3D Environment Modeler

Daniel Batista, Aaron Coville, Matthew Cummings, Sean Huskey

CREOL and Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — A 3D environment modeler was created using light detection and ranging (LIDAR) to render an image relative to the surroundings of the device. The device implemented LIDAR by using sequential equivalent time sampling (SETS). SETS effectively slowed down the signals such that they could be processed by a microcontroller. After processing, the data was encoded and sent over Bluetooth to and operators computer where it was rendered as a 3D point cloud. The laser beam was directed by rotating mirrors to capture a full field of view. This prototype has the potential to be integrated in multiple applications requiring a cheaper and smaller LIDAR system.**

***Index Terms* — Data communication, Free-space optical, Optical imaging, Optical propagation, Optical Pulses, Optical sensors.**

I. INTRODUCTION

Current LIDAR systems are expensive and no known small-scale LIDAR system exists in the market that can image its surroundings in a 3-dimensional manner. Ultrasonic detection systems cannot reach a precision that LIDAR can provide and this system may be integrated in other systems that require a view relative to the device. The prototype designed in this paper outlined a working model that may be further improved for future systems.

Figure 1 shows the hardware and optical implementation of the device. The detector and lens were attached to a printed circuit board (PCB) and lie on the same plane. The rest of the system is aligned and set up relative to the location of the PCB.

To view around the device, a slip ring was used. A slip ring allows current to flow from a static platform to a rotating platform. The slip ring chosen for this system contains a through-hole in the middle which allows light to pass both into and out of the system. Gears were implemented to turn 360° in azimuth. A gear was attached to the slip ring while another gear was attached to a stepper motor to the right of the slip ring as shown in Figure 1. The gears were driven by stepper motors which provide rotation. A mirror on top of the slip ring gear was attached to another stepper motor to turn 90° to view elevation. The mirror attached to the motor reflected the outgoing beam as well as reflected the incoming signal to locate a point in the environment.

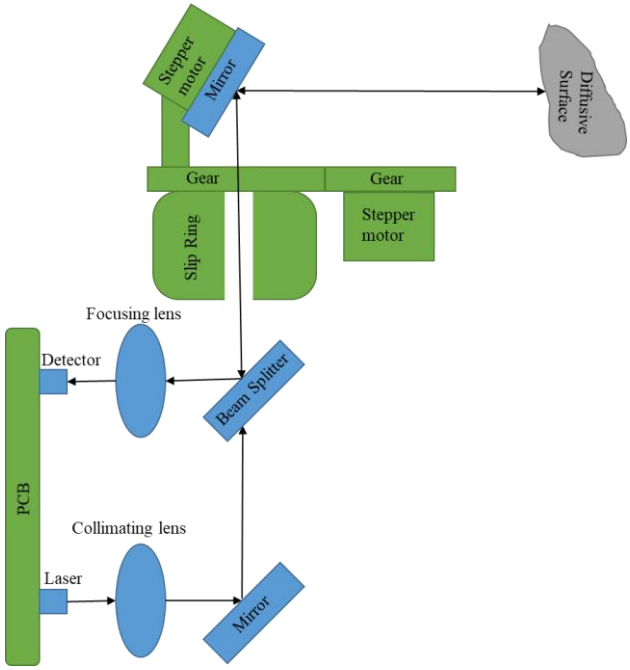

Fig. 1. The design of the optical and mechanical components of the system.

II. LIDAR THEORY

The methods of LIDAR that were considered are discussed in this section. The theory behind the specific method chosen is further expanded upon along with specifics of the implementation.

A. Considered Methods

Three methods of range finding were explored for this project:

- Optical Triangulation
- Continuous-wave (CW) Intensity Modulation Phase Difference Time of Flight
- Pulse-modulated Time Difference Time of Flight

Optical Triangulation works by shooting a beam of light and using its location on a sensor when focused through a lens to triangulate the distance it must have traveled. An advantage to this method is that high speed electronics are not required; however, it's range is limited to less than a meter and this limitation ruled triangulation out of consideration.

