

STUDENT PLAGIARISM: COURSE WORK – POLICY AND PROCEDURE

MTRX 2700 COMPLIANCE STATEMENT

INDIVIDUAL / COLLABORATIVE WORK

By submitting an assignment through the University Learning Management System (LMS),

I/We certify that:

1. I have read and understood the University of Sydney Academic Dishonesty and Plagiarism Policy;
2. I understand that failure to comply with the above can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);
3. This Work is substantially my own, and to the extent that any part of this Work is not my own, I have indicated that it is not my own by acknowledging the source of that part or those parts of the Work.
4. I declare that this assignment is original and has not been submitted for assessment elsewhere, and acknowledge that the assessor of this assignment may, for the purpose of assessing this assignment: a) Reproduce this assignment and provide a copy to another member of Faculty; and/or b) Communicate a copy of this assignment to a plagiarism checking service (which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking).
5. Information on plagiarism is available online at: [University of Sydney policy website](#)

Name(s):

James Cooper, Thejan Elankayer, Rishi Rangarajan, Naida Rasheed, David Young

Date: 31/05/2019

MTRX2700

Final Report



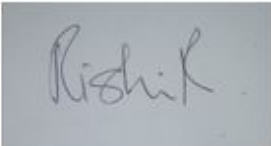

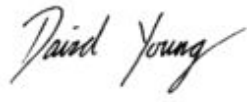
Estimative Capture and Perception Equipment

Group 8 (Wednesday Lab)

James Cooper - 480357216
Thejan Elankayer - 480383673
Rishi Rangarajan - 470369843
Naida Rasheed - 480376491
David Young - 480372574

Group Certification

We certify that the design, implementation, and documentation presented by our Group are, unless otherwise acknowledged, the work of our group members only, and that the individuals named below were responsible for completing the following percentages of the modules named.

Name	%	of the Module	Signature
James Cooper	25	Hardware	
	15	Software	
	20	Report	
Thejan Elankayer	20	Hardware	
	10	Software	
	30	Report	
Rishi Rangarajan	25	Hardware	
	15	Software	
	20	Report	
Naida Rasheed	20	Hardware	
	20	Software	
	20	Report	
David Young	10	Hardware	
	40	Software	
	10	Report	

Introduction	4
Document Identification	4
System Overview	4
Document Overview	4
Reference Documents	4
System Description	6
Introduction	6
Operational Scenarios	6
Test Mode	6
Scan Mode	6
System Requirements	7
Module Design	8
Module Requirements:	8
Conceptual Design	13
User Interface Design	15
User Inputs	15
Interface Outputs	16
Input Validation and Error Trapping	17
Hardware Design	17
Scope	17
Power Supply	18
Computer Design	19
Sensor Hardware	20
Actuator Hardware	23
Operator Input Hardware	24
Operator Output Hardware	24
Hardware Quality Assurance	25
Hardware Validation	25
Hardware Calibration Procedures	26
Hardware Maintenance and Adjustment	27
Software Design	28
Software Design Process	28
Software Quality Assurance	29
Software Design Description	30
Preconditions for Software	37
System Performance	38
Performance Testing	38
State of the System as Delivered	40
Future Improvements	40

Safety Implications	41
Conclusion	42
References	42

Introduction

Document Identification

This document will overview the design of ESCAPE (Estimative Shape Capture and Perception Equipment). The document is prepared by MTRX2700 Group 8 for the MTRX2700 Major Project.

System Overview

ESCAPE is a rotating, LIDAR (Light Detection and Ranging) based scanning system, which detects and plots the surface of a particular object within its plane of vision. The system has been used to plot rectangular and polygon shapes, and determine the area of the shape's face that is within the plane of sight of ESCAPE's LIDAR module.

Document Overview

This document has been created to detail the design of ESCAPE. It will include the system requirements upon which ESCAPE was built, a breakdown of its modular design, the hardware and software components, a review of ESCAPE's performance, and any safety implications associated with the system.

Reference Documents

The present document is prepared on the basis of the following reference documents, and should be read in conjunction with them;

Adafruit Industries, "ADXL Digital Accelerometer," ADXL345 datasheet,

STMicroelectronics, "L3G4200D MEMS motion sensor: ultra-stable three-axis digital output gyroscope," L3G2300D datasheet

Honeywell, "3-Axis Digital Compass IC HMC5883L," HMC5883L datasheet

HiTEC, "HiTEC General Servo Information," Servo Datasheet

PulsedLight LLC, "LIDAR-Lite v2 "blue Label"," LIDAR datasheet.

Common Acronyms Utilised in the Report

Acronym	Full-Form
ESCAPE	Estimative Shape Capture and Perception Equipment

LIDAR	Light Detection and Ranging
PTU	Pan & Tilt Unit
IMU	Inertial Measurement Unit
MCU	Microcontroller Unit
PWM Signal	Pulse-Width Modulated Signal
IIC	Inter-Integrated Circuit

System Description

Introduction

ESCAPE has the following modules to be able to detect and scan an object within its range; PC, DragonBoard MCU, LIDAR Module, Servo Module, IMU Sensor Module. The PC and MCU are used to input values in the system. The MCU implements software processes into each of the hardware components; moving the servo module, interpreting LIDAR scans, and converting IMU sensor measurements. The Servo module directs the line of sight of the LIDAR, such that the system can eventually cover the entirety of the object. The LIDAR module then takes a distance measurement from its current position to the object placed within its line of sight. The information from the Servo and LIDAR modules are then serially transmitted through the MCU and back to the PC to provide a combination of the position and distance of a particular detected point. A cumulation of these points is plotted on the PC to develop a final scan. These points are additionally utilised to calculate the area of the object being scanned.

Operational Scenarios

There are two operational scenarios within the system; a **test** mode, and a **scan** mode.

Test Mode

The 'Test Mode' is called for testing all the modules separately and ensuring all components are functioning within the system. The Test Mode is designed to work with each of the components, including the two servo motors, the LIDAR sensor and the IMU. The test for the two servo motors allows the user to write the angle for pan and tilt as a floating point value, and the pan and tilt servos will respectively get into those positions. The test mode for the IMU presents an interface, which reads the values from the IMU sensors. The servo motors are disabled, and the user is free to move the PTU to various position to confirm the readings. As specified in the project specifications, each IMU sensor has a particular function (Gyroscope measures both pan and tilt angle, Accelerometer measures tilt angle, and Magnetometer measures pan angle), and in the test mode, the user can test of each of sensors individually. The user is able to which sensor they would like to read the output from, and the LCD will display the requested information. The test mode for the LIDAR operates in the same way as that for the IMU. Instead of measuring the movements, the LIDAR test mode will display the distance measured by the LIDAR.

Scan Mode

The Scan Mode is called for detecting an object and evaluating the area of the shape. A series of user inputs determine the type of scan and set values for the associated parameters in this mode before executing the scanning. Currently, the module can execute two types of scans: 'Rectangle Scan' and 'Full Scan'. Each of these modes require the following parameters to be set by the user: Resolution, Number of Samples, and interface mode. For both types of scans, the servo moves in controlled increments, with LIDAR

samples take at each position. This data (the position/orientation of the module and the distance detected) is processed and then sent to the PC for processing. The PC is responsible for using this data to evaluate the area of the shape and plot it on MATLAB.

The two scan types are designed to approach the scanning task differently, and have different associated programs. The 'Rectangle Scan' is designed specifically for detecting rectangles, and is designed to quickly locate the edges of the object. The PTU scans horizontally to find the distance between the left and right edge - identifying the width of the shape. The module then proceeds to scan vertically to find the distance between the top and bottom edge - identifying the length of the shape. A more thorough scan is then conducted for locating any holes or surface features on the shape. The scanning algorithm also processes the data, including finding the median distance from a range of samples and sends this to the PC for further processing and presentation.

The 'Full Scan' mode samples the entire region in front of the device. In this mode, the module first scans horizontally along the centred axis to locate the first edge of the shape. Once the shape is detected (when the LIDAR sample shows distance within the range of 0.9m and 1.3m). After detecting this edge, the module begins the scan at from the bottom-left, and incrementally samples the distances at each new location. The median distance is calculated by the module and then sent to the PC. The MATLAB function on the PC will then filter the entire scanned region according to its distance to obtain a set of points within the range the object is expected to be placed.

Within the scan mode, the type of scan, resolution, number of samples and interface mode (PC or DragonBoard) are input into the system. From here, the servo begins moving according to the specified resolution.

A third scan, edge detection, was conceptualised. However, it has not been added into the system due to its complexity (it is unlikely that the utilised hardware would be able to execute this scan accurately). This scan will be detailed in the software section of the report.

System Requirements

1. The system should be able to detect objects within a range of 0.9 metres to 1.3 metres.
2. The system should be able to map any object within the above distance range and -50 to 50 degrees azimuth and -25 to 25 degrees elevation.
3. The system should be able to evaluate the area of the object within a 10% of its accurate value.
4. The system should be able to execute scanning and plot of solid shapes as well as those with holes, and accordingly calculate the area.
5. The system should be able to receive user inputs to specify parameters for a customised scan such as resolution, sampling frequency, etc.
6. The system should display real-time information for pan and tilt angles and distance measurement during the process of the scan.

