PIDAR:
3D Laser Range Finder

Andrew Watson, Jonathan Ulrich

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

***Abstract* — The Pidar provides 3 dimensional points to a client that represent a 3 dimensional environment. An infrared laser sensor provides distance data along a Z axis and angular data on an x axis. A motor rotates the laser mechanism around a Y axis; this supplies the system with the necessary data in order to create a 3D point cloud representation of an environment. The Pidar utilizes a Linux operating system in order to manage the flow of information from the external sensors to an independent client.**

***Index Terms* — Client-Server Systems, Image Generation, Laser Radar, Systems Software.**

I. Introduction

Since its inception in 2002 the Robotics Club at UCF has pushed students and volunteers to the cutting edge of technology and innovation. Through annual participation in multiple international autonomous robotics competitions and outreach programs the club has excelled in generating robotic platforms capable of increasingly complex tasks. These competitions, primarily hosted by the Association for Unmanned Vehicle Systems International (AUVSI), include a variety of different kinds of platforms such as surface, ground, and underwater based vehicles. While upon initial inspection it may seem that such platforms operating in completely different environments would be vastly different, they are instead very similar in accomplishing basic autonomy. Since all of the platforms require interaction with their environment being able to sense the surroundings accurately has proven difficult for the organization to manage across multiple platforms without extreme cost. Of the many sensors outfitted on the vehicles there is one which provides the ample amount of real time data necessary for full autonomy.

2D Light Detection and Ranging (LIDAR) scanners are used on the largest of the platforms fabricated in the robotics club. These sensors are great for obstacle detection and avoidance. While previous attempts at using the raw 2D data from these sensors for map generation has proven beneficial observing a 3D world from 2D data is never an ideal scenario. It is the goal of this design group of computer engineers to enable 3 dimensional sensing from the physical rotation of a 2D LIDAR for use on these platforms.

The Robotics Club at UCF provided a list of requirements to which the project should adhere. The physical requirements are to occupy less than a cubic foot and weigh less than 5 pounds. The 3D scan time should be 1.5 seconds per scan or better. The assembly should be capable of at least 160° horizontal at least 90° vertical F.O.V. Achievable angular resolutions on all axes should be at least 0.5° or better. Real time configuration of resolutions and update rates should be a feature of the project. Power will be provided by a single +12V power rail. The maximum power consumption should not exceed 24 watts and there should be onboard regulation for all components with the ability to sustain immediate power loss. Interfacing to the system should be accomplished through a common PC interface such as Ethernet or USB. All connections should be weatherproof and the system should be able to sustain long periods of operation.

Software requirements should allow for an “always on” operational mode. Once the power is primed, the system should begin reporting data. The software should also all be open sourced and well documented. This will provide future engineering students in the robotics club opportunity to work on top of the project and leverage it. The system is to operate in both indoor and outdoor environments with equal performance.

II. Overview of the Pidar

The Pidar will be processing and outputting a very large amount of data. It generates points in 3D that are representative of the area in front of the system. The laser sensor runs on a continuous rotation and will communicate to our microcontroller through a serial connection. The laser sensor provides points that represent a two dimensional plane coincident to the horizon of the sensor. This provides us with 1080 points of 2D data. This data corresponds to the magnitude and angle of the point along a single axis.

The servo motor is controlled through a separate RS485 communication method. Using a magnetic rotary encoder, the servo is able to provide us with constant feedback of its location. Concerning the electrical design of our system we must identify the structure required to facilitate its intended functions. The microcontroller unit (MCU) must accept input from the LIDAR sensor and orientation sensor as well as motor feedback. That input will need to

be interpreted together in order to ensure our resulting 3D points are correct. In addition to the sensor input to the MCU it must also output the interpreted data to our PC for visualization and data analysis. Figure 1 depicts the basic electrical subsystem of our project.

