Improving end joint sensing for robotic grasping (Using IMUs to estimate contact force)

Nao Ouyang

Abstract

Despite decades of research, robots remain unable to reliably grasp objects in environments not designed for robots. Thus, they are not able to work alongside us in human environments or outside buildings. One approach is to use tactile sensing to predict whether a grasp will succeed or not as well as build simulations to evaluate potential grasps. In this lab, we use an underactuated three-fingered gripper, which makes it difficult to estimate the location of the fingers and and the location of the force with respect to the robot frame of reference.
I investigated the use of an inertial measurement unit to provide such information. By characterizing the stiffness matrix K of the finger, I could investigate whether using an inexpensive inertial measurement unit (IMU) to estimate θ deflection with the linear equation $\tau = K\theta$, I could discover information about the force applied. I did very preliminary work, limiting my experiments to the case where we only desire to estimate the change from directly before to second or two after contact. I also I looked at using of off-the-shelf machine learning approaches to reduce the errors in our model. Finally, I investigated adding more sensors to the fingertip in order to better estimate contact location.

1. Introduction

State-of-the art robots are bad at grasping in unstructured environmnents, limiting their use to structured environments like factory floors. An active area of research in order for robots to expand to unstructured environments (such as kitchens or hospital environments), is the ability to grasp novel objects (objects the robot has not encountered before and may not have any prior information about). Grasp stability analysis plays a critical role in enabling robots to achieve high grasping success rates (in terms of not dropping objects). Through sensing, for instance tactile sensing, we can predict whether or not we will drop the object after grasping and prior to moving. If we find our grasp to be unstable, and assuming the object has not moved during grasping such that releasing our grip might e.g. cause it to tumble, we can simply release our grasp and try again.

Physical model approaches involve modeling the contact forces, contact location, and surface normals (which will differ from the force XYZ if the surface(s) are squishy). However, over the last few decades physical models have proven insufficient. This in part due to complex gripper-object interactions during grasping which are both difficult to measure and hard to model. Thus, one of the aspects of this investigation is to evaluate whether machine learning can be used to improve physical models. We can combine the good aspects of physical models (good generalizability, requiring significantly less data than in pure computer vision machine learning approaches) with

(not having to have extremely precise models). I tried to charactize the residuals in the physical models and determine if these residuals could be more accurately modelled by a black box machine learning approach rather than making the physical model more complex. This is an increasingly popular approach, which makes intuitive sense. We can leverage domain knowledge, instead of using an end-to-end approach for sensor data to stability prediciton.

Although there are a few compact 6 axis force torque sensors, they cost multiple thousands of dollars and are finicky and fragile. Thus, we would like to use an inexpensive nine degree-of-freedom (three-axis acceleromter, gyrometer, and magnetic heading) IMU instead.

During the course of this rotation, senior graduate student Qian Wan discoverd that the current tactile sensors are insufficent for estimating contact location accurately enough to develop a good grasp stability estimate. Thus, this project includes a brief exploration of a new sensor design using the same elements (barometric pressure sensors covered by a layer of elastomer, such as urethane), but more of them.

2. Theory

2.1. Linear Model

For the physical model, I am making a few simplifying assumptions:

1. There is a linear relationship between force and deflection, or alternatively torque and angle of deflection
2. The center of the axis of rotation was at the tip of the proximal joint (in reality, likely it is two or three mm shifted outward, and changes as the flex increases)
3. We consider the z axis to be defined respect to the surface of the finger, so that z=0 is always at the tip of the finger

Note that the consequence of #3 is that we cannot estimate z-axis torque.

2.2. Reference Frame

```
Coordinates
----> + x (roll)
|
|
v
+ y (pitch)

+ z (up out of page) (yaw)


Finger Positions
=======      ====================.
 {CL}  || --- || 15  12  9  6  3 ||
 {A} [xy=0]-- || 14  11  8  5  2 ||
 {MP}  || --- || 13  10  7  4  1 ||
=======      ====================.


  [[---]]
||  WEB  ||
||  CAM  ||
  ||--- ||
```

For the initial (3D stiffness matrix) experiment, Points 13 through 15 are at x=2.6 cm, and each x is spaced 5 mm apart. The y's are at 0.4, 0.1, and -0.2mm respectively.

For the final experiment comparing two stiffness, the x spacing remains roughly the same, while the y's are now at 0.3, -0.1, and -0.4 mm respectively.

2.3. Linear Model in the 1D case

In the one-axis case, the math is straightforward. Experimentally, this is when I am only collecting data down the center (y-axis) of the finger, which should result in almost no roll and pitch, limiting the analysis to yaw.

$$\tau = k \times F \tag{1}$$

For example, one datapoint might be

$$k = F/\tau \tag{2}$$
$$k = (20g \times 9.8m/s^2)/0.1 \text{ degrees} \tag{3}$$

where 'k' represents the stiffness of the finger. Let c represent the inverse of k.
Using least squares error, we may fit a line to find $\hat{c}$.

$$\hat{c} = \frac{1}{\hat{k}} \tag{4}$$
$$\theta = \tau\hat{c} + b \tag{5}$$

where b be a constant determined by the line of fit.

2.4. Linear model residuals

From the above, we may calculate the residuals of any of our estimated variables. For instance, from our actual data we may obtain an estimate for 'k'.

$$\tau_{data} = \hat{k} \cdot \theta_{data} \tag{6}$$
$$\tag{7}$$

Using this k we can go back and calculate estimates for the "true" torque, assuming our linear model was correct.

$$\hat{\tau} = \hat{k} \cdot \theta_{data} \tag{8}$$

We would then calculate our torque residuals as

$$\epsilon_\tau = \hat{\tau} - \tau \tag{9}$$

If we plot a graph of (torque residuals) vs (estimate residuals) and find that our points are randomly scattered around a straight line, then our model well-approximates reality as sensed by a noisy sensor.

However, our residuals may instead follow a parabola, in which case we would want to amend our model to have higher-order terms.

$$\hat{\tau} = \hat{k}\theta_{data} + c_1\theta_{data} + c_2\theta_{data} \tag{10}$$

and so forth.

We may eventually use machine learning techniques to fit higher-order terms.

2.5. Linear model in the 3D case

The 3d case is exactly the same, except now each of the variables are vectors or matrices.

$$\left[\vec{\tau}\right]_{3\times n} = \left[K\right]_{3\times 3} \cdot \left[\vec{\theta}\right]_{3\times n} \tag{11}$$

where

$$\vec{\tau} = \begin{bmatrix} \overset{3\times1}{\tau_1} \mid \overset{3\times1}{\tau_2} \mid ... \mid \overset{3\times1}{\tau_n} \end{bmatrix} \tag{12}$$

$$\tag{13}$$

$$\vec{\theta} = \begin{bmatrix} \overset{3\times1}{\theta_1} \mid \overset{3\times1}{\theta_2} \mid ... \mid \overset{3\times1}{\theta_n} \end{bmatrix} \tag{14}$$

2.6. Sanity Check: Simplify to 2D case

We know that $\tau = r \times F$

$$\begin{bmatrix} \tau_x \ \tau_y \ \tau_z \end{bmatrix} = \begin{bmatrix} \\ K \\ \\ \end{bmatrix}_{3\times3} \begin{bmatrix} \theta_x \ \theta_y \ \theta_z \end{bmatrix} \tag{15}$$

Further, we can use some of the simplifying assumptions we made about to model our system, in order to have an idea of what values our math should result in for k.

We may also see that we will only ever have x and y components for r; z components for F; and thereforce (by how cross products work) only ever have x and y components for τ.

Off-axis

$$r \approx \begin{bmatrix} r_x \\ r_y \\ 0 \end{bmatrix} \times f \approx \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} = \tau = \begin{bmatrix} \tau_x \\ \tau_y \\ 0 \end{bmatrix} \tag{16}$$

$$\tag{17}$$

On-Axis

In the even more simplified case, if we do not apply the force off-axis and there is only a pitch deflection

$$r \approx \begin{bmatrix} r_x \\ 0 \\ 0 \end{bmatrix} \times f \approx \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} = \tau = \begin{bmatrix} 0 \\ \tau_y \\ 0 \end{bmatrix} \tag{18}$$

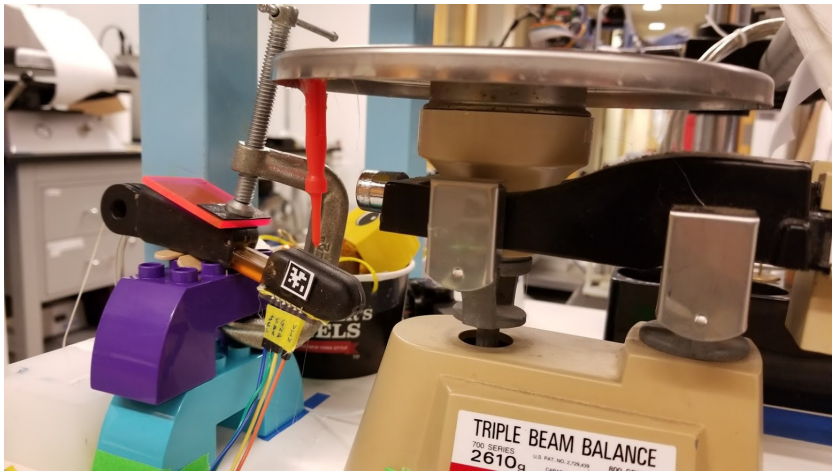
(19)

We **could** have enough nonzero values for the xyz components of θ that we will be able to calculate a nondegenerate estimate for k. This is thanks to applying the force off-axis, causing the finger to roll and create non-zero elements for θ_y (our roll) and θ_z (our yaw). However, after solving for K, I found that I ended up with a bottom row consisting entirely of zeroes.

3. Methods

3.1. Setup

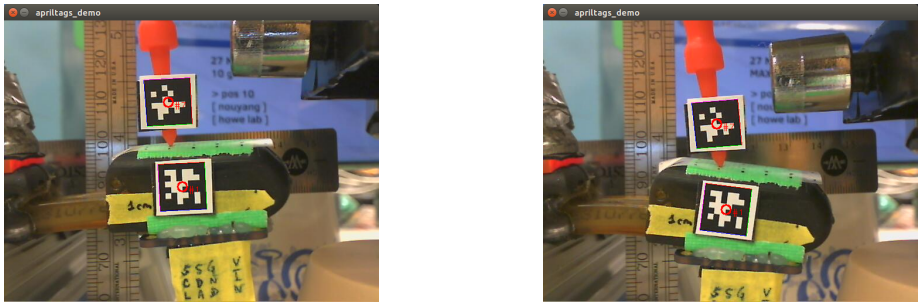
For the setup, I used a triple-beam balance as a way to apply force while retaining a vertical angle. I hot-glued a pumpkin awl to the bottom of the plate and put standard weights on top of of the scale. This caused the awl to contact the surface of the finger, and the finger would then deflect.



3.1.1. One-Dimension Data

In the 1D case, I simply placed a ruler behind the finger and used a webcam to take pictures. By mapping pixel counts to millimeters, I could then estimate the deflection of the finger. The linear fit from these very imprecise measurements proved linear enough that I then relied on the IMU going forward and skipped the step (which should be completed eventually) of characterizing the accuracy of the IMU versus a gold standard, such as a stereo camera (the Microntracker available in lab) or other methods (such as using Apriltags with a calibrated webcam).

In the 1D case I simply took pictures of the finger against a ruler in order to measure angular deflection.



(a) Zero measurement (b) Weight (torque) applied

Figure 1: I drew lines through the vertical (with reference to the camera image) center of the finger before and after weight was applied. The angle between the two is what I wrote down

3.1.2. Three-Dimensional Data

In order to move to the 3D case, I then created a grid of 15 points on the finger. For each of the 15 points, I applied several known amounts of force, with three samples at each force-point combination. Each sample consisted of a ”zero” measurement pre-contact and a measurement post-contact. In post-processing, I then subtracted the two to obtain my final sensor data.