7. The system should be able to plot the mapped points as it is being detected by the PTU in real-time and offline.
8. The IMU should be able to track the movement of the PTU as it pans and tilts to scan different positions.
 - a. Gyroscope should accurately measure pan and tilt angles up to 70 degrees in any direction.
 - b. Accelerometer should accurately measure tilt angles up to 70 degrees in either direction.
 - c. Magnetometer should accurately measure pan angles up to 70 degrees in either direction.
9. The PTU should be able to orient itself in any way in within the range: 20 to 160 degrees.
10. The user should be able to set the inputs and execute the scanning from the Board or the PC.

Module Design

The system can be broken down into the following modules;

1. Servo Motors
2. LIDAR
3. IMU
4. DragonBoard MCU
5. PC

The requirements for each module will be specified below.

Module Requirements:

1. Servo Motors

a. Functional Requirements:

The two servos should exhibit control over the range: 20 to 160 degrees each; one for adjusting the pan angle of the module and other to adjust the tilt angle. The servo motors should further be programmed to execute a horizontal scan, moving repetitively by a set small resolution (decided by the user) to cover the entirety of the shape being scanned.

i. Inputs:

The input written to the servo is the new position of the servo in the form of a PWM. This new position indicates the next orientation of the servo motor(s) for which the next LIDAR sample is to be taken.

ii. Process:

A function in the program takes the next angle to be written as an input and converts this into a PWM, which is then sent to each of the servos.

iii. Outputs:

No outputs are received from the servo motors.

iv. Timing:

The delay between each consecutive servo position is determined by the Scan Delay parameter that is input by the user.

b. Performance

The servo motors work accurately for resolutions up to 0.5 degrees. The servo has difficulty in incrementing its pan and tilt angle by lower increments than 0.5.

c. Interfaces

The Servo motors are interfaced to work with the Board. Each servo motor (pan servo and tilt servo) is connected to pins with PWM outputs

d. Design Constraints: The tilt angle and pan angle are restrained by the hardware design of the PTU. The servo motors can move the

1. LIDAR:

a. Functional Requirements:

i. Inputs:

The LIDAR was set to continuous sampling instead of edge-triggered sampling. Hence, no input was sent to the LIDAR to trigger the distance measurement.

ii. Process:

The distance measurement is output as a PWM signal. To convert this signal, a function is called which stores the timer count at the rising edge and then at the next falling edge.

iii. Outputs:

The LIDAR outputs a PWM signal, which contains the distance measurement as the duty cycle of the PWM. Rather than using an interrupt, the signal from the LIDAR is polled for, and a function is used to convert the signal with respect to the clock cycles into a value for distance.

iv. Timing:

The LIDAR's output is to be polled after the servo has moved to its next position. The LIDAR samples are generated continuously though, the function to convert the PWM is called after the servo function is executed.

v. Failure Modes:

Errors mainly occur due to the presence of noise. If a simple average of several distance samples were taken, then the noise would still affect the mean. As a result, a median filter has been implemented to obtain a more accurate reading, while filtering out the noise and avoiding it from interfering with the distance measurement.

b. Performance

The LIDAR sensor is able to sample distances fairly accurately in the range specified in the requirements (0.9m to 1.3m). However, at very close distances, the sensor does output some inaccurate readings. The LIDAR is

also prone to noise in the readings, but this can be filtered out while processing the data.

c. Interfaces

The LIDAR is setup to work with the Dragon Board, and is connected by two pins: Trigger and PWM Monitor. Using a jumper, the Trigger pin was connected such that it would continuously be at low (or 0), hence triggering the laser continuously.

d. Design Constraints:

Since the LIDAR could only rotate about its axis, points on either side of the centred axis (on which the sensor was mounted) had a greater perceived distance. Using trigonometric formulae, these perceived distances needed to be converted to true distances for accurate plotting of the shape. Otherwise for a scan run with the LIDAR centre pointing to the object's centre, a flat rectangular object would be plotted as a curve; with the distance to the shape's horizontal edges measured higher than that to the shape's centre.

2. Inertial Measurement Unit (IMU):

- a. Functional Requirements:** The IMU consists of three position sensors, accelerometer, gyroscope and magnetometer. The accelerometer should be able to measure the tilt angle of the PTU, the magnetometer should measure the pan angle, and the gyroscope should provide a measurement for both the tilt and pan angles of the PTU.

i. Inputs: No inputs

ii. Process:

Gyroscope: When the PTU is moved, the gyroscope measures raw values of the angular velocity of the unit (radians/second). This measurement should be offset by a value to account for error, and numerically integrated with a chosen timestep to provide the angular displacement.

Accelerometer: The raw values provided by the accelerometer measures the acceleration of the PTU in all 3 axis, x, y and z. The tilt angle should be determined from the accelerometer by using the x and z axis. Inverse tan of the ratio of these axis will provide the tilt angle.

Magnetometer: The magnetometer provides raw values on the relative position of each x, y and z axis from North. Inverse tan of the x and y values will find the pan angle in radians. This should then be converted to degrees, in order to be able to compare with the current servo position.

iii. Outputs: Tilt angle (degrees), Pan angle (degrees)

iv. Timing: The IMU sensors should have a delay to ensure readability of measurements. However, this delay needs to be synonymous with the change in the servo position in order to accurately provide a second reading for the position of the server. Furthermore, the delay should be small enough to

correspond with a small time-step change (dt), to ensure a more accurate numerical integration.

b. Performance:

The IMU sensors should be able to measure the pan and tilt angle of the PTU to accurately to within one decimal place of the degree. The degree measurement will need to be a two digit reading, so this level of accuracy is sufficient. The IMU is prone to a large amount of noise, hence smaller deviations measured had to be filtered out to prevent the readings from drifting over time. The Gyroscope, specifically, is unable to cope up with sudden movements at high velocities, it often leads to the reference point being offset, i.e. if the PTU is moved very quickly from 0 to 90 and back, it does not return to 0, and all readings henceforth, are offset by that value.

c. Interfaces:

The IMU sensors will interface with the DragonBoard via an I2C serial bus, thus synchronously sending data to the DragonBoard.

d. Design Constraints:

The tilt angle and pan angle are restrained by the hardware design of the PTU. As a result, the gyroscope accelerometer and magnetometer will only be able to measure a maximum of 90 degrees to -90 degrees (from the centre) in the pan axis, and less than 90 degrees to -90 degrees in the tilt axis (the wiring of the PTU restricts further movement in this axis).

3. DragonBoard MCU:

a. Functional Requirements: The DragonBoard should be able to communicate between the PTU and PC. Upon receiving inputs from the user, the DragonBoard should be able to read the LIDAR distance reading at particular servo positions, and then serially transmit this to the PC in order to plot an appropriate scan.

i. Inputs: User input either from the PC or on the board itself determining the following; Resolution of servo movement (pan and tilt angle changes (degrees), number of lidar samples, scan delay (ms), scan type (rectangular or full scan)).

ii. Process:

The user input modes should serially input from the PC into the DragonBoard. The DragonBoard should then output appropriate servo positions (determined by PWM signals) to the PTU. At each servo position set by the DragonBoard, the DragonBoard should input LIDAR values (again determined by pulse width measurement of PWM). These values should then be output from the DragonBoard to the PC, along with servo position values, to plot an appropriate 3D scan.

DragonBoard; This is a similar process to the PC. However, the user input values should sent through push-button values on the board rather than through the PC.

- e. **Outputs:** LIDAR distance measurements (this will be a polar measurement)
- f. **Timing:** The LIDAR distance measurement should be serially input to the DragonBoard before the next change in the servo position. The program needs to account for the fact that multiple LIDAR measurement samples may be requested by the user. These multiple samples will need to be transmitted before the change in servo position (this should not be a significant issue, due to a short range lidar measurement frequency of 1000 Hz) Furthermore, in order for live plotting to occur on the PC, the distance measurement should be output to the PC before the next change in the servo position occurs.

2. Performance:

The DragonBoard was setup to work with floats and doubles in the program by enabling the 32 bit floats and doubles in the project setup.

- 3. **Interfaces:** The DragonBoard should interface with the PTU via the I2C serial bus. It should also interface with the PC via the serial port. Additionally, a user interface should be created on the DragonBoard, to implement the previously stated input conditions.
- 4. **Design Constraints:** The DragonBoard is limited by the amount of memory available on the board. As a result, the program uploaded to the board will need to be concise, with any processes that can be done outside of the board being executed on the PC instead. Hence, the median filtering and noise filtering is executed via the program on the PC.

