POE Lab 1 - LIDAR

David Zhu
Jack Fan

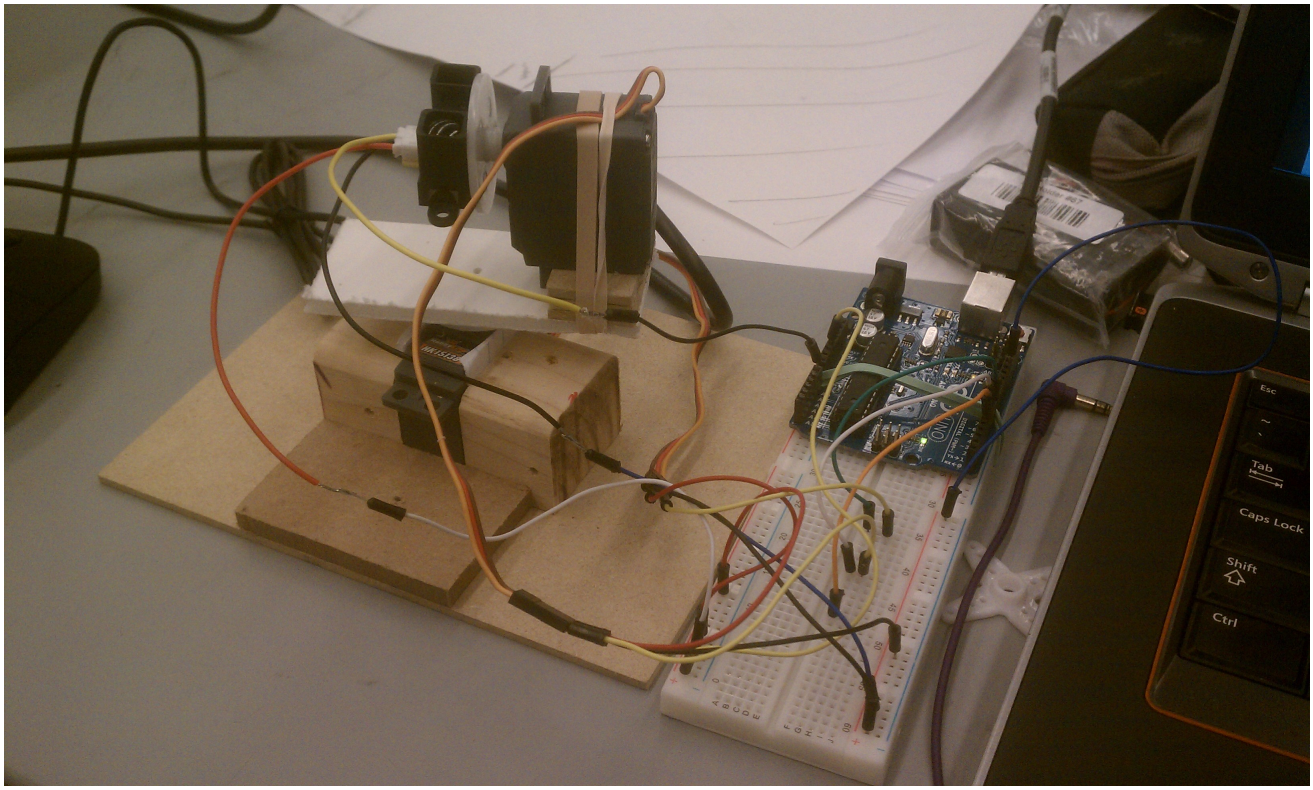
09/23/2013

1 Introduction

In this lab we create a small scale infrared LIDAR to capture an image of a small diorama. As part of Lab 1.5, we also use the LIDAR to image two other objects. Our LIDAR is a device that emits infrared light and measures the intensity of the reflection. Using this data, we can estimate the distances of objects and create a radar-like device, except with infrared.

2 Construction

The physical LIDAR is constructed with an infrared proximity sensor, two servos, and one Arduino Uno.



