Open Source LIDAR

Final Report

Light Seekers

Gaurav Bhalla

Yile Chen

Aamhish Rao

Paul Roy

Department of Computer Science

Texas A&M University

05/06/2022

Table of Contents

1 Executive summary (1-2 pages; 5 points)

We tried working with a research group to create a novel LIDAR system. LIDAR is a new technology that has many applications in various industries such as automotive, forecasting and GIS/GPS modeling. It uses time of flight principle to map out the surrounding areas. A laser is shot out and then sensed on the way back; based on the time difference we try to create a model of our surrounding environment.

The problem is that the research group is having issues with the MEMS mirror used in their LIDAR system. Commercial products do not have non-uniform sampling rate, so the research group has tasked us with creating a LIDAR system that can be programmed to have non-uniform sample rates and also be re-configurable. In all the various LIDAR sensors we looked over in our literature review none had the capability of non-uniform sampling. Being reconfigurable means that the system can be configured to work with a motorized spinning mirror or a MEMS mirror. The MEMS mirrors create low and high speed areas of sampling. Since the research group's mirror is not working properly, our main goal was to copy the MEMS mirror capability without using the MEMS mirror. However, due to supply chain issues and a COVID outbreak in Shenzhen, China, we did not get the OpenTOFLidar PCB in time. Therefore, we had to implement a slightly different LIDAR system with help from our advisor to meet the basic requirements for this project in the remaining time. Specifically, we implemented a solution with the SF30C under the motorized spinning mirror platform to demonstrate the functionality of our quasi-Lidar motorized spinning mirror platform. Then, we mapped the data from the SF30C under the spinning mirror in a point cloud map through the RVIZ software.

Overall, we wanted to help the research group to create novel LIDAR that can be configured and programmed to suit their research's needs. The first objective was to recreate a working LIDAR system based on the OpenTOFLidar project which we completed with the SF30C LIDAR system. We tried to follow the firmware and PCB files in the OpenTOFLidar source project to build this LIDAR system from scratch and test its capabilities, and practice non-uniform sampling. However, due to the PCB not getting here in time, we were not able to meet this objective. Specifically, we made a LIDAR system with the SF30C laser and sensor, which only has a uniform sampling rate. We managed to receive the OpenTOFLidar PCB in late April, but did not have time to test its non-uniform sampling capabilities.

Then we tried to develop a final design that is reconfigurable to work with a MEMS mirror or motorized spinning mirror system. We partially achieved this objective because we were able to get the motorized spinning mirror system working; however, we did not test whether we could place the MEMS mirror on the platform in a similar way to the motorized spinning mirror.

2 Project background (2-4 pages; 5 points)

2.1 Needs statement

We are given a repository with a PCB schematic, and layout files for the OpenTOFLidar. Our first goal was to try to recreate this system first resulting in a working model of the open source LIDAR. The research group uses a MEMS mirror which is difficult to use. Yet the MEMS mirror is useful as it can move in a manner which creates low and high sampling areas. We wanted to try to follow the MEMS high scanning and low scanning regions through having a non-uniform sampling rate LIDAR product that mimics these patterns. Moreover, after this, we would aim to improve the sampling speed and the scanning range.

2.2 Goal and objectives

The primary goal of this project was to work off of an open source repository to create a working LIDAR scanner and to use the data collected to map out the surroundings. Our secondary goal was to improve upon the working model on sampling and access. In terms of sampling we would have aimed to mimic the MEMS mirror using a rotating mirror of some sort. This would have created a non-uniform sampling of the environment. Moreover, we wanted to create a reconfigurable system; in which we can easily swap out the MEMS mirror for a spinning mirror.

2.3 Design constraints and feasibility

Financially, we were given a constraint of $600 dollars. However, we exceeded this budget by about $300 because we needed to buy the SF30C, which costs $299, to implement our alternative LIDAR solution due to the PCB not arriving in time.

Regarding environmental constraints, we tried to use as little lead as possible in the PCB. This meant that I looked for ROHS compliant components for the components on the PCB when the PCB assembler told me that a component was out of stock and they wanted me to find a replacement. However, we recognize that the solder used on the PCB was most likely leaded which is not environmentally friendly.

Due to manufacturability and constructability constraints, I could only order a minimum of 5 PCBs. I could not order 1 PCB. Although, I chose to only get 1 PCB assembled which definitely lowered the overall cost of the PCB manufacturing and assembly. So, in the end, we got 1 PCB assembled with all the components and 4 PCBs without the assembled components. The high extra cost of the assembled components were mainly due to the unexpected high prices of some of the components including the laser and diode, which was most likely due to the current global ship shortage.

We were given a time constraint of four to five months to finish this project. The largest bottleneck was the one we foresaw regarding the need to ship for parts from various suppliers due to the supply chain being backed up. While we did predict that we would have issues regarding the shipping of parts, we were surprised by the extent of the impact on our design. Due to issues from ethical, social and political issues in Eastern Europe and new COVID outbreaks in China, we had some of our parts including the OpenTOFPCB heavily delayed. This resulted in us having to change to our contingency plan LIDAR involving the SF30C laser altimeter in order to still proceed forward with our project.

2.4 Literature and technical survey

We went over several different products that we could have swapped out portions of the LIDAR system. For instance we identified a different scanner module, a different motor module or even completely swapped out the repository. Overall we felt that the OpenTOFLidar offered the most flexibility and cofngrubility with our own existing system.

- Luminar Iris [10]
    - https://www.luminartech.com/products/
    - 120 degree FoV, 26 degree dynamic vertical FoV
    - 600m max range, 250m at <10% reflectivity
    - High data fidelity, 1cm range precision, high precision reflectance
    - camera like resolution, >300 points per square degree
    - Detection and tracking
        - road & drivable space: 80m
        - lane markings: 150m
        - objects & vehicles: 250m
    - Environmental
        - dust&water ingress: IP69k
        - Vibriation: ISO 16750-3
        - Shock: IEC 60068-2-27
- Ouser OS0 [11]
    - https://ouster.com/products/scanning-LIDAR/os0-sensor/
    - All-weather performance
    - 3D perception in fog, rain, dust, and snow
    - Ingress protection
    - Low temperature cold start function
    - Shock and vibration specifications

    - Designed for 100,000 hours of continuous use
    - IP 68/69K rugged, waterproof
    - robust to shock and vibration, and temperature rated from -40 ºC to +64 ºC.
    - Vertical resolution: 32, 64, 128 channels
    - Horizontal resolution: 512, 1024, 2048
    - range: 120m
    - vertical FoV: 45 degree
    - precision: ±0.7 – 5 cm
- Velodyne Alpha Prime [12]
    - https://velodyneLIDAR.com/products/alpha-prime/
    - Best horizontal (360°) and vertical (40°) long-range sensor
        - 10% targets >300m typical
        - 5% targets >180m typical
        - Ground plane hits >90m typical
    - High resolution (0.2° x 0.1°) and point density at full frame rate
    - Strong performance with retro reflectors & sunlight
- Innoviz Innoviz360 [13]
    - https://innoviz.tech/innoviz360
    - Detection Range: 0.3m-300m
    - Maximum Angular Resolution (HxV): 0.05°x 0.05°
    - Maximum Field of View (HxV): 360°x64°
    - Programmable Frame Rate: 0.5-25 FPS
    - Configurable Scanning Lines: 300-1280 Lines per Frame
    - Ingress Protection: IP6K6K, IP6K9K, IP6K7
    - Operating Temperature: -40°C to 85°C
- Quanergy M1 LIDAR Sensor [14]
    - https://quanergy.com/products/m1/
    - angular resolution of 0.033°
    - 360 degree coverage
    - Range: Detect objects up to 200m at 80% reflectivity,
    - Point Cloud: Up to 75% more data per second
    - Rated up to IP67 against dust and water ingress
- SF30C
    - https://lightwarelidar.com/products/sf30-c-100-m

- Range 50m over natural targets and 175 for reflective
- No spin
- Max Readings for Serial 20010 and 625 USB
- Analog Range 256m (0V to 2.048V)
- Serial S30-14615
- All communications are done in standard ASCII format.

2.5 Evaluation of alternative solutions

- OpenSimpleLIDAR [15]

Designed by the same creator of the OpenTOFLIDAR of which our group wanted to base our main product design based off of. However, the OpenSimpleLIDAR does not meet the structure requirements of the design for the LIDAR system we need to design. Specifically, the structure does not allow for us to place the MEMs mirror and the motorized spinning mirror easily with modifications to the design. In the OpenTOFLIDAR's design, the motorized spinning mirror provides a desired design to redirect the laser light source in different directions. With the OpenSimpleLIDAR, spinning the laser and the sensor itself would not meet the design requirements to mimic the MEMs mirror movements to spread the laser's light in different areas. In the OpenTOFLIDAR project, we are able to easily modify the 3d printed platform and PCB to allow for placement of the MEMs mirror and a motorized spinning mirror. The system with the motorized spinning mirror can be configured by Mr. Di Wang and his fellow researchers to mimic the sample rates of different commercial sensors for testing purposes.

- Xaxxon's OpenLIDAR [4]

While the Xaxxon's OpenLIDAR has a clean and compact design, it does not allow placement of the motorized spinning platform in its smaller design. So, we would need to significantly modify the 3D printed platform. The photodiode and the laser components appear to be attached to each other. In order for the design to be used, we would also have to replace the laser and photodiode parts used in the design. This could take significantly more time than with the OpenTOFLIDAR as we would need to manage great changes in the PCB and 3d printed platform design.

- OSLRF-01 [3]

The OSLRF-01 is an open source photodiode sensor and laser module. Creating a PCB and 3d printed platform from the ground up to work this open source TOF sensor would take much more time than modifying the OpenTOFLIDAR design.

- TIDA-01187 [5]

The TIDA-01187 is a Texas Instruments LIDAR distance processor but does not include the laser and photodiode sensor. So, while it can process the return signal bounced off the object into a distance measurement that can be read in a laptop, it does not have the actual laser and sensor to send and receive the light. Choosing the specific laser and sensor to work with the TIDA-01187 system would take time. In general, the TIDA-01187 system is more appropriate and cost worthy for commercial applications and not for the testing scope of our project.

- Open Source LIDAR - Unruly [2]

The design of the Unruly Open Source LIDAR is too small and does not allow for flexibility like the OpenTOFLIDAR design. The photodiode sensor and the laser are attached to each other and would need to be replaced with another photodiode and laser modules that can be separate and configured correctly with a MEMs mirror, the motorized spinning mirror on a 3d printed platform. Moreover, creating the 3d printed platform would take significant time in comparison to modifying the existing 3d printed platform from the OpenTOFLIDAR design.

In general, the commercial LIDAR TOF sensor modules were not considered because a non-uniform sampling rate to mimic other TOF sensors for testing purposes cannot be achieved with them.