4. PC:

- 5. **Functional Requirements:** The PC should be able to create a 3D live plot of all incoming measurements from the LIDAR (in conjunction with current servo position), as well as a final, filtered scan, showing an accurate image of the target object. It should also be able to calculate the area of the scanned object.
 - a. **Inputs:** LIDAR distance measurements, pan angle, tilt angle from the Board. If the Interface Mode is set to PC Mode, then the PC should receive the input for the scanning parameters to send to the Board.
 - b. **Process:** The PC should convert all incoming LIDAR distance measurements between the specified range of 0.8 m to 1.2 m (in combination with servo positions) from polar to cartesian coordinates. It should then filter out excessively inaccurate data points and plot these points on MATLAB. Consequently, it should use this plot to calculate the final area.
 - c. **Outputs:** 3D plot of scan, live plot of individual points during scan, total area calculated
 - d. **Timing:** The live plotting on the PC should occur before the servo positions are changed. As a result, all LIDAR distance measurements need to be serially input into the PC within this time period, to avoid significant delay whilst live plotting.

e. Failure Modes:

- 6. Interfaces:** The PC will interface with the DragonBoard via the serial COMS port. Additionally, the PC will have the option of user interface, whereby users can input aforementioned key input values.
- 7. Design Constraints:** The PC requires a Windows operating system, since the DragonBoard operates on CodeWarrior, which is a Windows based software.

Conceptual Design

Servo Motors: The Servo Motor module has been designed to have two major inputs; one being an input of a pan and tilt angle to rotate to any given position within a range of 20 degrees to 160 degrees, and the other being an input of resolution by which the servo pan/tilt angle will periodically increment during a scan. The first input was chosen in order to test whether the servo was able to accurately move to all specified positions. The second input was chosen due to the fact that by specifying a degree of resolution, the angle with which the servo would change at each increment would be known. This could then be combined with the distance obtained from the LIDAR to gain polar coordinates of a particular point.

LIDAR: The LIDAR module was designed with significant consideration toward the number of samples taken in one particular increment of the servo motors. The number of samples taken by the LIDAR is dependant upon user input, with a higher number of samples giving greater accuracy, but creating a higher delay within the system. Additionally, the samples are then filtered out to one measurement by taking the median of each set of values. The use of median rather than average to find a final distance measurement filters out any outliers in the samples taken.

IMU Sensors: The IMU sensors are a back-up procedure to check that the servo positions of ESCAPE are accurate. However, they have not been integrated as a step by step checking module with the Servo module, as the IMU sensors or the Servo motors may have a small degree of inaccuracy. Individually, these small inaccuracies would not prevent the servo motor and thus the system from achieving a scan of an object. However, if the IMU sensors were implemented as a feedback loop to the Servo, any discrepancies with the sensors would cause the system to delay in its scanning, and would thus overcomplicate the process. Using the IMU data as a user back up (to be able to go through printed values if there are any significant errors in servo positions), was therefore a more feasible solution.

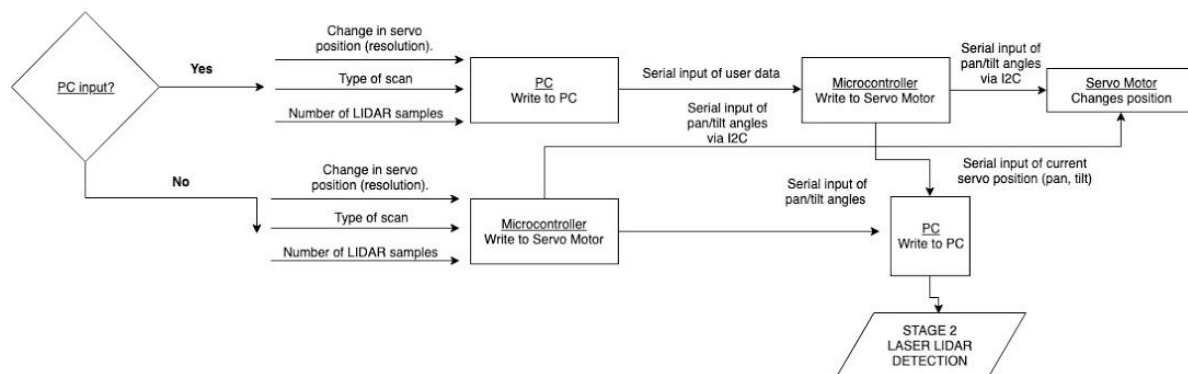
DragonBoard MCU: The DragonBoard was programmed to interface with the sensors, and present a user-friendly interface for any user to navigate through the menus and perform the scans. The DragonBoard also acts as an intermediate between the sensors and the PC, helping to convert all the raw data into processed data. The DragonBoard was also designed to form an important part of the interface. The keypad on the DragonBoard is used to receive inputs from the user, and set the scan parameters. The LCD and the 7 segment displays on the DragonBoard also help to display the current real-time information from the sensors,

including the current orientation in the form of pan and tilt angle and the distance measurement.

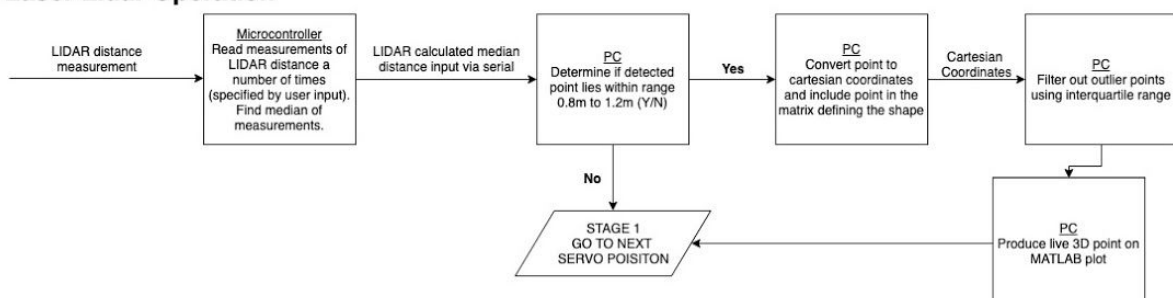
PC: The PC's primary function in the system is to post-process the data by using the information transmitted from the DragonBoard to produce plots of the shape/object that the PTU has detected. The PC's functionality was built on MATLAB, and a serial port was setup to communicate with the DragonBoard for receiving and sending data. The PC also works as part of the interface, as the user may choose to input the information before the scan through the PC itself rather than the DragonBoard. The PC was designed to have 3D real-time plotting and offline plotting. The PC is also given the functionality of median filtering, to reduce the impact of noise on the final plots.

The following is a functional block diagram of the conceptual design of the system.

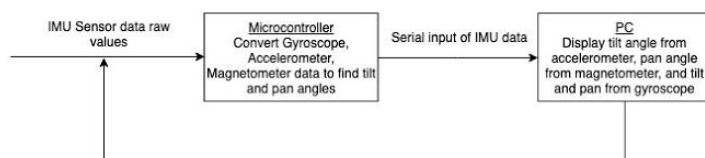
Stage 1 Go to Servo Position



Stage 2 Laser Lidar Operation



IMU Sensors



User Interface Design

The user has two methods to interact with the system. They may choose a board-based interface or a PC-based interface, which is determined by the user after program initialisation. The 'Interface Selection' allows the user to choose if they would like further inputs to the board is to be sent from the keypad on the board or the terminal on the PC. A series of inputs following this will set the parameters and type of scan, including setting the resolution, number of samples, and type of scan.

In board mode, inputs are received through the 16-button keypad on the right side of the HCS12 board. The number keys are used for numerical input or option selection, the asterisk key is used for entering a decimal point for float-supported user, the 'D' character key is used as an enter key and the 'C' character key is used as backspace. The output to the user is displayed through the blue two-row LCD display towards the top left of the board. When options are available for the user to select, they will appear on the LCD display in the format "1-Option". The number preceding the text indicates the keypad button press to select that option. When the board is open to receiving numerical input, a prompt string ">>" will display at the beginning of the second LCD row. When this symbol is visible on the display, the user can input a string through the keypad buttons, pressing the 'D' key to confirm.

In PC mode, user inputs are delivered to the board through the serial communications interface. The PC interface is built in MATLAB using the serial module and will communicate with the board to exchange input and output. User input is provided through entering the information into the MATLAB command window, presumably through the keyboard of the PC. When the program is expecting input, the command prompt ">>>" will be displayed. Three right arrows are printed to distinguish it from the default MATLAB command prompt ">>". Output will also be displayed through the command window. Options available for the user to select will be printed out in the same format as in board operation.

User Inputs

The user is required to enter a range of inputs before the system executes the scanning.

a. Interface Selection:

This determines whether the following inputs for the scan parameters are to be received from the keypad of the Dragonboard or the terminal of the PC. The interface mode is set to 'Board Mode' by selecting any button on the keypad and the interface mode is set to 'PC Mode' by sending any input from the serial terminal on the PC.

b. Resolution:

Resolution determines the increments at which the servo is supposed to increment. The resolution is a user input of a float that should represent the angle increments that the user would like the system to operate at. The input angle will then be converted into the respective PWM signals that are needed to be written to the servo motors to orient itself to the new position.

c. Samples:

Samples is the number of LIDAR samples that are required to be collected at each point scanned. The median of these values are taken to give the final distance value, which is then passed on to the PC. The user is expected to specify an integer value for this number.

d. Scan Delay:

Scan Delay is the customisable delay time in microseconds between each scan at each new location. This is also received as an input from the user from the Board or the PC, and is an integer value. The Scan Delay also gives the sampling frequency, the lowest value for this (maximum possible frequency) is 0 ms delay.

e. Scan Type:

Scan Type is the input that determines if the system will execute a Rectangular Scan or a Full Scan. The Rectangular Scan, as defined above is specifically for detecting rectangles, hence, this scan will be executed with the same procedure even if the shape is not a rectangle. Full Scan, on the other hand, is able to detect more complex shapes, and can be called by the user if a general scan is required.

f. Servo Angle:

In the Test Mode for the servo motors, the user may input angles as floating point values. These values will be converted into PWM signals, which are then written to each of the servo motors to change to the new position.