Continuous-wave Intensity Modulated Phase Difference Time of Flight works by comparing the phase difference between incoming and outgoing intensity-modulated CW laser light. A disadvantage to the technique is that the phase

wraps and this creates a limitation on the distance that can be measured which is inversely proportional to the resolution. In other words, the larger a distance is before wrapping, the less accurately that distance can be calculated. This problem, coupled with the need for expensive high-speed electronics, resulted in this method not being selected.

Pulse-modulated Time Difference Time of Flight works by shooting a pulse of laser light and seeing how long it takes to see the scattered light come back to a detector. The range is only limited by how powerful of a laser pulse that is sent out and how slowly the pulse diverges. This technique was selected because it is reliable in the range of 1 m to 10 m. High-speed electronics are still necessary for this method to sample fast laser pulses. The high-speed electronics constraint was resolved by utilizing an open source LIDAR design using sequential equivalent time sampling (SETS) to effectively slow down the fast signals.

B. Sequential Equivalent Time Sampling

The SETS technique works on the assumption that the signal being sampled repeats at a stable frequency and stays a consistent shape. If these assumptions are correct, instead of sampling a single pulse many times, SETS samples each pulse a single time. The sampling frequency is slightly slower than the frequency of the signal so that each sample happens slightly later in the pulse than the previous one. SETS is able to accurately reconstruct what a single pulse would look like but at a much slower rate.

The open source SETS design used is called the Open Source Laser Range Finder (OSLRF) [1] and has two main clock signal pins to be utilized by a microcontroller (sync and convert), as well an input for controlling resolution (control), and two outputs for the laser outgoing and incoming pulses (zero and return).

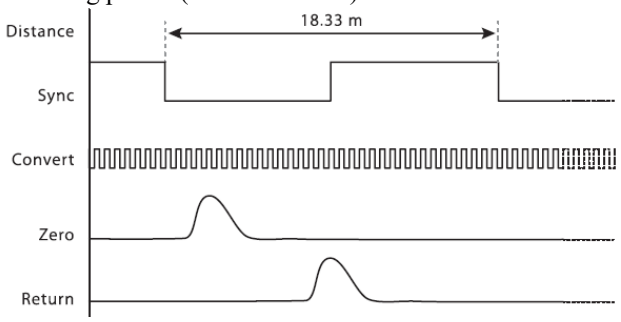

Fig. 2. The different OSLRF signals and their relation to each other in time.

The convert and sync pins are meant to be used to trigger a microcontroller on the falling edge of their signals. Sync is the master timing clock for the OSLRF; its falling edge denotes the beginning of the distance acquisition. Its frequency ranges from 3 Hz to 50 Hz depending on the voltage applied to the control pin. The convert signal is a 32 kHz signal that denotes when the microcontroller can read a sample point on the zero and return pins. The samples are created at constant speed, but the sync frequency is variable, so the effect of changing sync is to fit more sample points into a single distance acquisition. This decreases the smallest unit of time that can be resolved between the peak of the outgoing and incoming pulses, increasing resolution. The trade-off is that the overall distance acquisition becomes slower since sync is the master clock on the board.

The actual distance between the device and the target is calculated on the microcontroller. It does so by counting the number of samples between key points on the zero and return signals. The OSLRF documentation recommends either using the peak or the midpoint of the two half-maximum points as the reference for calculation. The equation for calculating distance given the signals from the board is

$$d = \frac{N_{return} - N_{zero}}{N_{total}}(18.33\ m) \qquad \textbf{(1)}$$

Where d is the distance between the laser and the detector, N_{return} and N_{zero} are the positions of the reference points of the return signal and zero signal respectively, and N_{total} is the total number of sample points in that sync period.

C. Laser Pulsing

To use Pulse-modulated Time Difference Time of Flight for LIDAR, the laser in use must be pulsed. The laser pulse was achieved through pulsing a current to drive the laser to turn on and off. The electrical pulse was designed by utilizing an avalanche transistor. When a voltage higher than the breakdown voltage of the transistor was applied the signal passes through. The result was a laser pulse with a repetition rate of 62.5 µs and a width of 30 ns. In general, the shorter the laser pulse, the better the resolution. A higher resolution can be attained because shorter pulses are more distinguishable in time.