Figure 2: Grid of 15 points, marked with silver sharpie applied through holes poked in a piece of tape.

3.1.3. Instrumentation

I simply hot-glued an IMU to the underside of the finger. I used was the Bosch BNO055 sensor. A breakout was available from adafruit.com for $35, which is relatively inexpensive. I chose this sensor not only for the built-in processing, and also because of the extensive beginner-friendly documentation developed by Adafruit for all of its products. It was recommended to me by a researcher at Right Hand Robotics, which is a spinoff of this lab. I then connected it to an Arduion Duo, which then relayed the sensor readings over serial to my laptop, running Ubuntu 16.04.

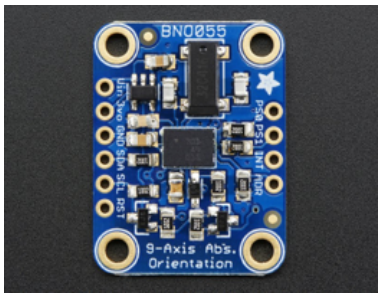


Figure 3: Adafruit BNO055 9-axis orientation sensor. ”Bosch is the first company to get this right by taking a MEMS accelerometer, magnetometer and gyroscope and putting them on a single die with a high speed ARM Cortex-M0 based processor to digest all the sensor data, abstract the sensor fusion and real time requirements away, and spit out data you can use in quaternions, Euler angles or vectors.” (quote and image source: adafruit.com)

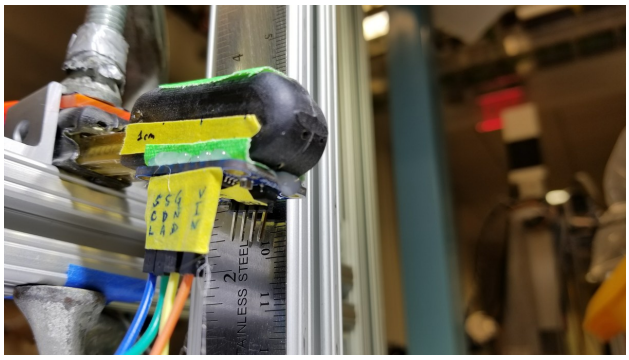


Figure 4: IMU attached to bottom of finger. I made sure that the wires hung loosely and did not interact with the tabletop and press up against the finger

I also attached an Apriltag (which looks like a QR code) and a Micontracker tag (which has a checkerboard like cross), both of which can simply be printed out on paper, and calibrated the webcam for use with the Apriltag C++ script. Although I have collected mictrontracker and apriltag data for comparing to the IMU deflection data, I have not had a chance to analyze this data yet.

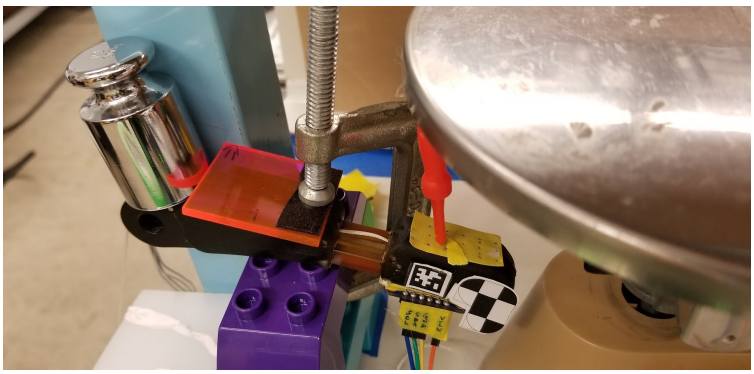
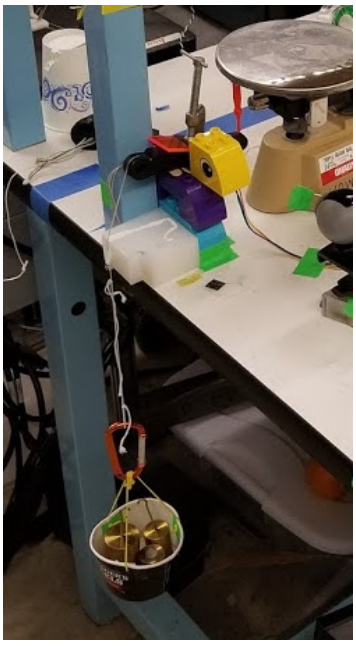


Figure 5: Apriltag on the left and Mictrontracker ”Xpoint” on the right

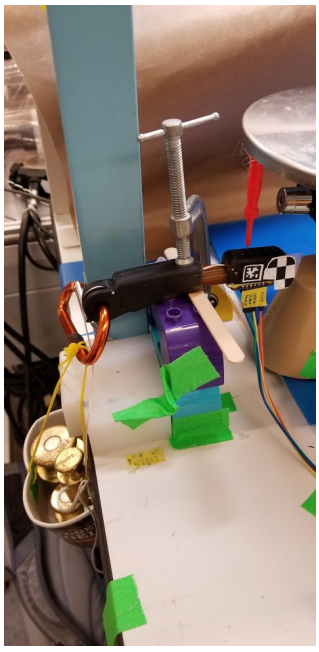
Unfortnuately, the IMU acclerometer would consistently refuse to calibrate if the sensor did not start parallel to the ground. The significant slope of the finger versus the clamp and 90 degree angle of the 80/20 meant that I had to approximate a slanted 80/20 setup by increasing the degrees of freedom on the L brackets (removing t-nuts). This led to a lot of worry on my part about the creakiness and variability in the data collection as I could not guarantee angle consistency, although in retrospect this is almost completely mitigated by the fact that I am zeroing before every reading.

Complications arose from the limited travel of the triple beam balance, which meant I could only measure small deflections (initially on the order of 4 mm; after discovering that I could take the end stop off the back edge of the triple beam balance, about 10mm). At the tip of the finger, I could only less than a hundred grams of ”force” (about 1 N), while 4N is a normal amount of force applied by humans to grasp things. This remained a source of frustration throughout the data collecting and ate up many hours of adjusting the experimental setup. Another source of delay was that it took several attempts to find the MiconTracker manual (found on the old desktop in lab), without which the Microntracker interface was difficult to use.

In order to apply load to pull on the tendon and change the stiffness of the finger, I decided on a simple setup where I hung a weight from the tendon.



(a) Loading the tendon by hanging weights in a paper basket



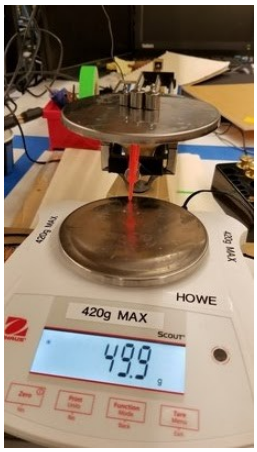
(b) Loading the end of the finger, to help bring the unloaded finger to parallel with the table.

Figure 6: Experimental setup for collecting data at two different stiffnesses (by loading the tendon, as would happen in order for gripper to grasp an object)

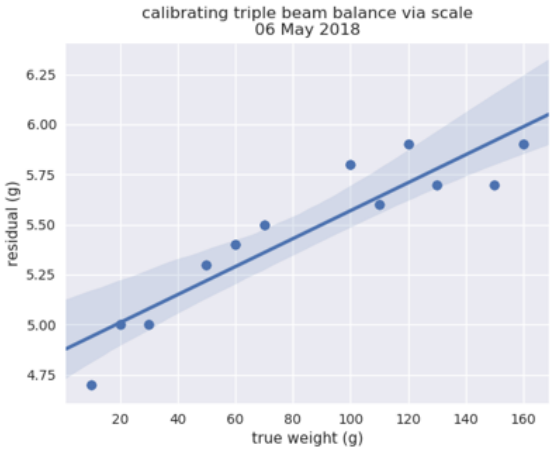
Over time, I trended toward collecting coarser and coarser data as I grew confident that mea-

surements still provided enough data for me to see overall trends in the data (such as the linearity of the data). I shifted from 2g increments, to 20g, and in the end 50g increments. This meant that for the finger when stiffness was applied, I had to change the setup so that the finger started at an angle and became parallel to the floor when load was applied to the tendon. After my deep grievances with producing angles on the 80/20 setup, I decided to use legos with popsicle stick props on one end in order to produce the correct angles.

Finally, I discovered at the very end that I could calibrate the scale to have a constant force offset. Thus, instead of collecting a zero measurement with the tip starting just above the surface, which meant that I had to wait for the oscillations to settle, I could put an offset which made the scale keel all the way to up when the weight was removed. This made zero measurements 2-3x faster, which reduced the cognitive load to remember which measurement I was at, which both reduced data cleanup and needing to recollecting data. This led to significant time savings.



(a) Scale calibration setup



(b) Scale calibration graph, with nominal offset of 5g. I observed that the offset is not constant and increases with increasing load

Figure 7: Scales and more scales: calibrating the weight applied with the triple beam balance using an electronic scale

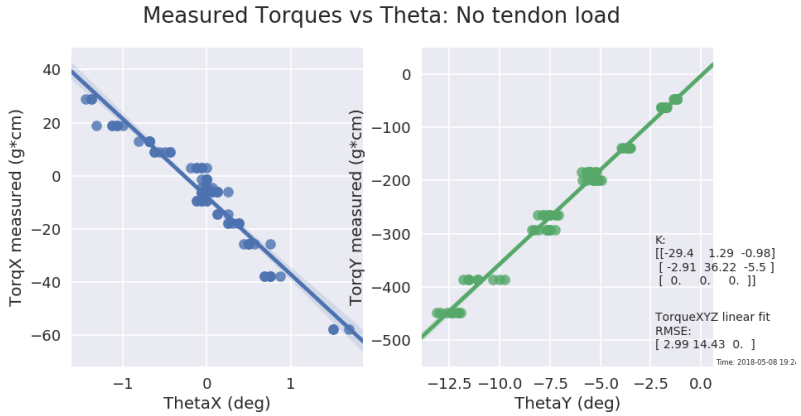
4. Data Analysis

4.1. Loaded tendon data

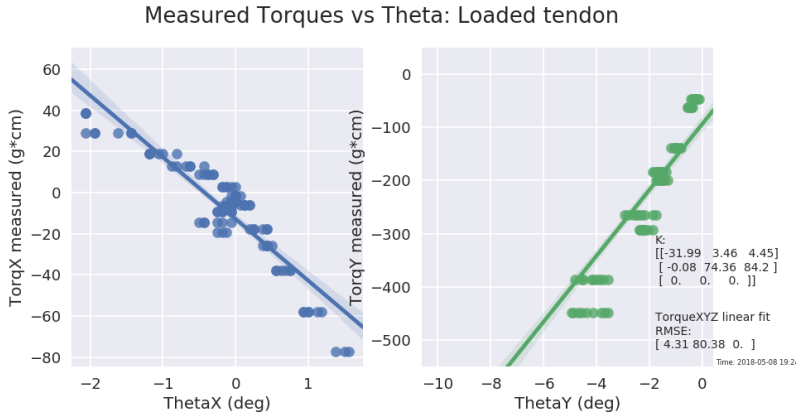
For the final dataset, I aimed to collect data quickly for two different finger tendon loads (which I'll refer to as relaxed and loaded respectively). I collected data in 50g intervals at two y locations and three x locations on the finger (total of 6 positions). Specificially, there were positions 4,5,6 and 10,11,12. x = [3.1, 4.1] and y = [0.2, -0.1, -0.4].

(Note that I defined the center line of the finger to be y=0, but as there is a mold line there, the data is collected from 0.1 mm off of the center line)

First, I conducted a quick sanity check. By plotting torque versus measured theta, I saw that the data follows a linear pattern, which is what I expected based previous experiments. See fig. 8.



(a) Relaxed tendon (no load)



(b) Loaded tendon (with 900g)

Figure 8: Please note that there are two different y labels! The left graph is of Torque X vs Theta X (roll), while the right graph is of the Y counterparts (Torque Y vs Theta Y)

For the Theta Y (yaw) case, the relaxed tendon fit has a root mean square error (RMSE) of 14.4 (gram centimeters), or in other words, if the average x position is 3.5cm, then the average error (for estimating y-axis torque) is about four grams.