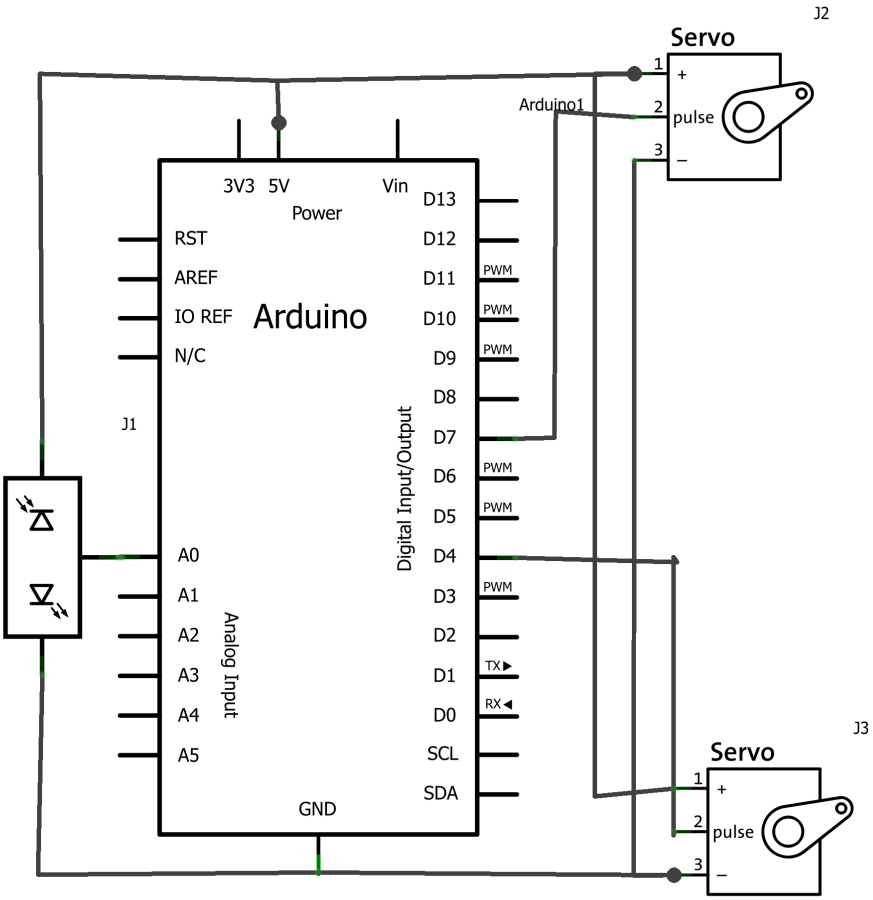
Physical LIDAR.

The sensor is mounted on the two servos in such a way that the sensor maintains as close to the center of axis as possible. We chose to place the sensor in this location because positioning it outside of the origin would cause future shift correction; maintaining this position makes calcuations easier and more accurate.

For the LIDAR configuration to balance the sensor position, the top servo must be offset. This creates a weight imbalance that requires a construction of a sturdy base. We aimed to avoid using hotglue as much as possible so that the servos can be recycled. Therefore, by balancing accessible construction versus speedy build, only the sensor and the plastic plates have been super glued.

The base servo is held tight in position to the base by paper insulation. We layered paper strips that allowed us to customize the thickness of the servo to fit with greater ease. The top servo is attached using some wood pieces as boost and a rubber band for adhesion.

2.1 Schematic



Device Circuit Schematic.

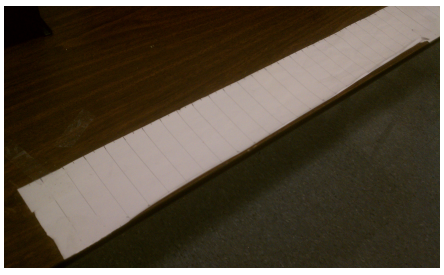
The servos and the sensor are connected to the Arduino Uno in a simple manner. The two servos read angles from ports D4 and D7. The proximity sensor sends distance data into A0.

2.2 Correction

Prior to running our project, we calibrated the proximity sensors so that input data becomes linear on our computer. This is due to the proximity sensor's limitations on sending back linear reads of its detected infrared. The proximity sensor provides a byte ranging from 0 to 1024 that represents its sensitivity to the infrared light that it sends. Because these values are not linear in relation to the brightness intensity, we created a measuring strip that collects the byte data. Using Excel, we fitted a power series onto the points and generated the following relationship:

$$Distance = 11194 * Readings^{(-1.151)} \tag{1}$$

This relationship is embedded in our Python code, which corrects the intensity reading on the computer after collection. Additionally, here is an image of our measuring strip.