III. 2D LIDAR SENSOR

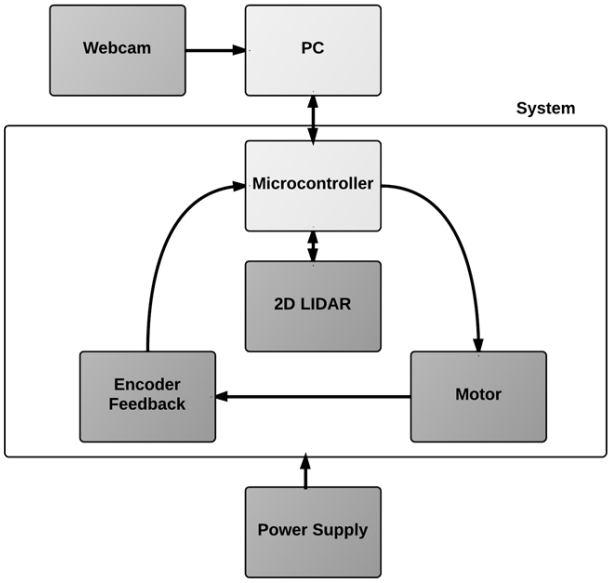


Figure 1. Pidar hardware block diagram.

The Hokuyo UTM-30LX 2D laser range finder used in our project allows us to scan distances up to 30m away with high accuracy. This LIDAR can be controlled through the provided SCIP protocol which can handle up to 12Mbps of data via a USB connection. The laser operates between 10.8 to 13.2 volts. Since our system is designed on a 12V rail there are no extra components needed to power this device. The UTM-30LX sensor can scan a 270 degree arc with an angular resolution of 0.25 degrees giving us up to 1080 distance values per scan. Given the sensor updates at 40 hertz this means our system must receive and interpret up to 43,200 distance points every second. More specifications can be found in Table 1. Each scan follows the same series of steps. In reference to Figure 2 step 0 is the first measurement point at which the scanning unit is enabled, though no data is sent until it reaches step A, the initial measurement step of detection range. This step is very important, as it does not occur until the UTM-30LX has reached the desired starting angle provided by the user. Step B, the sensor front step, is reached at the same time for each scan as it is at a point normal to the front face of the device. Step C, the end point of detection range, is the other user defined step. Similar to step B this tells the LIDAR when to stop recording the data it is scanning. Steps B and C are very important as they allow us to set our start and stop angles at any point outside of the dead zone.

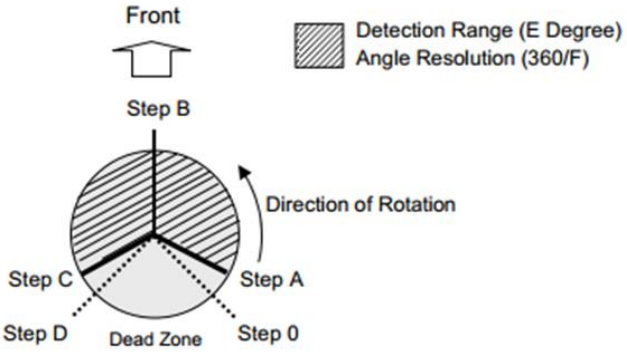


Figure 2. Hokuyo UTM-30LXscanning operation.

While the UTM-30LX has a maximum scan angle of 270 degrees between step A and step D, the LIDAR rotates through a 360 degree circle, such that it starts and ends at the same point each time. No matter the chosen scan angle, one full rotation will always take 25ms. Once one full rotation is finished, the UTM-30LX sends out a 1ms pulse via an open drain transistor output. The pulse is used for synchronizing 2D scans and motor position readings. The pulsing signal output is shown in Figure 3

Figure 3. Hokuyo UTM-30LX open drain output taken from an oscilloscope.

Table 1. Hokuyo UTM-30LX specifications.