3 Final design (5-10 pages; 5 points)

3.1 System description

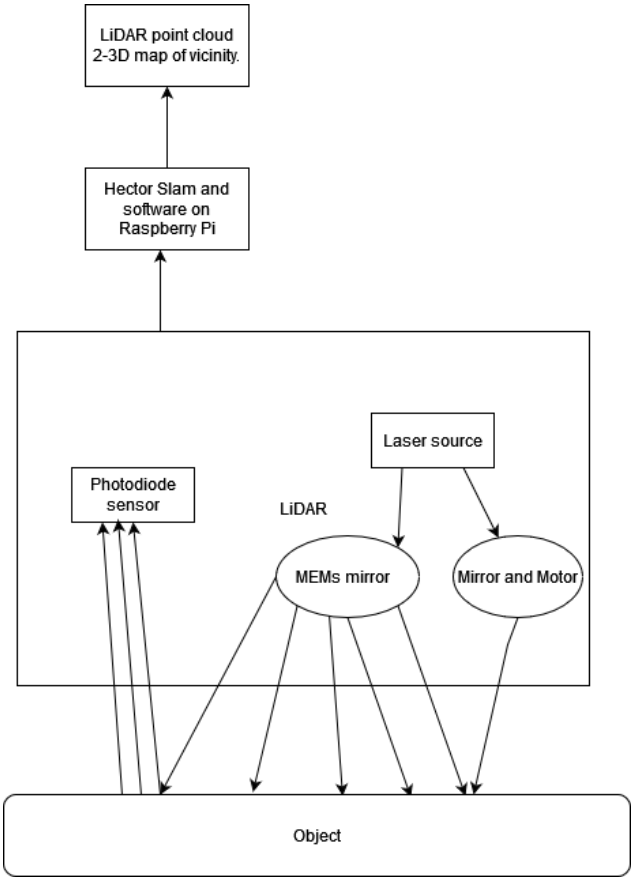
The following is a list of components that are on the block diagram below:

LIDAR Point Cloud: The LIDAR Point Cloud would essentially be the final product of our project, this would be a point cloud that would show the objects surrounding our LIDAR system and could then be used for systems such as autonomous driving or other autonomous motion.

HectorSlam/ROS Software: This layer would take the data from the LIDAR system and convert it to the LIDAR Point Cloud mentioned above using the HectorSlam software.

Laser Source: The laser source is the main component of the LIDAR and is used to track the object.

MEMs Mirror: The MEMs Mirror is an alternate to the traditional mirror and motor method of LIDAR and is electric and has the motor built into the mirror.



Mirror and Motor: The Mirror and motor are used to move the laser light around so that the photodiode in the SF30C can capture all of the objects in the environment. The motor will continuously be moving around so that new objects will be detected in real time.

Object: The objects we are trying to track in the environment. The system will come together as one in order to track the objects in the environment.

Photodiode sensor: The photodiode sensor is used to detect the light being reflected from the laser upon contact with the objects that are being detected in the environment.

The electronic components such as sensors and power components will be soldered together using the PCBway manufacturing and assembly service, while the physical components will be 3D printed and constructed together. We also plan on using an Arduino and Nvidia Jetson Nano and have the LIDAR module that we are using to be wired to that. The software will all be running on the Nvidia Jetson and Arduino and we will have the data readable on a monitor. More details will be discussed in the SF30C LIDAR system low level design.

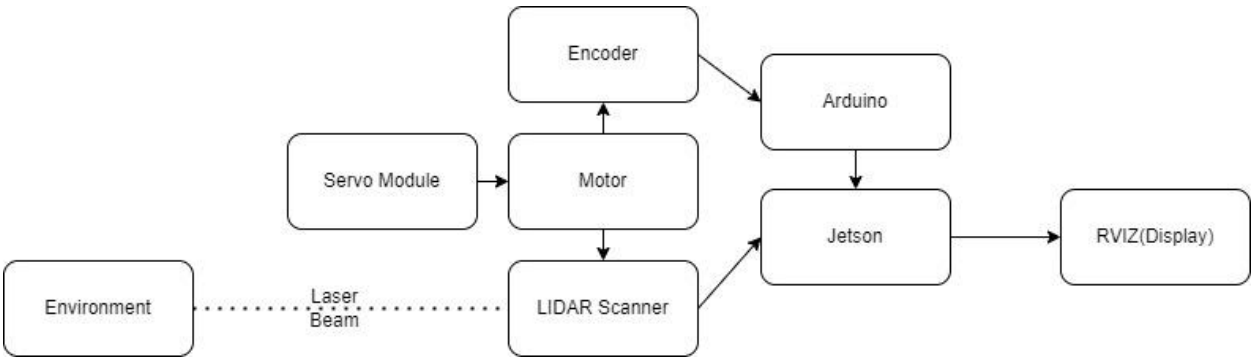
We wanted to have two alternate methods for the mirror and motor aspect to the project, we wanted to either use a MEMs mirror or use the motorized spinning mirror platform that we built.

3d printed mount which would mount the mirror to the motor and spin around. We will be using both methods and testing out which one would be better.

The overall system will have a fair amount of room for flexibility as we can add components as needed to the LIDAR module and additional motors if needed to the 3D printed design. The only issue is the processing time for getting components and having them assembled through the service as that is done in China and so due to this we are not planning on making many alterations upon the initial send off.

3.2 Complete module-wise specifications

- **SF30C LIDAR System Low Level Design**

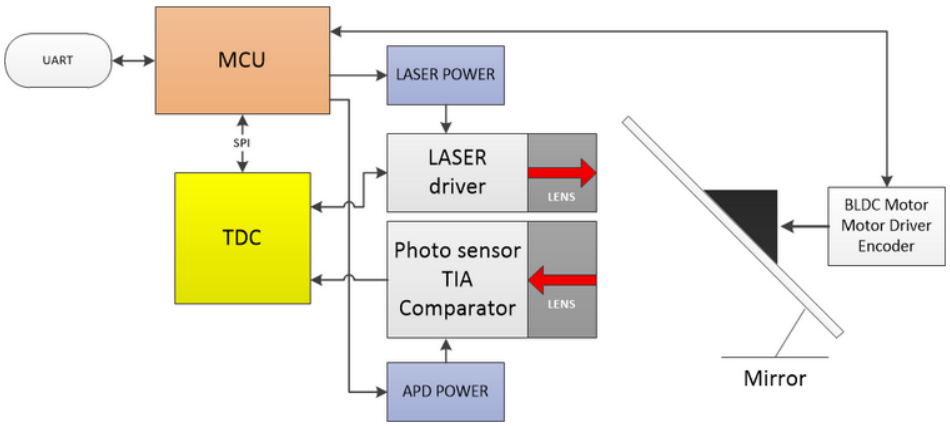


The diagram above details the low level design of the alternative solution that we implemented. The environment is the local area. The lidar scanner is the SF30C altimeter. The SF30C was connected to the Nvidia Jetson Nano and placed under the motorized spinning mirror platform.

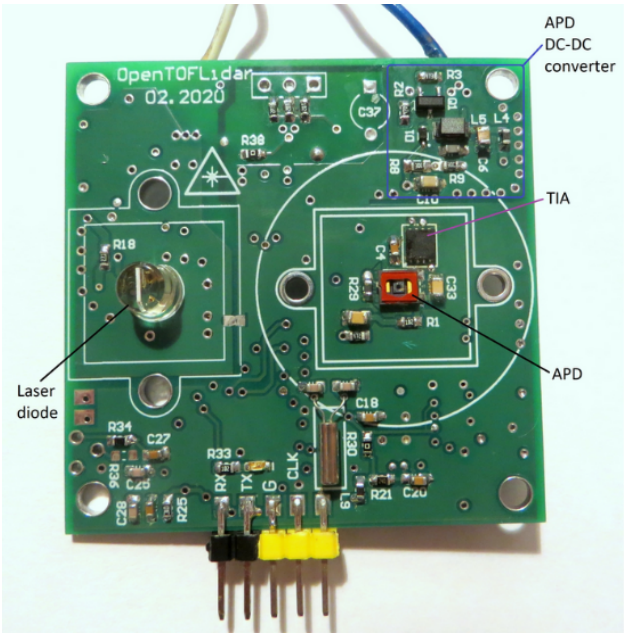
The motorized spinning mirror platform holds the motor attached to the mirror holder, and the encoder is attached to the motor's shaft. The brushless DC motor is connected to an ESC which is connected to a 12 V LiPo rechargeable battery pack and a servo module component. The motor

speed is controlled by a servo module component since we were not able to get the motor to run at slow speeds via the Arduino program. The Arduino program reads the speed of the motor through the encoder and sends the Revolutions Per Second (RPS) readings to the Nvidia Jetson. The Arduino Mega is connected to the Nvidia Jetson Nano through USB. The NVIDIA Jetson uses the RPS readings and the SF30C altimeter sampling frequency (readings/sec) to approximate the number of readings per revolution in a program. The jetson then uses the distance measurements recorded from the SF30C to create a point cloud map which is displayed through RVIZ.

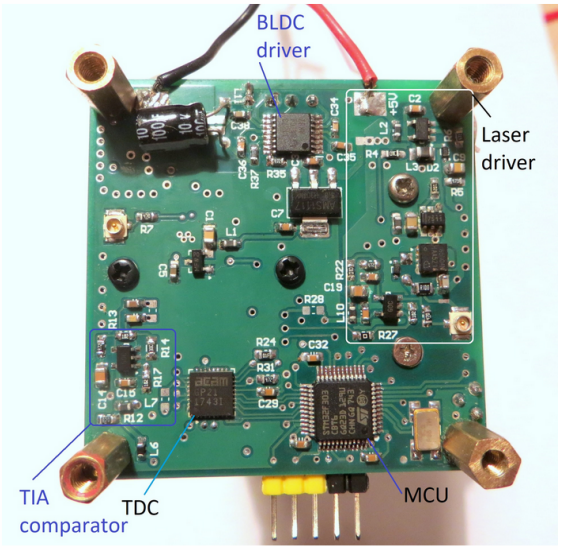
- **Circuit and logic diagrams**



Shown above is the low-level design for the OpenTOFLidar system [1]. The MCU communicates with and controls the laser and the photodiode. Specifically, the code executed on the MCU specifies the input voltage for the laser and the photodiode. Moreover the code that will run on the MCU also controls the motorized spinning mirror platform through the BLDC motor driver encoder chip. The MCU samples the distance (Time-of-Flight) data from the TDC through SPI through polling. However, the method to read data from the TDC can be improved through using interrupts instead of polling.

