ECE 785 Project Report

Sonar Point Cloud Builder

Ajay Rao (abrao2)
Neeharanshu Vaidya (nvaidya2)
Dhananjai Sharma (dtalipa)

Contents

1. Overview

The system uses an ultrasonic rangefinder (HCSR04) along with a 9-degree freedom MARG sensor to determine the locations of objects and to visualize this data on a point cloud viewer (Fig 1). The approach used by us in the project is as described below:

Upon start up the BeagleBoneBlack triggers the IMU sensor to find the attitude and heading of the system. After finding the system's heading the BBB sends a trigger via UART to FRDM KL25Z which interfaces with the Ultrasonic rangefinder sensor. Upon gathering information of the analog echo signal for 25ms, the KL25Z sends out the ADC output of the analog signals back to BBB via UART. In BBB the ADC values are analyzed to extract the distance of multiple objects from the ultrasound sensor.

The process is repeated whenever there is a change in the heading and attitude as detected by the IMU sensor. After moving the system through the area we want to map, the obtained distances are converted to points on the x,y,z plane using the heading information. All these values are written to a file in the XYZ format which is rendered on the online LIDAR point cloud viewer to get the point cloud image of the scanned area.

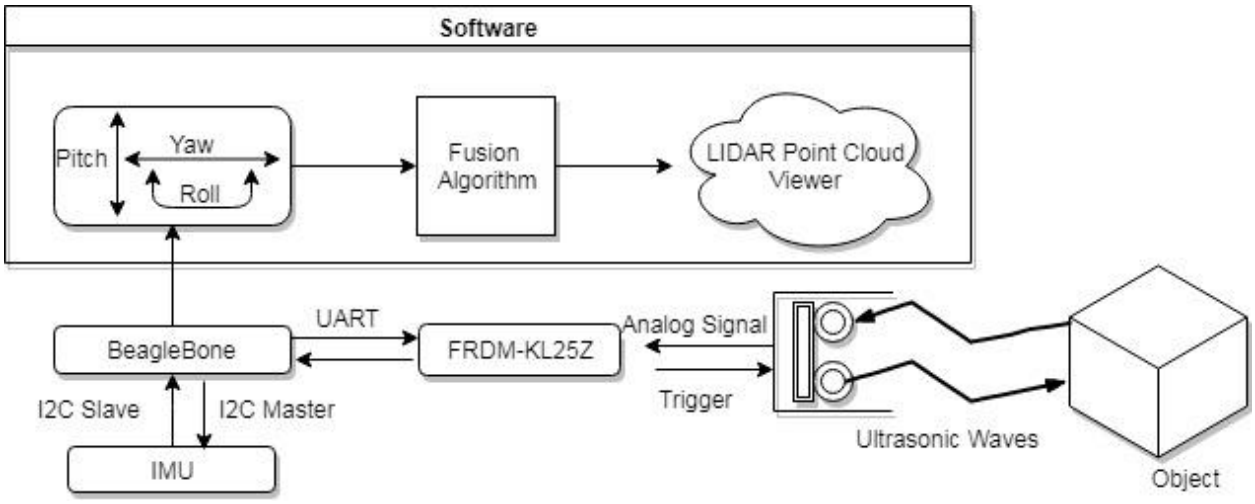

Figure 1: Block diagram of the system

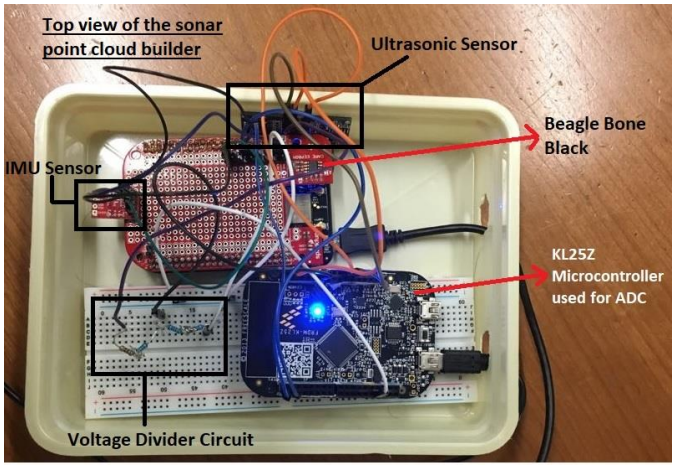

Figure 2: Top view of the hardware

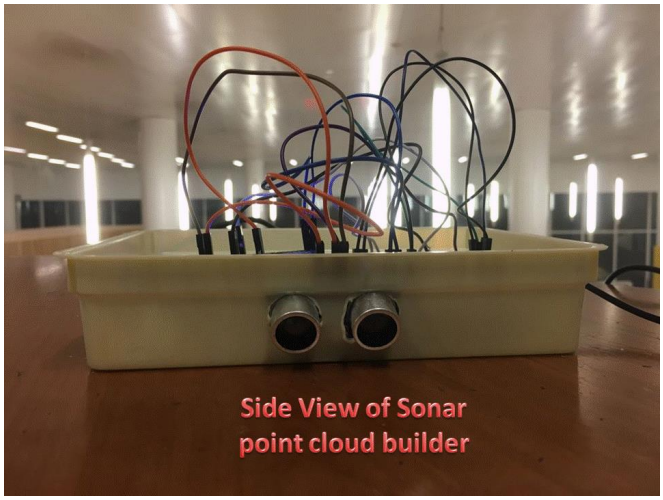

Figure 3: Side view of the hardware

2. Hardware and Software Design approach

2.1 Hardware: Ultrasonic Rangefinder (HCSR04)

2.1.1 Ultrasonic Sensor Overview

