Introduction

This goal of this project is to create a relatively cheap and lightweight 3D depth scanning system that can obtain a reasonable amount of data (~ 2 million points) in a short period of time (~ 5 minutes) for the mobility unit for NASA Big Idea Challenge. The write up goes into detail of the mechanical design, electrical components, and software necessary to be able to achieve such goal. Towards the end we will be presenting some scanning results, plus notes on improvements for future iterations.

Mechanical Design

The mechanical design can be split into two sections, the lower and upper rotation stage. The lower rotation stage consists of the microcontroller, quadrature encoder, and the DC motor with its transmission for lidar's continuous rotation around the vertical axis. The upper rotation stage mainly consists of the lidar and stepper motor, the latter of which is responsible for rotating the lidar against the horizontal axis.

Figure 1,2. Solidwork model in assembled/exploded view

Lower/horizontal rotation stage

The DC motor we chose for the lower rotation stage is RP360-ST, which should provide around 2300 RPM and around 40g/cm stalling torque at 5V. As the DC motor we are using spins at a much higher speed than my original RPM target at 5V, while also providing enough torque, the transmission

system has to be stepped down in order to reach the desired output characteristics. Due to the relatively low rotational speed of around 120 RPM, we were able to get away with using a geared system. The spear gear reduction ratio is 164 : 9 from DC motor transmission to the horizontal rotation stage (*or a reduction of around 18:1, first stage is 41 : 12, the second stage is 52 :13, and the last stage is 28 : 2*1). Another reason for a geared system is the longevity of the transmission, as the timing belt generally will wear down much quicker over time. Using belted transmission would also add complexity to the assembly. The downside of using a geared system would be the increased noise, but the issue can be alleviated by increasing the manufacturing precision, adding gear lubrication also helps in reducing the noise. No backlash was incorporated into the design to maintain measurement accuracy. This is achieved through the slight flexibility of the PLA material we've used for manufacturing.

Figure 3. Cross section view of the gearbox, top left(light gold colored) is the encoder gear (1:1), the left is the motor transmission (164:9)

On a side note, the current design is limited to 120 RPM due to the center of gravity not being perfectly aligned to the center of rotation, causing excessive wobble at high rotational speed. The encoder[1] used in this project is also not suited for high speed applications due to utilizing copper bushing (*rated at 100 RPM*) insteading of ball bearings for its rotating axle.

Upper/Vertical rotation stage

Since the upper stage where the lidar, stepper motor, and other electronic components reside has to make an electrical connection with the microcontroller on the lower stage, which stays stationary

[1] https://www.usdigital.com/products/encoders/incremental/shaft/S5

as the upper stage rotates continuously, a slip-ring has to be incorporated to be able to achieve such electrical connection without tangling the wires. The slip-ring that is incorporated into the design is rated at a maximum of 240 RPM, which is within the spec of upper stage's rotational speed during operation, and provides 12 leads from the two stages, providing ample connections for our use.

The vertical rotation stage uses a 28byj-48 which is incredibly common and cheap (~$5). It provides around ~300g/cm of torque at 5V in unipolar operation. The stepper motor is modified into operating bipolar mode which more than doubles the torque provided at around 700g/cm, more than enough to rotate the lidar unit (*weight in at around 35g according to spec sheet, the center of mass of the lidar attachment is around 3cm away from the point of rotations, giving us around 105g/cm torque required to drive the module*). The power consumption also dropped considerably using the bipolar configuration, which we will be mentioning hte the electrical section. Another reason this stepper motor was chosen was due to its high resolution at almost 2048 full steps per rotation (*the gearing ratio isn't a full 64:1 upon further inspection of the gearing ratios*).

To be able to home the stepper motor position, as the stepper motor has no absolute positional feedback signal, the design also incorporates a small optical interrupter module consisting on the right side of the upper rotation stage, consists of an IR led and IR photodiode/transistor (*not sure which one to call it, since it acts like a transistor but only has two leads*) . In order to prevent IR light being reflected by housing, black filaments were used to produce the housing, preliminary testing shows that the same housing printed with white filament would reduce the sensitivity by a significant margin.

Manufacturing & Result

Due to using 3D printing for manufacturing all components, a 0.1mm offset to the dimensions has to be considered during design. All components are printed with PLA with 20% infill to keep the weight down while maintaining structural strength. Due to our use cases the PLA should be fine for all conditions as no components within the unit would reach above a temperature of 60 degree Celsius. PLA is also the best material for the gear system due to its relatively high rigidity compared to ABS and PETG. Nylon is the only option that would provide better results. But again, due to the availability and manufacturing complexity, PLA is still the best choice for this design.