On the other hand, the loaded tendon fit has a much higher RMSE of 80.4 g*cm, which if valid, means that the average error is about 23 grams. This is a much larger error, and a bit unfortunate given that we care more about the loaded case in real grasping scenarios.

I then fit a line to the data, and double-checked my work again. The slope of this line gives us our stiffness tensor K. I can double check that we've conducted a linear fit by see that we have by comparing our newly created torque estimates to the original measurements. The datapoints do not fall precisely on a line because the K matrix and fit is in three-dimension. In picking a single dimension to plot, the projection results in datapoints that are not linear in one dimension despite a linear fit overall. The fig. 9 and fig. 10 reassure me that my code is working.

Next I looked at the residuals. Based on the previous experiment, I was expecting to see a linear relationship in the residuals between ThetaY and TorqueY. However, I did not see any such

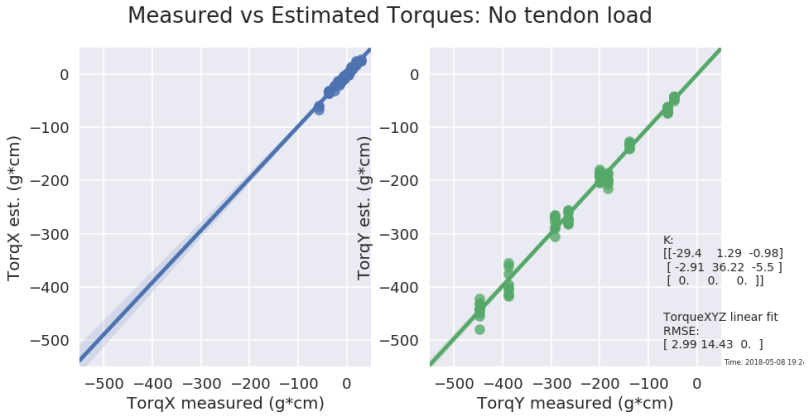


Figure 9: Relaxed tendon

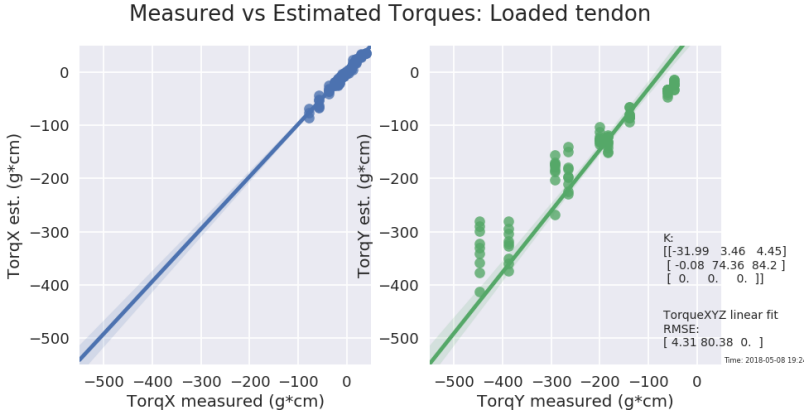


Figure 10: Loaded tendon

relationship in the relaxed case, which should have most closely matched the data in the previous experiment (see section 4.2).

However, in fig. 11 there does not appear to be such a pattern, and it seems like the residuals are somewhat randomly noisy.

Confusingly, as shown in fig. 12 a pattern did appear in the loaded case. Note also that the residuals in the loaded case are much larger. Further investigation will be required to figure out what is going on here.

Perhaps the most satisfying graph that came out of this experiment compares the two stiffness matrices in the relaxed and loaded cases. See fig. 13

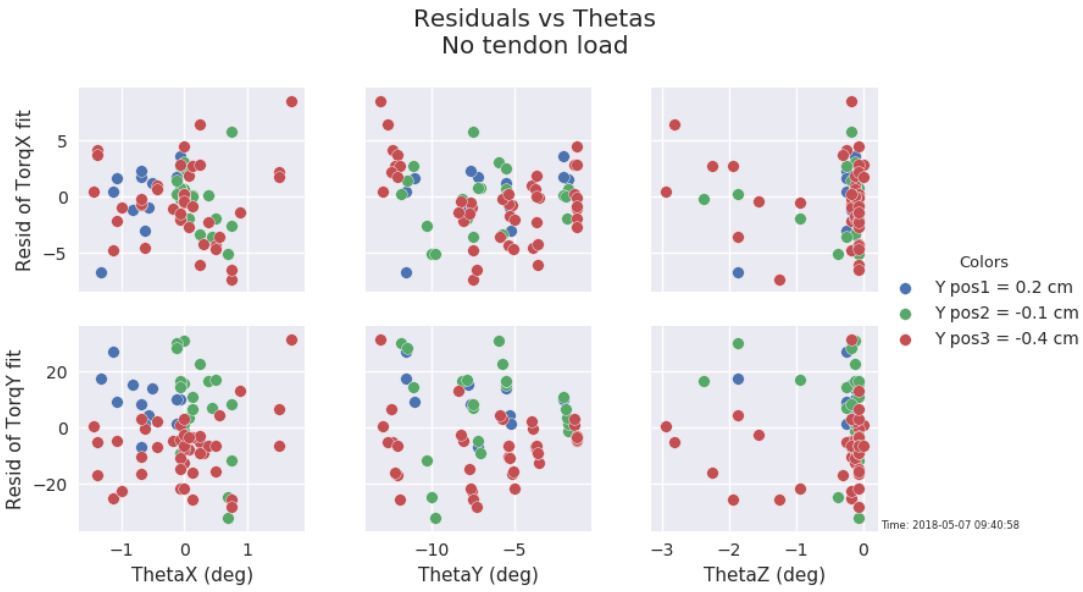


Figure 11: Relaxed tendon

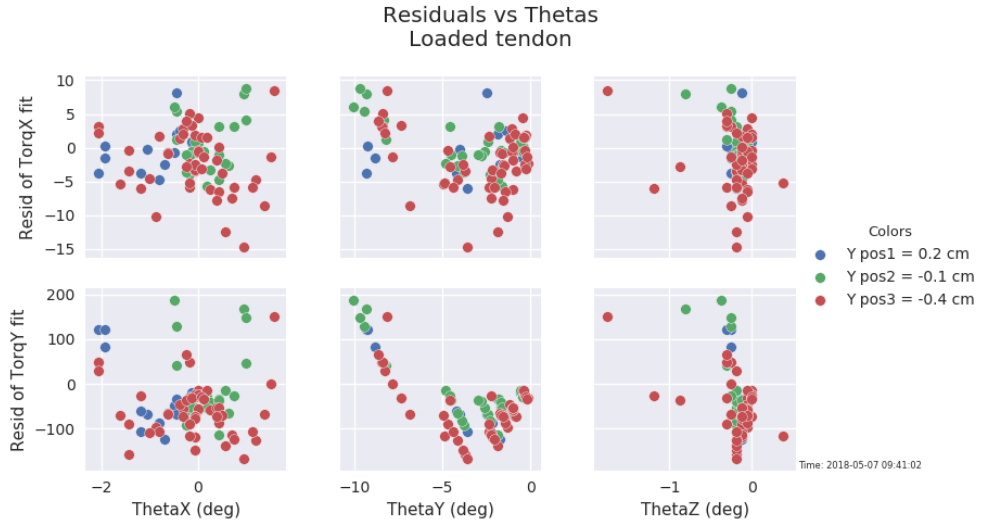
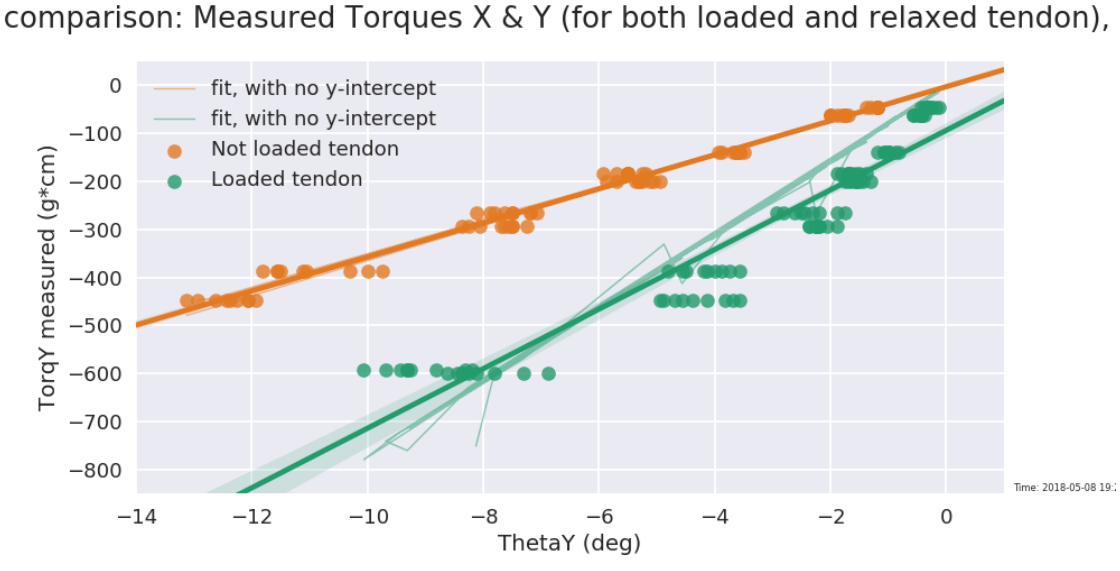
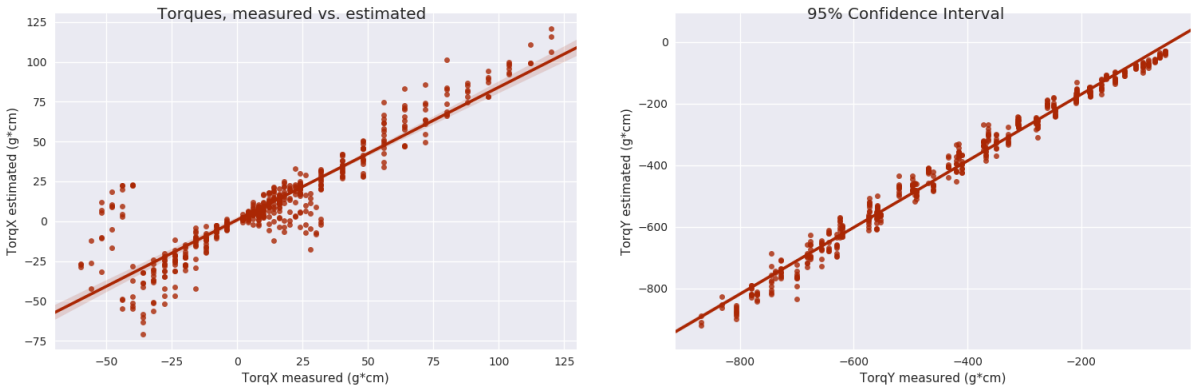


Figure 12: Loaded tendon. NOTE: Patterns in the thetaZ may be ignored as it was not used in the fit, due to the assumption that the contact forceZ is always zero.





(a) (b)

Figure 14: Measured torques on the X axis, and estimated torques on the Y axis. The left graph shows the X axis (roll) measurements, note the magnitude is much smaller compared to the Y axis torque (yaw) measurements. Please ignore the erroneous title on right graph.

4.2. Initial Experiment: 15 positions, with the relaxed tendon only

The experiment directly prior to the final experiment consisted of collecting nearly 500 points, mostly in 20g increments, from all 15 positions labelled on the finger (albeit a different finger, which did not have a tendon string in it – I had to change fingers because of that).

I started with a sanity check as before, and plotted the estimated vs measured torques. See fig. 14.