III. OPTICAL SYSTEM

This section discusses the components and the theory behind the optics of this project. The first subsection discusses correcting for parallax. Next, the components are presented. The final subsection addresses the limitations of the optical system.

A. Parallax Correction

Parallax is the movement of an image on the focal plane as the object moves. If not corrected for, this effect causes the LIDAR system to only be effective within a small range. Typically, this problem is resolved by reducing the distance between the laser and the detector. Alternatively,

the effect can be eliminated by having the image remain on the optical axis regardless of where the beam is directed by using a mirror and a beam splitter. The latter method was selected for this system.

To implement the selected correction, a mirror was placed at 45° in the collimated beam path along with a 50-50 beam splitter at 45° that was placed above that mirror aligned with the detector. The collimated beam reflects off the mirror, passes through the beam splitter, and passes through a slip ring before hitting the mirror that directs the beam around the room. The slip ring introduces loss in the returned signal by limiting how much light can pass from the beam-directing mirror down through the slip ring, as shown in Figure 3. This effect introduces a maximum distance between the detector lens and the beam splitter before the detector lens can no longer capture and focus all the light.

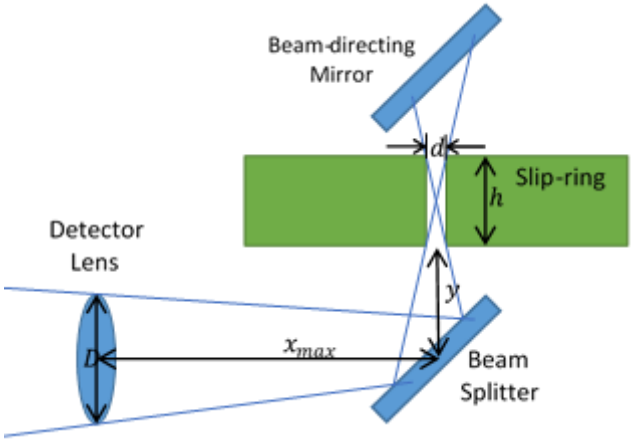

Fig. 3. This model allows a calculation to be made that determines the maximum distance between the beam splitter and the detector lens before light begins to be lost.

Using Figure 3, it can be determined that the maximum spacing between the beam splitter and the detector lens is

$$x_{max} = \frac{(D-d)h}{2d} - y \qquad \textbf{(2)}$$

Where x_{max} is the maximum distance between the detector lens and beam splitter, D is the detector lens diameter, d is the slip ring diameter, h is the slip ring height, and y is the distance from the bottom of the slip ring to the center of the beam splitter.

B. Optical Components

The optical alignment required the use of lenses to collimate the laser beam and focus the diffused beam onto the detector. Two bi convex ultraviolet (UV) fused silica lenses that used a broadband anti-reflection coating in the near infrared (IR) range were used. The lenses had an effective focal length of 25.4 mm with a 4.826 mm thickness. The lenses were placed in front of the PCB that contained the laser diode and the detector. To correctly align the system, a 3-axis translation stage was assembled for the detector lens. The three axes were necessary because the received signal was low in intensity and the detector area was 0.3 mm^2. Micrometer movement stages were used in the assembly and allowed the lens to be properly placed in the correct position relative to the detector. The laser diode has two different beam divergence angles being 9° in azimuth and 25° in elevation. A single axis translation stage was used to align the collimating lens. The beam then reflected off an aluminum mirror towards a 50/50 UV fused silica beam splitter. Another aluminum mirror was used directly above this system to reflect out to a diffusive surface.

C. Physical Limitations

In general, all the limitations that occur in this system result in a loss of irradiance on the detector which leads to a reduced maximum distance viewable by the system.