Light Source	Laser Diode (λ=905nm)
Application	Indoor/Outdoor
Accuracy	±50mm
Angular Resolution	0.25°
Scanning Range	270°
Detecting Range	0.1m to 30m
Scan Time	25ms/scan
Power	12v DC
Weight	370g
IEC Rating	IP64

IV. SCANNING METHOD

In order to attach the third dimension to our 2D data the Hokuyo UTM-30LX 2D scanner needed to be revolved about some axis to obtain the required output. Careful research into the three different possible configurations will enable superior results for real-time use. With the goal of implementing this axis of rotation at the point of measurement of the scanner, individual analysis of each technique will prove beneficial in examining potential design difficulties. Exploration of each arrangement will reveal not only impending downfalls but also advantages to each technique for the intended robotics application. Optimization of the data is crucial for data generation as frame rate will be critical on a moving platform. The available scanning methods are referenced by the naming scheme of rolling, pitching, and yawing scans. These methods are in reference to the lasers coordinate frame with the convention of positive x being forward out in front of the sensor, positive y being to the right of the sensor and positive z being down below the sensor. The selected method for obtaining our third dimension is the rolling scan.

The rolling scan implements a horizontal sweep and rotates the sensor around a vertical axis (x axis) coincident to the center of the sensor. By rotating the sensor in this method there is a single focus point in the front of the sensor. The density of measurement data collected by the sensor in this configuration is directly in front of the sensor. This method is intuitive as it is most similar to human vision wherein focus is generally right in front of the observer. However the majority of the initial scan coverage is offset from the center of the device to either side towards the 'peripherals'. Full 3D coverage is only possible via full 180 degree rotation. A system which implements the rolling scan methodology while retaining a revolution about the origin of scans is relatively straight forward. Since the mounting point of motion can be placed behind the sensor without obstruction of the raw scanning data. With the mounting system so close to the majority of the weight in the assembly lower torque motors become more viable and can therefore lower overall system costs. The laser scanner chosen does not have a full 360 view and thus the small window behind the sensor provides enough space for mounting motors and electronics without hindering data capture. Generating a full scan in around a second required a fast motor in order to achieve the desired frame rates.

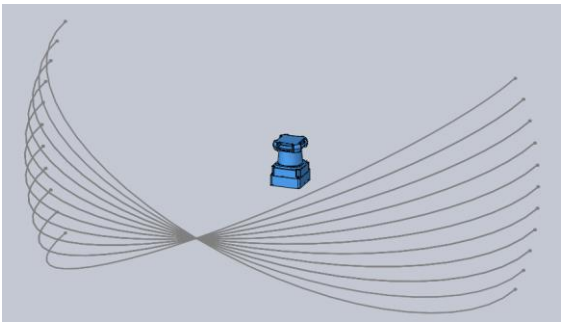


Figure 4. Rolling scan method.

A rolling scan can be implemented one of two ways; the first method would require a motor to alternate back and forth allowing the scanner to achieve a full view of the environment. The second method is to utilize a continuous rotation mechanism. This method allows the laser to continually move and collect data. We have chosen to implement the continuous rotation method in order to have the fastest and most consistent collection of data.

There are difficulties to using the continuous rotation method. One such difficulty is cable management. In order to communicate with our Hokuyo laser sensor, we must have high speed communication that is mostly reliable. We also have to power the device. In order to allow for free movement without sacrificing the data and power, we obtained a slip ring connector. The Mercotac Model 830 is an electrical slip ring that provides the laser with the power and communication that is necessary. The Mercotac model 830 slip ring utilizes a mercury channel that allows for constant electrical contact while being rotated with minimal noise, its specifications are shown in Table 2.

Table 2. Mercotac model 830 slip ring specifications.