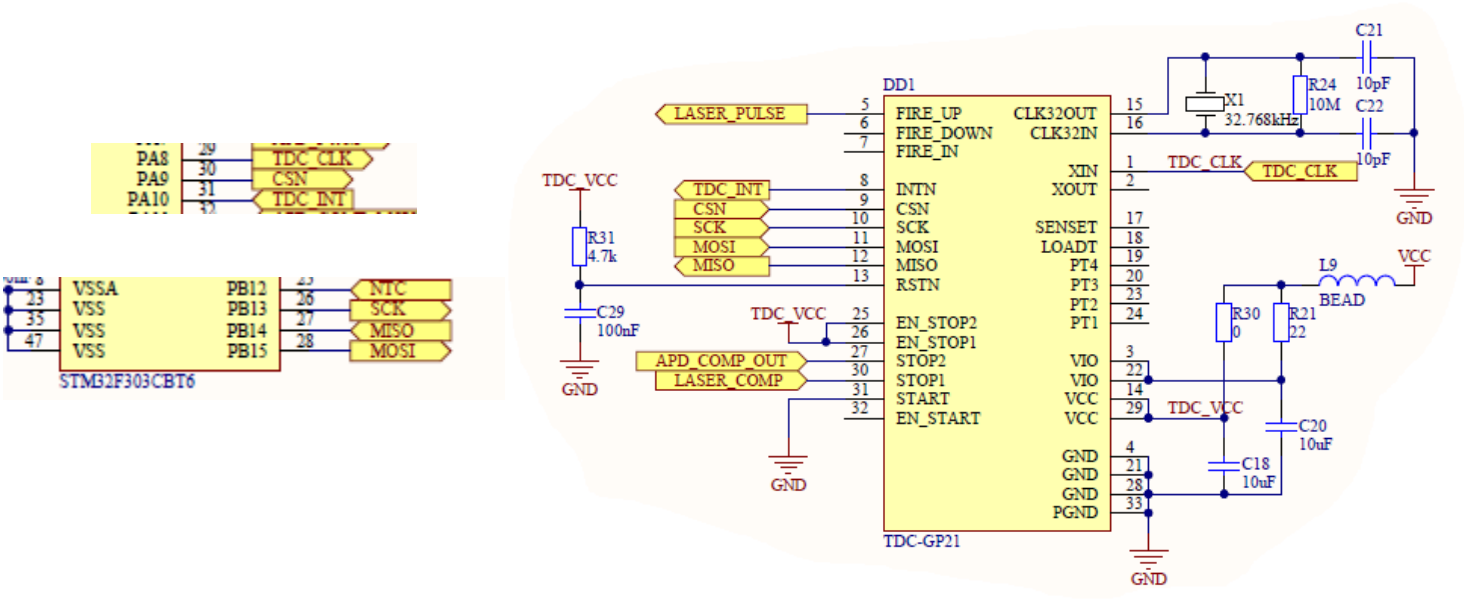


Shown above, is the OpenTOFLidar Author's annotated image of the top part of the PCB [1]. The main components on the top part of the PCB are the TIA, the DC-DC converter, the APD, and the laser diode. The transimpedance amplifier (TIA) amplifies the current received from the photodiode into a voltage. Since this PCB is a rangefinder, the TIA accommodates a large signal bandwidth to amplify the signal produced from the APD. The Avalanche photodiode (APD) is a special photodiode component that has higher gain and signal-to-noise ratio which allows for greater sensitivity of the diode compared to normal PN diodes which is very useful for LIDAR. APDs require a larger reverse bias voltage for operation to occur compared to normal diodes. Therefore, a DC-DC converter was used to form the high enough reverse bias voltage to use the APD. The laser diode produces a 905 nm wavelength of light but requires a large current of about 30A .The lens for the laser diode focuses the light produced by the laser diode allowing for stronger concentration of the laser beam, allowing the 905 nm wavelength light produced from the laser to reach farther.



Shown above, is the OpenTOFLidar Author's annotated image of the bottom part of the PCB [1]. The main components on the bottom part of the PCB are the MCU, the laser driver, the TDC, the TIA comparator and the BLDC motor driver. The brushless direct current (BLDC) motor driver receives a PWM signal through UART from the MCU code which controls the motor speed. The laser driver creates a high energy pulsed signal for the laser diode to output. The TIA comparator distinguishes the signal from the noise received by the APD through the reflected light. The Time-to-digital-converter (TDC) measures the time it takes to receive the reflected 905 nm light (TOF) from the start of one of the laser's pulses, allowing for the MCU to accurately sample distance values to objects in the surrounding area. As explained with the low level block diagram, the MCU is responsible for controlling and communicating with all the main components mounted on the PCB. The code executed on the MCU generates the pulse for the laser, reads in data from the TDC through polling to sample distance values, controls the motor speed, controls the power provided to the laser diode and the APD, and receives/sends the distance values to the Orange Pi PC through UART.

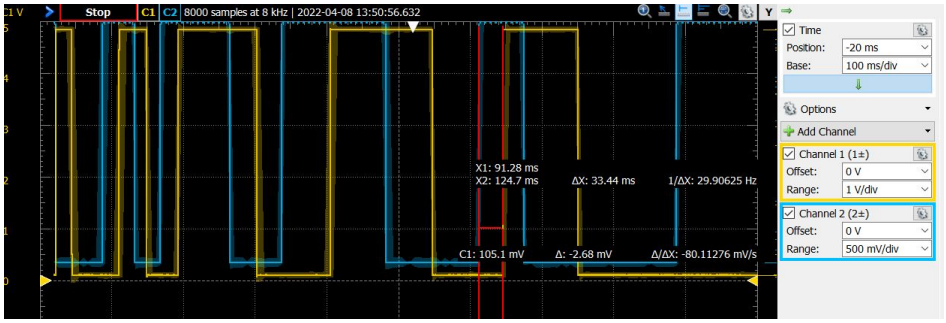
- **Interfaces and pin-outs**



A Serial Peripheral Interface (SPI) is used to communicate between the TDC and the MCU through polling. SPI is used for short distance communication between modules in an Embedded system. In this case, SPI is used for communication between the TDC (slave) and the MCU (master). In the parts of the schematic shown above, the MCU pins used in the SPI communication and TDC chip pins are shown through their corresponding schematic diagram. There is only one master and slave chip in this system. The master (MCU) pinout is shown on the 2 pictures on the top left and the slave (TDC) pinout is shown on the right. There is the TDC_CLK pin for the clock signal, the Chip Select Not pin (CSN) which is active when the signal is low to activate the SPI connection between the TDC and the MCU; Master in slave out (MISO) which is the channel for sending data from the TDC (slave) to the MCU (master); and Master out slave in (MOSI) which is the channel used for sending data from the MCU to the TDC for configuration purposes. Outside of SPI communication with the TDC, STOP2 - STOP1 is calculated via an ALU to determine the time of flight. From there, after multiplying by the speed of light, distance measurements are recorded and sent to the MCU.

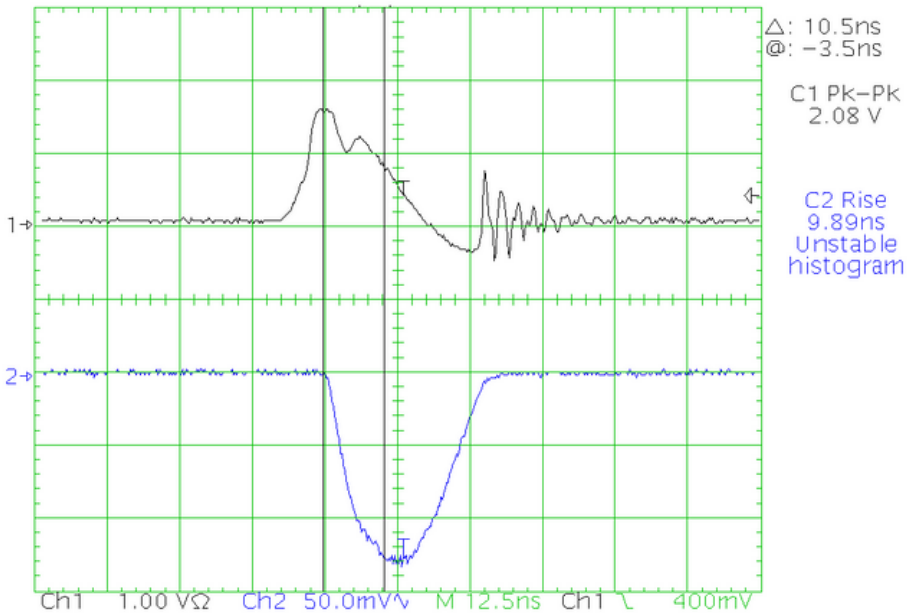
- **Timing diagrams and waveforms**

SF30C LIDAR Encoder - Channel A and B



In the screenshot of the AD2 oscilloscope screen above, the channel A and channel B signals from the encoder are shown above. As seen, channel A and channel B are about 90 degrees out of phase. The encoder and encoder disk are designed to have a resolution of 500 cycles per revolution (CPR). At 500 cycles, the encoder disk screwed onto the shaft of the motor would have completed one full revolution. Moreover, if the motor spinning switches direction from clockwise to counterclockwise, then channel A will be -90 degrees out of phase with channel B.

OpenTOFLidar PCB Laser and Photodiode Signals



In the image above, channel 1 is the laser pulse, and channel 2 is the signal received by the Avalanche Photodiode (APD). The signal in channel 2 is inverted because APDs are reverse biased [2]. By subtracting STOP1, which is the time the laser pulse is emitted by the laser diode from STOP2, which is the rising edge of the APD signal, the Time-of-Flight (TOF) is calculated. Then, the distance measurements to objects in the surrounding area is calculated and sent to the MCU by SPI.

- **Software processes with their inputs and outputs**

get_lidar_data.py

This file is used to get the distance measurements from the lidar. The lidar is connected to our jetson nano, and we obtain the distance measurements from the usb port.

display_ros.py

This file takes in the lidar distance measurements and creates a dummy lidar_message in ROS. It then launches ROS and displays the distance in rviz in real-time.

Encoder_readings.ino

The encoder and encoder disk is designed to be 500 cycles per revolution. So, this program reads in the signal from channel A and counts the square wave cycles 500 times, and also records the time for this to happen. This program then outputs the Revolution per second (RPS) speed of the motor on every revolution through the Arduino Serial.println() function. The Arduino program communicated with the Nvidia Jetson nano at a standard baud rate of 9600 bits per second.

- **Approach for Design Validation**

Our approach for design validation was to first test the distance measurements of the SF30C altimeter as if we were testing the OpenTOFLidar PCB. This meant validating the physical distances between the device and the object against the distance recorded from the altimeter device.

Then, we would test the speed of the motor through the Arduino program. Specifically, we would start at slow speeds and then slowly increase the speed of the motor. We would see if increasing the speed through the servo module's knob would increase the RPS speed value recorded by the Arduino program. Although this second test was more crude, it was the best test we could come up with to test the motor speed at the current time due to time constraints of the semester ending and final examinations.

4 Implementation notes (10-20 pages; 30 points)

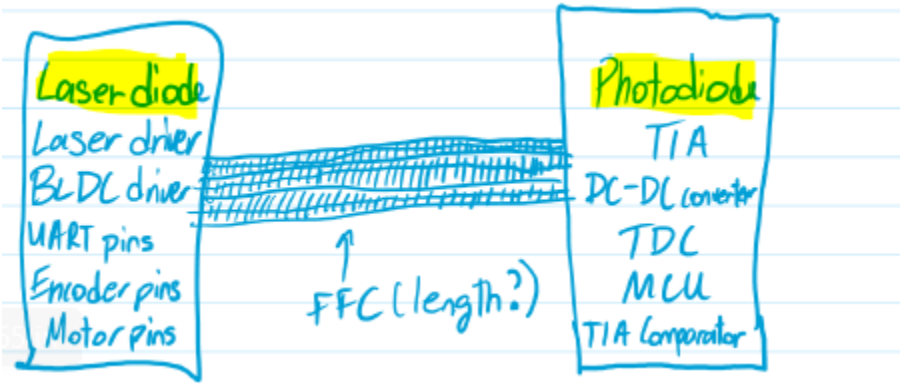
PCB:

The start of a Printed Circuit Board (PCB) is with a generalized schematic. The generalized schematic should output the necessary voltage and current parameters. Furthermore, the input and output impedances should also be theoretically determined. Knowing the results of adding specific components of particular values is the result of knowledge from training in circuit theory and circuit design. After, to create a more accurate schematic -to be associated with a PCB design- you look up parts for each component on part supplier websites like digikey or mouser electronics. I personally occasionally use octoparts which detail the availability of a specific part on different electronic part supplier websites. Then you import or create PCB schematic symbols with the PCB designer you are using (E.g. Altium Designer, Eagle Autocad, etc…). After all the schematic symbols are created or imported into the custom schematic library you are using, you would need to create the schematic by importing these symbols into the schematic and connecting them according to your general schematic design. Next, it is needed to create/import component footprints for routing in the PCB layout file. After, the designer should then route component footprints in PCB layout. Finally, to get the PCB manufactured, one should export gerber files and pick and place files to a PCB manufacturer's website. Then, one can work with the manufacturer to fix any design errors or find alternative components if the component listed in the Bill of Materials submitted is not present. Additionally, if one does not have the tools for electronic assembly it is recommended to use the PCB manufacturer's assembly services to properly solder all of the required electronic components onto the PCB.