Interface Outputs

a. 7 Segment Displays

The 7 segment displays are used to display the median distance that is measured from the PTU by the LIDAR sensor. This is a floating point value that is written to the LEDs using interrupts. Interrupts are triggered when a new distance measurement is taken, and the LED is updated with this new measurement.

b. LCDs:

This displays various kinds of information at different stages. After installation, the LCD helps to guide the user through the various options and set the parameters for the scan. In test and scan mode, the LCD displays information about the PTU's current orientation, including pan and tilt angle. In the test mode, the PTU is used to show the angles calculated from the gyroscope, accelerometer and the magnetometer individually.

c. PC Terminal:

The PC terminal is also an important part of the interface. Similar to the LCD, this works as both: a way of inputting parameters to the system and receiving the output from the system. While the terminal, like the LCD receives data about the pan angle, tilt angle and the distance, it also processes these outputs to help generate a plot of the object it has detected within the range. The system outputs several different plots, including raw data plots and processed plots, where the noise has been filtered and the coordinates have been translated from polar coordinates to rectangular coordinates. In PC Mode, the terminal prompts will also be able to guide the user to enter correct values for the scan parameters through the use of input validation.

Input Validation and Error Trapping

All user input is validated in the program to prevent errors in operation. This is achieved by placing user input receival in indefinite while loops, only breaking once an explicit condition has been satisfied. There are two types of user input: character and string. Character input is used for selecting between different options displayed on the screen. String input is used to enter numerical values. In character input, a button corresponds to each of the available options, and if button press has not been detected, the program will not proceed. In string input, the program will try to convert the input to a numerical value, and only proceed if it can successfully convert the value and if it falls within a certain valid range (specified according to the limits of the hardware).

To reduce chances of error, once the Interface Mode has been chosen, the other mode of input is disabled to avoid sending extra inputs to the system by mistake. For example, if Board Mode is chosen, then the serial interrupts used to receive input from the PC terminal are disabled for the rest of the session. Similarly, if PC Mode is chosen, then the keypad would be disabled to ensure that no inputs are given from the Board's keypad by mistake that may interfere with the session. The session needs to be restarted to be able change select the Interface Mode, but in one session, the Interface Mode cannot be changed..

Hardware Design

Scope

What we didn't design:

- The mechanical skeleton where the servos rest in and rotate.
- The electrical circuit connections between the servos, lidar, and the IMU.
- The connections between the microcontroller within the board and the external peripherals. Specifically for what we used was the LCD screen, the 7 segments, the HCS12 keypad, and the serial ports/ USB 1.0 ports.
- What we designed
- The software information connections between the seperate devices.
- The interactions between an external computer and the unit.

Below is a basic outline of the all components of the system (lidar, IMU, servo, HCS12 board, and external PC) and how they were integrated together both as hardware and software. But will less emphasis on mechanical design that is implemented in the system and more content connecting the given hardware .

Power Supply

The two general sources of power for the system. One is the serial port connect to the HCS12 board. This comes from an external PC and doubles as the way of importing code

into the system during development. However, this would not be the case in the product as software will not be changed however the Dragon HCS12 board still requires power for the microcontroller through the serial port. Most of the control of the power would be handled within the device that powers serial connection such as a USB2.0 connection on a computer. This power supply to the board should be approximately 5 Volts, 500 mAmps which is the common output of USB port charging.

The other power source is a powerpoint mounted AC to DC power converter that outputs a 5 Volts, 3 Amps. This is then connected to the external peripheral and it powers the IMU, lidar, and servos. This is dedicated power supply and grounding and fusing internally. The reason that the other devices required an external power source was because, servos require more power than either the board or the rest of the sensors and 5V 0.5A would not have been enough to ensure that they are effective.

Overall neither of the parts of the system use enough power or require enough sensitivity to provide large need any kind of large scale grounding or specific measures to limit the effects of static charge within the system. Instead we rely on the quality of electrical input to the system to ensure that it doesn't damage the circuitry.

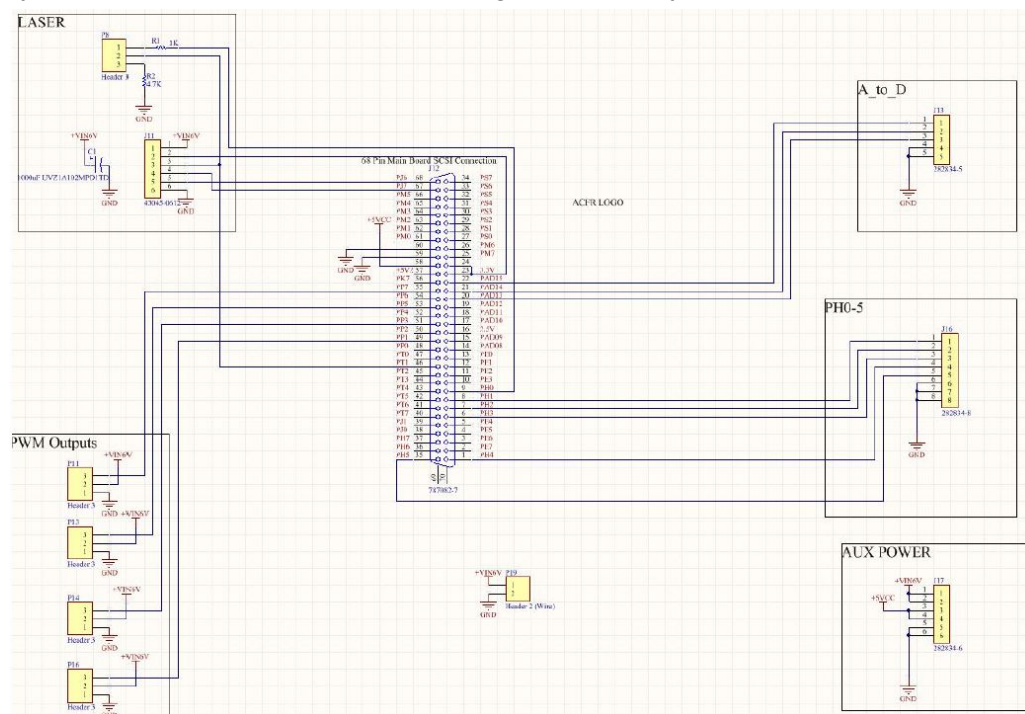


Figure of the Power /Connection circuit design of the Pan and Tilt Unit.

Note: When connecting the Dragon 12 Board and the Pan and Tilt Unit. To ensure the electrical interconnection between the unit of different power supplies. An order must be followed.

1. Connect the 69 Pin Ribbon Cable with both the board and the Pan and Tilt Unit.
2. Power the Dragon 12 Board (By connecting the USB to the board).
3. Connect the power to the Pan and Tilt Unit.

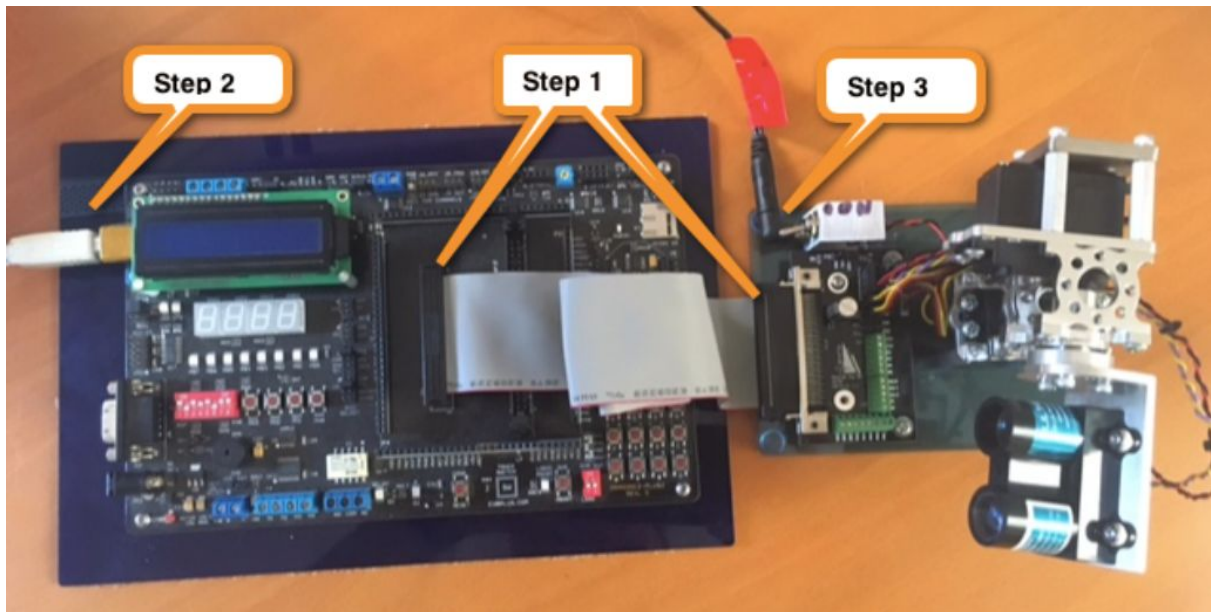


Figure shows the steps to connect the board and the Pan and Tilt Unit.