The final weight of the entire unit is around 387 grams, which should be light enough to be mounted on a conventional rover unit. Current design decisions such as choosing the 28byj-48 stepper motor, and using the S5 encoder from US digital, all helped in reducing the dimension of the unit. Few things that can be improved upon to further reduce the weight of the unit: Design a fully integrated optical quadrature encoder system would reduce the need for encoder housing. Shorter and smaller DC motors for driving the encoder stage should also help in size reduction.

Figure 4. Final mechanical assembly

Electrical System

The electrical system is very straight forward, as all components are operating under regulated power sources (*+5V USB source*) and consumes within the current limit of USB3 standard at 900 mA. The wiring diagram shown below provides a general idea of how everything is connected. Due to the experimental nature of the project, all components involved in the electrical system are off the shelf and thus no PCB was design (*although ideally for the sake of signal integrity and ease of assembly, one should be design if project is take into further consideration*)

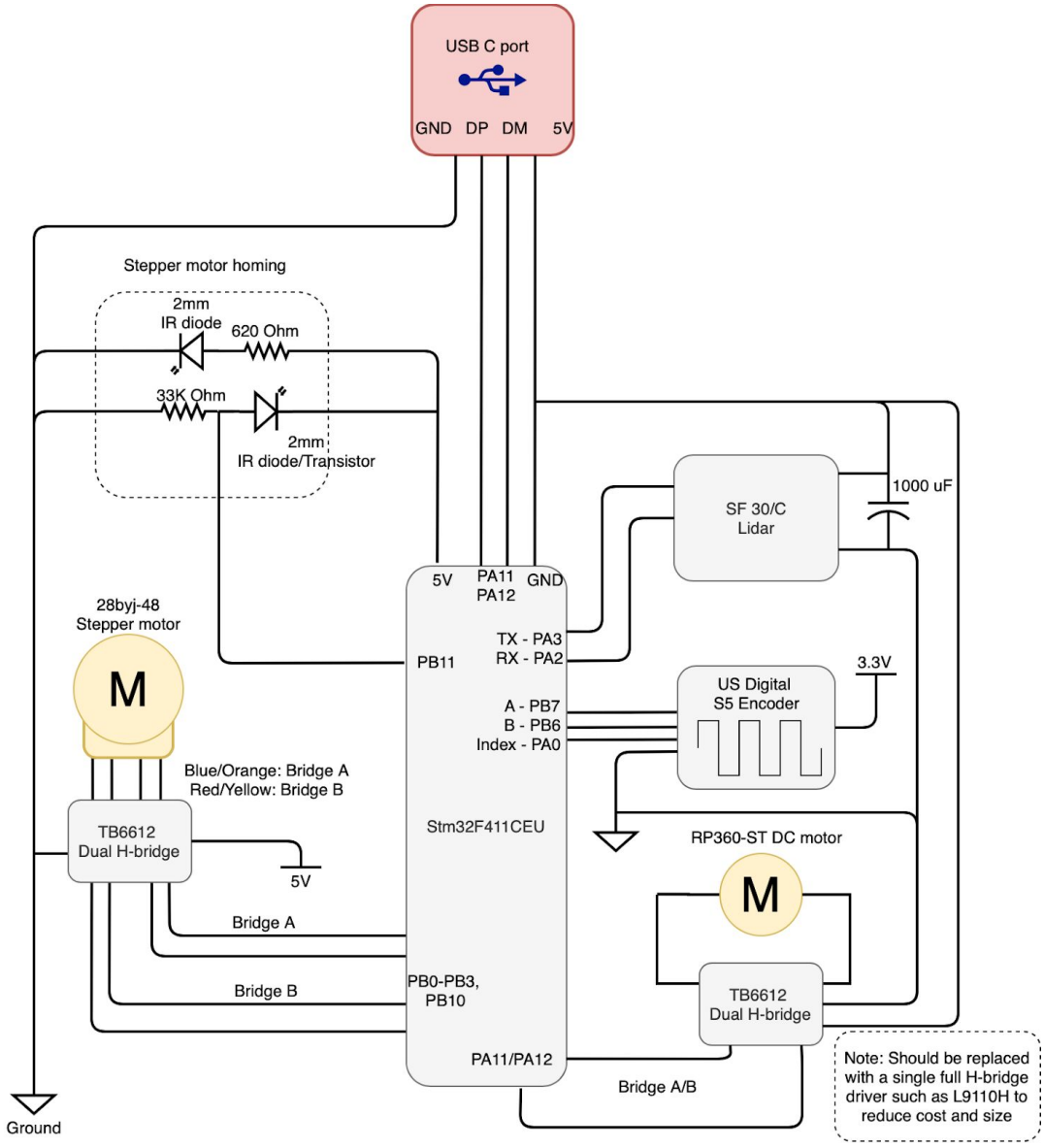

Figure 5. Electrical wiring diagram

For power, a 1000uf capacitor is applied across the 5v power rail for increased stability for both the lidar module and the microcontroller, as serial communications (*USART, USB*), are greatly affected by this. 1000uF should be sufficient to filter out most noise. Should the DC motor require higher rotational speed, a 12V voltage regulator can be applied across the 5V rail. We measured around 2.2 watts of power draw during the scanning operation.

Vertical rotation stage with originally decided to use an as5600 absolute magnetic encoder for closed-loop angular data acquisition, however, due to the slow transfer nature of I2C (*100kb/s, up to 400kb/s*) which would impede the USART transfer from the lidar unit. In this case, a homing device is proposed in place of semi-closed-loop feedback; as the stepper motor can provide enough holding torque and the fact that during scanning the lidar unit would be in a stationary position, losing step would be a non-issue. Therefore using the homing device the lidar would calibrate itself against the absolute position before each scan to maintain the vertical angular accuracy. However, in a future revision, a small compact quadrature encoder would be ideal as maintaining encoder count does not hold up the microcontroller process, and a closed-looped system would always be preferred for high accuracy and repeatability. On a side note, hobby-grade servos were considered, however, due to lack of repeatability from the initial test (*servo position can not be repeated with the same PWM duty cycle*), this idea was scrapped. High-end servo such as Dynamixel AX-12 was considered, but due to its weight, and the requirement of using serial communication (*which would prevent getting data from the lidar unit properly*)