For this project, I managed to learn how to order the OpenTOFLidar PCB with PCBway manufacturing service in addition to using their electronic assembly services since I did not have the proper electronic assembly equipment for PCBs. I learned how to order the OpenTOF PCB and ordered it in late February 2022. However, due to the supply chain shortage on electronic components and a COVID outbreak in Shenzhen, China where the PCB was to be manufactured, the order for this PCB and its assembly service was significantly delayed. Specifically, the order of our OpenTOF PCB was delayed such that I only received the assembled PCB in mid April 2022. Additionally, the PCB assembly team could not get two

components since they were out of stock. So, they shipped the PCB without these two components: a 32.768 kHz quartz crystal, and a 8MHz quartz crystal. Due to my other commitments in the remaining time till the end of the semester, I was only able to solder the remaining two replacement components I ordered recently onto the OpenTOF PCB. I was not able to fully test the OpenTOF PCB due to my other commitments to my team. In particular, I was working with my team to get the SF30C backup solution working, to document our project and to also work towards completing the modified PCB design.

I was not able to finish the modified PCB design, because it took significant time learning all of the processes that go into designing the PCB through online courses while also working with my team members to design and use the other hardware and software for the project. So, I was not able to complete the modified PCB; however, I was able to complete a basic plan on paper of the PCB I wanted to design. Moreover, after watching the videos of the two udemy courses on PCB design with Altium, I familiarized myself with the basic general processes of PCB design and also with using Altium Designer. Specifically, I familiarized myself with the PCB design through creating a practice schematic and layout of a diode circuit made in the first Altium design course. It should be noted that the standard units of the grid in the schematic and layout editor of Altium designer can be changed between the imperial unit of mils, metric unit of millimeters, or the US unit of inches. So, I developed the skills necessary to create the modified PCB design. However, I was not able to complete the execution of the modified PCB design due to the large time it would have taken to create all of the required schematic symbols, component footprints and do the routing in the PCB layout. Below, I also provide my plan for designing the modified PCB which connects two PCBs, one for the laser and one for the photodiode, which are connected by a Flexible Flat Cable (FFC):



Motor:

The Motor was initially perceived as a very simple task, as usually it is just two wires of DC Power that one would simply connect to a port on an arduino or some other voltage limiting source, but unfortunately we had a much more complicated experience with our specific motor.

The initial issue was that the motor was 12 volts, hence we could not simply use the arduino power supply to power up the motor. We needed some way to get the motor to run on 12 volts of power. The first idea we had was to use a step up transformer, which would convert the 5v from the arduino and convert it to 12 volts. Unfortunately, we did not realize that this would not work in our situation as the arduino only had a limited amount of current that it supplied and because of this, the transformer did not transform the 5 volts to 12 volts properly, causing the power supply to not be enough to actually power the motor. So, we decided to then use the power supply in the FEDC, as that could produce a proper amount of current and up to 12 volts, upon using the simple hack of having 6 volts on one supply node and -6 volts on the other supply note, causing the voltage difference to be 12 volts and so being enough to power the motor. However, this could not be taken outside of the FEDC, so we needed to come up with some other ideas that were more on the mobile side. We thought of maybe using an Analog Disovery to provide the voltage, as we could use the same -6, +6 voltage "hack" with this, but we ran into the same current problem. We also thought of using an opamp circuit to achieve the -6V but the circuit could not source enough current necessary for the motor to run from the Analog Discovery 2 or the Arduino. We then thought of using a standard 120V AC to 12 V DC power supply, that could be used for this very purpose, of converting the 120 volts from the very available plugs found in a common home or building and making them 12 volts of DC voltage at a current of 2V. This worked well, but still did not solve the issue of making the project completely mobile, so we ended up going with a 12V battery, and this accomplished exactly what we needed - gave 12 volts at whatever current the motor drew and worked perfectly.

Another issue we ran into was simply controlling the motor. We could not just control a brushless motor with a simple micro computer. These things did not even just have 2 leads like a normal brushed motor, they have three leads. Hence in order to use one of these motors, we need an electronic speed controller. The electronic speed controller needed a power supply as well as a PWM signal in order to control the motor speed. Doing this, essentially turns the 12v brushless motor into a servo, something that the arduino has countless different libraries for and has a multitude of support with. We could use a PWM Signal from either a servo control module or using the PWM signal found on an arduino, jetson or some other kind of microcontroller. However using the ESC was not as easy as one would think, we could not just use a PWM signal directly. Due to the fact that these motors can easily cause damage to humans and potentially break body parts, they had a safety mechanism in the Electronic Speed Controller that would make sure that the motor does not turn on without intervention, as if the motor is randomly turned on and

the user of the motor is not prepared, then the motor could easily cause some iverrisible damage. However, the fact that such a safety lock existed was not very documented and it took us a while to even realize it existed as we just thought that it was making random beeping sounds. Eventually we realized that we needed to hold the Servo controller at a low speed for a few seconds and then it would arm, which was symbolized by the motor no longer making any beeping sounds and that would mean that if we were to then supply a PWM signal that would represent the amount of voltage passed to the Electronic Speed Controller then it would run. Now, we were able to control the Motor through the Electronic Speed Controller using the Servo controller module but this did not solve the issue of us not being able to control the motor using the Electronic Speed Controller with an arduino or raspberry pi or some other kind of board, so that we could ideally control the speed using a computer, perhaps with the help of an encoder and a PID controller. So with the aid of some youtube videos we found online, we found a video in which someone did exactly what we were thinking of, they essentially came up with the sequence that would arm the motor, and we used their logic and were able to get the whole thing running without any issue. However, we found that for small speeds, less than 30/180, it would not run and we did not understand why that was the case.

The motor used a ribbon cable, and this was deemed as being very difficult to use, as the pins were very small to solder on to and it was very fragile, and it broke not only once but twice, and one of which was able to be fixed using solder while the other time was not able to be fixed as the motor snapped at a very early stage of the wire. We had come up with some very innovative ways to deal with the ribbon cable breaking, one of which being using sandpaper to basically get the copper connections out of the ribbon, as the ribbon based used a material to try and seal up the copper connections that were inside of it, similar to the insulation that is in a standard wire aside from the very obvious fact that the insulation in the ribbon can not be stripped with just a stripper, essentially making it so that the sandpaper acts as the insulation stripper in this situation and then basically soldering standard wires to that when the wire had torn.

The motor was also not initially compatible with the Jetson, as the jetson lacks a hardware PWM module and so we had to do some research and eventually found a hat that basically added this hardware support and we could use it for the motor controlling as well as the encoder controlling, as both of these required PWM connections and as stated before, the NVIDIA jetson lacked hardware PWM support and unlike the Raspberry Pi there was no software simulation in order to get this logic to work and so we had to order this board and wait a few more days before we were able to successfully attempt to try to integrate the motor and encoder which both used PWM to communicate with their boards with the NVIDIA Jetson control system.

Overall, the motor setup was a lot more time consuming and demanding than what we had imagined while coming up with the plan of using this motor and system to control the moving component of our lidar, but at the end we were able to come up with innovative solutions and learned a lot of motors and controlling them all in the process of tackling these issues.

Encoder:

The main requirement we had when researching the encoders for the project was it could fit with the motor we were using. First, we spent time understanding how encoders work to measure motor speed and direction. Then, we had to ensure that the encoder would fit onto the shaft of the motor. Since there were specific encoder disk sizes and encoder types, we had to look at the datasheet to get the right encoder disk and encoder combination (4mm diameter shaft hole) for the motor for the project. Then, we also had to record the resolution of the encoder we would get. In general, the higher resolution encoders are very much more expensive and some of them or their corresponding encoder disks that could fit our motor were out of stock. So, in the end, I chose a 500 CPR resolution encoder (HEDS-9140#A00).

Moreover, the encoder disk that met the requirement of fitting on our motor and that was in stock only worked with the 3-channel encoders. One of our team members accidentally got the 2-channel version of the encoder (HEDS-9100) from Taobao, so we had to order the correct version of the encoder (HEDS-9140) from Newark Electronics which was more expensive. However, the part got here in time since it shipped from the USA. We started the encoder work for the project only after we finished setting up the motor and the ESC to run properly. First, I revised my understanding of the Encoders, and then I looked at sample code to understand how I could effectively read the speed from the program. I did not completely understand the sample code due to its complexity. However, I understood that it used the attachinterrupt() function which could be used to effectively read a PWM pulse.

There are several types of encoders: mechanical, optical, magnetic and electromagnetic induction types. From my research, I found optical encoders to be generally cheaper and also provide higher resolution and accuracy than other types of encoders. Moreover, it is very easy to get the angle readings of the motor shaft in the course of its rotation through analyzing the state of Channel A or B. Specifically, the encoder disk, HEDS-5140#A11 has 500 Cycles Per Revolution (CPR) resolution. So, we could count 500 cycles per motor revolution where each cycle of the square wave signal (channel A or B) would count for 360/500 degrees.

Quadrature Encoder means there are 4 main signals coming from and going to the encoder: Channel A, Channel B, VDD, and GND. VDD and GND were input signals to the encoder, and Channel

A, Channel B, and Channel Index were the outputs. 2-channel quadrature encoders have only Channel A and Channel B while 3-Channel quadrature encoders additionally also have an Index channel. The index channel outputs a pulse every time the unique index mark on the encoder disk passes between the laser and the photodiode in the gap of the encoder which signifies the motor made a full revolution. This is useful for measuring the angular speed of the motor without the need to count the pulses of Channel A or Channel B. However, Channel A and Channel B are still useful for determining the motor direction. Channel A and Channel B both output 500 cycles of square pulses per revolution of the encoder wheel and are designed to be 90 degrees out of phase with each other. Channel A and Channel B being 90 degrees out of phase can help determine whether the motor is spinning clockwise or counterclockwise. For instance, if Channel A is +90 degrees out of phase with Channel B, the motor will be spinning clockwise, while if Channel A is -90 degrees out of phase with Channel B, the motor will be spinning counterclockwise. For our project, I chose to count the cycles of Channel A from the encoder to measure the RPS speed of the motor because I was having trouble capturing the speed from the index signal. However, if possible, I would recommend measuring the speed from the encoder signal and comparing it to that of the index signal to better validate the motor speed measured from the encoder through the Arduino program at high speeds.