Equation 2 gives a maximum distance allowed between the beam splitter and detector lens before losses are incurred. The only variable that is free to change is y, but it has a minimum value determined by the physical parameters of the optical setup. The diameter of the lens, the diameter of the slip ring, and the height of the slip ring are constant. Using the minimum value for y as well as all the other fixed values, a maximum distance of -76.2 mm was calculated. This negative value implied that the lens would need to be located before the beam splitter to capture all the light from the slip ring. Therefore, the minimum y set by the optical system is too large, and there is no way to avoid losing some irradiance at the detector lens to this problem.

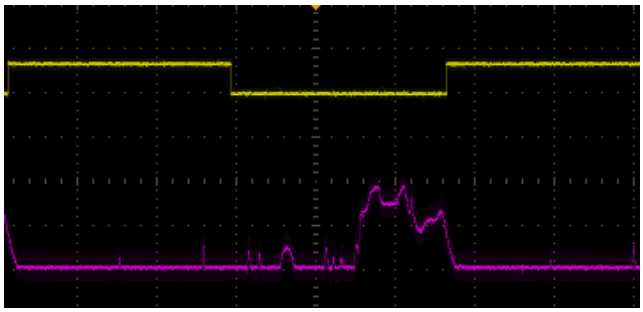

Fig. 4. Experimental evidence of false peaks in the return signal (purple) that are caused by the multiple reflections in the beam splitter are seen in this screenshot. The yellow signal is sync and is shown as a reference.

Another factor that contributes to irradiance reduction is the use of a beam splitter. The beam splitter being used is

a 50/50 splitter meaning that 50% of the original intensity is lost from the collimated beam before scattering off a diffusive surface. The returning irradiance is further cut down by an additional 50% when it is reflected by the beam splitter towards the detector. Additionally, the collimated beam passing through the second surface of the beam splitter causes reflection losses. The resulting series of reflections within the beam splitter creates signals that the detector picks up as false return signals. An example of this effect is shown in Figure 4.

The final source of loss identified in this system is related to how well the laser beam can be collimated. The beam is rectangular and diverges at different speeds along those two axes. Currently only a single lens is being used to collimate the beam, and with this single lens the beam can be collimated well enough to meet the 5 cm beam width at 9 m away recommended by the OSLRF documentation. A better collimated beam would allow for a further maximum detection distance.

IV. SYSTEM CONTROL

This section will discuss the control board of the system. The control board is an additional PCB which is responsible for reading the output signals of the OSLRF to calculate distance, controlling the direction of the laser, and communicating with the user application via Bluetooth. First, the PCB design will be discussed. Second, the interfacing between the OSLRF and the control board will be presented. Next, software design of the microcontroller will be discussed. Finally, the section concludes with an overview of the Bluetooth implementation.

A. Control Board Design

The control board has several responsibilities that need to be accounted for in its design. The first is regulating the power to the digital and analog components which make up the system. The OSLRF specifies that it requires 12 V to operate and will draw between 50 mA and 60 mA. The OSLRF's output signals are scaled to match a typical microcontroller which operates at 3.3 V. The chosen microcontroller is flexible with an acceptable operating voltage between 2.7 V and 5 V (limitations apply at lower voltages). The Bluetooth module only operates at 3.3 V. The stepper motor drivers, which will direct the laser, can operate between 5 V and 30 V.

With these specifications in mind, a 12 V supply is used to power the system. The control board passes the 12 V supply through a switching regulator which is configured to output 3.3 V as a supply voltage to the microcontroller and Bluetooth module with an efficiency of 90%[2]. Due to the microcontroller being powered by 3.3 V, all input and output signals are conveniently scaled to match. This simplifies the interfacing between the OSLRF, microcontroller, and Bluetooth module.

To save energy, the OSLRF does not receive power unless the device is scanning. To control the OSLRF's input voltage, a PNP bipolar function transistor coupled with an N-channel mosfet to create a high-side switch is used. The mosfet is controlled by the microcontroller at its gate to turn the OSLRF on when a scan begins and off when a scan completes. With the new switch and regulators in place, all components on the board receive their different operating voltages without risk of interfering with other components.