The large residuals on the left were alarming. To take a closer look at what was going on, I colored the graph in by X and by Y positions. In fig. 15 we see that all of the datapoints way off the line come from xpos4 and xpos5, specifically from y = -0.2cm (from fig. 15). This does not seem to be a fluke, as these are not sequential measurements (positions 12 and 15 respectively).

On the other hand, the yaw (torque Y) seemed like a very clean fit. fig. 16

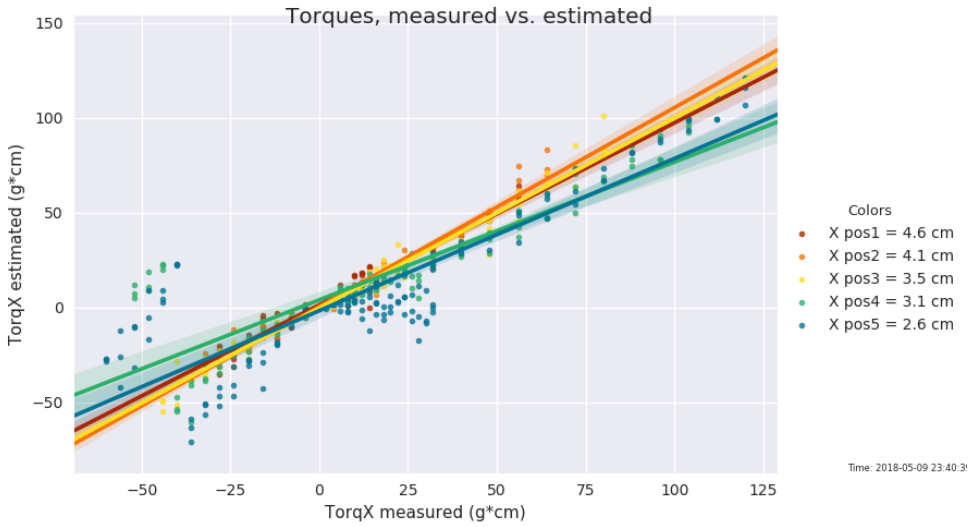
I then systematically plotted the residuals against the factors that might affect them: the torques, forces, and angular deflection amounts. For instance, it seems plausible that, as the amount of force applied increases, the residuals might increase. See

For additional graphs, please see Appendix E.

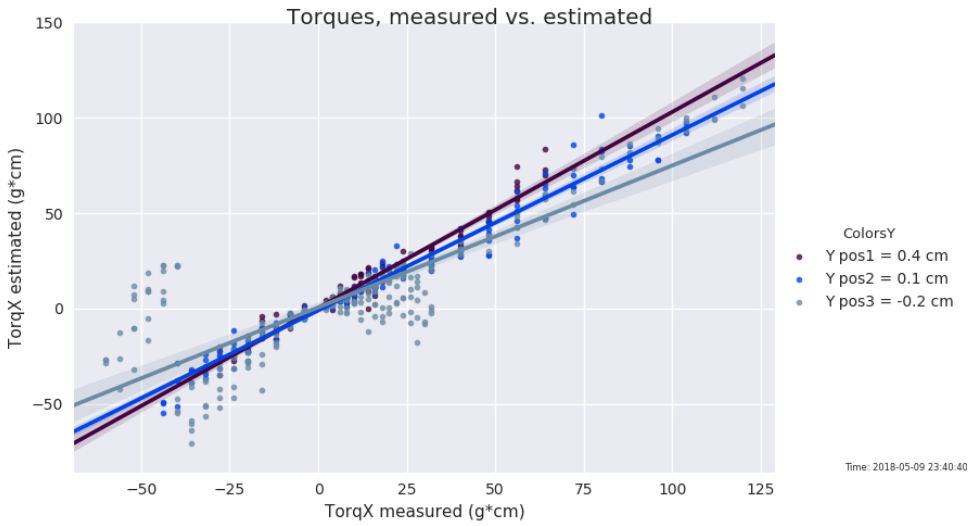
5. Attempted fit improvements

Finally, I briefly attempted to ”use machine learning or other methods to fit the residuals and improve the overall torque estimate”.

- For the first experiment (with 492 datapoints) the linear fit RSME was 36.63.
- With an additional term consisting of a linear fit on the residuals, where I allowed a y-intercept, RMSE = 26.65
- With a random forest fit, RMSE = 22.56



(a) Torque measured vs estimated, colored in by X position



(b) Torque measured vs estimated, colored in by Y position

Figure 15: Measured torques on the X axis, and estimated torques on the Y axis. The left graph shows the X axis (roll) measurements, note the magnitude is much smaller compared to the Y axis torque (yaw) measurements. Please ignore the erroneous title on right graph.

However, without any cross validation, it's likely that the improvements are due to overfitting and will not generalize at all, so really this is just a sort of hand-waving exercise.

I did briefly look into the generalization problem, both by calculating cross-validation statistics (now lost to time) and a brief inspection involving taking out one of the xy positions in the training data, training on the remaining torque-theta data, and then fitting to the thetas inthe dropped-position test set (see fig. E.29).

I also did a brief look into fitting the residuals on the new dataset (with both loaded and relaxed

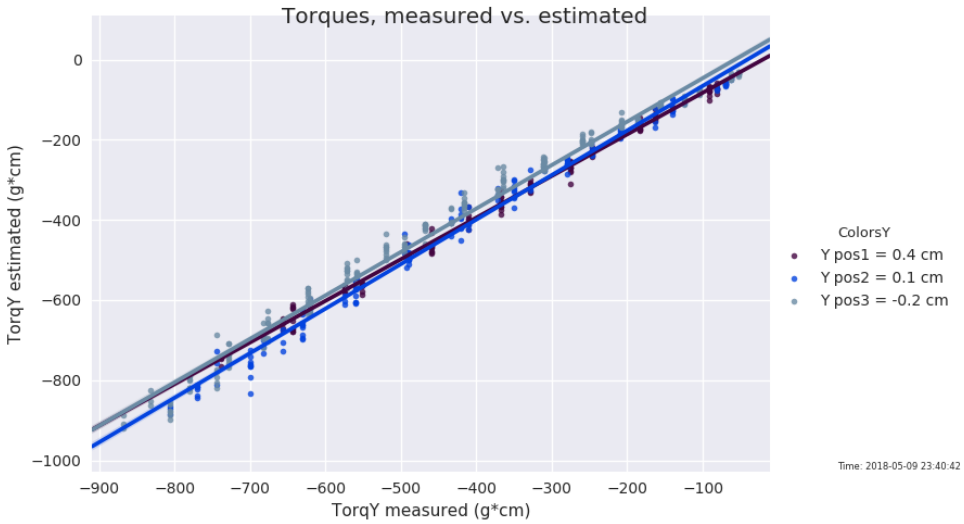


Figure 16: Compared to the torque X fit, the torque Y fit seems very clean.

tendon measurements).

Table 1: Brief investigation of fitting residuals on final experiment. "Black box" indicates fitting the model directly to the theta inputs and torque outputs. "Corrected linear fit" indicates taking a linear fit (with no y-intercept), calculating residuals, training another model on those residuals, and updating the model with that new residual estimate.

	Linear	Polynomial	Gradient Boost	Adaboost	Random Forest
Black Box	36.1	9.2	15.8	25.1	80.4
Corrected Linear fit	28.5	18.8	18.8	25.1	52.9

I graphed the results in fig. 19. Note that this is a very prelimary exploratory analysis, since I only used one dimension of data for the residual calculations (the K was still fit with all three dimensions). I found the graphs really interesting in displaying what each algorithm attempts to do. Of course, each of the machine learning algorithms can be tuned and should be tested to eliminate overfitting.

Notably, as shown in table 1, the polynomial fit directly on the data (torque Y vs theta Y **in the loaded case**) was very good, and did better than even models with additional residual terms.

6. Sensor Design

For datasheets, please see Appendix G.

I produced Arduino and python code that allowed me to view a plot in real time of the sensor data. I was able to read data from three sensors, plot data from two in real time (more possible of course), and test the addition of some elastomer on top of the sensor. See fig. 20c

I also integrated the Arduino with an RGB LED strip, where if you pressed on the sensor, more LEDs would light up.

Figure 17: Torque estimate residuals versus torques estimates, colored by their X or Y positions. There does appear to be a suspicious trend between the residuals and the torque Y measurement.

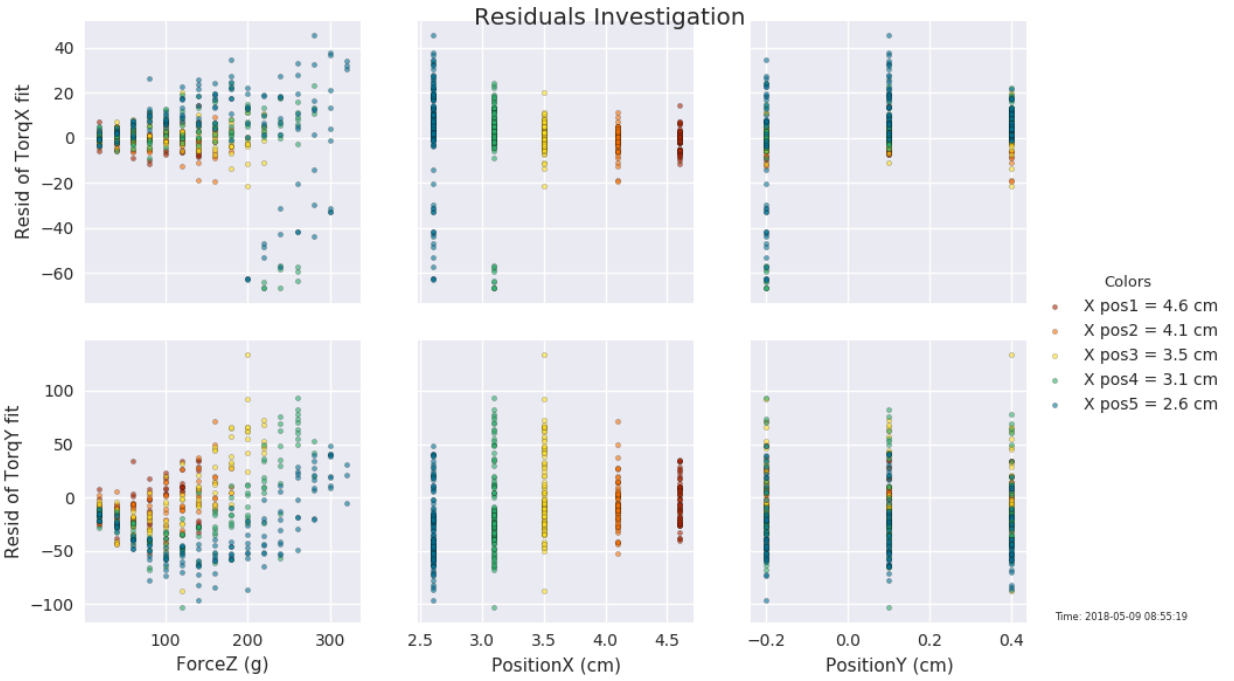


Figure 18: There seems to be generally somewhat random residuals, except there's a chunk of large residuals on the upper right graph of TorqX residuals vs force applied. I never resolved the cause of this. Additionally, note that the first X position, where the finger is least stiff, has a noticeably different set of residuals than the other positions.

