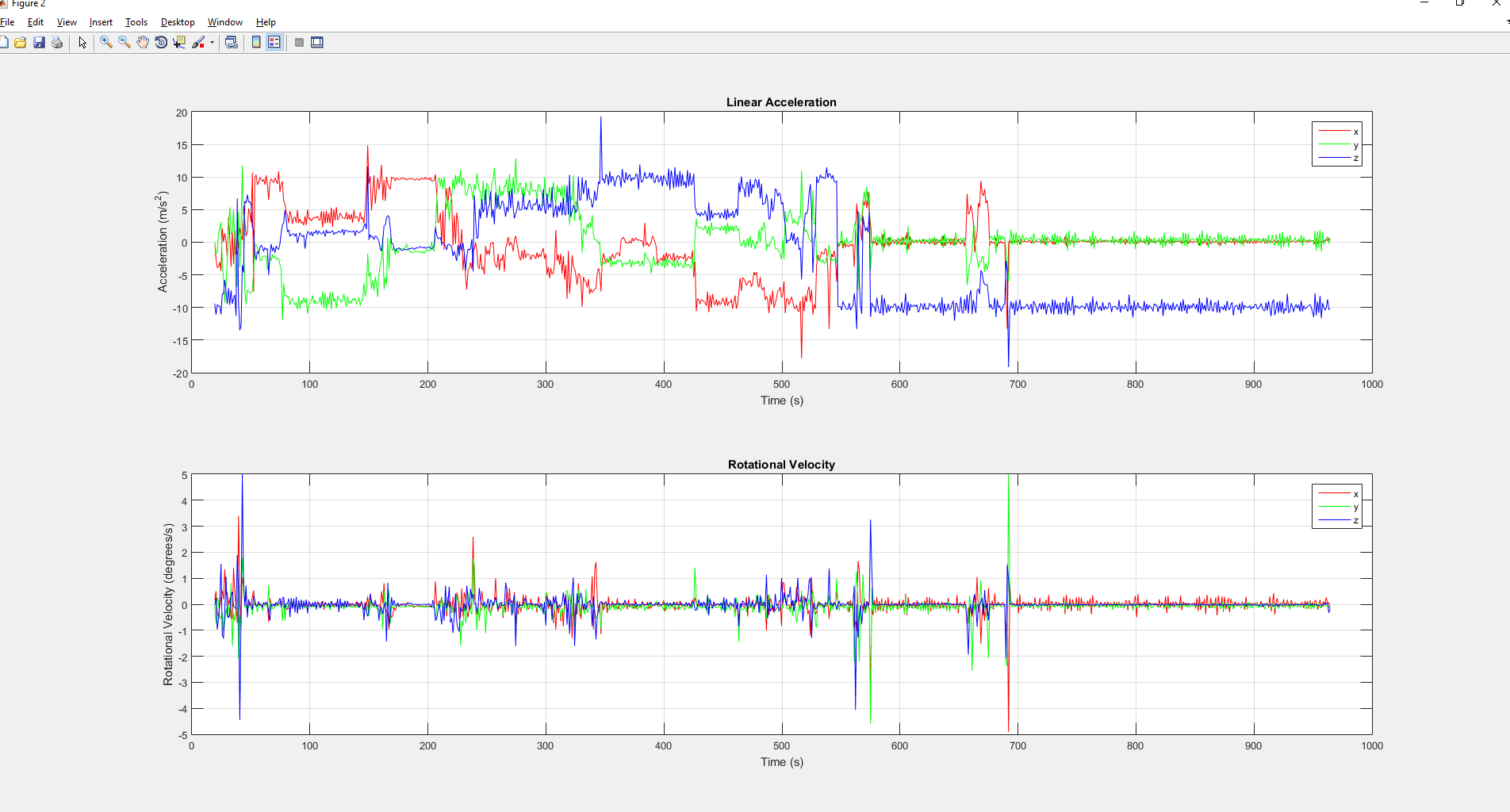
* Pressure sensors works in and out of water
* Steady state seems to change a bit after deformation.
* Depth in epoxy determines sensitivity.
* Drift may be due to temperature, water may have fixed it.

## Fully Encased 10-DOF and ESP8266

### Ocean Test

The fully enclosed system was dropped onto the ocean on a rope. The results are shown below and can also be found in “Data/Full System/Pre-processed graphs/”. Raw data is in ““Data/Full System/Raw data”.



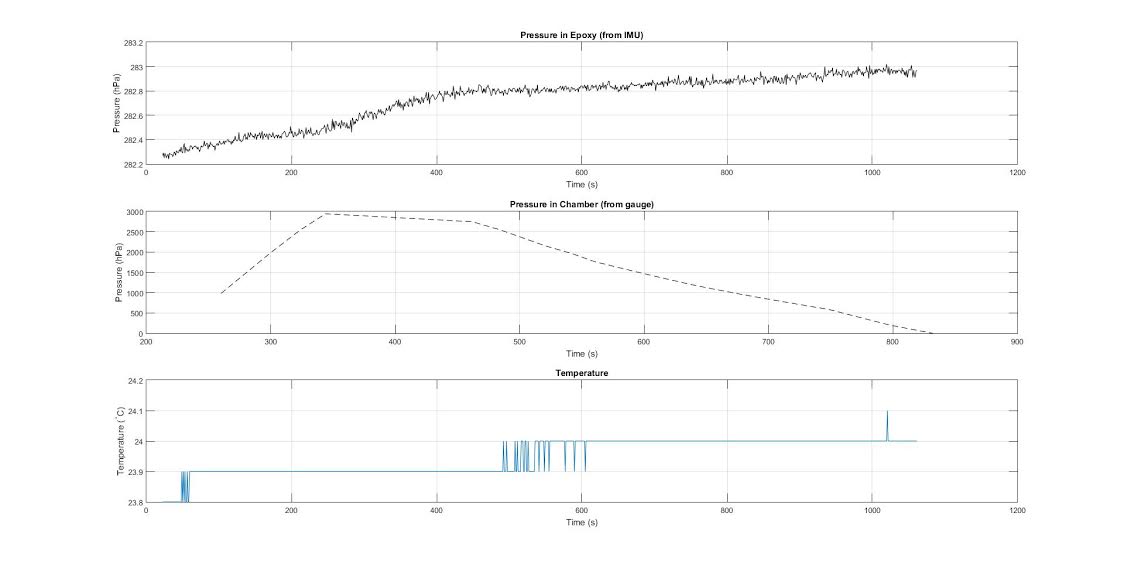


**Conclusions:**

* Pressure increases as we descend but never returned to normal as we started pulling it out.
* Pressure and temperature are strongly correlated. It appears that the correlation is about what you’d expect from the ideal gas law: going from 291 K to 292 K would increase pressure by ~0.3%, and we indeed see pressure go from ~268 hPa to roughly (268\*1.003)=268.8 hPa in that span.

### Pressure Chamber Test

The fully enclosed system was placed in the pressure chamber (in air, not in water) and the pressure was increased and then decreased according to the readings from the “Pressure in Chamber” graph. The results are shown below and can also be found in “Data/Full System/Pre-processed graphs/”. Raw data is in ““Data/Full System/Raw data”.



**Conclusion**

* The pressure recorded by the 10-DOF does not depressurize during our test nor does it react quickly to pressurization.
* Pressure essentially unusable in this current form. Unclear if epoxy is too thick, epoxy has properties that are hindering us, or if sensor itself will never work in epoxy.