The sound energy travels by causing disturbance in the medium it is travelling and this is called propagation of sound waves. Under normal conditions, the velocity of the sound is 330m/s. SONAR or Sound Navigation and Ranging is a non-contact distance measuring technique. In this technique, a high frequency sound wave is transmitted by a transmitter and the reflected echo from a target is captured by a receiver. As the velocity of the sound wave is known, by measuring the time of travel, the distance between the source and the target can be calculated.

2.1.2 Ultrasonic Sensor Configuration

The ultrasonic rangefinder is powered with 5V from the Freedom KL25Z board. The sensor sends out 8 pulses of 40kHz signal whenever it is triggered. The trigger signal is generated by the GPIO pin (PORT C PIN 1) of FRDM board. The pin is held high for 10us before it is pulled low.

The analog output pin of ultrasonic pins needs to be monitored for detecting multiple objects. The output from the ultrasonic sensor has to be converted to digital form using KL25Z ADC before it can be processed to extract the distance information. KL25Z ADC has a reference voltage of 3.3V, but the max output from the ultrasonic sensor is 5V. Hence the output from the sensor has to be stepped down before being fed to ADC. A voltage divider circuit was employed to achieve this as shown in Fig 2.

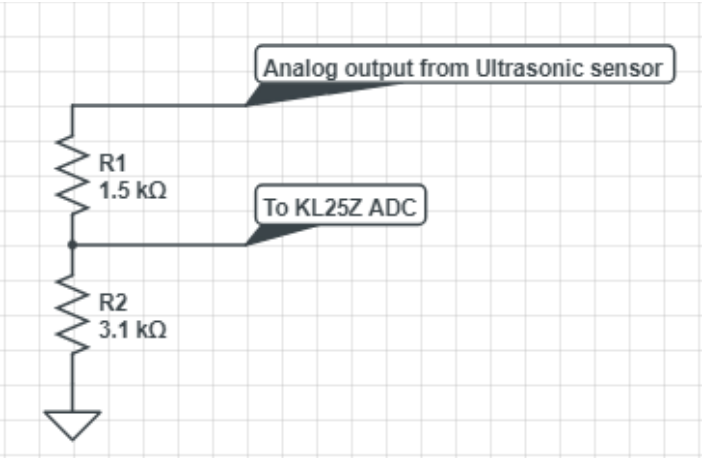

Figure 4: Voltage Divider Circuit used in the system

ADC channel 0 (PORT E PIN 20) is configured to sample the data from the analog output of the ultrasonic sensor. The ADC samples the analog output every 25us obtaining 1000 samples for 25ms. The total time of 25ms is obtained as follows:

Max range of ultrasonic sensor = 4m
Total distance the sound waves have to travel = 8m
Speed of sound = 330m/s
Total time taken for travelling 8m = 8/330 s = 0.02424 = 25ms

The sampled data is stored in an array first and then sent to BBB through UART communication. UART0 of KL25Z is configured at 115200 baud rate and the Tx pin of KL25Z is connected to the Rx pin of BBB. The sampled data is bundled into units of 5, where the first data is sent as is and the next 4 data are added and the sum is transferred.