The next responsibility of the control board is to direct the laser during the scan. To provide enough granularity in direction, stepper motors are utilized. The stepper motors chosen have 200 steps per revolution, or 1.8° per step. The design specifications call for at least 1 distance measurement per degree in the horizontal plane. To meet the design specifications, a gearing ratio was used to increase the number of steps required per revolution to 360. If the user wants to increase the resolution further, the stepper drivers used to control the motors support microstepping. In microstepping, the motors can achieve up to 8 times the number of total steps per revolution for a final horizontal resolution of 2880 points. In the vertical plane, no gearing was used. Instead, the stepper motor controlling the vertical plane's resolution utilizes microstepping to provide 0.9° per step. As with the horizontal resolution, this can be increased further by adjusting the microstepping by a factor of 2 or 4.

Next, the control board must be able to read and process signals from the OSLRF. The details of the OSLRF's pins are discussed in section II.B. There, the convert signal was specified to be 32 kHz. The return and zero signals are read from analog pins. A microcontroller was chosen which can run its analog-to-digital converter (ADC) at 32 kHz along with a sufficiently fast master clock which allows for any additional processing between sampling the input signals. The ATMega 328p-pu microcontroller was chosen because, according to its datasheet, it can be configured to meet these demands [3]. It also has the added benefit of being used in the Arduino Uno, which made finding resources and documentation on how to use it properly easily accessible.

Finally, the control board must communicate with the user and the user's application. To communicate to the user, a simple power LED and state LED are used. When the system is waiting for a command from the user, the state LED is on. When the system resets, the state LED slowly blinks 3 times. When the system has a message that needs to be retrieved via Bluetooth, the state LED is off. To communicate using Bluetooth, the HC-05 Bluetooth module was selected. The module connects the user

application and the microcontroller, simplifying the process of sending and receiving messages wirelessly.

B. Embedded System Software Design

Connecting the two PCBs together revealed unexpected issues. The OSLRF's output pins were subject to more noise than what was expected. In addition to the noise on the signals, the sync signal has a non-negligible transient response which can trigger a false positive on sync. In other words, the sync pin can drop in voltage far enough before it achieves a steady state to cause the microcontroller to register that a new sync period has begun. In addition to the noise and transient response issues, the ADC and signal processing algorithm on the microcontroller were too slow in practice to capture every sample provided by the OSLRF.

These issues are addressed in both hardware and software. The software is discussed in section IV.C. To address the issues in hardware, low pass filters were applied between the OSLRF and the control board. The design of the low pass filters were limited by time constraints, thus only passive filters were used. Despite these limitations, the noise was reduced by more than 50%.

C. Microcontroller Software Design

The driving force behind the software design on the microcontroller is the requirement that the system be responsive to user commands at any moment during operation. To ensure the software is responsive, the microcontroller is treated as a state machine where all states connect to each other through a check for user input. Because this check for user input occurs during the transition between states, the system can respond to user input as quickly as it can finish its routine to change state. If user input is found, the target state can be changed to one that handles the new command. This way, the system can respond to user input at any time during operation.

The state of the device is implemented as an enumeration. States are handled by specific functions which are referred to as state handlers. The state of the device is then mapped to its corresponding state handler using the enumeration as an index into an array of function pointers. Every state handler returns the enumeration value of the next state that needs to be executed. As explained above, before accessing the next state handler, the state can be changed in response to any user input.

After adjusting for user input, the enumeration is used as an offset into an array storing all state handlers as presented in Figure 5. Whatever state handler is accessed is then immediately executed. This design has the added benefit of eliminating long chains of if; else if checks or burdensome switch case blocks that make code less maintainable.

```
State (*stateHandlers[])() = {
    /* references to state handlers */
}

void loop(){
    if(Serial.available()){
        /* Adjust state according to user input */
    }
    state = stateHandlers[state]();
}
```

Fig. 5. Example implementation of responding to user input between state transitions.