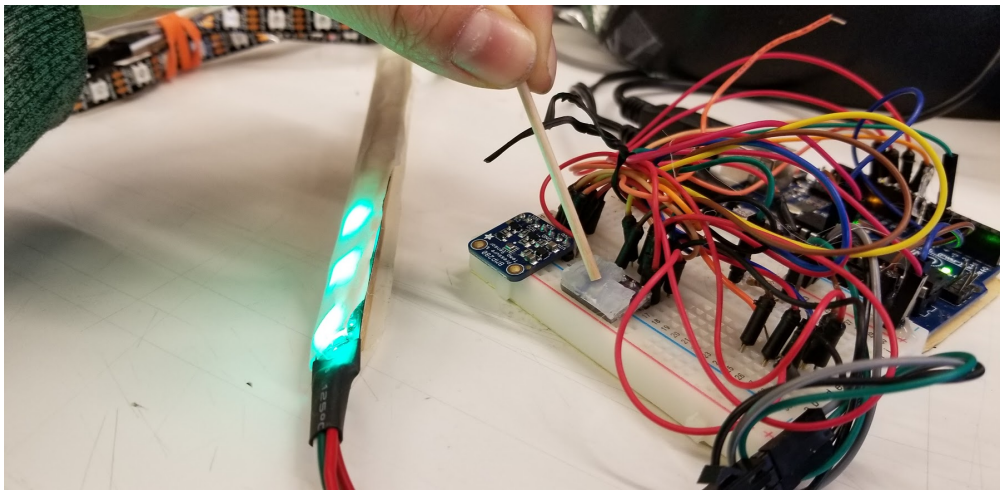


Figure 21: LED setup. The left (blue) PCB is an Adafruit breakout available for $10. The generic breakout can be had for less than a $1 (it has fewer components, e.g. does not support 5V input

The sensor showed a large amount of hysteresis, which should be investigated and addressed.

Note that the setup required use of a separate power supply, otherwise the Arduino would refuse to talk to the computer (for real-time plotting) when the LED strip drew load on the 5V line.

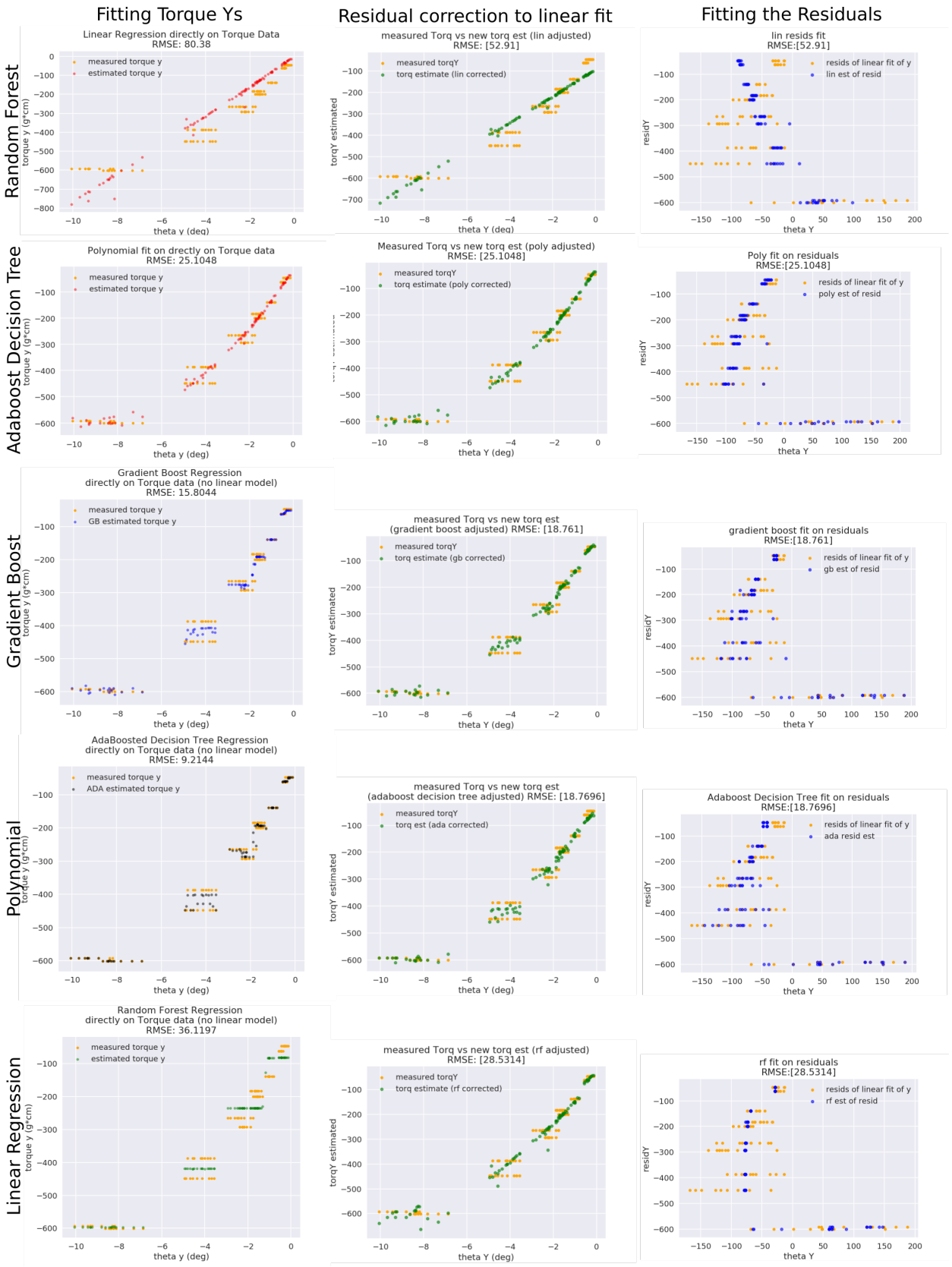
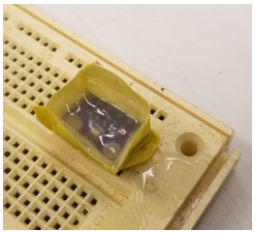
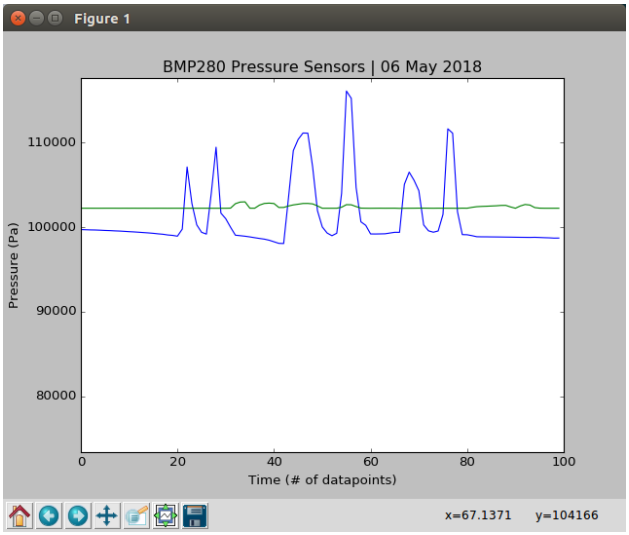


Figure 19: Visualization of the black box approach (leftmost column), then a linear model with residual correction (middle column), and finally a look at how each algorithm is interacting with the residuals (rightmost column)



(a) Mold for casting (I found some leftover Dragonskin 20 in lab

(b) After degassing in a vacuum chamber and curing



(c) Sensor readings as plotted in python on my laptop

Figure 20: Sensor prototype.

Evaluation of the sensor is ongoing. It has a built-in temperature sensor, which can be used to compensate for temperature changes (for instance, if you press on it with your finger, the heat compounds the reading and also resuls in a noticeably tail as the heat dissipates relatively slowly.

7. Knowledge

In the process of this research, I learned about how accelerometers are calibrated, which I found very interesting (although ultimately, thanks to the built-in sensor fusion and calibration routines of the BNO055 sensor, I did not need to write my own calibration routines). The common presence of the gravity vector across meaurements at diffent angles makes this calibration possible.

I learned the most about exploratory statistical data analysis. Where previously I had not even heard of residuals (only scalar measurements such as mean-square-error), I learned about how they could be compared to a horizontal basis against which they would be randomly distributed assuming random noise around a proper physical model.

Finally, I learned more about the I2C and SPI protocols for receiving (and writing) to multiple sensors. I began work on creating a custom printed circuit board to create a 3x4 array of sensors, and decided to switch and learn to use a newcomer in the PCB design space, which has CERN support: the free and open-source KiCAD tool.

8. Conclusions

The stiffness of the finger was remarkably linear, which meant that machine learning and higher order terms had little to contribute to improving the fit between measure theta deflections

and force estimates. Thus, the IMU appears to have much to contribute to estimating forces applied to the finger. However, caution is necessary, since in actual grasping, as Qian demonstrated, the axis I measured becomes the stiffest axis when the tendon is pulled taut to grasp an object.It bends relatively little compared to the other axes.

Qian also showed me how the finger relied extensively on mechanical mechanism intelligence (as opposed to full actuated parallel grippers, commonly used in computer science labs). I saw that complete failures tended to occur because the finger had failed to detect contact when contact had already occured. This suggests the better instrumentation of the finger is critical. The IMU by itself can only suggest contact force, and the tactile sensors are needed to tell contact location. To evaluate the contribution of the IMU to force estimates and not just torque estimates, we must build better location sensors than the current 1x4 array.

8.1. Future Work

The microntracker should be used to evaluate the IMU. Evaluating the use of Apriltags with the microntracker would help reproducibility by others (or future grad students), as webcams are cheap and readily available, and documentation for apriltags is freely available online.

A full sensor array on a custom PCB, manufactured into a finger unit, should be developed and characterized. By collecting data and then fusing this with the IMU data, we could physically evaluate the grasp stability preductions, producing an important tie between theory and practice to ensure the two complement each other. The important of this has been mentioned above.

9. Thanks

Thanks to Qian who fielded many of my naive questions about grasping and and had to deal with my relative over-enthusiasm, Buse, who I enjoyed comisserating with, Alperen teaching me how to be safe in lab, Yash for many admirably terrible jokes James Weaver for very late night encouragements, and to my friends at MD309 and my fellow first-year SEAS PhD students. And thanks to Prof. Robert Howe for patiently giving me a fairly gentle introduction to the world of research, and offering to advise me.

Appendix

Appendix A. Apriltags

I decided to investigate apriltags, which helped me refresh on my knowledge of C++. I struggled to find an appropriate ROS driver, and ultimately was able to get the C++ working (despite some issues with the openCV version on my laptop) with the help of Patrick from Prof. Kuindersma's lab.

Example data output

```
2 tags detected:
Id: 1 (Hamming: 0) distance=0.079741m, x=0.000532, y=0.006102, z=-1.487915, yaw=-0.134615,
    pitch=0.071828, roll=-0.041146
Id: 7 (Hamming: 0) distance=0.079741m, x=0.000532, y=0.006102, z=-1.487915, yaw=-0.134615,
    pitch=0.071828, roll=-0.041146
14.9312 fps
```

My issues with the openCV version were simply solved by specifiying the openCV version the compiler should use, as I had two versions on my laptop.

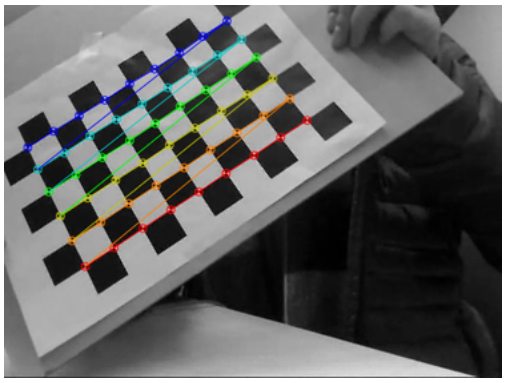
```
nrw@earlgrey:~/projects/apriltags$ vi CMakeLists.txt
    (line 14) find_package(OpenCV 2.4.9.1 EXACT REQUIRED)
```

Fig. 1: Applications and users of AprilTags. From top left, clockwise: robot-to-robot localization and identification on MAGIC robots, object localization for Boston Dynamics' Atlas robot, testing of virtual reality headsets at Valve, and tracking individual ants to study their social organization [4].

Figure A.22: Applications of apriltags. Figure from: Wang, John, and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection." IROS 2016.