Connections	8
Voltage AC/DC	0-250
Amp Rating	2@4 / 6@30
Max Freq MHZ	100
Contact Resistance	<1 milliohm
Max RPM	200
Temp Max (F) / Min (F)	140 / -20
Rotation Torque	1000 gram-cm
Circuit Separation	>25 megaohm

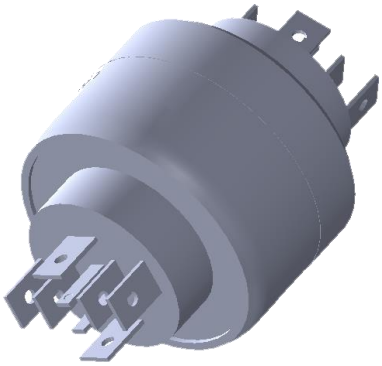


Figure 5. Mercotac model 830 continuous electric slip ring.

Since we are utilizing continuous motion driving the rotating assembly will require a motor that permits for continuous 360 degree rotation and provide accurate position feedback. After looking through different servo motors, stepper motors, and DC motors, we decided on a servo motor. The servo we chose is the Dynamixel MX-28R servo motor. This servo includes a built in magnetic encoder that will allow us to read the real position of the motor at any point. The encoder provides a 12 bit resolution for a position accuracy of ± 0.088 degrees. This level of accuracy would allow for our application as well as the others to obtain high resolution 3D scanning. This servo also has 360 degrees of rotation to allow for a full field of vision. This servo has a built in driver that allows for the serial multi-drop communication. The onboard processor monitors position, load, input voltage, and temperature and can report any metric via a single request. The communication speed can be adjusted between 8000 bps to a maximum of 3Mbps allowing for faster reports. This servo works on 12V and has a maximum operating current of 1400mA.

The Dynamixel MX-28R servo will provide motion to the rotating mechanism. This servo will provide the rotational motion on the x-axis of the laser scan. The servo is controlled using a serial connection from our microcontroller. It allows for setting of the angular position on the motor as well as providing feedback to the microcontroller. The MX-28R uses a RS485 asynchronous serial method of communication. It allows for commands to be sent or received at any time and is aimed at providing daisy chaining capabilities. Communication with the servo is done using an instruction packet. In order to read or write data the memory address must be referenced. The two sections of memory are the EEPROM and RAM. The EEPROM can be used to program values on the servo that will not be erased power is removed. Alternatively volatile RAM is only used for operating values and its values will not remain in memory without power.

In order to utilize the laser, slip ring, and motor, a mechanical assembly was required. The assembly must handle the wires that come from the laser and attach to the slip ring. We had to make sure that the wires were not caught in the rotation, as this would have devastating consequences for the system. Direct drive from the motor to the laser was not possible. In order to rotate the laser, a specially designed geared shaft was created as seen in Figure **6**. This allows the wiring from the Hokuyo laser sensor to have power and communication that is undisturbed by the motion.

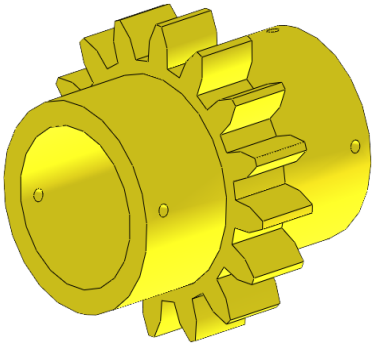


Figure 6. Geared cylinder for Hokuyo rotation.

The gear in Figure **6** on one side attaches to a mount designed for the 2D laser with the other side attaching to the Mercotac slip ring. A matching gear is attached to the Dynamixel servo and is placed directly underneath the previously described gear. The gears needed to match in size exactly so that we could maintain the one-to-one alignment with our reported points on the servo. Because there are only 2 gears in this assembly, this means the reported feedback information is only applicable to the servo and not the laser. In code we had to adjust the values in order to reflect the changes of the laser; essentially considering the inverted gear meshing.

The laser mounting assembly was also created specifically for the Pidar system. One of the requirements was that the power and communication cables could not be severed on the Hokuyo laser. In order to maintain the cables a special wrapping area was made that allowed the cables to be managed in a way that was uniform and that also kept them safe from the moving mechanisms.