2.2 IMU sensor: MPU9250

2.2.1 IMU sensor Overview:

The MPU 9250 is a 9-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer. It consists of three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for digitizing the accelerometer outputs, and three 16-bit ADCs for digitizing the magnetometer outputs. Communication with all registers of the device is performed using either I2C at 400kHz.

2.2.2 IMU sensor Operating Principle

Gyroscope: When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. The full-scale range of the gyro sensors may be digitally programmed to ±250 at 8,000 samples per second.

Accelerometer: Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2g.

Magnetometer: The 3-axis magnetometer uses highly sensitive Hall sensor technology. The magnetometer portion of the IC incorporates magnetic sensors for detecting terrestrial magnetism in the X-, Y-, and Z- Axes, a sensor driving circuit, a signal amplifier chain, and an arithmetic circuit for processing the signal from each sensor. Each ADC has a 16-bit resolution and a full scale range of ±4800 µT.

2.2.3 MPU9250 interfacing

The below block diagram shows the pin outs of the MPU 9250 sensor. The sensor was soldered to the BeagleBone Black to use I2C0. The BeagleBone acts as a master and provides clock to the I2C bus whereas the MPU9250 is a slave that responds to the requests sent by the master.

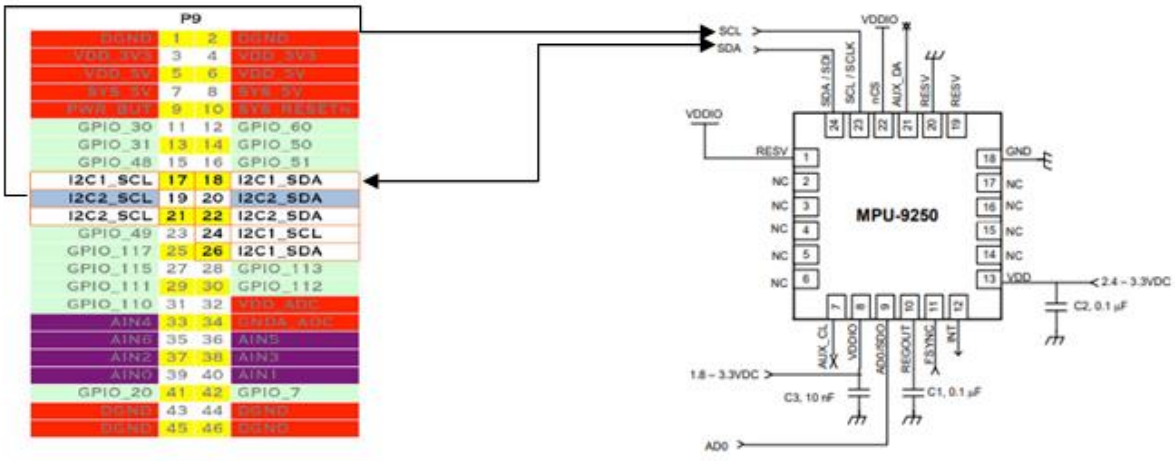

Figure 5: MPU9250 and BeagleBone Interfacing

The MPU9250 consists of Accelerometer and Gyroscope on a single device address whereas the Magnetometer is on a different slave address. Hence, to perform I2C read from all the 3 sensors, the I2C bypass enable needs to be configured. The sensor data registers contain the latest gyroscope, accelerometer, magnetometer, and measurement data. They are read-only registers and have 2 bytes for each axis of data. Hence, the sensors must be read in a single I2C read request for the registers to provide concurrent values.

2.3 Software Design: Ultrasonic Range Finder

2.3.1 SONAR Raw Data

The SONAR sensor provides output of objects at various distances. The software must calculate the time between the falling edge of the trigger and the first analog pulse to retrieve the distance between various objects.