To calibrate the camera, I was able to find a (very!) well-documented python script online that allowed be to calibrate the camera simply by taking a few pictures of a checkerboard.

```
(venv) nrw@earlgrey:~/projects/video2calibration$ ./calibrate.py example_input/chessboard.avi
     calibration.yaml --debug-dir out
```



Using the output of the calibration script, I could then set the values in the apriltag example code to give me (fairly accurate) estimates of the xyz location of the finger in real-world units.

```
nrw@earlgrey:~/projects/video2calibration$ ./calibrate.py ~/Videos/Webcam/2018-03-26-112657.webm calibration.yaml --debug-dir out

Performing calibration...
 RMS: 0.442700776066
 camera matrix:
 [[ 666.78668352    0.          343.73827809]
 [   0.          665.79103853  227.19081685]
 [   0.            0.            1.        ]]
 distortion coefficients:  [  6.06301194e-02  -1.94620209e-02   1.45555284e-04   1.24410189e-03
 -2.51439333e-01]
```

I did not use the distortion coefficients in the apriltags measurements. These coefficients, I believe, correct for fisheye and higher order distortions in the camera images.

```
public:

  // default constructor
  Demo() :
    // default settiwgs, most can be modified through command line options (see below)
 [...excerpted section...]
    m_width(640),
    m_height(480),
    m_tagSize(0.00944), // in meters
    m_fx(667), // in pixels
    m_fy(666), //
    m_px(344), // principal point
    m_py(227),
```

Appendix B. Project Management

In order to improve time estimates in the future, I find it helpful to look back at the original timeline.

299r Rotation: Harvard Biorobotics Lab
Spring 2018

Force Estimation with Inertial Measurement Units and with Applications of Machine Learning

Milestones

Week starting on Monday

1. **12 Feb**
   1D – Literature search, setup, collect preliminary data for F = kx (1st order model), in one axis of reflection (no twist of finger – directly up and down)
2. **19 Feb**
   1D – Analyze data, improve with machine learning
3. **26 Feb**
   3D – Collect off-axis deflection data (with twist). Test assumptions of linearity – A. Fix spot and change force, see if deflection angle changes B. Change spot, given same force, how much does the finger deflect at different points along the finger?
4. **05 Mar**
   3D – Collect data and analyze (with lower order model, then higher model with machine learning)
5. **12 Mar**
   3D – Analyze and write up conclusions
6. **19 Mar**
   Tendon – Setup for tendon measurements, where finger is curled
7. **26 Mar**
   Tendon – Collect tendon data
8. **02 Apr**
   Tendon – Analyze data, iterate on machine learning
9. **09 Apr**
   Tendon – continue analysis
10. **16 Apr**
    Spacer week
11. **23 Apr**
    Writeup
12. **Monday, 30 April 2018**
    Group presentation

From this chart, we can see that I wasn't able to get much traction on my milestones the first month. Then I spent half a month figuring out how to set up and analyze an experiment (for instance, understanding how to analyze data and plot residuals, as well as the point of fitting residuals). Finally, I performed an abbreviated version of the first 6 weeks squeezed into the last six weeks, and turned in the last two milestones a week+ late.

Although this seems like a dire in terms of project management, I did spend a lot of time on items related to research but not listed directly as milestones. I spent a lot of time learning about the grasping field. Additionally designing and prototyping a new sensor was not on the milestones. As a result, by taking a longer-term view, I personally do not feel as disappointed as I might be,

given my personal goals such as making progess toward understanding the research process and learning concrete skills.

Appendix C. Outside Resources

Here is a well-formatted version on Wikipedia describing the 3D stiffness tensor.

With respect to an arbitrary Cartesian coordinate system, the force and displacement vectors can be represented by 3 × 1 matrices of real numbers. Then the tensor $\boldsymbol{\kappa}$ connecting them can be represented by a 3 × 3 matrix $\boldsymbol{\kappa}$ of real coefficients, that, when multiplied by the displacement vector, gives the force vector:

$$\mathbf{F} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} \kappa_{11} & \kappa_{12} & \kappa_{13} \\ \kappa_{21} & \kappa_{22} & \kappa_{23} \\ \kappa_{31} & \kappa_{32} & \kappa_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \boldsymbol{\kappa}\mathbf{X}$$

That is,

$$F_i = \kappa_{i1}X_1 + \kappa_{i2}X_2 + \kappa_{i3}X_3$$

for $i = 1, 2, 3$. Therefore, Hooke's law $\mathbf{F} = \boldsymbol{\kappa}\mathbf{X}$ can be said to hold also when $\mathbf{X}$ and $\mathbf{F}$ are vectors with variable directions, except that the stiffness of the object is a tensor $\boldsymbol{\kappa}$, rather than a single real number k.

Appendix D. Hexo: Online documentation

I documented my work on a blog, http://orangenarwhals.github.io/, using the Hexo flatfile content management system. I chose this because it allowed me to host for free on github, allowed for easy version control of content, and also had an admin plugin that allowed me to drag-and-drop images, which is crucial for documenting experiments where I have a lot of graphs. I was also able to get Latex to work on the site so that I could have equations rendered on there.

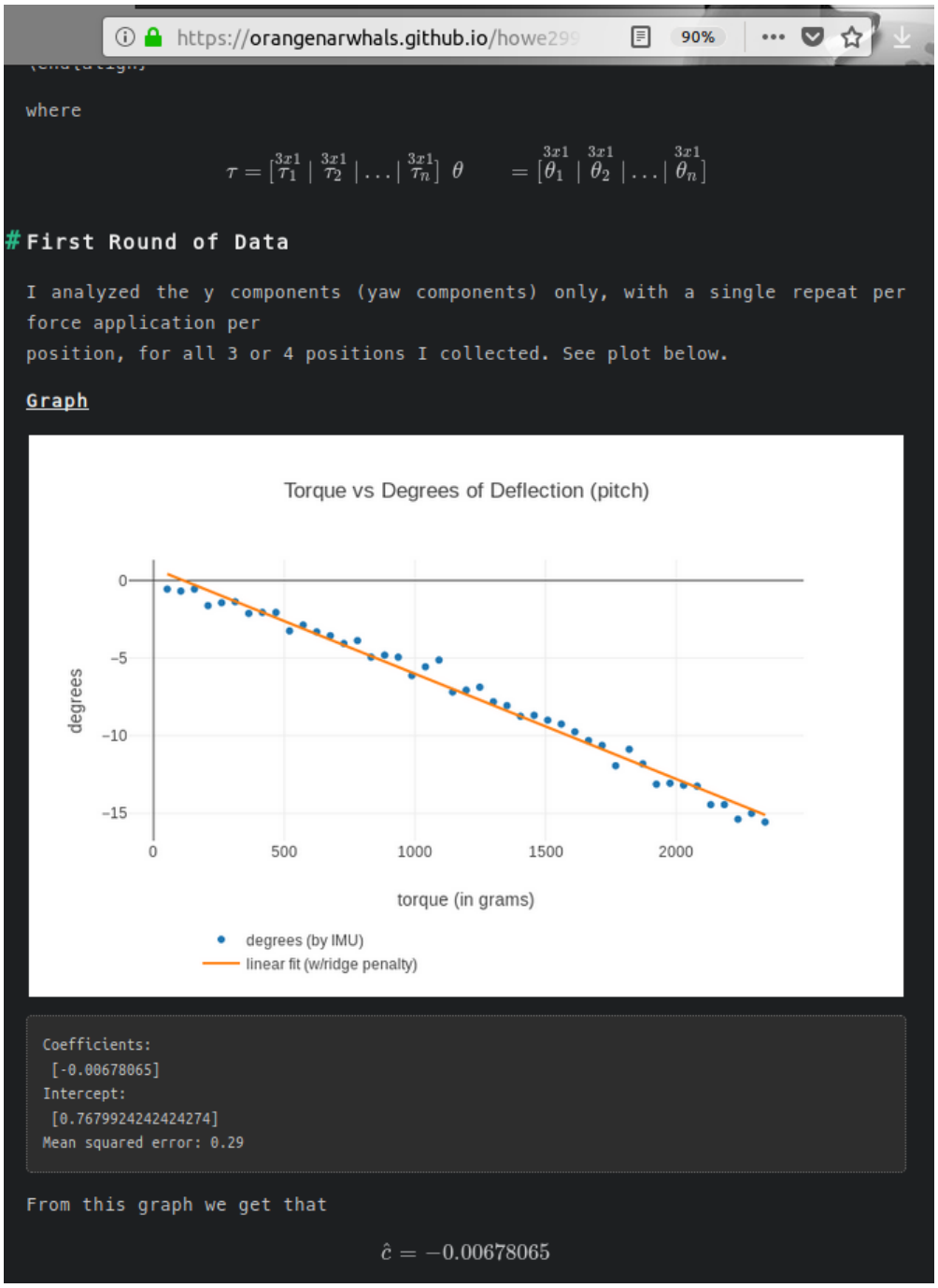


Figure D.23: Getting MathJax (a javascript in-browser version that renders Latex equations in your webpages) to work was a major source of frustration.

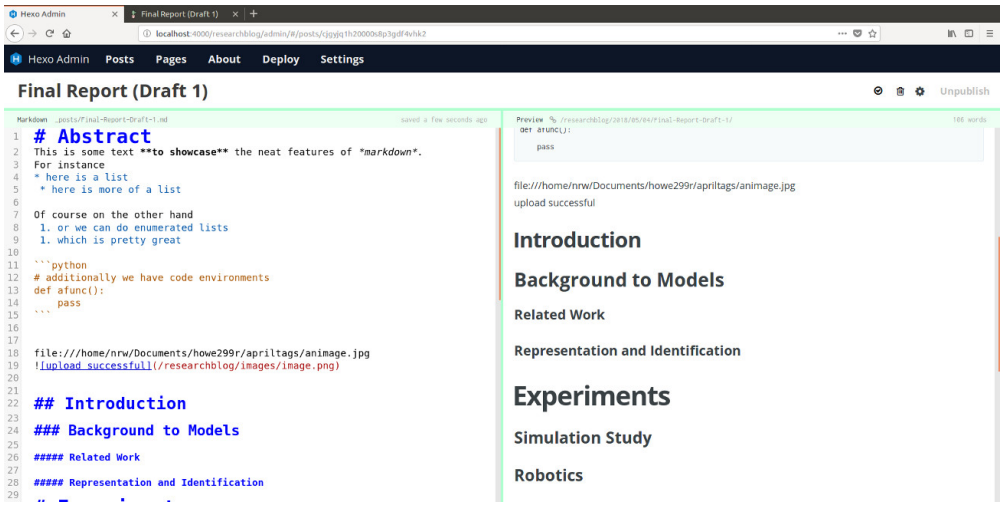


Figure D.24: Hexo uses markdown, which is a nice simple markup language (although gets annoying really fast if you violate the KISS and try to do fancy things like include latex). It has a very nice GUI interface for editing posts.

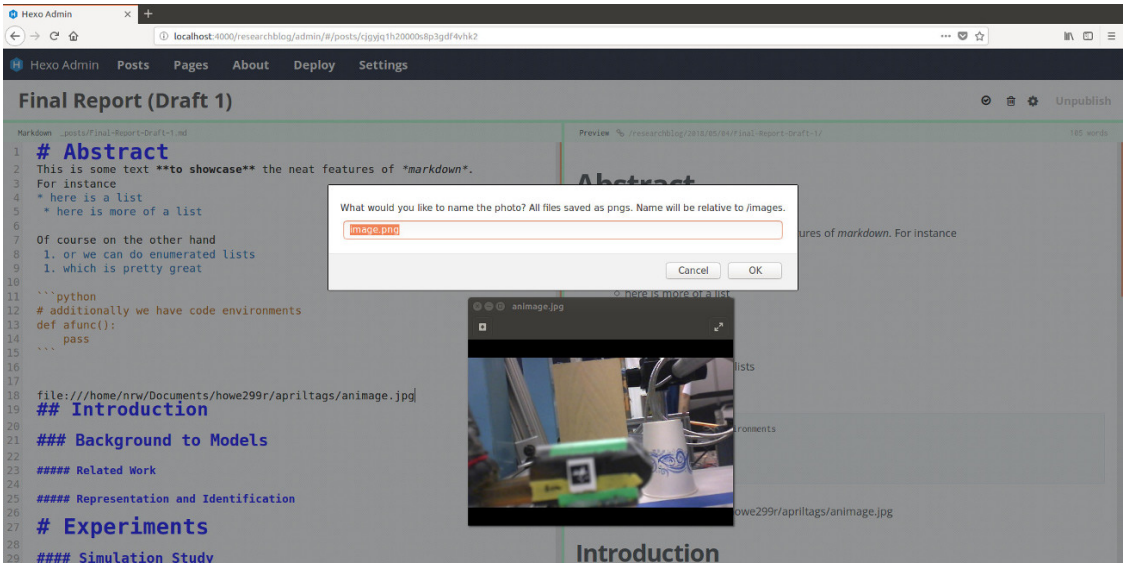


Figure D.25: The particular feature I was looking for was a drag-and-drop image manager. Although I could not get drag-and-drop plugin working, the default Hexo Admin editor allows you to copy paste (ctrl-c ctrl-v) pictures into the editor, and name them appropriately, which I considered good enough (and much better than the normal flat-file editor requirement of manually typing in the file names and locations.

