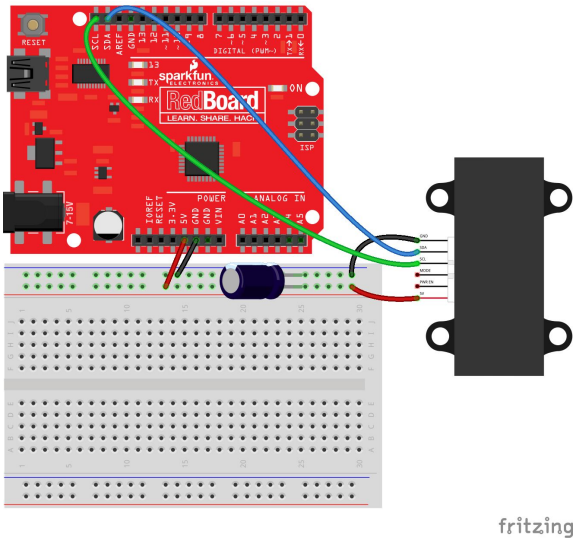
3/16/2018

- Got the LIDAR-Lite v3 sensor! Hooray! (Also got the MPU-6050 (Accelerometer/Gyroscope) breakout board and the MAG-3110 (Magnetometer) breakout board)
- Catherine is letting me use her Arduino until I can get an Arduino of my own
    - I NEED TO ORDER AN ARDUINO (either a nano or an uno; probably an uno tbh but we'll see)
- Verified that the LIDAR-Lite v3 does the following:
    - Does NOT have an attached accelerometer or gyroscope (hence why we needed that accelerometer)
    - Uses an I2C or PWM Interface (we'll be running I2C on an Arduino; it's what the other sensors use, too)
    - Only measures distance measures using a ToF (Time of Flight) measurement: (time between sending signal and receiving) * (speed of light)/2
        - Does nothing else, that's it's data format
    - Comes with a lovely set of rainbow wires that have their own uses:
        - Red = power source (AREF port)
        - Orange = Power enable (Internal pull-up)
        - Yellow = Mode control
        - Green = I2C SCL
        - Blue = I2C SDA
        - Black = Ground (-) (GND)
- Made a "LIDAR-Lite v3 Safety and Warranty Considerations" doc
    - Planning on using this as a non-paper reference for warranty info, and also as a safety reference for eventual "Safety and Booting Protocols" related to the LIDAR sensor
    - Main safety takeaways:
        - The sensor uses a Class 1 laser, meaning that it's safe to look at but we really shouldn't look directly into where the laser is coming from while it's on
        - We need to turn it off when we're not using it, to minimize the risk of hurting our eyes
- Learned what SDA and SCL stands for, and how that relates to I2C format
- Have read to the "Settings" subheading in the "Operational Information" section of the LIDAR-Lite v3 Operator's Manual
    - Need to read further
- Uploaded LIDAR-Lite v3 Operator's Manual PDF into LIDAR folder in Rover Team Drive
- Uploaded I2C Specifications (version 2.1, January 2000, Phillips) into LIDAR folder
- Downloaded Arduino IDE onto Mac running MacOS High Sierra (v10.13.3) using SparkFun tutorial (https://learn.sparkfun.com/tutorials/installing-arduino-ide) and downloaded Arduino IDE from Arduino website (https://www.arduino.cc/en/Main/Software)

- Downloaded FTDI USB Serial drivers (so I can actually run an Arduino on my Mac) from SparkFun (see following link: https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers/mac)
    - Successfully downloaded FTDI drivers!
    - Had an issue where I thought that the FTDI drivers weren't working, but it turned out that the USB switch power cable that I was using to connect my USB power + data cable was just giving my computer a power input, which was not what I wanted if I wanted to code on the Arduino
- Successfully communicated with the Arduino using the "Blink" Example (File > Examples > 1. Basics > Blink)!