Computer Design

There are two primary computational devices in this system. The first being the Dragon HCS12 board and secondly an external PC.

The HCS12 board is connected to the LIDAR sensor by a 69 pin cable. Each of the different sensors and actuators have allocated pins to the cable and each of them have a corresponding ICC bus connection that is used by the PC to ensure effective transmission of data from and the PTU to the board.

The external PC is connected using a USB serial port on the board to the computer. This connection is two direction as the PC can determine the setup of the session. And the board will always output the current angles and distance through this serial connection to the PC.

The board that must be connected to the PTU must be the HCS12D Family (MC9S12DG258B). As this is the only type of microcontroller that has been tested and other boards will be similar but there is no assurance that it will work.

The requirements for the PC is very minimal. It requires a way to receive a serial signal, commonly through a USB port, and also the capacity to run MATLAB. For MATLAB the graphing would be dependent on the specifications of the PC taking into account the RAM and processor.

Sensor Hardware

The two primary sources of sensor information in the product is the lidar-lite v2 and the GY801 Inertial navigation module. The following is a detailed description of the sensors and the uses for each of them.

LIDAR-Lite v2:

Description: This is a sensor that sends out a laser then measures the time that it requires for it to return to the sensor.

Input: None, this sensor constantly reads distance and doesn't require any bus activation as it isn't connected to the bus.

Output: A PWM signal that can then be converted into a distance reading. All of the data can be directly read from one of the pins from the 69 pin ribbon connector cable. Thus to access the data all that is required is to poll the current signal being outputted.

GY801 Inertial navigation module

Description: This is an all in compassing unit that contains 5 separate sensors.

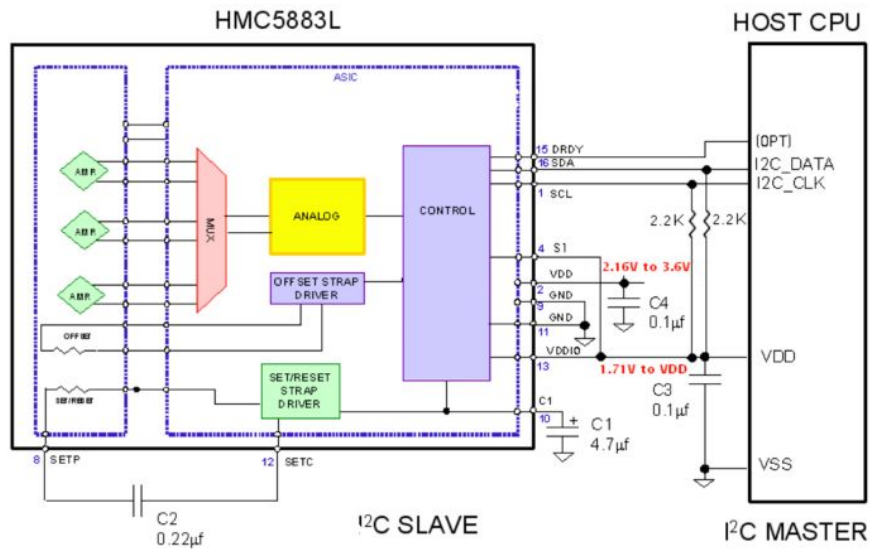
1. 3 Magnetometer (HMC5883L): This measures the magnetic field strength in three different orientations
2. 3 Accelerometers (ADXL345): This measures the acceleration of the chip in three different orientations.
3. 3 Gyroscopes (L3G4200D): This measures the velocity of 3 different rotations in the chip.
4. A Thermometer (Note Unused)
5. A Barometer (Note Unused)

Input: Each of the sensors needs to be called using the IIC bus and given a correct initialization of what kind of results that you want from each of the sensors then there is no needed extra input information.

Output: When the IIC corresponding to each of the sensors is called it will output a signed 16-bit number. None of the outputs are filtered and so there would need to be gain conversions from the different sensors to their actual values. All of the information is converted locally from analogue to digital by the sensors and so all of the outputs are digital outputs.

The following are schematic diagrams provided by the manufacturer.