In particular, since I deal with many pictures for a given post, Hexo had a built-in configuration to have a folder per post, which allowed for relatively easy image management.

```
~$ vi _config.yml
    post_asset_folder: true  # REQUIRED even if you make folder by hand!!
```

I will be using Hexo in the future for my own projects (e.g. a portfolio website).

Appendix E. Extra Graphs

Here are a few graphs I didn't include for the sake of space, as they do not add to the conversation.

Figure E.26: A graph of the torque (linear fit) residuals vs. the measured torques. The datapoints are shaded in by their Y position

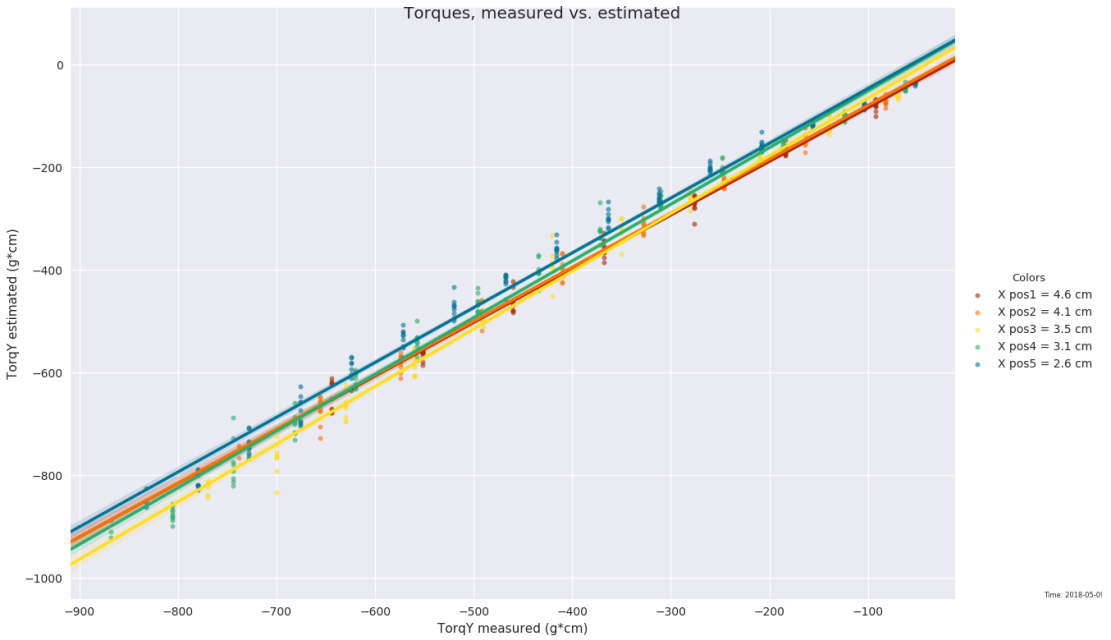


Figure E.28: From the experiment with all 15 positions, we see that the Torque Y fit (measured vs estimated) is very clean.

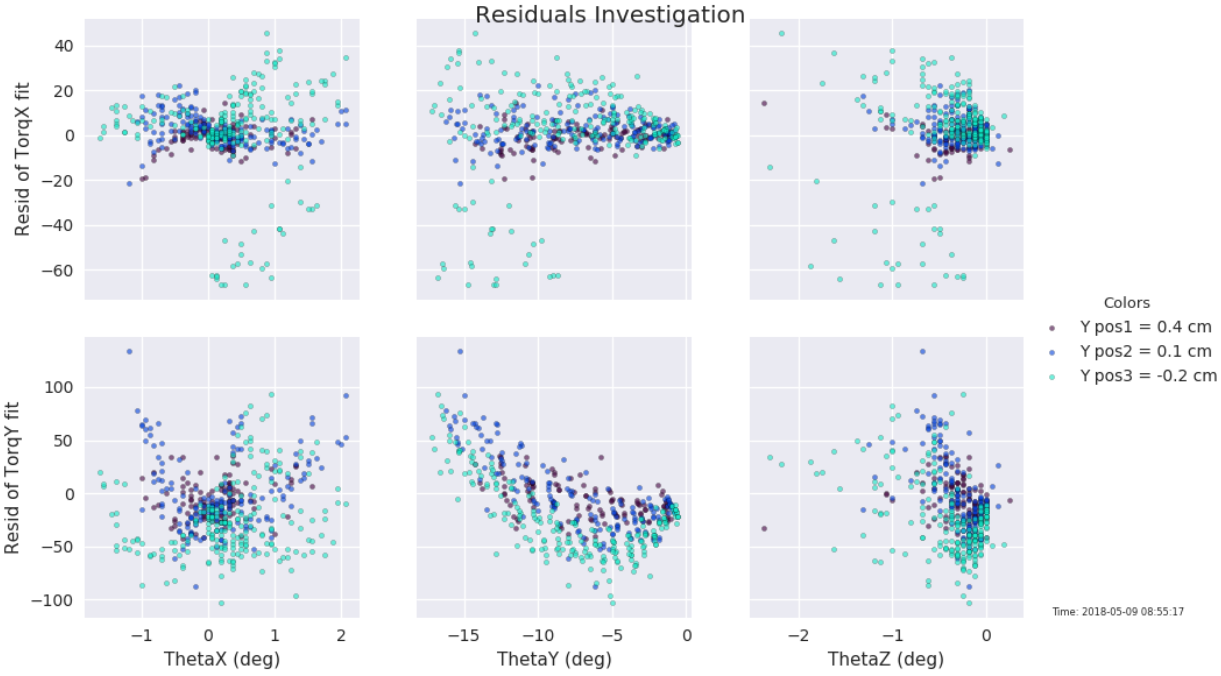


Figure E.27: A graph of the torque (linear fit) residuals vs. the measured angular deflections. The datapoints are shaded in by their Y position

Appendix E. Extra Graphs

Figure E.29: I accidentally tested the generalizability of the residual fit, instead of the overall torque fit after using residual corrections. Thus this graph is relegated to the appendix. In faded blue we have the training dataset, and in green we have the data we removed to be the test set.The faded blue and the green together comprise all 492 datapoints I measured. In the red, we have the predictions made by the linear fit (on the training data, for the test data) and the same in purple for the random forest fit. The residuals seem somewhat more polynomial than linear, so the random forest appears to do a bit better.

Appendix F. Additional Setup Pictures

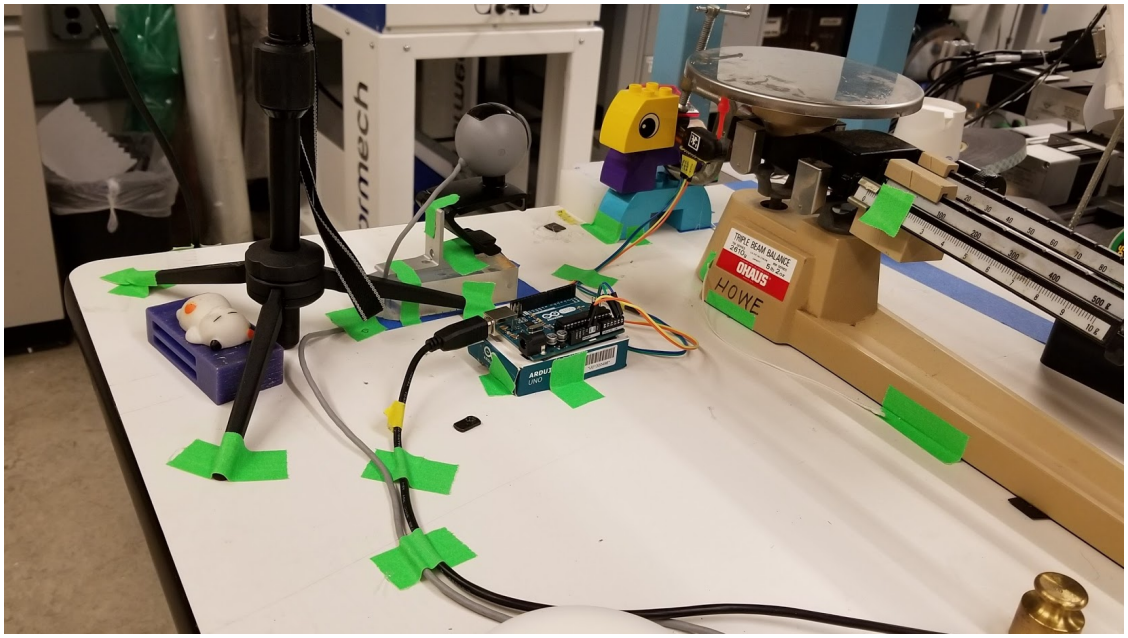


Figure F.30: Gratuitious amounts of tape were used as strain relief, to ensure that randomly bumping into or accidentally pulling on any of the power or sensor lines would not destroy the setup of equipment. Next time, I would just buy some white masking tape right away.

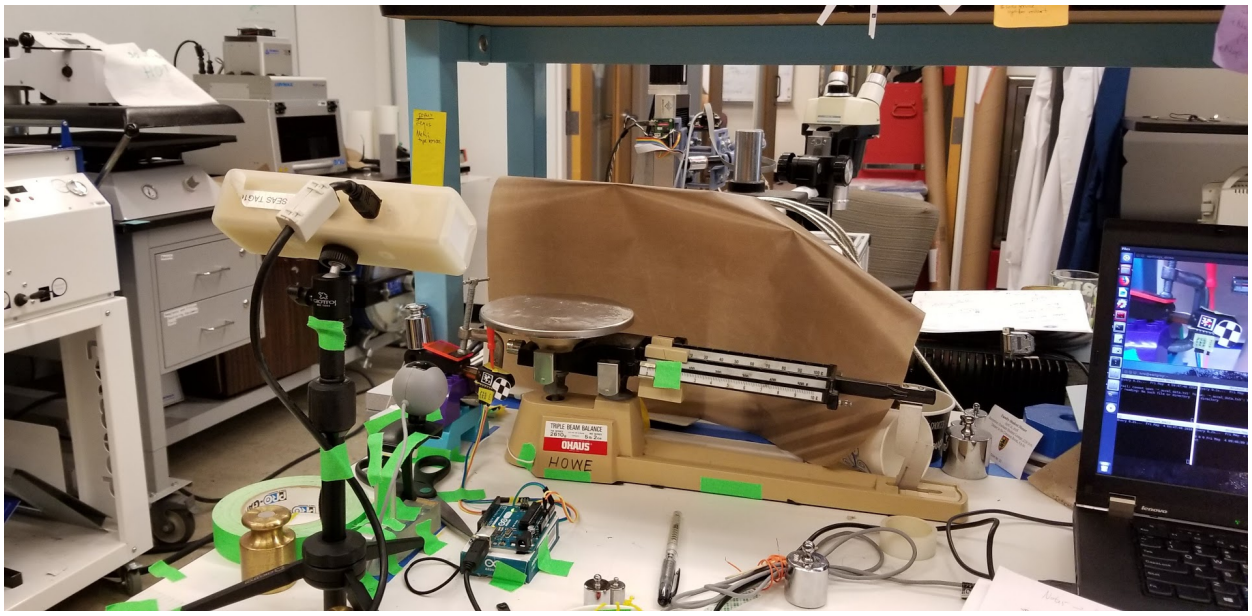


Figure F.31: An overview of the entire setup. This image is not included above as it's extremely messy and confusing, but it shows the relative placement of the microntracker, the webcam, the IMU, the arduino, and the triple beam balance.