Device Circuit Schematic

2.3 Code

This project is basically divided into two parts. We coded the Arduino and also have a Python code to control the Arduino. The Arduino would take in two numbers and pass the values to the two servos respectively. The servos would be moved to the exact position we want them to be and the infrared sensor would return three values to the arduino. These values are then averaged and sent back to the laptop through the serial port which was opened using our Python code. Our Python code does all the calculation while the Arduino code simply receives two numbers and then trasmits one number back. The value our Python code receives would be passed into a function that changes it to the actual distance (in inches). The function is a best fit function based on the data of the calibration of our infrared sensor. The distance is then stored into a list along with the positions passed into the Arduino. After looping through the positions we want our infrared sensor to take distance, spherical coordinates are changed into cartesian coordinates. Finally, we use a Python module called matplotlib to plot the points in 3D space. The feature of our code is that it asks for the two servos' starting and ending position, and then everything including all the calculation and plotting is done automatically. We enter four numbers, wait, and then receive a 3D plot.

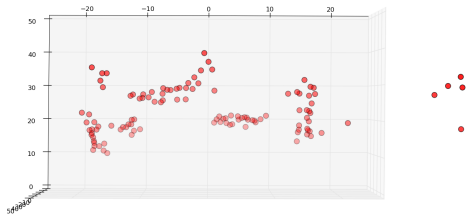
There is another Python script written that complements the original. This secondary script takes in the coordinates saved on the computer by the main script and replots them with convenience. This way, we can modify our graphs without having to retake our image.

3 Results

We collect serveral different images using the LIDAR. Following are the different figures that we created.

Picture of the physical diorama.