The 28byj-48 stepper motor is modified in bipolar mode and thus requires two H-bridges to drive it properly. The TB6612 was selected due to its compact size. Current (*TB6612 is rated at 1.2A per channel, or 2.4A to drive a bipolar stepper motor*) wasn't a concern as the stepper motor was selected due to its low power consumption. The stepper motor's current draw is measured at around 300 mA in unipolar mode and 100mA in bipolar mode (*for some very odd reason, most likely the coils had to provide more magnetic force to be driven by the stator*). The stepper motor also runs a lot cooler in bipolar mode, thanks to the significantly lower current draw.

The photo interrupter module is fairly simple consisting of an IR led and IR photodiode. The line out between the 33 kOhm resistor and the photodiode provides an on and off signal whenever the photodiode received brightness above a certain threshold. During On, the line should provide a 5V signal to the microcontroller as an external interrupt signal; in the case of STM32F411, the pin to which the line is connected to is 5V tolerant, thus depending on the microcontroller one is using, one should step down the voltage of this signal line. The voltage across the IR led is maintained at 1.1V with a 630 Ohm resistor in series. Both are hooked up to a 5V rail. During operation, both draw around 13 mA.

Software

The firmware for the microcontroller uses HAL library, and for simplicity of hand off, is moved to STM32CubeIDE for the IDE environment. The current 10k reading is already limited by the baud rate and the operations between reading the lidar data from the serial port: such as send data out to USB virtual com port, and the worst possible offender, using sine and cosine operation on floating points. This was an even greater issue before moving from STM32F103 to STM32F411, which does not have a floating-point unit built-in, requiring at least 10 times the number of instruction cycles to perform floating-point related tasks. Thus the main overhead would mostly come from the simple floating point operations from the microcontroller (*which needs more analysis by determining the instruction cycles in between receiving data points*).

USART

Setting the baud rate to the maximum 1440000 bauds seem to have a slight benefit of improving reading speed and error at over 10kp/s reading. Setting an oversampling of the usart port to 16 (*OVERSAMPLING_16 in HAL library*) also helps improve the performance by drastically reducing the errors being read from the lidar module through the usart port compared to oversampling of 8 bits; this is due to the lack of impedance matching and the generally noisy dataline (*originally shared with the DC motor*) due to using slip-ring.