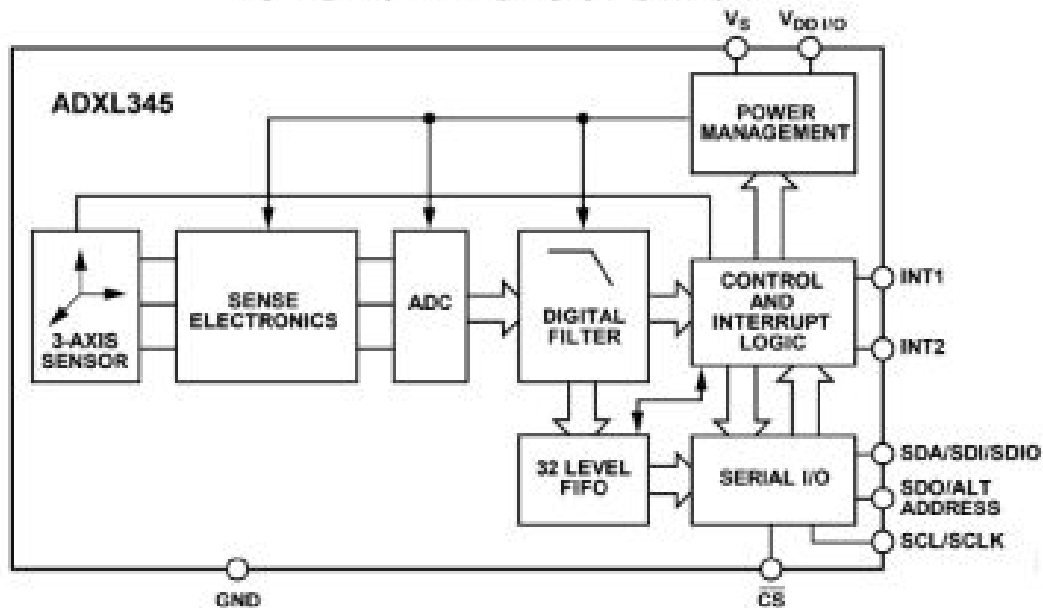
INTERNAL SCHEMATIC DIAGRAM



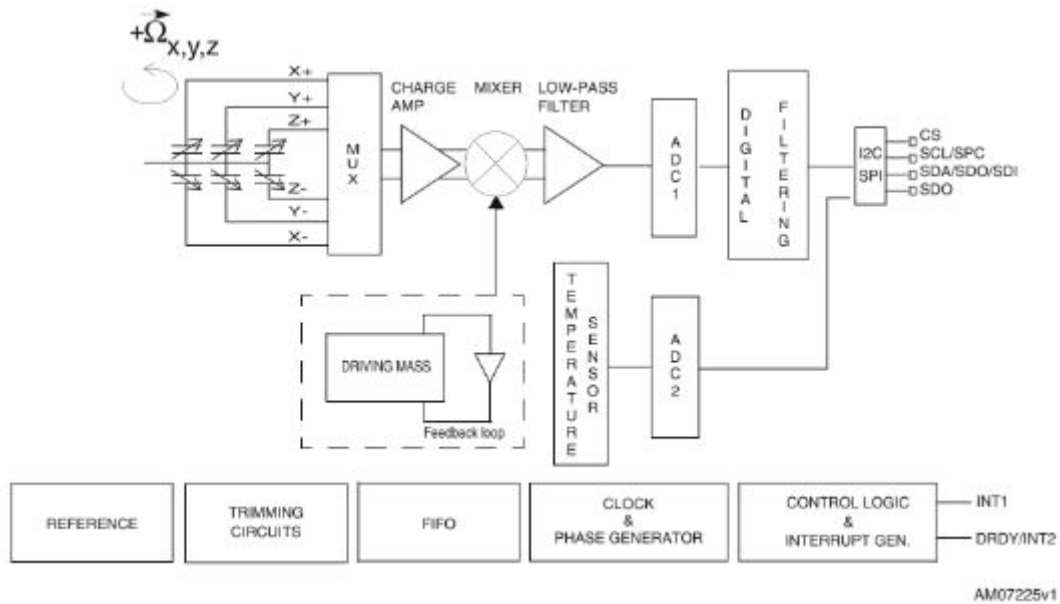
www.honeywell.com

Accelerometer FBD

FUNCTIONAL BLOCK DIAGRAM

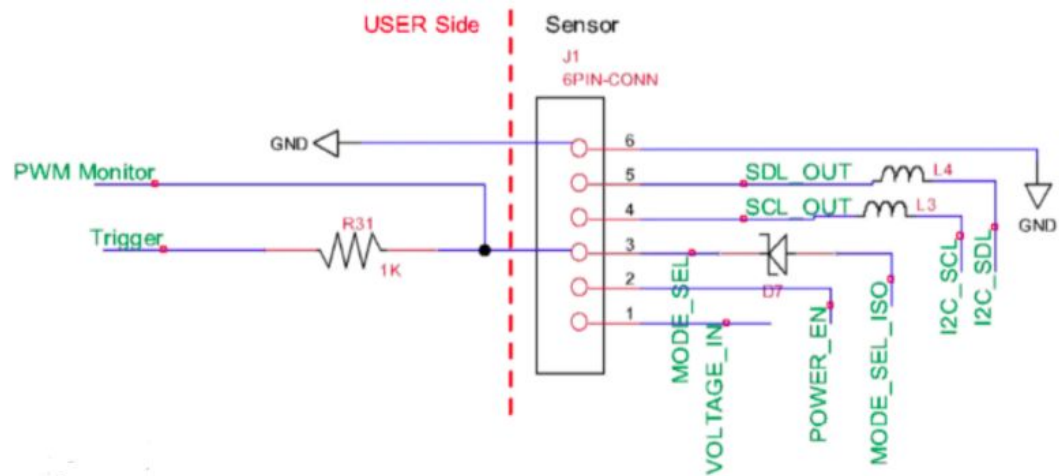


Gyroscope

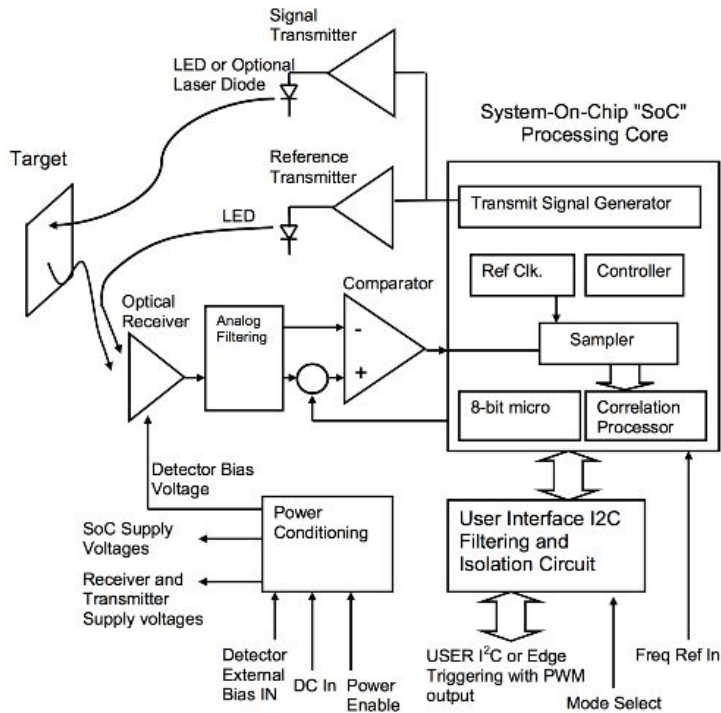


LIDAR: System Diagram and Block Diagram

(Sourced from LIDAR datasheet.)



LIDAR-Lite Block Diagram



For further in-depth information on the sensors refer to the referenced datasheets.

Actuator Hardware

The one actuator in the system is a pair of servos (HITEC HS-422 Servo) where one corresponds to pan and the other to tilt.

Overview: Servos are a kind of step motor and it keeps track of it's position. This allows it to rotate an object into a specific range of rotational orientations.

Input: Each of the servos require a PWM signal. The corresponding angles converted to PWM signal is mapped to around 5 - 15% duty cycle.

Output: There is no feedback from this actuator about it's position to the system. But given a certain PWM signal it will rotate a rod to a certain angle taking whatever is connected to the rod with it to that position. In the case of this production one of the servos would change the tilt of the lidar and the other would change the pan angle of the lidar.

Operator Input Hardware

This product has two types of input into the system to choose the type of scan or test. They are directly onto the board through the keypad and the PC's keyboard. Both have the same options and the alternative between them is just ease of use for the user.

HCS12 Keypad

This is the inbuilt HCS12 keypad. It is a grid of 4x4 buttons that allow the user to select inputs into the system. Only one of the buttons on the board is hard coded and that is the reset button which is not part of the keypad. Each of the buttons have a symbol next to it for ease of use. It forms a 3x4 grid which contains each of the numbers in the typical phone grid style layout. And then the letters A-D on the left side of that. The pressing of each of the buttons creates a digital change that needs to be picked up by the software.

PC Keyboard

The alternative option for inputting instructions into the system is by using the computer and sending commands through the serial port. Through the USB connection between the computer and the board.

Operator Output Hardware

There are two ways that the system displays the current data that it is measuring. One way through the digital outputs of the board (LCD display and 7-segment Display) and the other through outputting the data through the serial port into the PC, where it is then displayed on the screen.

LCD Display

This is an integrated peripheral into the HCS12 board. It has its own direct port connected to the processor and doesn't require any extra set up. It has two output rows that can be selected and once written to will remain visible until a clear or overwrite.

7-segment Displays

This is another integrated peripheral into the board. It is a lot simpler display mechanism and such requires a more significant software approach as the 7SEGs are linked up in parallel and any signal written to one is also written to the rest of them. There are 4 in total on this board. So it requires the turning off of the other 3 7 SEGs and a flickering between the different digits.

SCI1 Port

The last way that the product can output the data is through the serial port as it sends all the data to the computer which then displays the information on the screen in different formats depending on the PC's options.

Hardware Quality Assurance

To ensure that the quality of the hardware we completed validation and calibration for each of the different parts of the system. This was to ensure that each of the different parts worked and were getting accurate results. Similarly we completed integration testing for the combination of all the hardware to ensure that all of the aspects connected together and working as intended. This ensures quality as it has been completely tested and the overall product has been tested together.

We have also completed full reset testing to ensure that the system works even after being turned off in different circumstances. This provides hardware assurance as random problems do not provide extensive issues to the overall product.

Hardware Validation

The primary way to ensure the validity of each of the different devices was to create a test function that went through each of the different hardware requirements and then ensured that they were working. We completed this for the IMU (including gyroscopes, accelerometer, and magnetometer) for the LIDAR, and the servo.

IMU Validation: We used a function that constantly collected the raw IMU data (gyro accel and magnet all obtained individually) and then outputted the raw results through the serial port into a terminal connected to that serial port on a PC. Then we evaluated the expected results as a pattern and made assumptions to see if the results for all of the sensors were logical as they changed. This was before the inclusion of any gain constants to shift all of the results into accurate result. Then after we had validated that the results were reasonable we completed calibration procedures (Outlined below) on the sensors and added in required constants in order that the output angle measurements were approximately reflecting reality. If any of the sensors couldn't be complete the second part of the procedure and not be calibrated it was assumed that it wasn't being valid.

Servo Validation: We needed to do was put in sample PWM frequencies and if the servo changed position and stayed at that angle, then that would be a validation that the servos were working, but not necessarily accurate. In order to ensure that the servos were working and accurate, calibration needed to be completed in order to get the correct angle from the input. If after calibration the servos did not go to the correct angle then the servos would not be working.

Lidar Validation: To ensure validation we connected the PWM output signal from the lidar into an oscilloscope. As the lidar constantly outputs the PWM signal whenever turned on, then the live plotting from the oscilloscope shows the trends in the device. This would show if there was an error in the signal as in if the duty cycle was not constant over a constant distance for the lidar. Also manual calculations could be performed on the duty cycle in order to ensure that the output was working as expected.

Hardware Calibration Procedures

Servo Motor Calibration

According to the General Servo Information for Hitec servos, the servo operates through receiving a 50hz PWM signal, where the angle is determined by the pulse width. The theoretical formula is $DUTY = 2250 + 12 \cdot (\text{angle} - 90)$, where the duty cycle is a number between 0 and 100, and the angle is between 0 and 180 degrees. However, implementing the theoretical formula was found to be highly inaccurate and instead the formula was determined manually. The calibration procedure is listed below:

1. A testing program was written to output various duty cycle values to the servos.
2. The duty cycles corresponding to the 0 and 180 degree angles were found.
3. By adjusting the formula using the tested values, the duty cycle was correctly mapped to the angle.

The final formula obtained through this process was $DUTY = 2350 + 15.85(\text{angle} - 90)$. This ended up being extremely accurate; no error was observable.

LIDAR Calibration

After implementing the software to read the PWM signal on PT1 and interpret the pulse width as a distance, the distance values were found to have a constant offset of 30mm above the actual distance. The software was adjusted for this error by subtracting 30mm from each distance reading if the reading is above 30mm (to avoid the possibility of negative distances).

Gyroscope Calibration:

In order to calibrate the gyroscope we needed to clean up the raw gyro data. This was primarily done through finding the average static value that existed in each of the different orientations. The raw values for our system ranged from 65 to 85 on average for the x orientation. Then we offset this figure to be centred about 0 by subtracting 75 in this case. Then we passed the data through a high pass filter. This was to ensure that the static possibilities would not change the overall angle. In our sensors we tested to see if the output was between -100 and 100. If so we cleared the value to zero. We were able to do this because the values that we identified when the gyro was in motion was in the thousands and would not be cleared by the high pass filter.