V. Controller

The control system we designed had to be accessible and expandable for future members of the UCF Robotics Club. It also had to be fast enough and have enough memory to process and communicate all of the points in a point cloud twice a second. For this task we decided to go with a development board called the Raspberry Pi and hence the project name “Pidar”. The Raspberry Pi is a small embedded computing platform with the design goal of bringing computing to students in all parts of the world. It is outfitted with a 700 MHz ARM1176JZF-S ARM 11 processor featuring the ARMv6 instruction set architecture. With 512 MB of ram this computing platform is capable of running a full Linux operating system. With a power rating of 300 mA and a size of 3.37 inches by 2.125 inches this microcontroller capable computer carries with it a small footprint and a lot of I/O potential. With a total of 8 GPIO (General Purpose Input Output) pins, I2C busses, and full 3.3v and 5v rails this is a great platform for embedded systems. Built into the Raspberry Pi is a full 10/100 Mbps Ethernet port, and two USB 2.0 ports. The Raspberry Pi is powered from 5V line fed through one of the GPIO pins. The benefit of using a Linux environment is that the code can be edited and compiled on the device. Since the immediate intention of this device is to travel to competitions, it may be beneficial to have the work environment on hand to make any changes necessary. The system will work as described however every time without any necessary changes. One of the GPIO pins will be attached to the Hokuyo synchronous open drain line. This pin will trigger an interrupt service routine (ISR) that will be used within our code. The serial communication to the Hokuyo will be handled directly using a USB port. The motor will be controlled using RS-485 protocol through an FTDI chip directly connected through the second USB port on the Raspberry Pi. This setup has proven to be modular and easily manageable.

VI. Software

The Pidar is written predominantly in C++ using and leverages the Qt framework. This language proved to be the best for our requirements as it can directly manage memory, is object oriented, and is compatible with our external libraries. The Pidar relies heavily on multithreading for handling all the different devices, components, and I/O operations. The Pidar is an event driven system. Once the Hokuyo laser sensor signals the new scan has completed, our interrupt service routine then kicks off a set of functions to complete the process.

Each of the devices has its own class. The laser class implements a call back method to update its values to a parent controller. As the laser scans its data the callback is called to update. This will allow the class to hold the data that is retrieved. The laser class also includes methods to initialize the laser. This starts the communication and sends the commands necessary to get the data from the laser. The laser provides 2 dimensional points in polar coordinates representing the magnitude and angle of every point. Each of the scans will provide roughly 1080 points. These values are raw data and cannot provide a 3D representation on their own. The laser class also extends the Hokuyo library. This provides an easy-to-use interface to the custom SCIP protocol implemented by the manufacturer in order to communicate with their sensor.

The motor class allows us to control the Dynamixel. This class also extends its manufacturer’s library however its initial design was not efficiently implemented for our needs. Many changes had to be made in order to get it to work effectively and minimize the overhead for threaded operation. The motor class allows us to initialize the motor set its speed and direction among other functionality. It also allows us to read the position. This motor class also utilizes the callback architecture to handle triggering of new data to a parent class.

The environment we are using is essentially Linux on a chip. The Linux environment is an application level operating system intended for users. Access to interrupts at the user level in an operating system level was necessary for the project and thus a library called WiringPi was utilized. This library was developed with kernel level access that allows users to have nearly direct access to the GPIO pins. The ISR implemented is triggered when the Hokuyo 2D laser sensor finishes a scan. The high value it outputs lasts 1 millisecond upon which our interrupt is triggered on the rising edge. This alerts the WiringPi library at the kernel level which then triggers a ISR in our program with negligible delay.