Alternatively, we could also have done as our advisor Mr. Di advised and read in the angle data of the encoder and the analog voltages from the serial port of the SF30C synchronously through the Arduino ADC's sampling period after configuring the motor to spin at the right speed of course and send that data as a LIDAR message to RVIZ; however, I do not think we did not have the technical knowledge or time to complete this. We initially tried to do thi; however, we were not able to parse the analog voltage readings of the SF30C altimeter due to an unknown error. Thus, for our project, we focused on reading the SF30C distance readings through USB to the Nvidia Jetson nano.

ROS (to create hector slam map):

ROS is the robot operating system, it is a specific set of frameworks used to communicate different hardwares with each other, generally implemented in order to create robotics systems. The system is especially useful to abstractify hardware and devices in order to simplify messaging of data between different systems. These sensors, devices and processes work in tandem as nodes, to communicate and make decisions. This can be done in a variety of ways, such as robot models, perception, localization, mapping and planning.

The system is resembled as a graph structure made up of nodes and edges, called nodes and topics respectively. A main process named master registers and starts up the nodes, but afterwards the nodes work on their own, this decentralization of nodes works well in systems with off board computations or multiple separated networks. The node represents some process running. The nodes are the specific programs running in isolation and can take some data and make a transformation which can then be sent out. The topic, which is called an edge, is the communication between the nodes. This communication is done with a publisher and subscriber model in which data such as sensor data, motor control, states and so on.

For our specific program we used Noetic Ninjemys which is the latest form of ROS 1. We used this distribution to set up our ROS node and edge system. We ran the entirety of the system from four main nodes which communicated with each other. The four nodes were called roscore, rosrun, python and rviz. We also used a rosbag in a specific case. The first node is called roscore, this is just a setup command which begins a collection of nodes and programs that are pre-requisites of a ROS-based system. The system is a massive collection of background nodes that would be too large to go over right now. The second command is rosrun static transform publisher. The function takes a series of parameters and uses them to transform based on a *x,y,z* and *yaw, pitch and roll* transformation. These are placed from a *frame_id* to a *child_frame_id.* These frames are eventually used by a rviz to visualize the sensor data. I will go over rviz in the last section as it is complicated enough. Finally the last program we run is the python program. This is the program that takes the values gathered from our sensor data. This is the laser scan message mentioned earlier.

In order to somewhat make sense of the data at hand we had to create a visualization of it. To do this we came up with two options, using hector slam and rviz. We ended up primarily using

RVIZ

Hector Slam is a mapping visualization software used with ROS systems commonly. The hector slam node takes a bagfile as its input. Bagfile can be conceptualized as a recording of the output of some edge in a given ROS system. The bagfile can be accessed as many times as needed in order to visualize and present data multiple times. Through hector slam you are able to map the robot as it moves through an environment, as well as the certainty of certain barriers and a mapping of the movement path that the program works through. In addition the robot's current orientation and path can be determined. This is done by working in tandem with two packages called hector_geotiff and hector_trajectory_server.

Rviz is the program we ended up using in order to visualize the sensor data in real time. RVIZ is short for ROS visualization, and it is a tool used to visualize whatever the robot is seeing or doing.

For our purposes it took a frame named laser_frame and superimposed the distances we calculated on to it. The frame is an important instance for visualization. The two frames that matter are the fixed frame and the target frame. The fixed frame is the reference frame which denotes the "world". The other frame that matters is the target frame which is similar to a reference frame in graphics. This frame is the camera or in our case the sensor's view. Both frames can be switched in and out of. The other parameter that is worth mentioning is the use of different views. Views/Displays are groupings of sets of data. This is useful in order to not overwhelm the viewer with too much information. In our case we only wanted to show the laser scan message as our data and were able to remove items such as frame_posoiton, axes and grid.

Running Code Jetson:

In this section we continue to discuss the software system as well as the computing platform used in our project. In order to run the SLAM software that is used for processing the lidar data and visualizing the results, we needed a computing platform that was powerful yet portable enough to fit properly in our design. We do not need a very high end computer, which is usually very expensive and takes up a lot of space. Instead we want to have a device that has a nice balance between computing power, physical size, and price. In our initial design stage, we came up with three options for consideration after researching the market for devices that would suit our needs. The three options were Raspberry Pi, Orange Pi and Jetson Nano.

The Raspberry Pi is a low cost computer about the size of a credit card. It can plug into the monitor and be operated by a keyboard and mouse. It is capable of tasks of a regular computer or laptop but at a much cheaper price. More importantly, the Raspberry Pi is used in a wide range of open source DIY projects. It has a large global community of developers and lots of hardware and software support. The operating system running on the Raspberry Pi is called Pi OS and is similar to linux. It comes with various hardware configurations which are suited for people with different needs.

The Orange Pi is a similar product to the Raspberry Pi, but it is much more configurable in terms of operating system. Users have more options of operating systems to run on the Orange Pi, such as Android, Ubuntu, Debian, etc. It also comes with different versions of hardware, which provide different

computing power based on the user's needs. The OpenTOFLidar open source project also used the Orange Pi, so we thought it would be fine to use it as well.

The Nvidia Jetson Nano is a small and powerful computer that can power small, power-efficient AI systems. It has the power to run multiple neural networks in parallel for machine learning applications such as image classification, object detection, segmentation and speech processing. It runs a variation of linux OS, similar to Ubuntu, and can be used like a regular computer for a variety of tasks.

After researching and reviewing the three options for the computing platform, our most preferred choice is the Jetson Nano, as it can run Ubuntu and it is best in computing power amongst the three options. We initially planned to borrow a Jetson Nano from a team member's organization, but the circumstances didn't work in our favor. The organization wanted to sell us the Jetson Nano instead of borrowing it, but it would not be possible to apply for reimbursement in this case. Therefore our alternative was to use the Orange Pi. We did not choose the Raspberry Pi in the end because it did not run Ubuntu Linux, and ROS/SLAM software were best supported on Linux compared to Pi OS. After our Orange Pi arrived, we first had to flash the operating system onto the SD card, in this case we wanted to use Ubuntu Linux. However, after following a few tutorials to set up the Orange Pi, we were not able to get Linux running on it. Moreover, online support on the Orange Pi was limited, and there wasn't a large developer community online. Therefore, we had a hard time finding any resources that could help us set up the Orange Pi properly.

Just when we were stuck on configuring the Orange Pi, I remembered that a friend of mine had a Jetson Nano, and he gladly borrowed it for us. Setting up the Jetson was fast and straightforward, as the instructions were well documented. It also had a large developer community online, so debugging any issues was easier as there are many resources available online. After setting up the Jetson Nano, we started developing code. The first major step was to install necessary software packages, this included ROS and some other python libraries.

For the software side of our project, the first part was reading in data from the lidar and obtaining the distance measurements. The Lidar is connected to the Jetson Nano via USB, and there was some sample code provided by the lidar manufacturer on running the lidar and reading distance data in python. We tested the sample code, and added our own code to have it running properly. By now we were able to successfully obtain real-time distance measurements from the lidar. For the next part, we initially intended to control the motor via a python program. However, we also needed to obtain readings from the encoder,

which was not possible via python. Therefore, we decided to not control the motor from the Jetson Nano, but rather use the arduino instead. After obtaining the angle measurements and rotation speed of the motor, we can proceed with creating the ROS lidar message node and run ROS on the Jetson Nano.
For the final part of the software component, we incorporated the laser distance measurements with a ROS laser scan message node, and used RVIZ to generate the point cloud map. We first went through some official ROS documentation and online tutorials to understand the laser scan message and how to create it. Then we started with some official code provided by ROS, and implemented additional features based on our needs. We also had to integrate the laser distance measurements into the ROS code. In the end, we were able to integrate the hardware components with the software components, and the Jetson Nano serves as an important piece of the lidar system.

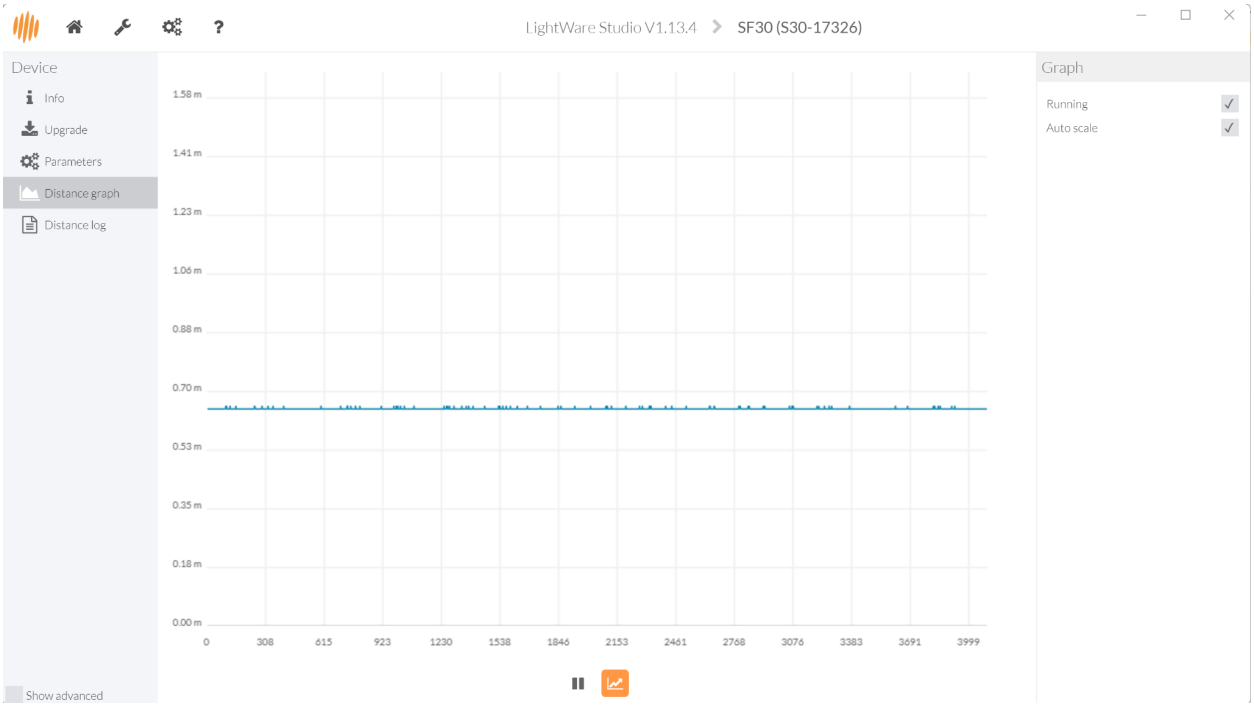
5 Experimental results (5-10 pages; 20 points)

- **A description of the tests you performed to validate the system**

Our approach for design validation was to first test the distance measurements of the SF30C altimeter as if we were testing the OpenTOFLidar PCB. This meant validating the physical distances between the device and the object against the distance recorded from the electronic components on the altimeter device. Then, we would test the speed of the motor through the Arduino program. Specifically, we would start at slow speeds and then slowly increase the speed of the motor. We would see if increasing the speed through the servo module's knob would increase the RPS speed value recorded by the Arduino program.