As introduced in the previous section on interfacing with the OSLRF, an issue was found where the processing algorithm is too slow and the system loses samples during sync periods. This is addressed by first configuring the ADC to operate at a speed which is more than double the rate of convert.

Then, the sampling state handler does not actually run the ADC or the processing algorithm for any sample which is known to be irrelevant to the distance calculation. To calculate the distance, the position of the return signal's peak is compared to the position of the zero signal's peak. To find these positions, the microcontroller samples to find where the strength of the return signal is half of its maximum. This point will occur both on the left and right sides of the maximum value. With the half-maximums found, the midpoint between them is computed and the resulting position is used as the signal's peak.

It is known that the peak of the return signal must occur after the peak of the zero signal. Because of this, all return signal samples up to the position which corresponds to the peak of the zero signal can be ignored. Additionally, once the position of the right-most half-maximum is found, no more sampling is required. By running the ADC and sample processing only on the samples which are relevant, the number of lost samples is reduced to a minimum. Assuming the lost samples are evenly distributed while the ADC is running, and the width of returned signal responses do not vary too much, inaccuracies in the distance calculation can be reduced with a linear calibration factor.

There was one final factor considered in the software design. Once any two half-maximum values are found, the microcontroller stops running the ADC. In testing, it was found that this approach is vulnerable to premature termination due to the presence of local maximums in the return signal. To get the algorithm to push past local maximums, a minimum distance between the left and right half-maximums is enforced. The minimum effective

distance between the left and right half-maximums was calibrated during testing.

D. Bluetooth

To render the data provided by the system, the data was sent via Bluetooth to an auxiliary PC application. From the embedded system's perspective, the Bluetooth module is simply a source of input signals and a destination for data. Beyond reading and writing the ingoing and outgoing pins (respectively), the system has no further decisive interaction with this module. The responsibility of establishing a connection, initiating communications, and querying for new data falls upon the PC application.

The Bluetooth is implemented by utilizing several C libraries developed under the BlueZ communication stack. The process is as follows: first, the PC application opens a socket to its local Bluetooth adapter. That socket is then used to query the local area for available devices to connect to, looking for the Bluetooth adapter it assumes exists on the embedded system. Once found, the two adapters are connected and the PC sets the embedded system's microcontroller into idle mode until a laser scan is requested. Once the scan starts, the embedded system writes data to the module as it is generated, and the PC periodically checks to see if a new point is available. Whenever new points are received, they are rendered, and this process continues until the scan is completed.

V. PC INTERFACE

The PC interface section describes the application designed to allow users to interact with the embedded system via a computer. It also describes the other systems which run in the background to support this application.

A. Graphical User Interface

The graphical user interface (GUI) of the PC application was made using the Qt C++ framework, chosen for its cross-platform versatility and support for strong event-based design. The GUI's elements are organized into four sections and are shown in Figure 6.

The first section is an OpenGL scene. This is where the data received over Bluetooth is rendered. Camera controls and three-dimensional axial markings are provided to give the user a better sense of the three-dimensional space the point cloud is rendered in.

The next section is a read-only text console used to describe the state of the application at various stages. These include initialization, acquisition of data points, rendering, and connecting with a remote Bluetooth module.

Then there are several buttons: a "Connect" button used to begin the device query, a "Start" button used to start a laser scan once connected to the device, a "Simulate" button used to load a simulation into the program (more details on this are provided in section V.D), a "Clear" button used to clear all points currently rendered in the OpenGL scene, and a "Pause" button.

Finally, a combo box allows the user to switch perspectives in the OpenGL scene. The two options are "Total" and "Central." "Total" shows the user the scene from a distant outside perspective. Using the arrow keys rotates the user's camera around the origin while keeping the view pointed inward. This is the default perspective. "Central" fixes the camera at the origin and looks outward, taking the hypothetical perspective of the embedded system. The arrow keys swivel the camera in place to look in different directions.