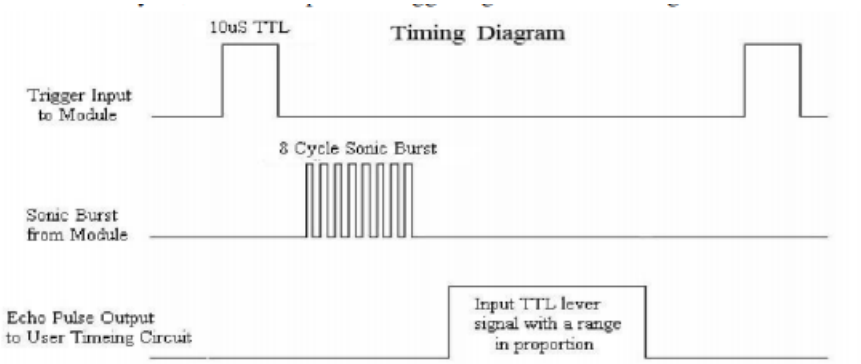

Figure 6: Timing output from SONAR sensor

The major difference between the timing diagram provided above and the one that was used for this project is that the Echo pulse output was not used. The limitation of the echo pulse causes a distance to be calculated from the nearest object. However, for multiple objects to be detected, the analog output was used.

2.3.2 Flowchart

The below flowchart describes the algorithm for detecting an object. The algorithm data acquisition is performed over FRDM-KL25Z and processing is performed over BeagleBone.

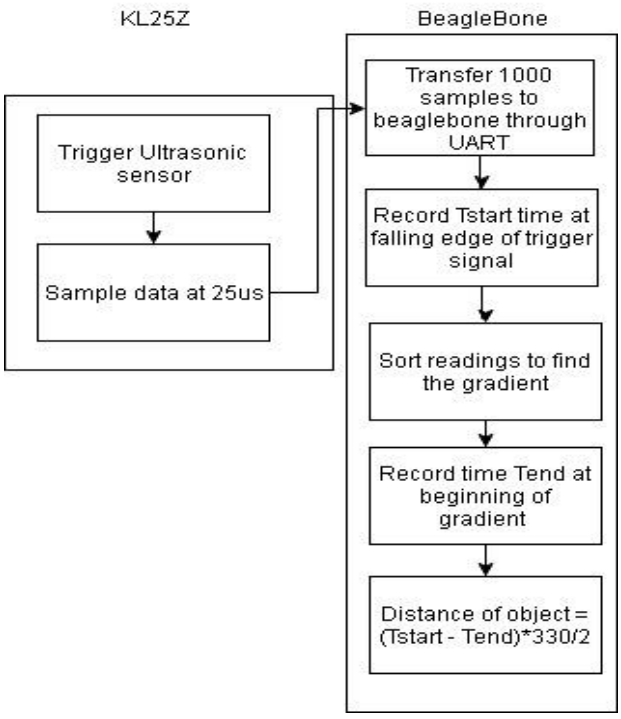

Figure 7: Flowchart for processing SONAR data

2.3.3 Processing of SONAR data

The SONAR data is transferred in a batch of 1000 samples from the FRDM-KL25Z to the BeagleBone. The BeagleBone further breaks down the samples to detect objects at different distances. Here are the steps followed by the processing algorithm:

- Break down the samples into chunks of 100.
- Calculate the max value from these samples.
- Use the max value to set threshold such that all analog values above this threshold correspond to an object.
- Detect the slope of the samples.
- The slope represents presence of an object in the vicinity. Time taken for the beginning of this slope from the falling edge gives us the time taken by the sound waves to travel to the object and return.
- Distance (in cm)= (time at falling edge – time at beginning of the slope)*330/ (2 * 10000).