Other methods to further improve the usart performance include replacing the current onboard crystal of the microcontroller with one that has better stability, as asynchronous communication relies heavily on clock stability (*the greater the variance of clock pulses every cycle, the less in sync between the two devices, and thus more error bits would be generated, meaning more bits has to be resend which would lead to increased transfer time*). The built in crystal oscillator of the microcontroller is disabled in favor of the external crystal (*rated at 25Mhz*), as during the preliminary test the built-in crystal provide a performance of around 18ppk, which is decent, while the external crystal oscillator provide stability value of around 25 ppm, which should provide a much more error free system *(this is crucial for serial communication and onboard timer delay*).The best crystal one can buy (*in this case, Digikey, Mouser, and Arrow*) would provide a performance of around 9 ppm (*variance pulse every million oscillation*), which should provide a much better USART performance than the current 25 ppm module. On the other hand, a custom designed controller board that would allow the incorporation of crystal oscillator, instead of a crystal, would further increase the clock accuracy by another order of magnitude, would provide even better communication stability (*usually at around 1ppm, at 500ppb the cost of crystal is still decent at around $2 a piece*).

Communication Method

To effectively reduce the transfer overhead from the microcontroller to the host system, three methods are used: 1. By rounding all xyz values to the nearest centimeter, as the lidar unit itself was only able to provide centimeter resolution, thus no resulting resolution would be lost. 2. By packing the xyz value into a more compact string format, this allows a reduction of total bytes sent during each measurement cycle from 18bytes down to 12 bytes.

To further elaborate on how the data are packet, the figure below provides an example of how the X coordinate value is packed into the 4 byte data packet.

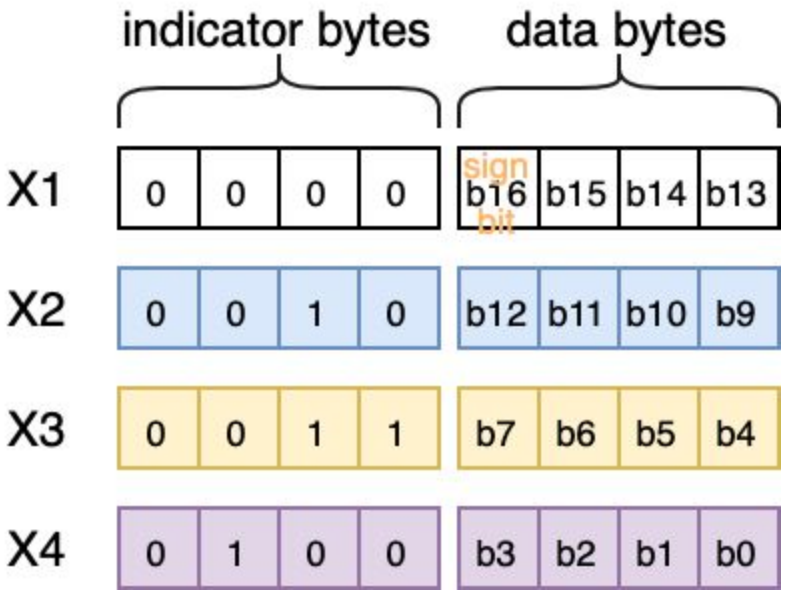

Figure 6. X coordinate distance packing bitmap

The highest 4 bits of each packed bytes indicate which byte it is of the 12 byte total data that will be sent (*X1: 0x00, X2: 0x10, X3: 0x20, X4: 0x30, Y1: 0x40, Y2: 0x50, Y3: 0x60, Y4: 0x70, Z1: 0x80, Z2: 0x90, Z3: 0xA0, Z4: 0xB0, the added 0 is for ignoring the lower 4 data bits*). The corresponding bytes would be used on the receiving end for increasing the accuracy of the datas received. The indicator bytes would also provide other functionality: 0xC0 indicator bytes would indicate the end of scanning. This custom packing method not only allows the reduction of data packet size (*from 18bytes per measurement cycle down to 12*), but also allows us to ditch sprintf (*and my own BCD implementation*) for faster data packing methods (*just a few dozen instruction cycles with bit*

manipulation). Although this does increase the complexity of the host side a bit by unpacking the data back to 16 bit integers.

The main reason to perform the calculation of the coordinates on the microcontroller is to reduce the complication of using the system, and to maintain consistency of lidar performance. However, the calculator can be offloaded to the host system to possibly further increase the microcontroller read speed from the lidar unit (*depending on the host system speed*). And in this case, as the planar rotation stage only counts up to 3600 and the stepper motor to around 1024, we can also further reduce the data packet size from the microcontroller to host pc as the total bytes required drops from 12 down to 9.

Coordinate Calculation

The math behind calculating the xyz coordinate is fairly simple. The encoder feedback for the horizontal stage counts up to 3600, which gives us a 0.1 degree resolution for every step. The stepper motor cover step 0 to 1024, corresponding to vertical angular position of -π/2 all the way to π/2. The scan starts at a step value of 350 in the software.