Then we implemented an integration method to convert the angular velocity into a change in angle. We used a trapezoidal method. After this was implemented we need to apply constant gain to this number in order to get the output in angles. To complete this we found the change in angle for a 180 turn. Then used: $\text{CurrentGain} / \Delta\text{Angle} = \text{CorrectGain} / 180$. I.e. $\text{Getting Correct Gain} = \text{CurrentGain} \cdot 180 / \Delta\text{Angle}$.

Then we implemented this iteratively and used this numerical method to get a close overall angle. Then we did some manual calibration in order for the most accurate and reliable angle change.

Note: Calibration needs to be complete separately for the two orientations as the sensors are not uniform. The gain value would be different for both of them depending on the circumstances.

Accelerometer Calibration:

When we performed calibration to the accelerometer we arctan ed the raw data and then applied a basic gain of converting radians into degrees. After this we were getting almost perfectly accurate results and required no other change in the gain of the system. This might not always be the case if one of the orientations of the accelerometer of the was damaged or something similar.

Note: Since we used arctan, the units of the raw data cancelled, and we are left with an uncalibrated raw angle.

Magnetometer Calibration:

To calibrate the magnetometer we performed arctan to the raw values in order to get a pan angle then we performed a similar action to the gyroscopes and completed the iterative process of current angle/ gain and correct angle/ gain in order to get an appropriate gain value. Also the gain should take into account the conversion from radians into degrees.

Note: Since this instrument deals with magnetic fields it has interference with metal objects that are in its surroundings. Therefore the calibration of the device would need to be completed in the designated work environment for the best results.

Hardware Maintenance and Adjustment

The routine adjustments for this product is rather minimal. Each of the sensors and actuators should be routinely inspected after calibration to ensure that they still calibrated. If any of the sensors are not accurate, then a full calibration of the sensor would need to be completed again.

To increase the maintainability of the device a test mode was implemented in the software that allows an operator to perform on site testing of each of the separate sensors and actuators. This would be completed in a full inspection of the sensor. A quicker way to routinely maintenance the product would be to use a designated sample and then complete a test on it for each time and if it doesn't lie within a specific accuracy of the predetermined accuracy then a full test should be completed on the system.

Software Design

Software Design Process

The shape detection algorithm underwent significant development and changes over the 5 week process. Prior to the proposal presentation, an algorithm designed specifically for minimising the time required to scan in an ideal scenario was devised.

This method was designed under the assumption that noise from the LIDAR would be minimal and that the positional data would remain accurate throughout the entire scanning process. The basis of the design is that the system scan diagonally in a downward-rightward direction from the top left corner. Once the system detects an edge of the shape, it will begin tracing the outline of the shape and storing the positions of each edge point it traces. While tracing this outer outline, it will record the minimum x and y values and maximum x and y values of the shape. Once the scan returns to the position at which it started tracing the outline of the shape, it would begin scanning the rectangular region bound the minimum and maximum x and y values for any holes in the shape. Once the system detects a hole, it will trace that hole, and store the locations of each of the hole edge points, and once the hole has been completely traced, the system will return to the start of its most recently interrupted sweep and continue scanning for more holes, employing the same process as described above for each new hole discovered. The positions of all the hole edge points would be stored in one matrix, and the positions of all the edge points of the outer edge would be stored in a separate matrix. Then, using the Matlab 'imfill' function, the shape enclosed by outline of the outer edges was filled in matrix 1, and each of the holes in matrix 2 were filled, then matrix 2 would be subtracted from matrix 1, to get a matrix with a depiction of the actual shape. Each point in this matrix depiction represents an area determined by the scanning resolution selected. The total area of the shape could be estimated by finding the number of points then multiplying that by the area represented by each point.

The edge tracing algorithm in itself is quite complex and will be described in detail below. Essentially, at each given point in time, there are two options for the scanner, move vertically, or move horizontally, and it will alternate between these two options. However, the direction in which to move vertically (up or down) and horizontally (left or right) must be determined. To do this, we must take into account the result from the previous scan and the current scan. If the scanner has moved horizontally and there is no significant change in distance from the previous scan and the current scan, the direction in which the scanner moves vertically must be toggled. Conversely, if the scanner has just moved horizontally and no significant change in distance measured by the lidar has occurred, the direction the scanner moves in the horizontal direction must be toggled. However, if there is a change in distance between the previous scan and the current scan, the scanner will continue to alternate between vertical and horizontal movement, without toggling the directions of either of these options.

However, this form of shape mapping was realised to be somewhat unviable given the limitations of the equipment. Since this algorithm requires an idealistic environment with little to no noise and errant measurements, it would likely fail to accurately trace the edges of the shape with the equipment used, as random errant measurements would throw it off and cause it to differ from the required path. Resultantly, after initial experimentation with the equipment, it was decided a full scan implementation would be more accurate.

This sort of scanning method would be very slow, however it should prove to be relatively accurate. It would entail scanning the entire range of motion of the pan-tilt servos and filtering out the points that aren't within a desired distance from the lidar, resulting in a plot of most of the points on the shape, plus a few random points from incorrect measurements from the lidar. Dissimilarly from the previous algorithm however, the points would not be stored in a matrix with scanner's x and y positions representing the index of the matrix, and the depth measurement as the value of each index, but instead would be stored in three separate arrays for pan angle, tilt angle and depth measurement that grow in size with each measurement. These arrays would then be converted from polar coordinates to cartesian coordinates using a method that will be discussed in greater detail in the *Software Design Description* section. Once the arrays have been converted, they are used as x,y and z coordinates to form a 3 dimensional-plot of the shape using the matlab function scatter3.

Once significant progress was made with the above method, it was realised that the team now had enough time left before the presentation date to design some more time-efficient methods of scanning the object. As a result, two new forms of scanning were birthed. The first of these, was the 'Rectangle Scan' method, which was a scanning algorithm designed specifically to find the area of a rectangular shape placed in front of the LIDAR. The second scan, from now on referred to as the 'Full Scan' method, would work for any shape placed in front of the lidar, but took several shortcuts to greatly speed up the process from the original full scan method. Both of these methods will be discussed in detail in the *Software Design Description* section, as they both pertain to the final design.

Furthermore, the process of development for interfacing an intuitive MATLAB UI and on Board UI was quite quick.

Software Quality Assurance

Throughout the project integration process, the code was extensively tested by ensuring that each separate module was functioning exactly as intended. Upon completion of the project, the final code was re-evaluated and certain parts were rewritten to minimise code repetition and maximise efficiency. In the program file, the code has been documented and commented in great detail, such that any unfamiliar user should be able to understand the program. Overall, the code is of considerably high quality in both code and documentation standards.

Software Design Description

Final Software design:

As aforementioned, the two system final designs for scanning were the Rectangle Scan method and the Full Scan Method.

Rectangle scan:

This method was made purely for efficiency when calculating and plotting an area for a rectangular shape. It works on the assumption that the shape placed in front of the lidar would be vertically centered, ie, that it would be in line with the lidar across the horizontal plane. Given that these conditions were met, the algorithm would start by pointing the lidar toward the centre-left most part of its scanning range, then would gradually scan across to the centre right, reading in the distances as it went. As it did this, it would record points that were within a distance of 0.8m and 1.2m - as it is specified within the specifications sheet that the shape would be placed within this range - and would calculate the distance between the leftmost and rightmost points that fall within that range. When a range of 0.8m to 1.2m is specified, we are referring to the the point observed by the lidar being between a perpendicular distance of 0.8m to 1.2m. The conversion process for converting from polar to cartesian measurements will be discussed in more detail further on in the section. Once this distance has been recorded, the scanner will move to the midpoint of those two points and move down to the bottom of its scanning range. It would then gradually move up to the top of its scanning range while repeating the same process as the horizontal scan, recording points between 0.8m and 1.2m and calculating the distance between the bottom and top of the rectangle. Once these two horizontal and vertical scans have been completed, the area can be calculated by simply multiply the horizontal distance by the vertical distance. And to find the distance between the shape and the LIDAR scanning array, the median of all the recorded distances would be taken. It was decided to take the median point as it is a measure of centre that is least affected by outliers, of which there were bound to be many due to noise from the LIDAR measurements. Finally, to generate a plot of the shape, a MATLAB function was developed, which takes in parameters (x,y,w,h) which correspond to the corner of the rectangle in coordinates, which will be given by the recorded minimum horizontal and vertical values from the scan and w and h refer to height and width of the rectangle, which in turn will be given by the recorded horizontal and vertical distances from the scan. This rectangle would then be plotted in a 3-dimensional plot using the scatter3() MATLAB function.

Full Scan:

This method improves on the original Full Scan method by reducing the region that the LIDAR must scan. From the specifications, we are given that the shape will be within the bounds of 40cm by 40cm. Again, this algorithm assumes that the shape will be centred about the same horizontal plane as the LIDAR.