To control the laser, motor, and interrupt classes, we developed a controller class. This object creates instances

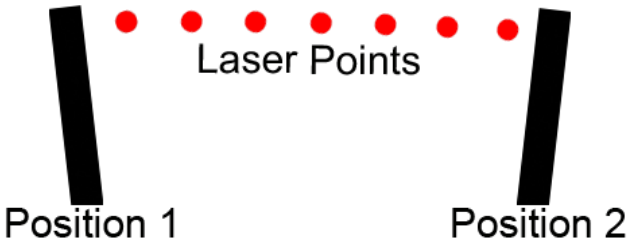


Figure 7. Point interpolation between two motor positions.

of each of the aforementioned classes and handles communication, control, and assembly of all information. Since we have continuous motion during capture of the 2D scans interpolating 3D values must account for this curving motion. By monitoring the motor position before and after each of the scans it is possible to compute a linear regression on every point in the 2D scan. This process is outlined in Figure **7**. These positions will also have to take into account the fact that the motor has been reversed due to the rotational assembly. Once each rotational position is calculated for every point in a 2D scan a 3D data structure is filled with the scan. This data structure (vector) is a set of 3 floating point values representing the spherical coordinate location of every point (R, Theta, Phi). That vector is then placed into a queue of scanned points that is ready for transmission. Once the control class has successfully completed its task, it waits for another interrupt.

In a completely separate thread, the transmission of points is processing the queue. The queue is constantly being monitored for new data and once new data is found it is quickly transmitted. The original idea was to use TCP communication and transmit points upon request. However after implementing and testing that method we found that it was necessary to use UDP transmission. While this method does not guarantee correct receipt of the scan data it was found to be the most efficient in transmitting the large amounts of 3D data. The transmission thread submits the points to a broadcast address. This allows anyone on the network to listen to the server. Since this method does not need a client to run, it is completely independently and will continue to run regardless of any problems with any listening client. The data that is contained in our structure is converted into a char array and sent out byte by byte over the network. On the receiving end the client listens for a transmission and will reconstruct the structure on the other end. This is not ASCII data but is binary data. In order to cut down on transmission time the group consciously did not use strings as they would have increased overhead.

In another thread we have a UDP listener. This thread is listening for commands on a separate port that can come from any of the clients. The commands are received and processed to see if they are valid. Once a valid command is received it sets a value, if needed, and then sets a flag that notifies our main thread that a change has been requested. When the processing is complete it will send out a return value of “OK” if the command completed successfully. If the command is invalid it will return “INVALID”. If a value is requested, it will return the value. The commands thread works by using ASCII. Since this isn’t a time critical transmission and there is nowhere near the same amount of data being transmitted, it was appropriate to use ASCII strings to send transmissions. This also helps with troubleshooting over a network as you can easily monitor packet data and look for errors in plain text through any packet monitoring software.

The last thread being utilized is the main thread. This thread will perform timed routine management of the program. It will check that different items are running correctly and if not it will attempt to correct them. Due to the use of our control class, if we find that a thread is misbehaving, we restart that thread and fix the problem. The main thread also handles the altering of the devices. Since the main thread holds the instance of our control class, it has access to perform the different tasks capable of each of the devices. It can report the speed, temperature and position of the devices and can change the motor speed. Since we cannot afford to risk the client overpowering our control thread, we have it on a timed check. The system is using a last in first out method to set the server controls. All controls can be changed once per cycle. The cycle speed can be configured by the user.

The configuration file is stored on the server in a plaintext document in using XML formatting. The document allows the user to configure the default settings of the Pidar without having to recompile or even rerun the program. This will allow users to be able to SSH into the Pidar and edit it as needed. Once the XML has been read, client commands will override those for the duration of its uptime. The Raspberry Pi is configured to automatically boot up once it receives power. The system is logged in and the application is executed. This gives users the ability to have quick uptimes upon providing power. In order to tell the system to safely shutdown we are utilizing an external logic circuit that will tell our Raspberry Pi to shut down. Upon operating system shutdown the external logic will remove the power from the Raspberry Pi entirely. This system is using an Atmel chip and communicates via two of the GPIO pins.