The conversion of the yaw and pitch is show below:

$$\Psi(yaw) = (cnt_{enc} * \pi) / 1800$$
$$\Theta(pitch) = ((cnt_{stpr} - 512) * \pi) / 1024$$

And the conversion from yaw, pitch and distance to origin to cartesian coordinate:

$$X = cos(\Theta) * sin(\Psi) * Distance$$
$$Y = cos(\Theta) * cos(\Psi) * Distance$$
$$Z = sin(\Theta) * Distance$$

Result & Analysis

The LiDAR module once fully operational takes around 6 minutes and 30 seconds to complete a scan consisting of around 5 million points on average with packed data format, and around 4 million points on average with ascii bcd data format. The final output data count also depends on the number of invalid data received based on the electrical noise of the environment. Below are a few samples of the scan result we were able to achieve. The point cloud datas are visualized in Meshlab.

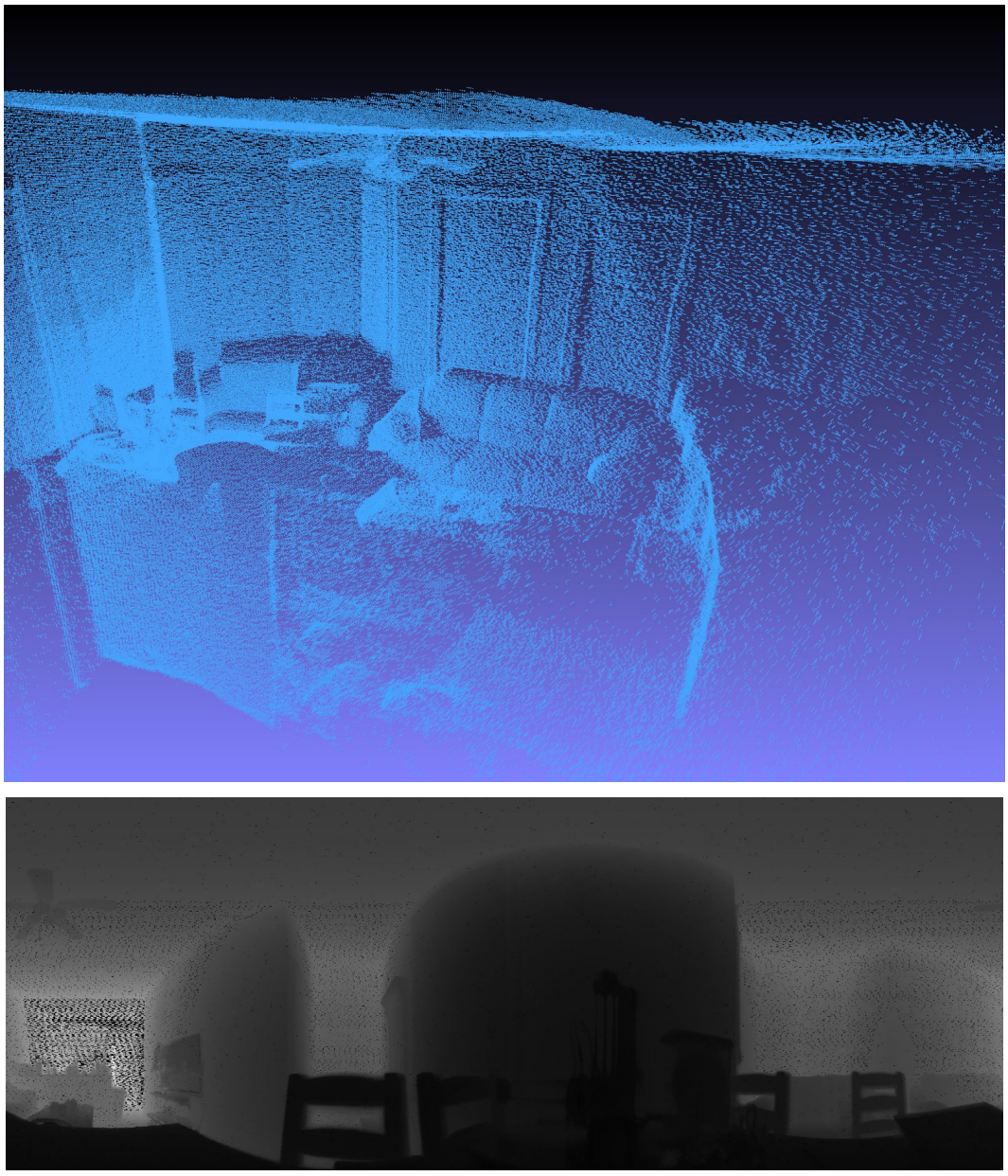

Figure 7,8. Depth scan of a random living space somewhere in boston, along with translated depth map (~4 million points)