3/20/2018:

- Got two 680µF capacitors yesterday from Chloe, so no worries about ordering these capacitors
    - We'll be applying these capacitors to our intro LIDAR sensing circuit
- Set up LIDAR circuit according to the below configuration:
    - IT WORKED, WE GOT DISTANCE MEASUREMENTS!!! Easy peasy



- Started looking into how we're gonna visualize these point data using a program
    - Questions asked:
        - How do we get these measurements in xyz? Do we use the IMU?
        - How do we take these converted xyz data and turn them into a point cloud?
        - How will these data be used to influence the rover's movements?
- Started looking into SLAM (Simultaneous Localization and Mapping) algorithms and learning about that
- Realized that just relying on the altitude and azimuth angles relative to a starting point are not gonna be enough if we want to determine xyz points WHILE WE'RE ALSO MOVING BETWEEN MEASUREMENT LOCATIONS (which is why we're looking into SLAM)

- 3D SLAM algorithms seem to be developing (but compared to 2D versions it seems a little more basic)
- Learned that we're gonna want to hook up are IMU to be moving with the sensor so we know where the sensor is in 3D space when it's scanning, which we can then use to determine where the measured data points are relative to the scanner (we're still looking into SLAM because SLAM allows us to cleanly integrate the scanned map into one cohesive map, instead of individual maps taken at different points that we would then need to blend together, which could end up going horribly wrong especially if we don't know the scanned area)
- Watched the following videos:
    - https://www.youtube.com/watch?v=OJNNm6iMOKk - hour-long talk that gives a really good overview of LIDAR in terms of self-driving robots (which is very necessary for Elif's part of the project)
        - Also provided info on SLAM and how to plot point clouds
    - https://www.youtube.com/watch?v=RHPAJ3kjc3Q - some guy's first version of a 360° scanning mount, which is good because he actually discusses the pros and cons of this mount; however, he doesn't show what data is coming from it, and how he might program that
        - Interestingly he uses a reflectance sensor in order to reset the scanner to a starting position, but it means that it doesn't scan very smoothly :(
        - This version with gears is LOUD
        - Slip rings are IMPORTANT if we want this thing to work at all
    - https://www.youtube.com/watch?v=zizrY-cOS9k - the same guy's second version of his 360° scanning mount, but this time with a weird servo add on so it can go up and down as well as spin in 360° (that being said it didn't get that much movement so who's to say how effective it really was)
        - He switches from gears to a timed belt system, which is a heck of a lot smoother and less noisy (and the same kind as used in this more showy, but less useful video: https://www.youtube.com/watch?v=gCpCGkwwy8I)
        - Servo is very awkward because it's moving the whole platform up and down
        - The whole thing is jumping around a lot because it's resetting to a known location using the reflectance sensor at every rotation, so it stops and starts again

3/21/2018:

- Found an interesting and free point cloud analysis software called SlugView ("Slug" because it's from UC Santa Cruz - go Banana Slugs!) that's listed on OpenTopography, which is a NSF-funded data facility that is available to support PI's and projects that use lidar
    - Link to OpenTopography: http://www.opentopography.org/
    - Link to OpenTopography Tools webpage, which includes SlugView: http://opentopo.sdsc.edu/tools/listTools

        - OpenTopography SlugView description: http://opentopo.sdsc.edu/tools/viewTool?toolId=613
            - SlugView Description: “This is a Point Cloud visualization tool developed in conjunction with the University of Santa Cruz. SlugView started out as a general point cloud viewer. Neva Ridge Technologies added numerous capabilities primarily to support visualization of point clouds with associated numeric attributes. The data can have up to 2 associated attributes which can be used to control which points are actually displayed. Multiple point clouds can be displayed at once and can be displayed sequentially as an animation. Input data formats are primarily text files. Each line in the text file would contain X, Y, Z and optionally red, green, blue, and additional attribute values.”
            - Windows, Mac, and Linux operating systems (needs Java 8, Open GL)
            - Manual is listed here: https://websites.pmc.ucsc.edu/~seisweb/SlugView/manual/manual.html
            - Looks like SlugView can be run in the CLI or in MATLAB, at least according to the manual
    - SlugView download and project page: https://websites.pmc.ucsc.edu/~seisweb/SlugView/Slugview.html
- While scrolling through OpenTopography’s website, I found that they hold workshops (which I can’t attend) on how to use lidar and other 3D mapping technologies. This was important because it introduced me to a new potential fix to our 3D mapping issue if we’re given multiple pieces of data at different positions around something: Structure from Motion (SfM)
    - SfM is “a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals” (https://en.wikipedia.org/wiki/Structure_from_motion)
        - “Local motion signals” could maybe be acquired through an accelerometer or IMU unit located on the rover body
    - Only seems to be a thing for images, so we could potentially apply it to stereo cameras on mast and then the 3D images could be mapped over LIDAR scans, but it doesn’t seem to be relevant to LIDAR point clouds
- Looked into the differences between Stepper and Servo motors:
    - Servo motors would allow us to smoothly move over a specific distance (in degrees if we’re using a rotational actuator), and then to return to a home position, and repeat that same thing
        - Typically these move in 180°-ish degrees, but you can buy special ones for 360° motion