The client application is independent of the Pidar server. The application is also written using C++ and also leverages the Qt framework. Qt is cross-platform GUI builder and allows for design and deployment of applications across different operating systems including Linux, Windows, and Mac. The client application provides an interface to the Pidar as well as visualization of its Pidar output. Since the Pidar outputs a single scan at a time, it is up to the client to reconstruct the full 3D point cloud. We have developed our client application to render the visualization in multiple ways. The first method is clear the points off the screen as soon as soon as a full scan has completed. This will only show the user a single

2D scan. With this method there are fewer points, roughly 25,000, that are shown at any time. This means that the image to the user's eye may seem sparse. To a robotic vision application, it will still suffice to identify objects. Users can use other modes that will continuously add points to the frame. This method works very well for static areas with few moving objects. The system can then create a nearly complete 3D representation of the environment by obtaining millions of points to construct. Another viewing method available in the client allows for viewing of a single scan at a time. This only shows the one or two scans that were received last by the Pidar. This enables the user to visually see what the scanner is reading and demonstrates how it works.

The colorization of the imagery is done using RGB data that corresponds with the distance of the point in reference to the Pidar. This is a technique commonly implemented in heat maps but can work equally well for visualizing distances. This depth coloring allows the user to distinguish the different objects from one another as it gives contrast to the image. The client also provides the option to visualize points in only a single color. This gives the user the ability to just see the shapes of the objects and can give a cleaner looking image if the user wants to just see edges. Additionally the client can utilize RGB data that is collected from an attached webcam. The camera and the Pidar need to be aligned and calibrated ahead of time in order to perform this task correctly, but can allows the user to overlay the image on top of the point cloud and view real imagery in 3D.

The client application utilizes a library called the Visualization Tool Kit (VTK). VTK includes tools for displaying 3D environments, objects, and meshes that can also represent points. This library is the underlying framework for our visualizer in the client application and can allow the user to interact with the point cloud from the Pidar. By moving the visualizer window users can see the scanned environment from all possible viewpoints. VTK requires us to format our data in a predefined way so that the image can be displayed using their visualizer. The visualizer works by displaying points using their X, Y and Z values. However, the data collected form the Pidar is not in that form and thus must be converted for display purposes. The distance of the point is easily known when using the spherical data format. The distance to any object is provided without any further calculation. The conversion of points can be done by applying trigonometry. These computations can be seen below.

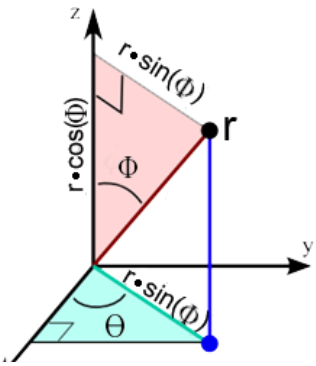


Figure 8. Spherical representation of a 3D point on the Pidar.

$$X = R * \sin(\theta) * \cos(\varphi)$$
$$Y = R * \sin(\theta) * \sin(\varphi)$$
$$Z = R * \cos(\theta)$$

Before the point cloud can be finalized, an RGB value must be applied to each point. Since each point is individually colored, it allows us to show a gradient to represent distance. This is performed by an RGB selection algorithm based on distance. Alternatively, this is where the live image representation will be set. The webcam image is processed into a data array. Each pixel of the image will correspond to a location and contain an RGB value that represents that place in real space. This RGB value is taken from the webcam image and placed into the point cloud point RGB field. This enables us to see true images in 3D.

Additional options in the client allow the user to pause the image so they can further inspect the point cloud. This method will disregard any of the point clouds that have come in since. A clearing option allows the user to clear the existing point cloud at any time and refresh it with only new data. The speed can be adjusted of the Pidar by adjusting a slider on the client. The range is from 1 to 60 rpm. This will allow the user to fine tune the speed to