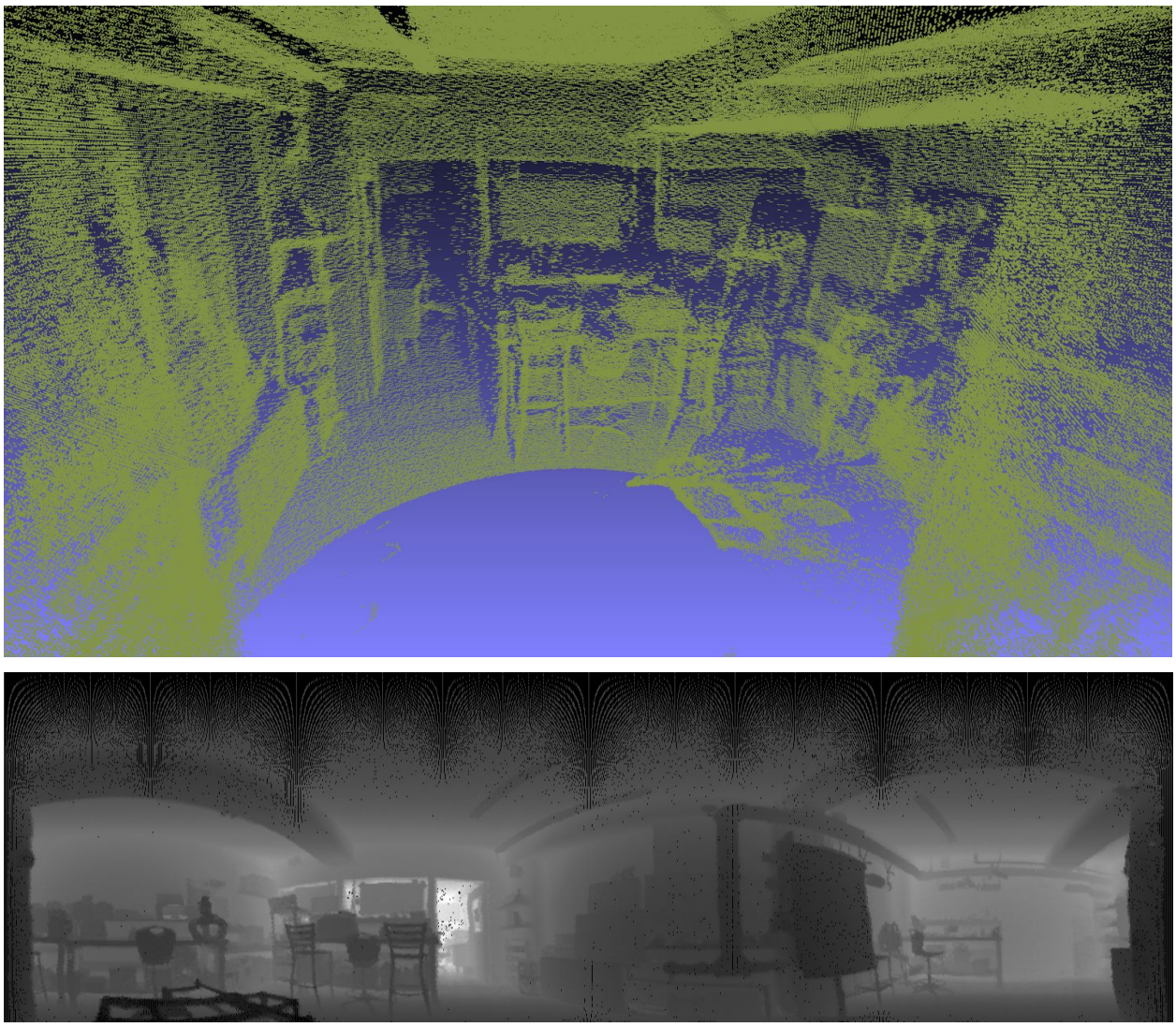

Figure 9,10. Depth Scan in front of a random lab on campus along with translated depthmap (~4 million points)