2.4 Software Design: IMU Sensor

2.4.1 IMU Raw data

The raw data from IMU consists of ax, ay, az, gx, gy, gz and mx, my and mz. The values range as shown below:

Table 1: Sensor Range

Sensor	Range
Accelerometer	-2 to +2
Gyroscope	-250 to +250
Magnetometer	-4800 to 4800

2.4.2 Flowchart

The software for IMU is as shown below:

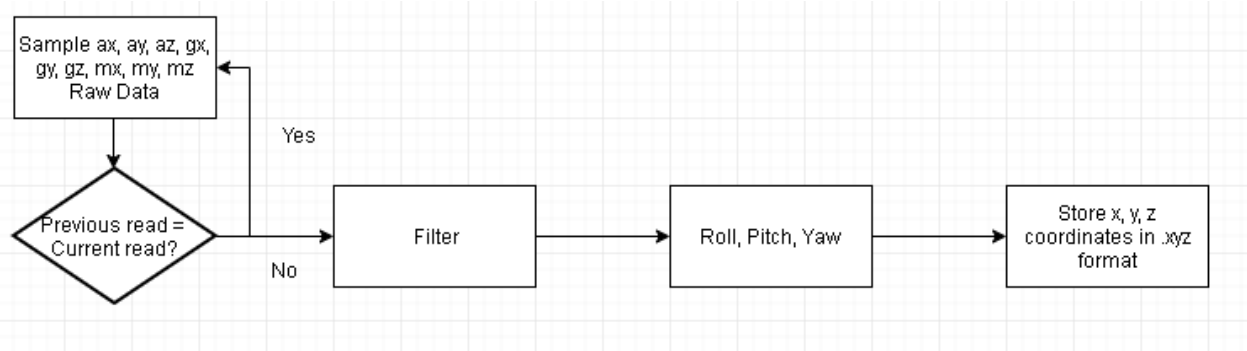

Figure 8: Flowchart for reading IMU data

The raw sensor data is read over I2C as explained in 2.2.3 and passed through a filter.

2.4.3 Why use a filter?

As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. The accelerometer data is reliable only on the long term, so a "low pass" filter has to be used.

The gyroscope is unstable too because of the integration over time, the measurement has the tendency to drift, not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term.

2.4.4 Filter Working

The filter combines measurements from an accelerometer and gyrometer into a better angle estimate than either could provide on its own.

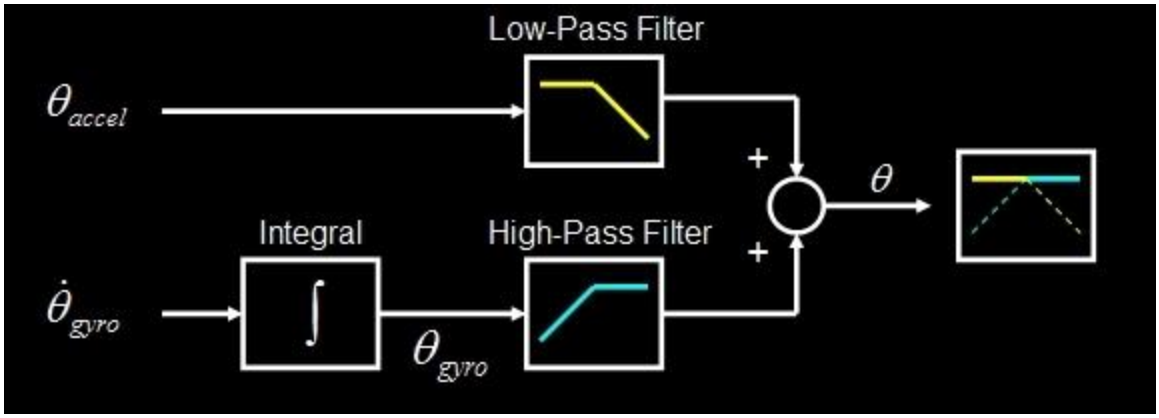

Figure 9: Component Filter using accel and gyro data

The filter treats the Roll and Pitch axis separately. It keeps track of the absolute angle by integrating the output from the gyro on that axis, and corrects the angle with a small part of the angle from the accelerometer on that axis when it is within a certain range. This angle is then fed into the control PI loop resulting in roll/pitch angles.

The final roll, pitch and yaw values are converted to radius, theta and phi which are the rate of change of each of those corresponding parameters. The radius, theta and phi values are plugged into the below equation to obtain x, y and z values.