Appendix G. Sensor Datasheets

With regards to the BMP280 sensors: I am using the Adafruit libraries with the generic sensors. Both breakouts have schematics available online.

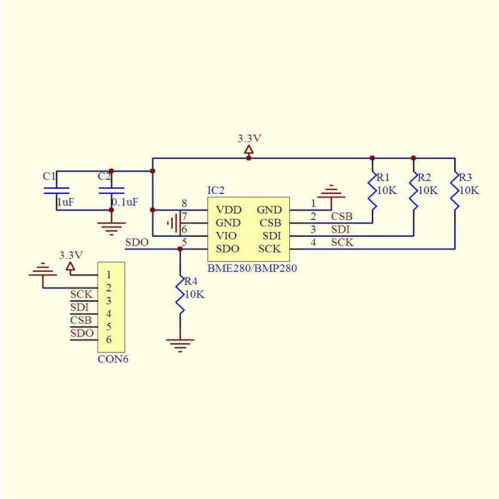
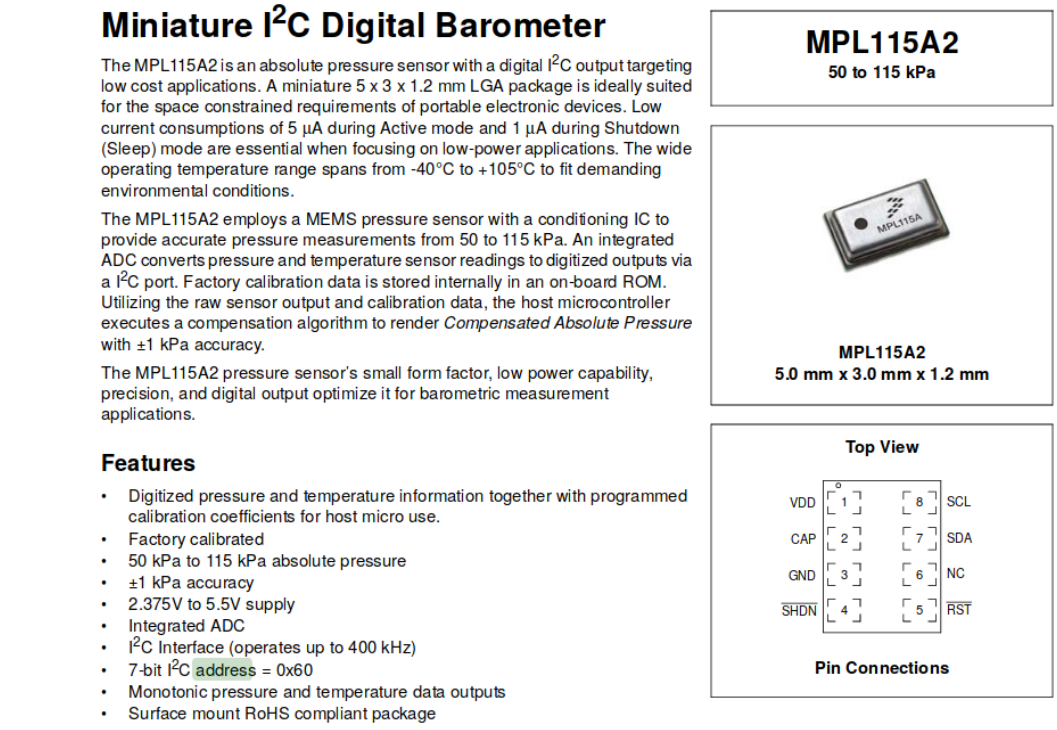


Figure G.33: Schematic for generic breakout. Note that aside form the sensor there are just 2 resistors, 2 capacitors, and some header pins. Source: rainway87 seller on ebay



Miniature I²C Digital Barometer

The MPL115A2 is an absolute pressure sensor with a digital I²C output targeting low cost applications. A miniature 5 x 3 x 1.2 mm LGA package is ideally suited for the space constrained requirements of portable electronic devices. Low current consumptions of 5 µA during Active mode and 1 µA during Shutdown (Sleep) mode are essential when focusing on low-power applications. The wide operating temperature range spans from -40°C to +105°C to fit demanding environmental conditions.

The MPL115A2 employs a MEMS pressure sensor with a conditioning IC to provide accurate pressure measurements from 50 to 115 kPa. An integrated ADC converts pressure and temperature sensor readings to digitized outputs via a I²C port. Factory calibration data is stored internally in an on-board ROM. Utilizing the raw sensor output and calibration data, the host microcontroller executes a compensation algorithm to render *Compensated Absolute Pressure* with ±1 kPa accuracy.

The MPL115A2 pressure sensor's small form factor, low power capability, precision, and digital output optimize it for barometric measurement applications.

Features

- Digitized pressure and temperature information together with programmed calibration coefficients for host micro use.
- Factory calibrated
- 50 kPa to 115 kPa absolute pressure
- ±1 kPa accuracy
- 2.375V to 5.5V supply
- Integrated ADC
- I²C Interface (operates up to 400 kHz)
- 7-bit I²C address = 0x60
- Monotonic pressure and temperature data outputs
- Surface mount RoHS compliant package

MPL115A2
50 to 115 kPa

MPL115A2
5.0 mm x 3.0 mm x 1.2 mm

Top View

Pin Connections

(a) Note the single I2C address given, indicating this sensor has one fixed I2C address. However, it has a "RST" line which can be used to turn off I2C communications for a chip, in effect acting as a chip select.

BMP280

DIGITAL PRESSURE SENSOR

Key parameters

Pressure range	300 ... 1100 hPa (equiv. to +9000... -500 m above/below sea level)
Package	8-pin LGA metal-lid Footprint : 2.0 × 2.5 mm², height: 0.95 mm
Relative accuracy (950 ... 1050hPa @25°C)	±0.12 hPa, equiv. to ±1 m
Absolute accuracy (950 ...1050 hPa, 0 ...+40 °C)	typ. ±1 hPa
Temperature coefficient offset (25 ... 40°C @900hPa)	1.5 Pa/K, equiv. to 12.6 cm/K
Digital interfaces	I²C (up to 3.4 MHz) SPI (3 and 4 wire, up to 10 MHz)
Current consumption	2.7µA @ 1 Hz sampling rate
Temperature range	-40 ... +85 °C
RoHS compliant, halogen-free	
MSL 1	

(b) Details about the BMP280 sensor, which is half the size of the MPL115 A2 sensor.

The 7-bit device address is 111011x. The 6 MSB bits are fixed. The last bit is changeable by SDO value and can be changed during operation. Connecting SDO to GND results in slave address 1110110 (0x76); connection it to V_{DDIO} results in slave address 1110111 (0x77), which is the same as BMP280's I²C address. The SDO pin cannot be left floating; if left floating, the I²C address will be undefined.

The I²C interface uses the following pins:

- SCK: serial clock (SCL)
- SDI: data (SDA)
- SDO: Slave address LSB (GND = '0', V_{DDIO} = '1')

(c) An example of how to find out whether the I2C addresses are fixed for a given chip. Here we see that the BMP280 has 1 bit (2 addresses) to choose from.

Figure G.32: Details about the two sensors: MPL115A2 on the current Takktile fingers in lab, and the new sensor I'm evaluating, the BMP280, which supports both I2C and SPI protocols.

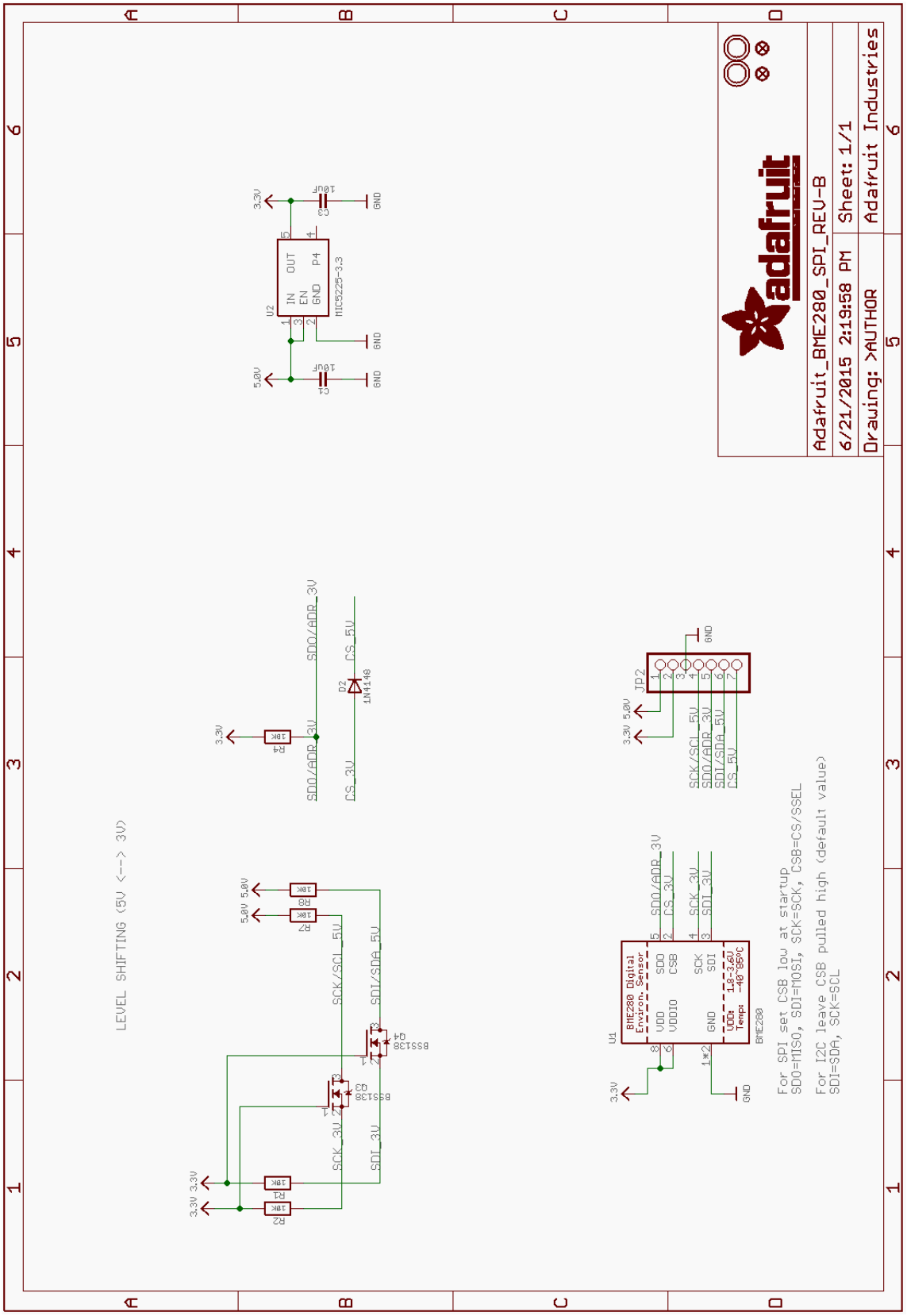


Figure G.34: Schematic for Adafruit breakout. Source: adafruit.com

Thanks for watching. Tune in next semester for another episode of

GRAD SCHOOL:
RESEARCH IN PROGRESS!