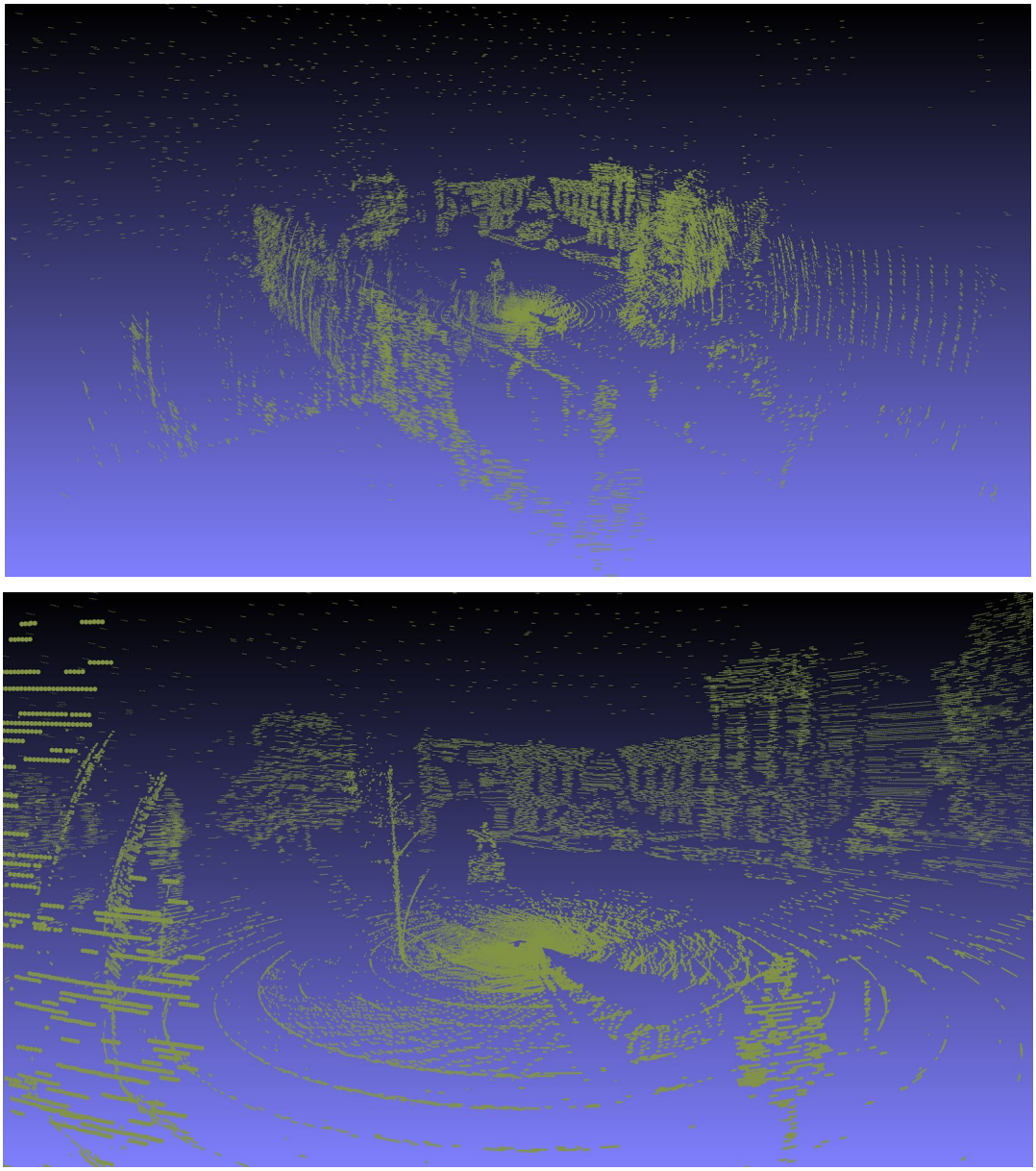

Figure 11,12. Depth Scan in front of the Boston MFA (~1.9 million points)

As one can observe, the geometries are mostly clearly defined and thus the measurements and the proceeding calculations are taken quite accurately. The depth map provides a more direct visualization of the scanning result. However, from the last two sets of figures shown above, one can observe the smearing effect. This is due to the latency between the measurement being taken and the assumed angular position of the lidar. A higher resolution encoder can also help with object details at measure objects further away.

Future Work

A second microcontroller used for pid control of the dc motor for more consistent spin (and data acquired) would be ideal. A more stable rotation stage would also allow for faster spin & faster scan as Lidar generates enough measurement to saturate the encoder resolution at 10k/s. Integrating an IMU may also reducing measurement error during scanning operation due to unwanted mechanical vibration, although this would need faster microcontroller, such as STM32H753 or the IMRXT1062 used on the Teensy 4.0, also ideally using a microcontroller with more than one core for multithreading, as I2C would hang the USART communication, RTOS might not help as switching task during I2C would probably lead to corrupted data.