Pseudo-code: (standard math library trigonometric functions)

From `(radius, theta, phi)` you can find the point `(x,y,z)` :

```
x = radius * sin(theta) * cos(phi);
y = radius * sin(theta) * sin(phi);
z = radius * cos(theta);
```

Figure 10: Formula for calculating (x,y,z)

2.5 Software Integration

The software integration involved weaving the UART, Ultrasonic and the IMU module to output a file in .xyz format which can be read by the LIDAR point cloud viewer. The .xyz file is saved from the python script which reads current date and time to name the file.

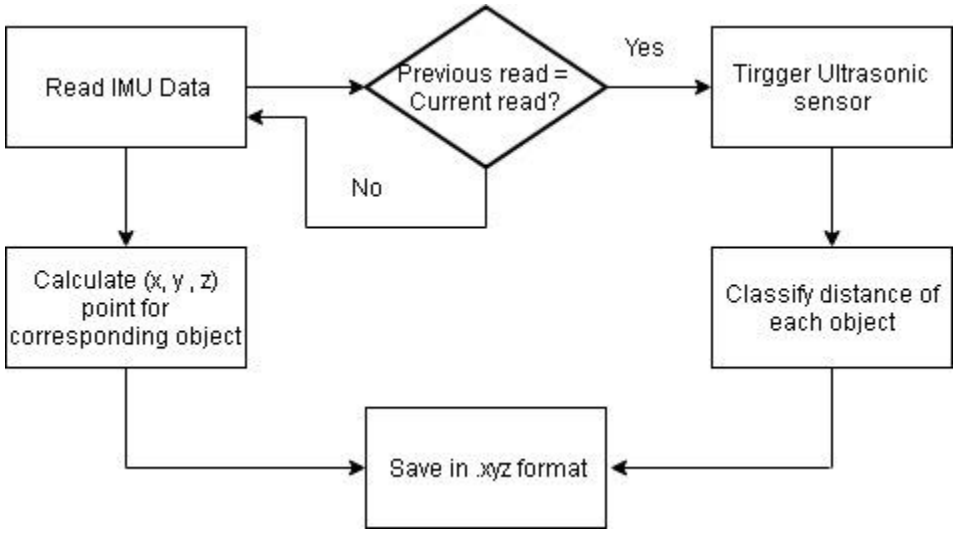

Figure 11: Flowchart for integrated software

2.6 Point Cloud Viewer

The LIDAR point cloud viewer generates a point cloud based on the .xyz file provided. The current scope of this project is limited to manually generating the point cloud. The below steps are followed in order to view the point cloud:

1. Extract the .xyz file from the path: /home/debian/Final_Project/Point_Cloud_Data on the BeagleBone.
2. Visit http://lidarview.com/
3. Select “XYZ” format and choose the file whose point cloud has to be viewed.
4. Click on Render.

3. System Testing

The system was tested with the below setup:

Figure 12: Test Setup

The setup consists of the below objects at distances which were measured to be:

Table 2: Object and distances

Object	Distance wrt the sensor (in cm)
Marker	41
Duster	58
Coca Cola Cup	51
Water Bottle	82

4. Results

The figures below represent the output on the console of the beaglebone. The python script prints out values of theta and phi along with the distances it has calculated. As we can see, the distances in the table and the console are close values.

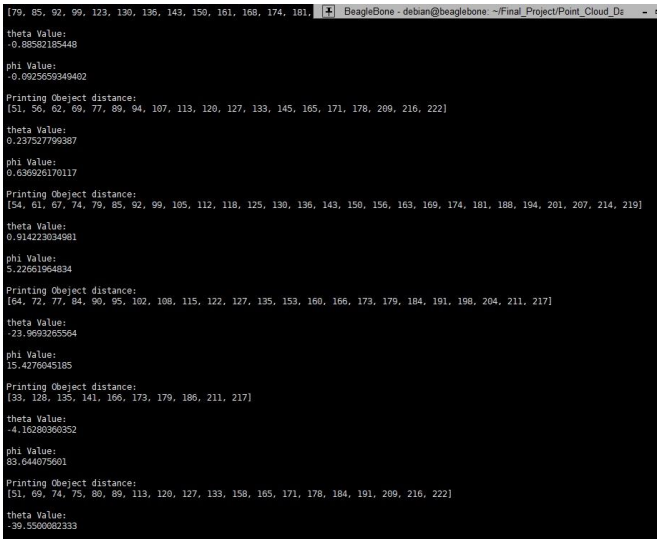

Figure 13: Output on the console

The above output also provides a .xyz file which when fed into the lidar point cloud viewer generates a 3D map of the surrounding. The correlation between the objects placed and the point cloud can be observed as shown in the figure.