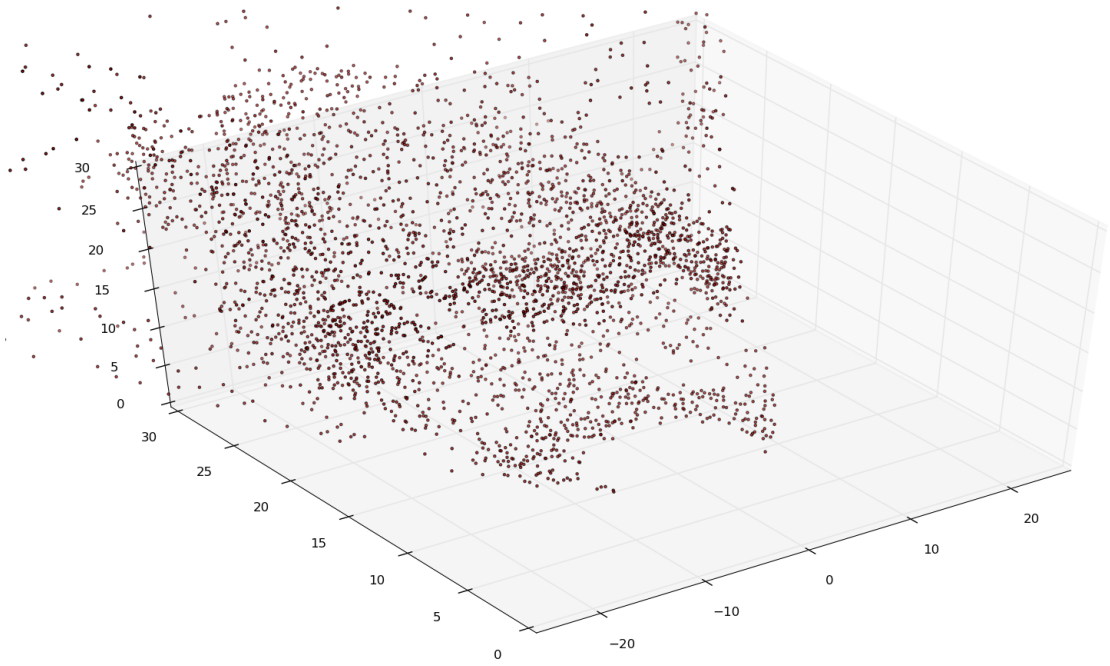
3.1 2D Scan



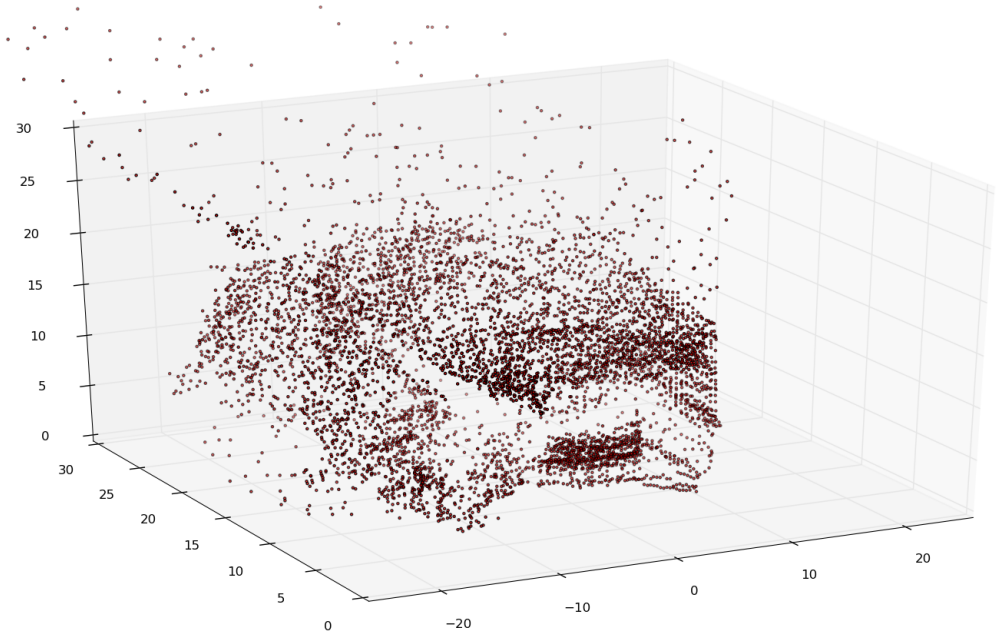
2D scan of the diorama.

3.2 3D Scan of Diorama

Following are two 3D scans of the diorama. One is taken at far range, and another at close range. Notice the change in resolution as we move the LIDAR forward.



Far scan of the diorama.



Near scan of the diorama. Notice the higher resolution.

3.3 Real Setting 1 - Fire Hydrant

We took the LIDAR outdoors and placed it in front of a fire hydrant. The resulting image is highly blurry, but it produces the general shape.

Actual hydrant.

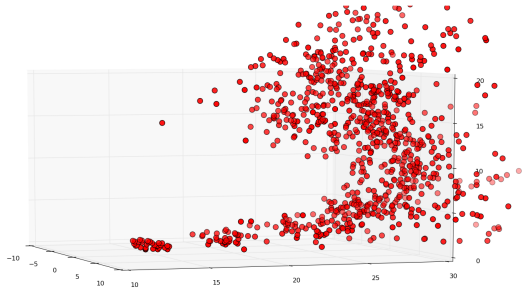
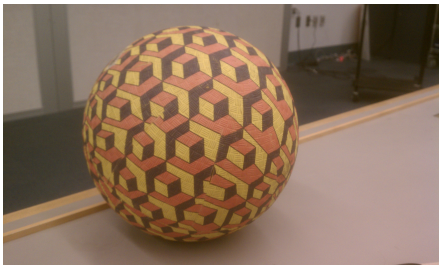


Image of hydrant.

3.4 Real Setting 2 - Dodgeball

When we applied the LIDAR to a dodgeball, it created a round face. There are some floating error points that make the whole image look like a comet.



Actual ball.

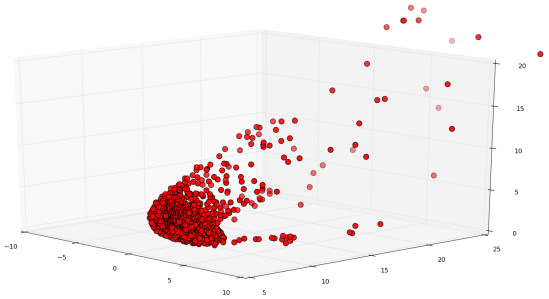


Image of ball.

4 Discussions

Initially, instead of creating a fancy physical mechanism, we decided to focus more on the software system. We let the computer do all the computation and leave the Arduino with the most basic job of controlling the servos and getting readings from the infrared sensor. One on the biggest challenges we faced was figuring out the communication between PC and the Arduino through the serial port. After some fidgeting, we learned that there is a receiving buffer(64 bytes)on each side of the serial port. Data sent from PC is stored in the buffer on arduino and vice versa. The data stays there until the PC or Arduino actively reads it. After understanding this interaction, the rest of the project is focused on minor details such as leaving room for the characters representing CR(carriage return) and LF(line feed). Another nice feature we have for our project is that the infrared sensor is located directly above the shaft of the base servo, allowing it to rotate about the origin of our spherical coordinate system.

One of our biggest unsolved challenges was to create a mesh in Python for our images. Sometimes the scatter plots look too blurred, and a surface plot would have been more defined. However, python's 3D mesh was a bit hard to implement and we decided to conclude this project before then. Were we to have more time, we would refine our program to reject more random points and create the image mesh. We also wanted to explore more with higher resolution, but that requires changing the fundamentals in how the Arduino processes.

Through this project, we gained extensive knowledge in Python Arduino communication and data processing.