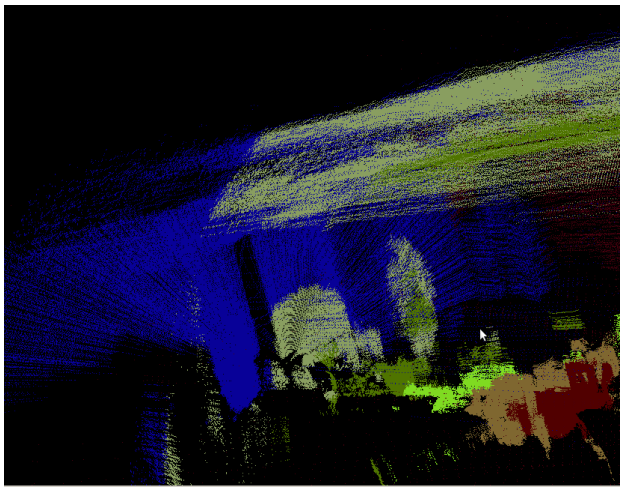


Figure 9. Early version of RGB point cloud representation.

balance between refresh rate and resolution. The client that has been created is only a test bed application. The actual client design for the end use will have to be integrated onto the autonomous robotic application. Data will then have to be processed and analyzed to allow for navigation and identification of objects.

VII. HARDWARE

The Pidar is designed to be a weatherproof system. The box was custom built to house our components and allow access to our cables. The laser, motor, and controller mounts were designed in Solidworks and all printed using a Makerbot 3D printer using PLA plastic. This allows us to have a enclosure that is impermeable to rain and harsh weather. The Hokuyo 2D laser is weather proofed already and no modification was necessary.

The electrical system is powered with a 12V input. This will ideally be a battery system provided by the host. The Pidar uses two separate power systems. The laser and motor are both powered directly through via the +12V input. It is assumed that the power input is a regulated clean +12VDC power supply. The Raspberry Pi is powered using a 5VDC input. So we have used a LMZ14203 switching regulator step down module. The part has been found capable of driving a 3A load which is sufficient for our power requirement of 6.75 Watts on the 5V rail. The IC is capable of handling from 6V to 42V input and is available in a TO-PMOD-7 package. This package type has 7 large pins all oriented on the same side of the plastic housing making it very easy to surface mount onto a PCB.

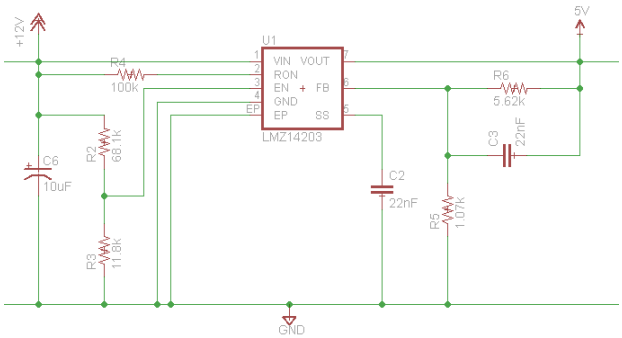
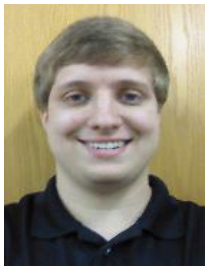


Figure 10. Power switching circuit.

VIII. CONCLUSION

The Pidar system is a complex system which provided us many design hurdles to overcome. The 3D data that is generated will prove to be most beneficial to its intended end use and provides a terrific framework for future designs.

IX. ENGINEERS



Andrew Watson

Andrew is a computer engineering student at the University of Central Florida. Currently employed as a research assistant his career interests include embedded and autonomous systems development.

Jonathan Ulrich

Jonathan is a computer engineering student. He currently is employed by the University of Central Florida. He will one day open an engineering firm and save the world.

X. ACKNOWLEDGEMENTS

We wish to thank the UCF faculty and the UCF Robotics Club for all of their support and guidance.

XI. REFERENCES

[1] Oliver Wulf, Bernado Wagner, "Fast 3D Scanning Methods for Laser Measurement Systems" *Proceedings of the International Conference on Control Systems and Computer Science,* vol. 1, pp. 312-317, July 2003.