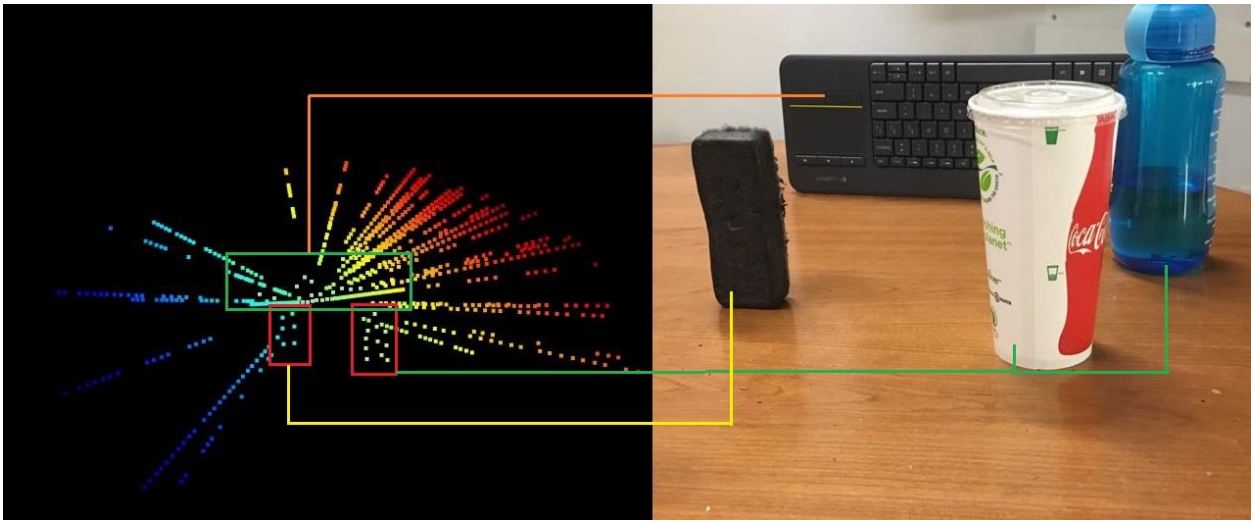

Figure 14: Correlation b/n point cloud and objects

5. YouTube Link

The video of the operation of the device can be found here:

https://www.youtube.com/watch?v=N7aGDDu8S-4&feature=youtu.be

6. Challenges

There were numerous hurdles that needed to be passed before we were able to get the desired results. Few of them are listed below:

1. Configuring UART in BBB and writing C program for the duplex communication. After trying for 2 days modifying the example script in ECE785 GIT folder and still failing to establish a duplex connection with KL25Z, we looked at writing python code for UART communication. We then developed a python wrapper script to configure and maintain an UART communication.
2. Understanding the analog echo output and developing an algorithm to measure the distance of multiple objects.
3. Stabilizing values out of the IMU.
4. Fine tuning filter to obtain reliable quaternions.

7. Future Scope

Currently the system does not plot the surroundings in real time and also does not automate the process of sending data to point cloud server. In the future system, BBB must be able to connect to internet via Wi-Fi and should be able to process the data on each scan and upload the data to point cloud server on real time basis. As the BBB has a HDMI port, we can connect a monitor to it and display the plot of scanned objects in real time.

8. References

- Complementary filter design on the Special Euclidean group SO(3) ('07) [.pdf] – by Grant Baldwin, Robert Mahony, Jochen Trumpf, Tarek Hamel, Thibault Cheviron
- Fun with the Complementary Filter [link] and The Balance Filter (Jun. '07) [.pdf] – by Shane Colton
- A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications (Dez. '09) [link] – by Starlino
- https://stackoverflow.com/questions/30619901/calculate-3d-point-coordinates-using-horizontal-and-vertical-angles-and-slope-di