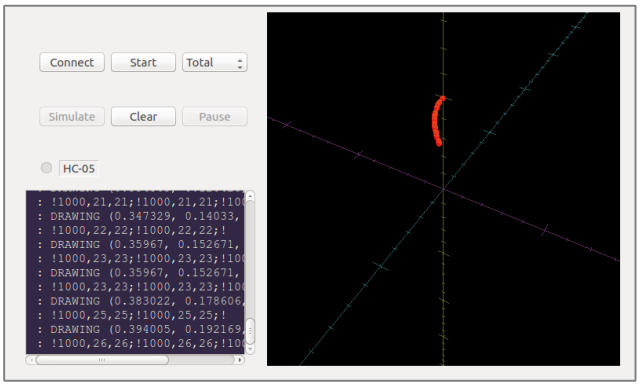

Fig. 6. The GUI of the application. In this screenshot, the application is connected to HC-05 and is receiving data wirelessly.

B. Qt C++ Framework

Qt is the event-driven, object-oriented framework that the application was developed in. It provides basic templates to all the GUI elements built upon in the application, as well as a native implementation of event handling. This is done by implementing special programmer-defined functions called signals and slots.

The system works by first defining signals, which accept a set of parameters but have no behavior on their own; they must be sent to slots by using the emit keyword. A slot is compatible with a signal if they share the same set of parameter types, and can otherwise be called on their own as if they were normal member functions. The connection of a signal to a slot happens by using a Qt function called connect(). After being connected, any time a signal is emitted, all slot functions connected with that signal are called using the parameters provided to that signal. This process is illustrated in Figure 7.

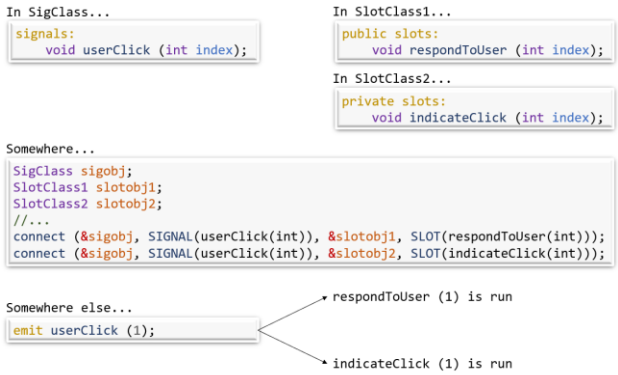

Fig. 7. Qt's SIGNAL and SLOT system

C. OpenGL

As a cross-platform graphics library with extensive documentation and native support in the Qt framework, OpenGL was the selected software to construct the virtual 3D scene.

For rendering primitives, the OpenGL functions glBegin() and glEnd() are used. In performance-intensive applications these functions can be proven inefficient, but given their straightforward functionality and the relatively low graphical runtime requirements of this project, these functions have proven sufficient.

Each "point" rendered in the scene is in fact not a GL_POINT but instead a cube made of GL_QUADS. This allows the data to be seen much more clearly in the scene, and the nature of the 3D shape also gives a useful depth perspective not present when rendering single-pixel points. To further aid with clarity and depth perception, dimensional axes are created using GL_LINES. These axes each intersect the origin and have markings every 10 cm in virtual space. The units in the scene are set that 1 virtual unit corresponds to 1 meter in physical space.

To direct the camera, the function gluLookAt() is used. This is a shortcut function which loads a fresh translation matrix onto the model-view stack and then modifies that matrix to translate the camera to a given position and point it at a second given position [4]. For this application, the up direction is defined as a unit vector along the z-axis.

D. Simulation

To confirm the functionality of the software in the absence of the embedded system, the ability to load a simulation into the program was added. In this context, a simulation is defined as a text file containing a list of spherical coordinates, formatted in a manner consistent with what the embedded system would generate. When reading this file, the program will convert each line from a collection of ASCII characters into three-dimensional Cartesian coordinates and will then use those coordinates to render a point in the OpenGL scene.

VI. DISCUSSION

The current state of the project is that the device is not responsive to objects further than around 1 m away when the targets are diffusive.