While math does take significant amount of instruction cycles[2][3] (around 14 instruction for division, around 60 instruction cycles for sine and cosine), offloading the math by sending the raw encoder values, stepper motor step count, and distance measure to host for xyz calculation would perhaps further reduce the latency, reducing accuracy loss through rounding floats (although not significant), and reduce packet size per measurement cycle. Implementing a sine table on the microcontroller would also further increase the data acquisition rate and reduce the problem with losing data at 20kp/s.

Another optimization method for additional accuracy is to use the synchronization pin on the SF30/c lidar and feed it to the microcontroller as an interrupt pin which should be detecting falling edges. This will allow for a more synchronized between angle readings corresponding to the lidar reading. However, this feature is not usually present on other single channel lidar and thus wasn't implemented in the current implementation (on the other hand, lidar lite provides an external trigger pin which would allow consistent lidar reading at specific angle).

[2] http://www.micromouseonline.com/2011/10/26/stm32f4-the-first-taste-of-speed/

[3] https://www.st.com/resource/en/application_note/dm00047230-floating-point-unit-demonstration-on-stm32-microcontrollers-stmicroelectronics.pdf

There is a synchronisation signal that goes into a low state at the instant when a distance measurement is taken. This signal then goes high just before the result is transmitted on the serial port.

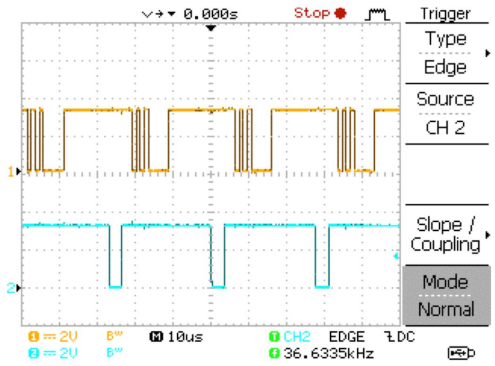

Figure 7 :: An oscilloscope screen capture showing the serial port data on the orange trace and the synchronisation signal on the blue trace

figure 13,14. Synchronization pinout diagram and example output waveform[4]

A microcontroller with double the clock speed (around 120MHz, the current one runs at 60 MHz due to USB clock and PLL math limitation as PLL doesn't provide the fractional values to allow the STM32F411 to run at 100 MHZ) should be able to provide enough instruction cycles between each USART receive for the microcontroller to read the incoming LIDAR data properly. A cursory estimate of the current instruction cycle limitation: 60Mhz with 10k/s second reading gave us around 6000 instruction cycles between each reading, while seemingly alot one have to realize USART communication between the lidar to the microcontroller, and the virtual comport USB from microcontroller to host machine would take up most of these instruction cycles due to the there nature (at 1440000 baud rate a 16bit data packet would take up around 666 instruction cycles, and USB would take up another 480 instruction cycle minimum at 12Mb/s with USB full speed connection. But to the lossy nature of USART and USB communication, around 3 times of the ideal value under normal operation is more or less accurate, and thus we are limited to around 2000 instruction cycles or less). Thus one can see the importance of reducing data size but also microcontroller raw speed (with an I.MX RT1062, which is used on Teensy 4.0, we would have at least ten times the instruction to work with, which is more than enough. Not to mention the RT1062 provides 480b/s USB high speed connection, further alleviating the data rate problem).

The V+ of the IR Led/ IR transistor can be wired up to a digital pin on the microcontroller, same with the dual full H-bridge enable pins. Disabling these while outside scanning operation would reduce power consumption

[4] http://documents.lightware.co.za/SF30%20-%20Laser%20Altimeter%20Manual%20-%20Rev%209.pdf

Afterword

This is a cursory overview of the design of this simple rotating lidar unit. Many design decisions are passed due to time constraints that I would like to discuss further in the future if I decide to revisit this (such as custom PCB for minimizing the electronic footprint/ reducing signal noise). The Solidwork Assembly also lacks the documentation of specifications of the nuts and bolts used for assembly (Mostly M3 and M2 Nuts and bolt, the M4 used for the adapter plate connecting the ball bearing strut and the rotation arm).

Some of the non technical challenges I had faced include difficulty accessing a manufacturing source reliably, shorts caused by improper wiring, and debugging the microcontroller ports (using an oscilloscope).

The lidar can also operate in 2D LiDAR by disabling the vertical rotation stage. However, the current rotational speed of the horizontal stage isn't fast enough for any practical use.