- **Numerical results of these tests**

1) Validating distance measurements

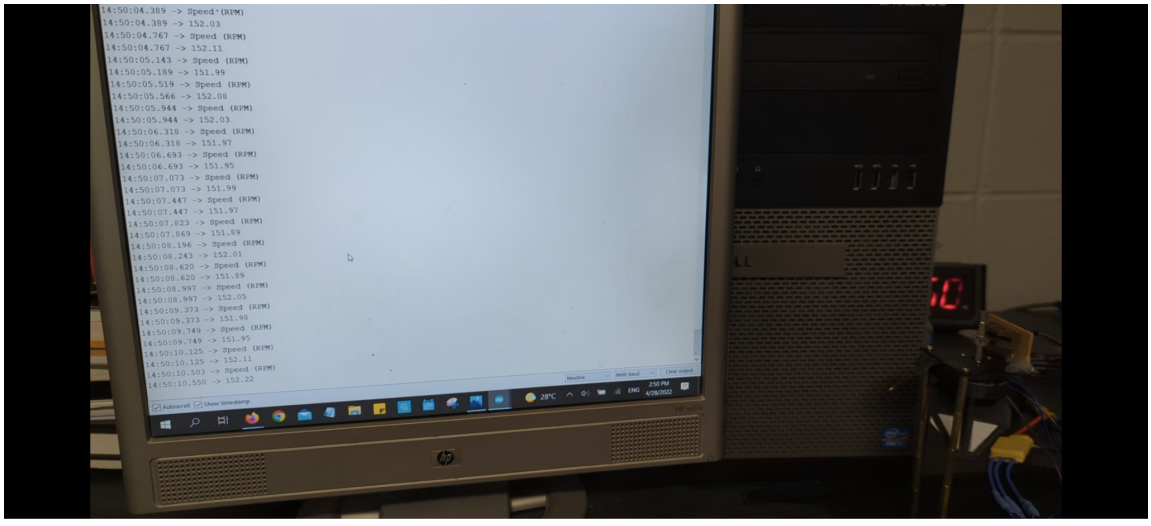


Validated physical distance measurements against distance readings of the SF30C altimeter (through Lightware Studio as shown above). The purpose of this validation was to test the SF30C altimeter's distance readings as if we were testing the OpenTOFLidar PCB's distance readings.

We validated various sizes of distances readings of the SF30C that was possible in the room where we were working, as shown below:

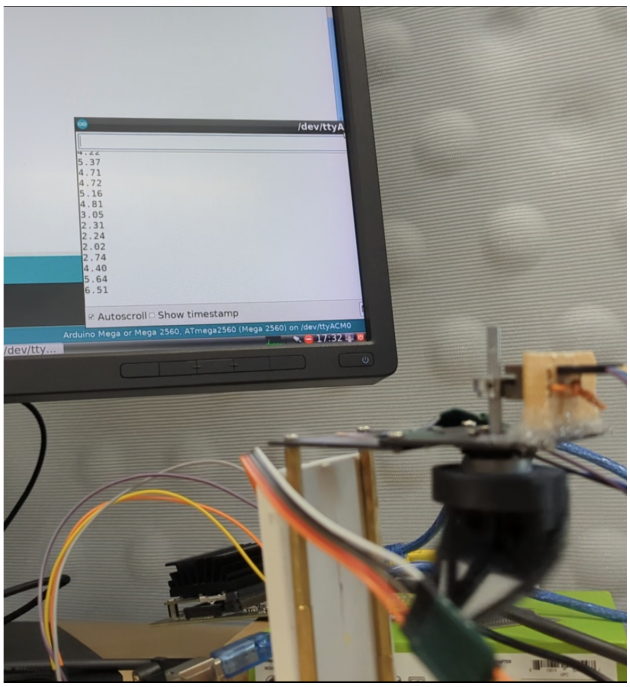
Physical Distance (m)	Lightware Studio for SF30(m)
0.66	0.66
0.96	1.04
0.36	0.38
0.53	0.59
0.61	0.68
0.25	0.27
3.72	4.3
0.63	0.62

2) First, I tested the Revolution per minute (RPM) motor speed:



As shown above, the arduino program reads the signal from the encoder attached near the encoder disk screwed onto the shaft of the motor. The Arduino program then computes the RPM speed based on the time it takes for channel A or B to output 500 cycles.I

validated the RPM speed by checking if it was mostly consistent and that it increased when I turned up the knob on the servo test module.



We needed the revolutions per second (RPS) speed values to compute the num_readings variable value in the ROS python program which displays the point cloud map through RVIZ. So, I then tested the motor again with their RPS readings in the same manner I tested the motor for the RPM readings I described above.

- **A thorough analysis and discussion of these results**

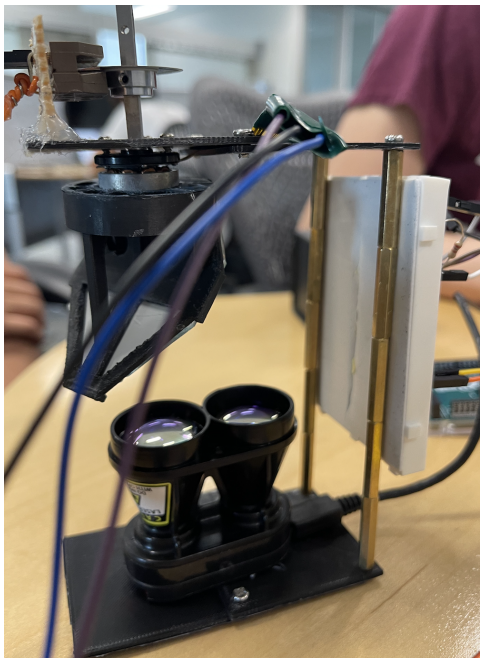
While it can be said validating the distance measurements of the SF30C altimeter was redundant since it is a commercial product, we believe that this validation is still within reason because there is always a chance an electronic system may be faulty. However, in this case, we determined that our SF30C altimeter device was not faulty as it always gave us mostly accurate readings. The distance measurements of the SF30C were always within +/- 0.5m of the physical measurements

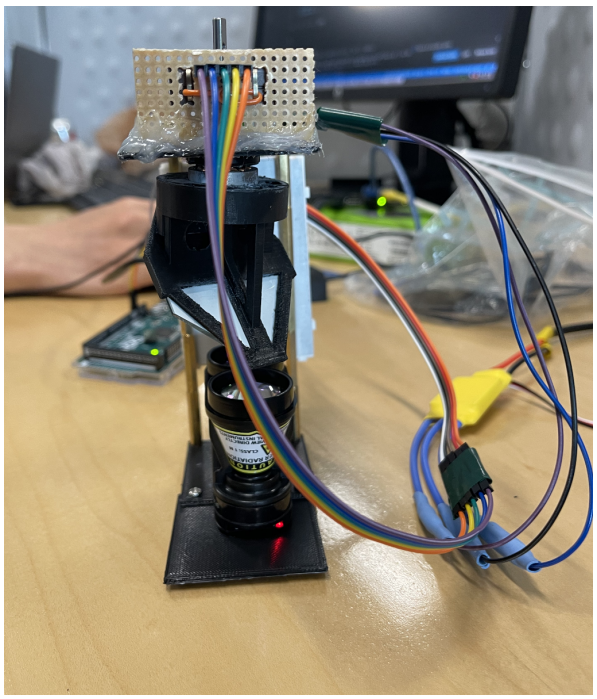
recorded with only 1 or 2 measurements with an error of 0.5m. So, it can be concluded that the SF30C distance sensor is mostly very accurate.

The speed of the motor was changed through turning the knob on the servo test module connected to the Electronic Speed Controller (ESC) which is connected to the motor. The motor speed was read from the encoder system placed near the motor through an Arduino program that I designed. At slow speeds (1 - 2 Revolutions Per Second), we were able to verify the RPS of the motor displayed on the Arduino serial monitor with the RPS recorded by our eyes. To test high speeds of the motor, we qualitatively analyzed the serial monitor to see whether the arduino program was able to accurately record the increase in RPS speed of the motor from its initial lower speed. We found that the Arduino program worked as planned by displaying increased RPS values when we increased the knob on the servo module and decreased RPS speed values when we decreased the knob on the servo module.

The readings of the motor were mostly uniform which indicate that the motor speed is mostly consistent at any position of the knob on the servo test module component. We were not able to validate the encoder's ability to record the motor speed at high speeds very quantitatively. In the future, I suggest modifying the Arduino program to validate the encoder's motor speed recorded from channel A or B with the encoder's recorded motor speed recorded from channel Index (I) through the Arduino "Encoder.ino" program.

- **A collection of pictures of your final system**





6 User's Manuals (5-10 pages; 20 points)

(1) Hardware:

I. Altium Designer

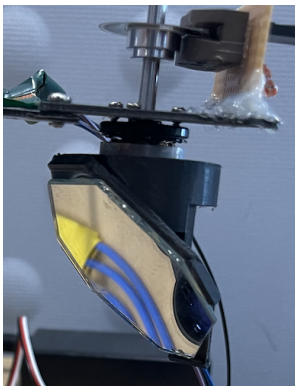
As a graduate or undergraduate student, one has free access to the Altium Designer through their student designer license. Therefore, you can use Altium designer for free as a student. To get used to Altium designer, I recommend Robert Feranec's courses on udemy regarding getting started with Altium designer and PCB design with Altium . For instance, I was able to follow along with getting started with Altium designer class in Altium Designer and finish this over the 1 week of the spring break. I was also able to watch most of the tutorials on the PCB design with Altium class; however, I did not have time to follow along in Altium designer. I thought his language and instructions were mostly clear most of the time.

II. Encoder

To choose encoders, first determine what type of encoder you are looking for. I was looking for optical rotary encoders. So, I researched this topic and found Broadcom encoders to best meet my requirements. Moreover, Broadcom's rotary encoders were sold on digikey and mouser electronics. After learning the shaft diameter of the motor we are using was 4mm, I bought the appropriate encoder and encoder disk that was available. Due to possible supply chain shortages of these incremental optical rotary encoders and their disks, I could only find certain encoder and encoder disk pairs which could fit our motor. So, I was not too focused on the resolution of the encoder system. However, if one wants to get more accurate speed, a higher resolution encoder would be useful if it is available. As a result, I chose to get the 500 CPR resolution encoder that was available. This means that after 500 cycles of the square wave from channel A or B the encoder disk would complete one revolution. The encoder stood up near the motor by being threaded to a piece of phenolic protoboard that was cut to fit the top of the motorized spinning mirror platform. The piece of phenolic protoboard with the encoder's pins threaded through, was hot glued to the top of the motorized spinning mirror platform such that the encoder disk could go through the opening of the encoder (between the photodiode and the laser) appropriately.

III. Mirror cutter

In case the mirror on the mirror holder breaks, we have an extra mirror piece. However, it is necessary to first cut this mirror piece to fit the current mirror holder. So, one should use the high quality glass cutter we bought from amazon to cut the mirror to fit the mirror holder as instructed here . Currently the shape the mirror holder supports is not a rectangle as seen below:



So, alternatively, one can 3d print a mirror holder to hold the rectangular mirror as it is.

IV. 3D Printing

Our product comes with the printed out design, but if additional prints are required, the stl files are attached and can be reprinted as needed.

The FEDC allows for printing of the files, you can go to this link in order to submit a request to print out the files. The FEDC gives a budget to those who have FEDC access, which can be applied for in this link.