        - Controlled with a specific type of pulse train signal: pulses occur at 20mSec (50 Hz) interval, and vary between 1 and 2 mSec in width. Uses PWM (Pulse Width Modulation) hardware (found on many microcontrollers); @ center of mechanical range when pulse = 1.5 mSec
        - Has low RPM (60 RPM max) so stepping motors are better for discrete movements (ie. alt movement on our LIDAR sensor)
        - Used this video to get an understanding of what a servo motor is (Titled "Servo Basic Concepts" from Yasakawa America's Youtube Channel): https://www.youtube.com/watch?v=t0dSaugIXF8
            - This explanation was albeit for industrial users but its still informative for the basic process, and distinguishes between linear and rotational actuators
        - This video ("Electronic Basics #25: Servos and How to Use Them" *GreatScott!*) is a bit more accessible for someone who's only planning on using tiny, cheap servos for my project: https://www.youtube.com/watch?v=J8atdmEqZsc
        - Needs a driver to run
        - Have a three-pin plug: ground, voltage, and the command line (pin colors are not consistent across servos, so need to check documentation)
    - Stepper motors can easily move in 360° and can easily move at specific degree increments
        - Looked at this video ("Electronic Basics #24: Stepper Motors and How to Use Them" *GreatScott!*) to get a sense for stepper motors and how to use them: https://www.youtube.com/watch?v=bkqoKWP4Oy4
        - Can be combined with a timing belt to get really smooth motion
        - Needs a driver to run, which can be purchased on SparkFun
        - While you can buy steppers that have 200 or 400 steps per rotation, you can increase the number of steps by half-stepping or micro-stepping which gives you less torque as you increase the number of steps
- In my endless search to figure out my xyz points given just a distance measurement, I decided to look into how I might get these data
    - Originally I'm thinking that I should use an IMU sensor that's actually moving with the moving sensor so that we get acceleration and gyroscope data

4/6/2018:

- Ok so since writing in this last I have:
    - Received an Arduino Nano, but upon closer inspection it might be a clone
    - Written a detailed update to the LIDAR section of my proposal
        - Compiled scan time estimates assuming the sweep of phi = 45 degrees
        - Figured out calculating relative distance using a reference point
        - Figured out how to calculate xyz points given theta, phi, and distance data
        - All this is effectively written in the proposal
- Still having issues getting Arduino Nano to upload a code

    - Downloaded another FTIR driver for Mac (v. 2.4.2) - didn't work
    - Consulted forums - didn't give me progress
- In exasperation I tried running the Blink example (Arduino > File > Examples > 1. Basics > Blink) on the original Uno that Catherine gave me, and I got the following error:
    - /Users/jocelynreahl/Library/Arduino15/packages/arduino/tools/avr-gcc/4.9.2-atmel3.5.4-arduino2/bin/avr-g++: no such file or directory
- Decided to uninstall Arduino software in an attempt to fix it because I couldn't find the Arduino15 file in the Library on my computer (Macbook Pro running OS X High Sierra)
    - Wes and I tried doing a bunch of stuff on the terminal and it… didn't work. None of it worked
- Started to give up and log into a computer running Windows 7 to try downloading Arduino IDE on it, but realized that when I originally uninstalled the Arduino IDE, I forgot to delete the hidden Arduino15 file in Library
    - Had Wes delete it before I redownloaded the new IDE, and IT WORKED HOLY CRAP
        - By "it worked" I mean that the UNO is working on it again; the nano (which I'm certain is a clone now) still isn't working
    - Tried downloading Arduino IDE on classroom computer, and it wouldn't let me unless I was an admin, so in theory I could log in under Wes's profile, but that starts getting into some weird territory that requires a lot of effort on both ends
    - Ended up just having Jenny get an original Nano from the Arduino website, because nothing seems to be working and the following forum post implies that even if I did install the correct drivers for the clone, it still wouldn't work on Mac OS Sierra and above unless I did some crazy troubleshooting that is super time consuming - https://arduino.stackexchange.com/questions/34775/arduino-nano-not-visible-in-serial-ports-mac-os
    - Got Jenny to get the Nano from this website: https://store.arduino.cc/usa/arduino-nano

4/10/2018:

Tasks for today (if servo motors AND Nano haven't arrived yet)