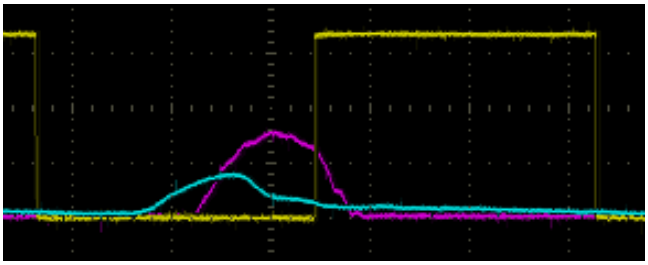

Fig. 8. A screenshot of the sync (yellow), zero (blue), and return (purple) signals measured off the board.

Figure 8 demonstrates what a typical set of signals from the OSLRF looks like, and in that specific screenshot, the target is within 1 meter of the device. This is not within the range specified by the OSLRF documentation for accurate results. This means that it has not been demonstrated that the device gives an accurate distance for a diffusely reflecting target. In general, the device has not been responsive enough to respond to targets further away than one meter to test how accurate it is. There is strong evidence pointing towards the cause of this being that too much power is lost due to the topics discussed in section III.C. There is reason to believe that working to reduce this lost power will increase the maximum range of the device into the region where accurate distance measurements should be possible. A more collimated beam could also help to reduce this problem.

To test the accuracy of the device without using a diffusive target, a reflective surface was placed at various distances from the device and the reflected beam was

directed back towards the device. Unfortunately, the device showed a non-linear error in distance prediction. These incorrect distances were verified to be independent of any software errors by calculating the distances manually on an oscilloscope. The calculated distances were always observed to be less than the physical distances and the error between the calculated and physical distances increased the further away the reflective surface was placed. If a non-linear curve could be fit to the error, then it could be corrected for in software. This correction is not ideal and a deeper understanding of the cause of the error will need to be investigated.

The alignment of the optics proved to be challenging. Constructing a chassis for the optical components which retains the proper alignment proved infeasible.

VII. CONCLUSION

The goal of this project was to demonstrate that the SETS technique can be used to implement LIDAR without expensive, high-speed electronics. The authors believe that this goal was achieved in the sense that the device did respond to pulses of light, however inaccurate the accompanying distance measurements may have been. The failure of the device to work within the distance range that was specified in the project goals is believed to predominately be a result of not considering the effects of power loss carefully enough. The authors strongly believe that redesigning the optical system more carefully to account for losses in power would increase the maximum detectable distance of the device.

Even considering errors in accuracy caused by the power loss issues, the processing algorithms and software design in the microcontroller have been demonstrated to account for imperfect signals during testing. Further work is required to account for dropped samples while the ADC and sampling process is being executed.

The Bluetooth module has been verified to communicate reliably with the user application. This was done using a dedicated testing circuit which sends a stream of points with the same formatting as what the embedded system provides when prompted over Bluetooth. Points received from all sources (including simulations) render correctly to the OpenGL scene, and all other GUI elements operate as expected.

ABOUT THE AUTHORS

Daniel Batista – Graduating with a B.S. in Photonics Engineering and planning on continuing graduate education as well as working full time with Northrop Grumman as an Optical Engineer.

Aaron Coville – Graduating with a B.S. in Photonics Engineering and starting work with Harris after graduation as an Optical Engineer.

Matthew Cummings – Graduating with a B.S in Computer Engineering and starting work with Lockheed Martin as a Junior Software Engineer after graduation.

Sean Huskey – Graduating with a B.S. in Computer Engineering and currently working as a Systems Developer with HostDime Global Data Centers.

REFERENCES

1. Optoelectronics, L., *OSLRF-01 Laser Rangefinder*. 2014.
2. Instruments, T., *LM5165-Q1 3-V to 65-V Input, 150-mA Synchronous Buck Converter With Ultra-Low IQ*. 2017.
3. Atmel, *ATmega328/P DATASHEET COMPLETE*. 2016.
4. Martz, P., *OpenGL Distilled*. 2006: Addison-Wesley.