V. Motor and ESC

The motor comes with a ribbon cable that needs to have wires soldered to it, these wires will need to the ESC. The order of the outside wires does not matter but the center pin of the motor and the center pin of the ESC should be connected. The ESC then will need to be attached to the power supply using the lead black and red wires and the PWM cable(skinny 3 headed cable) will either be attached to the arduino or the Servo control module, depending on what configuration the user would prefer. If using the arduino code to run the motor, the only arduino code that needs to be

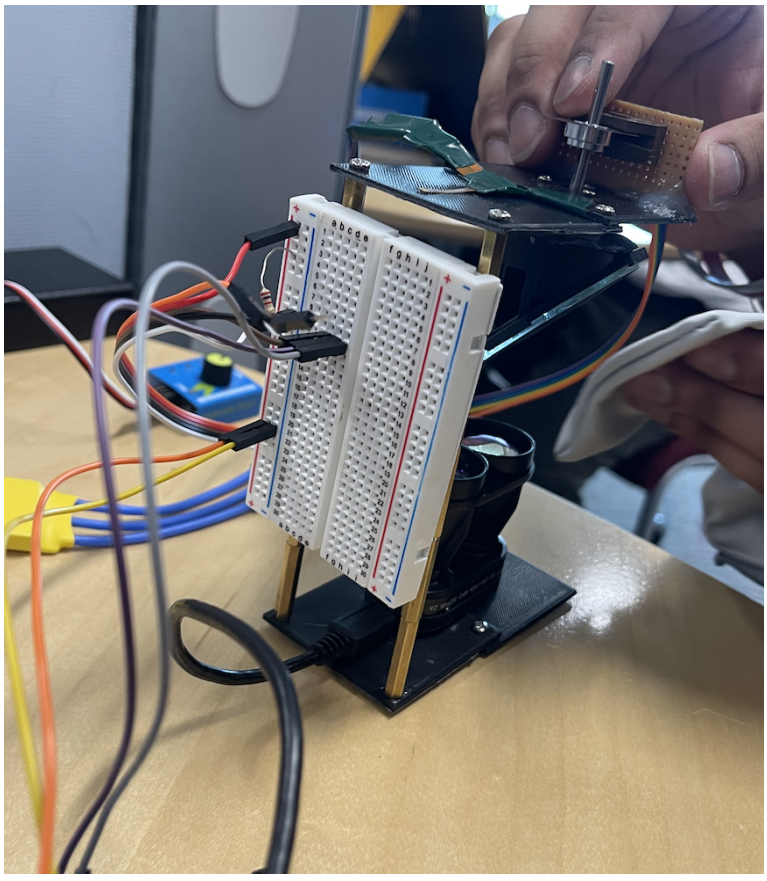
modified is the servo.write line in the very end because it can be changed to a different speed if desired by the operator. Pattern can also be set up for a non-uniform speed pattern for the motor.

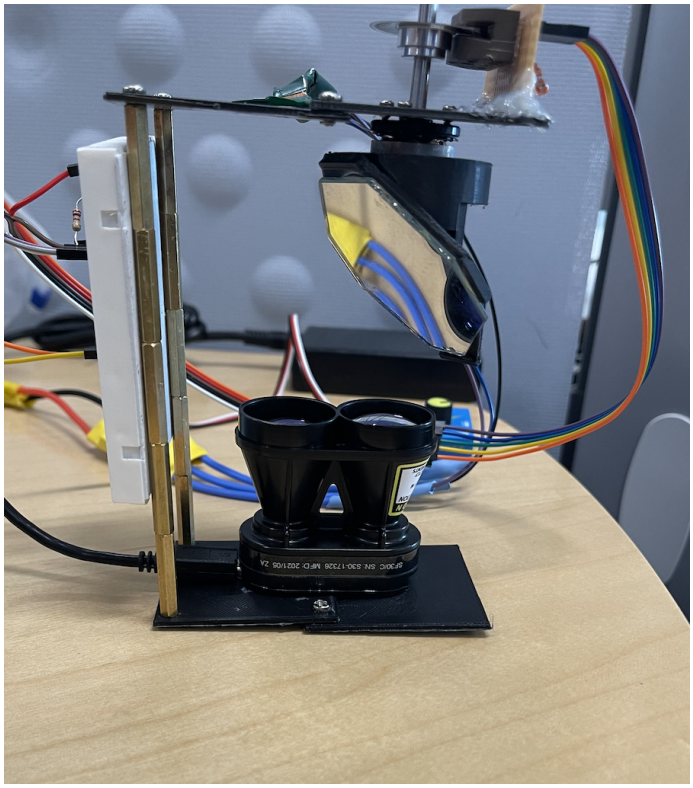
VI. Power Supply

The unit comes with both a battery as well as a wall adapter as well as adapters for each that would convert the plug into two leads that can be soldered or connected to the two larger red and black wires that are part of the ESC. The wall adapter and battery both accomplish the same mission of essentially providing a 12 volt source, and so other ways of getting to these 12 volts such as using two analog discoveries are also acceptable if connected properly to the ESC.

VII. Mechanical

The 3d printed parts will need to be screwed together as per the holes given, with 5 male to female hex standoffs to separate the top and bottom holding components, please see the diagram in the bottom that shows how everything is put together.





(2) Software:

I. ROS - The robot operating system was an integral part of our process. In order to get it up and running from our system. We did our setup on a Ubuntu linux system running on our virtual machine and also one running on our Jetson microcomputer. To begin with we first set our Ubuntu repository settings to allow for restricted, universe and multiverse.

Then we went ahead and set up the permissions to allow for software packages from packages.ros.org. Using the command

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Next we need to set up and apt install

```
sudo apt install curl # if you haven't already installed curl
```

```
sudo apt update
```