1. Get MPU 6050 and MAG 3110 sensors wired up and running
    a. Includes downloading necessary Arduino libraries and reviewing datasheets for each sensor -- Almost done for MPU 6050
    b. Also obviously includes getting them wired to a breadboard -- Done for MPU 6050
2. Learn to solder @ WeLab at 11:00am -- Done

Tasks for today (if Nano has arrived but not servo)

1. Same as above
2. Get LIDAR hooked up on Nano

- Pulled up MPU 6050 datasheet (http://43zrtwysvxb2gf29r5o0athu.wpengine.netdna-cdn.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf) to start learning about MPU specs:
    - Called a 6-axis motion tracking device (ie. 3 axis accelerometer + 3 axis gyroscope + Digital Motion Processor™ (DMP))
        - Note that gyroscope is tracking yaw, pitch, and roll (ie. rotational motion), and accelerometer is tracking translational motion in x, y, and z
    - Dedicated I^2C bus ⇒ working with Arduino Uno/Nano
    - Says that w/a 3 axis "compass" (which I'm reading as a magnetometer for now) you can get a 9-axis MotionFusion™ output (this is thanks to the DMP)
    - Designed to interface w/non-inertial sensors like pressure sensors (and maybe LIDAR sensors, ooooOOOOooo)
    - Has 3 16-bit analog-to-digital converters (ADC's) for digitizing gyroscope outputs and another 3 for digitizing the accelerometer outputs
    - Users can program the gyroscope and accelerometer to track fast and slow motion:
        - Gyroscope: ±250, ±500, ±1000, and ±2000°/sec
        - Accelerometer: ±2g, ±4g, ±6g, ±8g and ±16 g
    - Low power thanks to 1024 byte FIFO buffer ⇒ allows the system processor to read the MPU data in bursts instead of constantly
    - Gyroscope specs (besides user-programmable stuff):
        - Gyroscope operating current: 3.6mA
        - Standby current: 5μA
        - User self-test
    - Accelerometer specs (besides user-programmable stuff):
        - Accelerometer normal operating current: 500μA
        - Low power accelerometer mode current: 10μA at 1.25Hz, 20μA at 5Hz, 60μA at 20Hz, 110μA at 40Hz
        - Orientation detection and signaling
        - User-programmable interrupts
        - High-G interrupt
        - User self-test
    - General MPU 6050 specs:
        - Auxiliary master I2C bus for reading data from external sensors (e.g., magnetometer)
        - 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
        - VDD supply voltage range of 2.375V-3.46V
    - So apparently there's a temperature sensor that's just randomly on here????
- So SparkFun doesn't have a ton of stuff besides GitHub links, so I'm gonna check the Arduino Playground page on the MPU 6050: https://playground.arduino.cc/Main/MPU-6050#intro

    - Reading raw accelerometer and gyroscope values is easy, but getting a magnetometer on there is a little trickier
        - The MPU can act as a slave to the Arduino if it's connected to the SDA and SCL pins on the Arduino, but the MPU also has its own auxiliary I^2C pins (called AUX_DA and AUX_CL) for the second (aka sub) I^2C bus
- The following link provided great info on how to set up MPU circuit and how to get the necessary libraries for them: https://maker.pro/education/imu-interfacing-tutorial-get-started-with-arduino-and-the-mpu-6050-sensor
- Set up circuit, will get code hopefully understood next time (and hopefully servo motors will come by then too

4/12/2018:
Task:

1. Getting the MPU 6050 running w/Arduino example code
2. Getting MAG 3110 library downloaded
3. Maybe getting MAG 3110 running w/Arduino example code if possible

- So I triple checked the circuit for the MPU, but when I tried running the example program it kept telling me that the MPU wasn't connected, so I'll be following up this later

4/13/2018:
Before class:

- Was looking into installing MAG 3110 library and I realized that I've been installing libraries wrong so I moved all the libraries to the proper location (from "Documents > Arduino > libraries" to "Applications > Arduino > Contents > Java > libraries")