Set up our keys

```
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

Find our specific package

apt search ros-melodic

Source ROS to environment

echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc

source ~/.bashrc

Build Dependencies

sudo apt install python-rosdep python-rosinstall python -rosinstall-generator python-wstool build-essential

Initialize Rosdep

sudo apt install python-rosdep

sudo rosdep init

rosdep update

II. Environment setup

III. VNC

IV. Python3

V. UART Permissions

In order to properly receive data from various ports you need to configure the system to allow for various USB ports. In order to do this we use a command called:

sudo chmod

For our specific device, the name was ttyUSB0 file. From here we would use the chmod command to change the specific file permissions. We can view all the usb-type devices using a commands called

lsusb

VI. RVIZ

The program we used to view the readings from the messages was RVIZ. Once we had the LIDAR terminal setup and communicating our data into the main program, it was fairly simple to set up.

First we would need to set up the frames. The first frame to set up is the global frame. This would be set to a laser frame. This is used as the point of reference for the points to be placed around. Second we would select a topic. In our case the topic would be our laser message. Topic refers to the collection of points that the program is concerned with looking at.

In order to configure the display check the third 'box from the top. Here we can specify items like size of individual points as well as things like the shape of points. We can have options like flat squares, points and lines. Then we can also specify configurations such as how the points appear and disappear here as well.

VII. Lightware Studio

The specific hardware we used was the SF-30C from lightware studios. In order to specific hardware components and parameters we used a program called lightware studio provided by the main company. In this we could perform various tasks. For instance we could output the specific readings and then plot them over a graph. In this graph we had options to change axes to things such as voltage and time.

In addition to this we could also make specifications on things such as the sampling rate, distance, maximum reading distance and other parameters set up in this system.

(3) Operation Instructions for the Average User

1) Charge the 12V battery pack to the wall through the charger cable until the light on the charger turns green.
2) Connect the 12V battery pack to the Electronic Speed Controller, which is a long wire connected to the motor.
3) Connect the servo module to the Electronic Speed Controller.
4) Connect the encoder jumper cables for channel A of the encoder to a 2.7KOhm pullup resistor on the breadboard according to page 7 of this documentation for the encoder.
5) Connect two jumper wires from the channel A signal on the breadboard to pins 2 and 3 on the Arduino.
6) Connect VDD from the encoder to VDD on the breadboard power rail. Connect 5V from the Arduino to VDD on the breadboard power rail.
7) Connect GND from the encoder to GND on the breadboard power rail. Connect GND from the Arduino to GND on the breadboard power rail.
8) Connect the Arduino to a Nvidia Jetson Nano.

9) Startup the Arduino IDE on the Nvidia Jetson Nano or another microcomputer device, and run the Encoder.ino program located in the Backup_firmware folder of our OpenSourceLidar github.
10) Connect the SF30C to the Jetson and use Lightware Studio to configure the SF30C's sample rate to be 625 readings per second and over USB. Then close the light ware studio and place the SF30C under the motorized spinning mirror platform.
11) Run the ROS python program in the Backup_firmware folder of our OpenSourceLidar github repository (after installing all of the required packages listed in the user manual).

7 Course debriefing (2-4 pages; 10 points)

Provide a thorough discussion of your *team management* style. If you were to do the project again, what would you do the same, what would you do differently?

We had routine meetings 3 times a week and made sure to communicate with each other as per what we were doing and make sure everyone was on the same page. I feel as if this routine schedule was very important to the success of our project and if I were to do this project again I for sure would continue to keep this touine and I feel that it was very important. The one thing I would change would be the fact that we once met on very short notice and that caused some issues with everyone participating in that meeting, so I felt that providing a 24 hour notice before meetings would be helpful. I noticed that 24 hour notices were given before most subsequent meetings.

Are there any particular *safety* and/or *ethical* concerns with your product(s)? What steps did your group take to ensure these concerns were addressed? Are there any additional steps you would have taken if you were to do the project again?

Since we were working with a lidar that contains a laser inside, one of our major safety concerns is avoiding the potential danger of the laser beam harming our eyes. To ensure this concern is addressed, we took extra precautions to make sure that the lidar was not pointing directly at anyone when it is turned on. We also purchased safety glasses that could protect the eyes from the laser beam. With these two considerations in mind, we were able to avoid any safety incidents throughout the project development. We think we have taken sufficient steps to address the safety concerns in this project, and no additional steps are needed.

Did you test your product(s)? Do they work as proposed? Can you think of any relevant situations in which you haven't tested your product(s)? If you were to do this project again, what additional *verification* and *testing* procedures might you add?

The motorized mirror platform reflected the SF30's laser readings and we were able to get readings from the encoder. Finally, we were able to get a mostly accurate point cloud map through RVIZ from our ROS python program in very close to real-time as shown in our detailed 30 second demo video.

8 Budget (2-4 pages; 5 points)

The parts were a bit on the expensive side due to the fact that we had to import a lot of them from China, and this ended up being a problem in terms of shipping time and a covid outbreak causing production to be delayed. We did have to buy some components for our backup plan, which ended up being more expensive than the original components, but this was only the case because of the issues mentioned above. The most expensive time that caused our budget to be about $300 over was the SF30C altimeter device which cost about $299.

Compared to the proposed budget, we did not plan to buy the SF30C altimeter, the mirror cutter, more hex standoffs, a ratchet screwdriver, and a battery pack. We also did not plan to go over the budget by much. However, our total expenditures were $967.86 which was $367.86 over our budget of $600. The budget is shown below:

No.	Name	Retail Price $	Description	Taobao Price ¥	Note	Taobao URL	URL
12	Photodiode Lens	$4.48	Lens, CS mount, 25mm focal length, F1.2	28			https://aliexpress.com/item/32303375838.html
13	CS lens holder	$0.24	Holder for CS lens	1.5		【淘宝】https://m.tb.cn/h.fNTqmms?tk=SyBE23GHtNo「CS金属镜头底座20mm安装孔距安防监控摄像机配件CS镜头座」点击链接直接打开	https://aliexpress.com/item/32815023477.html
14	BLDC motor	$0.64	BLDC, outrunner, 20mm diameter, KV > 200	4		【淘宝】https://m.tb.cn/h.fm6chvO?tk=733o23GF56t「【仓兴达】高速云台电机 2010 无刷电机 KV值2500 滚珠轴承」点击链接直接打开	https://aliexpress.com/item/4000323714513.html
15	Scanning Mirror	$1.60	Customized shape, Alum, front surface mirror (12$ for 4 mirrors)	10		【淘宝】https://m.tb.cn/h.fOUAm6T?tk=S1yH23Gw2a1「无框大块钢化玻璃镜子定制切割鞋柜玻璃镜面定做防爆贴墙小镜子片」点击链接直接打开	https://aliexpress.com/item/4000246071735.html
		$4.80	soft mirror	30		【淘宝】https://m.tb.cn/h.fl63whX?tk=HLhs23Gx2ry「软镜子 超薄手工镜子diy材料镜子贴安全不碎非玻璃软镜子可剪切」	

16	Stand-offs	$1.28	25mm M3 Hex Stand-off x 8, price for 20pcs	8	10 male	【淘宝】https://m.tb.cn/h.fl6V45D?tk=5gSm23GBfNx「六角铜柱 单头铜螺柱 25mm铜柱 隔离机箱铜柱 铜柱螺母 铜柱M3」 点击链接直接打开	https://www.aliexpress.com/item/4000449592477.html
		$0.24		1.5	5 female	19微子那可地有一小多着这为信 https://m.tb.cn/h.fNTG5K5 ~这个商品很不错，我还有一张【3】元，限量优惠l券给你，快来看看！【(5个)M3*25mm 双通六角铜柱 电脑机箱主板螺丝母杆钉子紧固隔离】	
	Stand-offs	$0.00	10mm M3 Hex Stand off				
17	Laser lens	$4.80	25mm M12 F2.0 lens (Free delivery)	30		【淘宝】https://m.tb.cn/h.fNTuGhR?tk=SI0h23uZAjO「500万高清镜头 4mm6mm8mm12mm16mm25mm1/2.5英寸M12口监控器材配件」 点击链接直接打开	https://aliexpress.com/item/32913215547.html
		$2.96		18.5		【淘宝】https://m.tb.cn/h.fl6U6sX?tk=zIgO23u0t1z「3MP 25mm小镜头300万25MM数字高清镜头M12单扳机25毫米远焦镜头」 点击链接直接打开	
		$2.88		18		【淘宝】https://m.tb.cn/h.fm64kBz?tk=mevI23ub0ty「25mm高清监控摄像机定焦镜头单板机红外镜头长焦镜头m12镜头」	
18	Laser lens holder	$0.16	Plastic holder, 10mm height, 20mm hole distance (price for 10pcs)	1		【淘宝】https://m.tb.cn/h.fNTvrDI?tk=J19J23u1Ypu「M12镜头座20mm安装孔距7 8 10 14 15mm高塑钢金属塑料小镜头座」 点击链接直接打开	https://aliexpress.com/item/32953675441.html
		$0.16		1		【淘宝】https://m.tb.cn/h.fNTvrDI?tk=J19J23u1Ypu「M12镜头座20mm安装孔距7 8 10 14 15mm高塑钢金属塑料小镜头座」	
19	M12 lens adapter	$0.19	Price for 1pcs	1.2		【淘宝】https://m.tb.cn/h.fm6f8Fw?tk=UMC623u1y79「小镜头增高环 延长环 M12小镜头加高座 变长环 增高座总长10MM」 点击链接直接打开	https://aliexpress.com/item/32962439240.html
		$0.48		3			
		$0.00				【淘宝】https://m.tb.cn/h.fm65WEC?tk=6qyr23u1aRV「M12*0.5镜透增高环 转接圈 延长圈 镜头座加高神器」 点击链接直接打开	
		$0.00					
		$0.00	**Note - this total price is calculated without delivery price**				

		$0.00					
		$0.00					
_1_	[AD500-9 TO52S1F2]	$0.00	Chinese Avalanche photodiode (NOT proved but similar to AD500-8), TO-52 package, interated interference filter	0	with filter	【淘宝】https://m.tb.cn/h.fm65y0n?tk=oIHy23uX9b7「AD500-9 TO52S1F2 APD雪崩光电二极管 905nm 带滤光片 全新原装」 点击链接直接打开	https://aliexpress.com/item/32735300921.html
		$0.00		0	without filter		
_1_	[MTAPD-07-013]	$0.00	Original Avalanche photodiode (proved), 3-LCC package	0		【淘宝】https://m.tb.cn/h.fNTDkj9?tk=TCMj23u279q「MTAPD-07-013 [M] SENSOR PHOTODIODE 905NM LCC-3」	https://www.digikey.com/products/en?keywords=MTAPD-07-013
	Interference Filter	$0.00	Optional Interference Filter, 905 +-10nm				https://www.ebay.com/itm/Optical-Filter-905nm-band-pass-filter-895nm-915nm-Dia-11-5mm/262976990433
	Interference Filter	$0.00	Optional Interference Filter, 905 +-5nm, Free delivery	0	diameter 11.5mm, thickness 1mm	【淘宝】https://m.tb.cn/h.fNTCf51?tk=lSQX23ud7kq「905nm窄带滤光片滤波镜片 微光夜视红外光通光镜面玻璃材质可定制」 点击链接直接打开	https://aliexpress.com/item/32963619780.html
		$7.51		46.96	x2	【淘宝】https://m.tb.cn/h.fl6Sey5?tk=zWB723u3emY「窄带滤光片785nm 808nm 830nm 850nm940nm980nm窄波段带通滤光片」 点击链接直接打开	
		$0.00					
	Optical Encoder Disk	$55.19	HEDS-5140 #A11	newark.com			https://www.newark.com/broadcom/heds-5140-a11/code-wheel-500ppr-3ch-4mm/dp/58Y4782?CMP=AFC-OP
	Optical Encoder	$54.00	HEDS-9140 #A00	93	x2		https://www.digikey.com/en/products/detail/broadcom-limited/HEDS-9140-A00/1001290
	Laser Safety Glasses	$16.86	905nm protect eyes. 7 for everyone	105.4	x7		
	PCB Assembly	$30.00	PCBway				
	PCB Manufacturing	$49.00	PCBway				
	PCB Components	$170.00	PCBway				

	Perfboard (ensure material is Fr-2 or Fr-4 not metal)https://learn.adafruit.com/collins-lab-breadboards-and-perfboards/learn-more	$14.99	min. dimensions (W by L) - 1" by 2" or 4cm by 6cm				https://www.amazon.com/AiTrip-Perfboard-Breadboard-Prototyping-Electronic/dp/B07XYL451Y/ref=sr_1_4?crid=UP6CPVO17KSL&keywords=perfboard&qid=1646026983&sprefix=perfboar%2Caps%2C116&sr=8-4&th=1
	breadboard jumper wires	$7.99	10 breadboard jumper wire cables of female-female, 10 male-female, and 10-				https://www.amazon.com/Elegoo-EL-CP-004-Multicolored-Breadboard-arduino/dp/B01EV70C78/ref=sr_1_2_sspa?crid=38IJYJ9RHIEDQ&keywords=breadboard+wires&qid=1646156803&sprefix=breadboard+wire%2Caps%2C112&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFUQTJRM1k1T1Y5OUImZW5jcnlwdGVkSWQ9OTA2NzMyMjJSRk9ZNkhLQ0FVQk8mZW5jcnlwdGVkQWRJZD1BMDUzMTg0NzNMTjhVQ1ZVMUg0OEQmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl
	HiLetgo CP2102 USB 2.0 to TTL Module Serial Converter Adapter Module USB to TTL Downloader	$0.00	Not necessary for now.				https://www.amazon.com/IZOKEE-CP2102-Converter-Adapter-Downloader/dp/B07D6LLX19/ref=sr_1_2?crid=ZE3BZNLLWZ75&keywords=HiLetgo+CP2102+USB+2.0+to+TTL+Module+Serial+Converter+Adapter+Module+USB+to+TTL+Downloader&qid=1646156924&sprefix=hiletgo+cp2102+usb+2.0+to+ttl+module+serial+converter+adapter+module+usb+to+ttl+downloader%2Caps%2C108&sr=8-2
	Orange Pi	$39.44	https://www.taobao.com/list/item/666143511704.htm?spm=a21wu.10013406.taglist-content.2				
	Glass cutter	$12.96					
	Glass cutter oil	$9.98					
	M3 Screws	$6.48					
	PWM Driver	$13.99					https://www.amazon.com/Acxico-High-Power-Brushless-Controller-Regulator/dp/B0863HZN9L
	Step up Converter	$10.89					https://www.amazon.com/Aceirmc-Current-Converter-Adjustable-Regulator/dp/B082XQC2DS?th=1
	SD Card	$7.98					https://www.walmart.com/ip/onn-128GB-Class-10-U3-V30-MicroSDXC-Flash-Memory-Card-with-Adapter/772835319
	Power Supply	$11.99					https://www.amazon.com/Power-Supply-Lights-Adapter-Transformer/dp/B07GFFG1BQ?th=1
	Male to Male plug Adapter	$6.99					https://www.amazon.com/RLECS-5-5x2-1mm-Connector-Female-Adapter/dp/B07XDTDMVJ

	DC Pigtails Cable	$10.81					https://www.amazon.com/Pigtails-Female-Connector-Pigtail-Security/dp/B08PYWN3T7
	Rechargable Battery Pack	$28.99					https://www.amazon.com/TalentCell-Rechargeable-3000mAh-Lithium-External/dp/B01M7Z9Z1N
	Jetson Hat	$16.99					https://www.amazon.com/MakerFocus-Resolution-Interface-Compatible-Raspberry/dp/B07H9ZTWNC
	Ratchet	$16.99					https://www.amazon.com/MulWark-Profile-Ratchet-Quarters-Screwdriver/dp/B07D4DZB1T
	Arduino Mega	$35.93					https://www.amazon.com/ARDUINO-MEGA-2560-REV3-A000067/dp/B0046AMGW0/ref=sr_1_3?keywords=arduino+mega&qid=1650982728&sr=8-3
	Hex Standoffs	$7.99					https://www.amazon.com/Hxchen-Female-Hexagon-Standoff-Pillars/dp/B07WKL7TF1?th=1

9 Appendices

Constraints and Engineering Standards Index
For

Project Title: OpenSourceLidar
Team Members: Aamhish Rao, Paul Roy, Yi-Le (Allen) Chen, Gaurav Bhalla

CONSTRAINT	PAGE	PAGE	PAGE	PAGE
Economic	4	37		
Environmental	4			
Social	5			
Political	5			
Ethical	5			
Health and Safety	4			
Manufacturability Constructability	4			

ENGINEERING STANDARD	PAGE	PAGE	PAGE	PAGE
1.ASCII standard	7			
2. Standard Baud Rate	17			
3. Imperial units	18			
4. Serial Peripheral Interface	14			
5. USB	11	7	35	