Tasks are the same as 4/12/2018

- Downloaded MAG 3110 library
- Verified that I have a Servo library and looked at examples
- Spent a LONG time troubleshooting the MPU again:
    - Tried reconnecting circuit in 10,000,000 different ways; still getting the error "MPU 6050 not connected" on Jeff Rowberg's program
    - Soldered headers on MPU 6050 and MAG 3110 in the hope that doing that would fix my problem (one of many forum discussions on Arduino seemed to have soldering do the trick) - didn't work but hey at least I have headers on my sensors now
    - Lots of forums seem to use an Arduino program called I2C scanner that checks for I2C things that are connected, and Meba gave me the link to it - used the I2C scanner (link to code: https://playground.arduino.cc/Main/I2cScanner?action=sourceblock&num=1) and the MPU isn't showing up on the scanner, but the MAG is, meaning that it's either a board or a port issue (forums that mention this: http://forum.arduino.cc/index.php?topic=142490.0; https://forum.arduino.cc/index.php?topic=255033.0)

    - Also consulted I2C dev forum on this (written by Jeff Rowberg, who seems to be the guru on the MPU 6050) and it didn't really have answers for me besides the ones I had originally tried: https://www.i2cdevlib.com/devices/mpu6050#help
    - I might have found some code that's helpful because it actually defines the I2C addresses of each MPU component!
        - Original address: http://www.14core.com/wiring-the-mpu-6050-sensor-accelerometer-gyro-with-nano/
        - Code in this page seems to have been originally written by arduino.cc, but when I googled the name of the code it isn't showing up with Arduino so who knows what this code's actually gonna do
            - I have a tad bit of hope though because this code has a lot of documentation, defines literally everything, and was originally written by arduino.cc; and also given all of the other written things about this sensor and when they were written (2012-2014ish), the last updates on the code were done in 2013 so it would make sense for it to exist at this point in time, too
- Got the MAG 3110 up and running an example bit of code - Wes wants me to use the xyz measurements of the magnetic field to inform rover placement, but I don't know how feasible that is and I'm still going to work on the MPU
- Servos arrived! Didn't get a chance to work with those at all today but I will definitely be doing that this weekend.

4/14/2018:
Tasks:

1. Make a prototype of LIDAR scanner
2. Troubleshoot MPU 6050

- Prioritized Task 1 for today and was working with Meba
- Meba and I downloaded a plan from GrabCAD that's for a 48 tooth spur gear
- We then edited that plan on Meba's computer to be laser-cuttable (ie. 2D)
- We then cut out two of these spur gears on ¼" plastic (I forget the exact name of it at this time) - the gears work really well together!
    - This was a challenge initially because the computer that we used to do the laser cutting kept saying that it couldn't cut because there was no COM port to be found; it eventually solved it on its own (I think we just had to be more patient with it)
- Played around with the continuous rotation servo that we bought, and we ran into a problem: the servo doesn't seem to move very accurately, it only seems to spin
    - After a lot of troubleshooting we realized that continuous rotation servos are basically glorified DC motors with no position feedback - might need to purchase a different motor, like maybe a stepper - but for prototyping this isn't critical just yet

- Managed to build a prototype for the mount that allows the LIDAR scanner to move in azimuth. Altitude will be managed this weekend!

4/15/2018:
Tasks:

1. Investigate stepper motors and potential other solutions to continuous rotation and feedback

- Learned today that not only do continuous rotation servos not provide feedback, but neither do steppers
- So this made me really worried about getting the 360 component to work without calculating the angle theta swept per increment of time, but I learned that there's a way to get around this problem: an encoder
    - Encoders hook up to the back of a motor and basically keep track of where it is
- Learned through further research that a company called Parallax made something called a Feedback 360° servo motor, which is basically a continuous rotation servo motor but with the feedback component that we all know and love
    - Link to product page: https://www.parallax.com/product/900-00360
    - There are cheaper versions on Amazon for less
- Finding the feedback 360° was cool, but there isn't a ton of arduino documentation on it
    - Found some code by JavelinPoint on an Arduino Forum that apparently gets the job done for both reading position and continuously rotating: https://forum.arduino.cc/index.php?topic=516986.0
- Current plan is to run all this by Wes and see what he thinks - I think he might approve because stepper motors and the continuous rotation motor clearly don't work without an encoder which is a pain in the ass to get
- So all in all, I would like to purchase the Parallax Feedback 360° servo motor ASAP, and will be emailing Jenny about it soon: https://www.amazon.com/Parallax-Feedback-360%C2%B0-Speed-Servo/dp/B074XKBWJ5/ref=sr_1_2?ie=UTF8&qid=1523845543&sr=8-2&keywords=feedback+360+servo
- Also downloaded Fritzing to begin making my circuit diagrams, but it's definitely gonna be a learning curve to use this so we'll see

4/19/2018:

- Began working on prototype code that can have LIDAR running w/servo motors and little breadboard
    - Was worried today that maybe we didn't need the little breadboard, but I just remembered that we also need to connect a 680 µF capacitor to this circuit for the LIDAR so honestly thank god
        - That reminds me, I should double check to see if the servos are gonna be affected if this is in the circuit… probably not, right? - the LIDAR GetDistanceI2C example seems to indicate that it's just there to mitigate "inrush current" when the sensor is on, whatever the heck that means

    - Gonna try getting this code to be pretty comprehensive, so that it's almost like an example piece of code for anyone working with this in the future
    - Planning on including Fritzing diagram in this code, both for future readers' sake and also my own

4/26/2018:

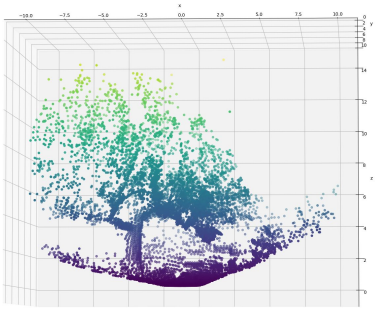
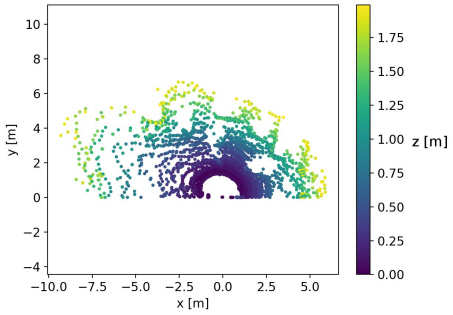
- Ok so since last writing in here I've:
    - Started creating parts in SolidWorks that will eventually be 3D Printed/Laser Cut and then assembled (this has been the bulk of my work)
    - Started working on Fritzing diagram for the circuit I'm making
    - Received the Arduino Nano in the mail! Finally! And I also received the Parallax 360° servo motor, but I haven't gotten a chance to try coding with it (I am PRAYING that it works and that I don't have to make a last-minute stepper motor order, because that would be the literal worst)
    - Seriously started working on my Arduino code for the instrument, and now I'm considering how it'll communicate with the Raspi (and I'm also assuming it's getting its power from it as well, and it's connected to a power source)
    - Haven't gotten a chance to more troubleshooting on the MPU, which is… concerning

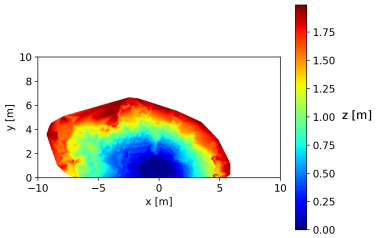
5/2/2018:

- I LASER CUT AND CONSTRUCTED THE HOUSING! I still need to get the 3D printable stuff 3D printed in Clapp, and given all of my other work it's looking like I would need to do that tomorrow (Thursday)
- I've backed up all of my SolidWorks files on a USB as a "just in case" and also as a necessity since my computer doesn't run solidworks to begin with
- Had a team meeting yesterday with Wes and Jenny, and the major issues with LIDAR that we still have are:
    - We still need to construct the housing and 3D printed thing, and get it spinning
        - To "get it spinning" we need to make a decision about the Parallax 360 Servo and whether or not we can get it to work. Jenny has been looking into it but it doesn't seem like it would get done within our block of time, so the new plan for now is to just use the continuous rotation servo and estimate the approximate theta's covered
    - We need to get some kind of accelerometer hooked up, and if it's not the MPU 6050 then it'll be an ADXL 343/345 that we hopefully have lying around.

http://www.charleslabs.fr/en/project-3D+Lidar+Scanner
https://processing.org/download/
https://arduino.stackexchange.com/questions/1758/whats-the-most-accurate-way-to-time-stamp-analogread-data?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

5/10/2018:

- I FINALLY GOT IT WORKING AND EVERYTHING
- I ditched using the continuous rotation servo in favor of a regular servo so I could more effectively use Charles Grassin's code
- Implemented this both in the WeLab (which worked) and then implemented it outside in the outdoor classroom
- I copied the data (Using command + A and then command + C) from the serial monitor into a text file and then sent these along to Wes to plot in python
- Wes plotted these and the data are BEAUTIFUL!
-  PNG of outdoor classroom scan
- 

PNG of top-down view, viewed below 2 m (to avoid tree branches
- 

PNG of DEM constructed from the above top down view