Scanning will begin with the lidar point to the centre left most point within its scanning range, then gradually scanning across to the right until an edge is detected. An edge is said to be detected, if three consecutive points are found to be within a perpendicular distance of 0.8m to 1.2m from the lidar, (Again, the conversion from polar to cartesian measurements will be discussed later). Once this left edge of the shape has been detected, the scanner will move to point to the centre rightmost point of its scanning range, and repeat the above process except this time scanning toward the left, to find the right edge of the shape. Once the left and right edges are found, the mid point will be determined, then the scanner will move to the bottom most point in its range that is vertically aligned with this midpoint, and will then begin scanning upward to look for the bottom edge. Once the aforementioned conditions for edge detection are met, the scanner will then move up to point at the top of its scanning range. It will then scan downward to look for the top edge of the shape. Once an edge on each side is found, a much smaller rectangular region in which the shape is than the full scanning range can be defined. By defining each edge to be the scanning resolution + 2 degrees away from the centre, errors caused by slightly concave and angled shapes can be avoided. As such it is assumed that the shape does not have any extremely steep slopes or is quite concave, otherwise a different method for determining a smaller scanning region must be determined.

Once this smaller region is determined, the scanner will move to point at the bottom left of this region, and will then begin to scan to the right. As it scans, each distance measurement and pan/tilt angle from the servos are sent through serial communication to the computer to be processed by matlab. MATLAB will sort, convert and filter the raw data through a process that will be described in detail later. Once the scanner has gone as far right as it can for a single scan, instead of moving 1 resolution up, and scanning back toward the left, it will immediately return to the left and then move up one resolution and again begin scanning toward the right. This is because it was found that the LIDAR yielded more accurate results when this was done. This process would then be repeated until the scanner reached the top of the smaller scanning region. Then, matlab would process all data and create a final plot of the shape.

MATLAB Data processing:

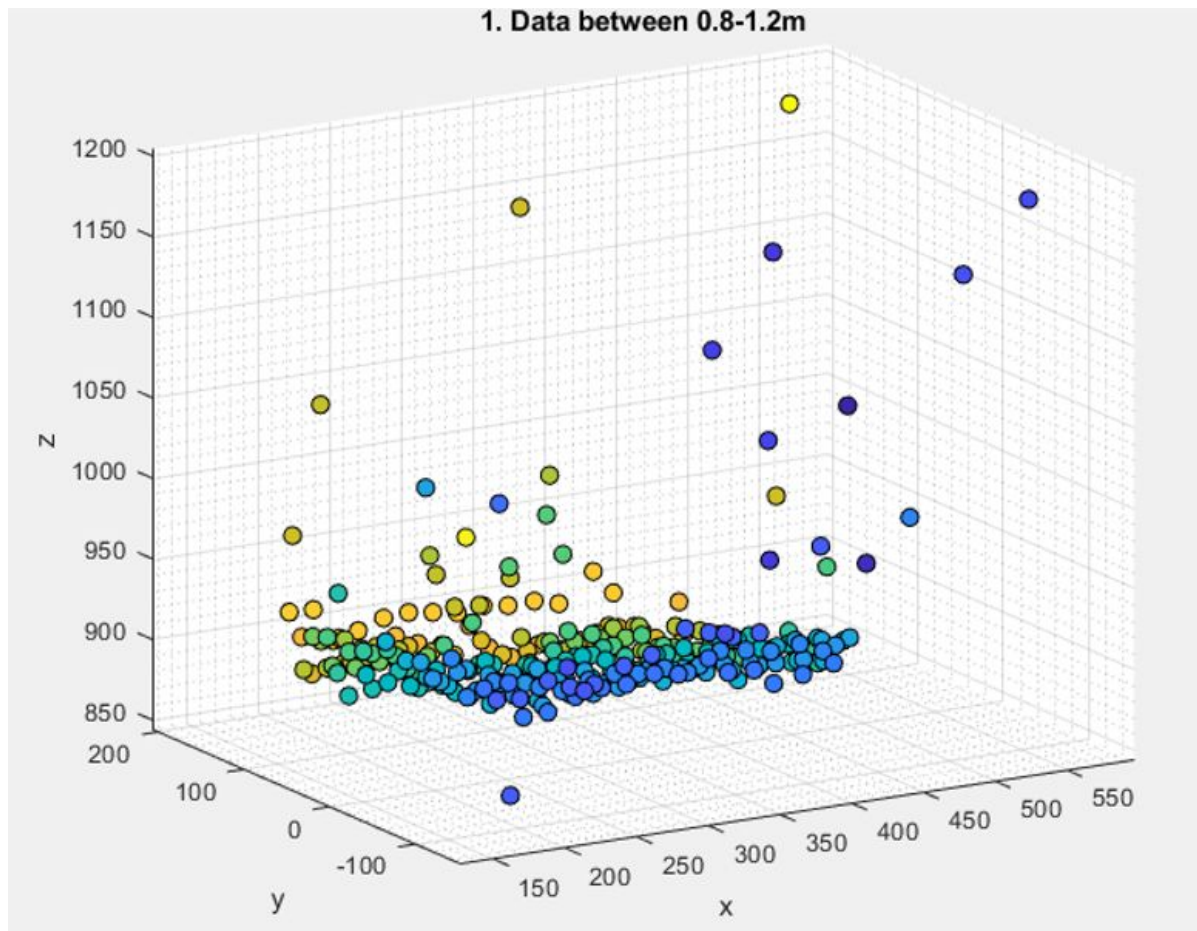
Processing the raw data from the board occurs in three main steps: converting the data from polar coordinates to rectangular, filtering the data, projecting the data, then evaluating the area.

Polar to Cartesian Conversion

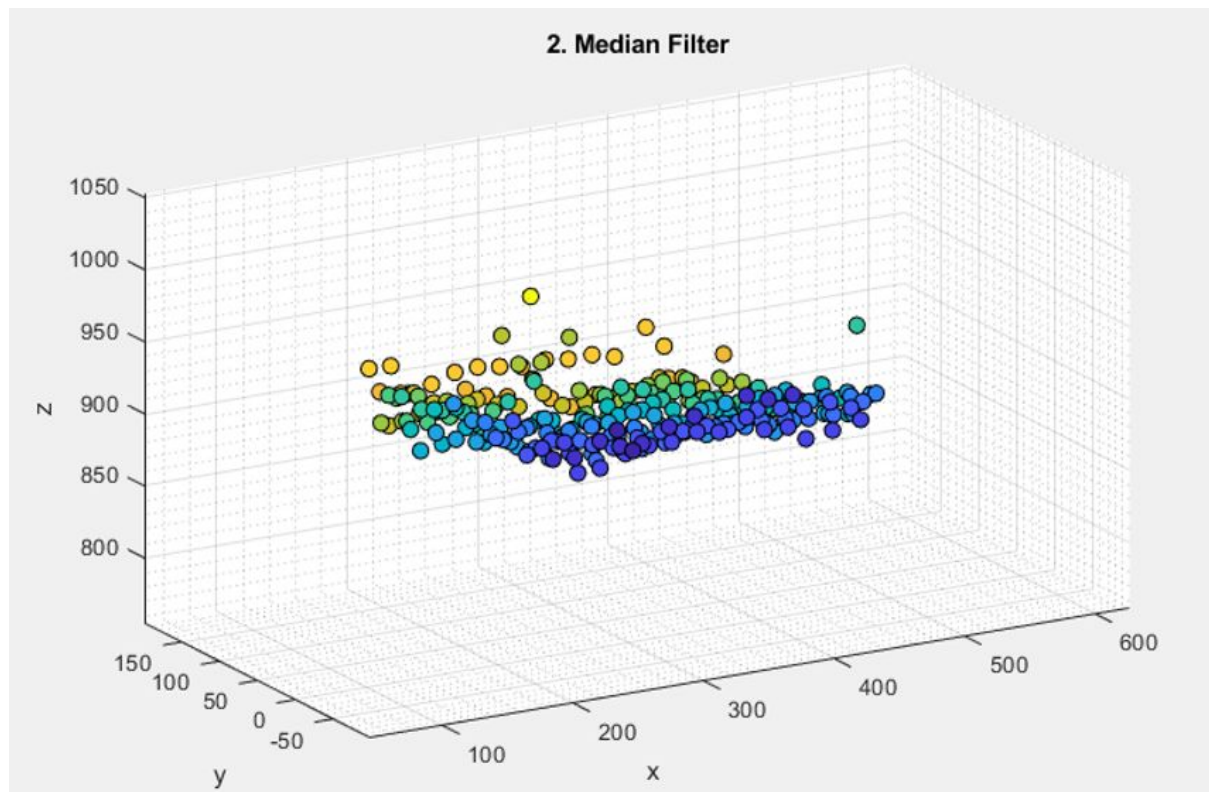
From the board, each data point is defined in space through the polar coordinate system: two angles (pan and tilt) and a distance from the origin (r). This was converted to Cartesian coordinates.

Data Filtration

The filtering of the converted data employs various common statistical methods. The first filter applied removes all data outside the range of 0.8 to 1.2 metres. This step removes the background of the scan. The result of this filter on an example shape is shown below.

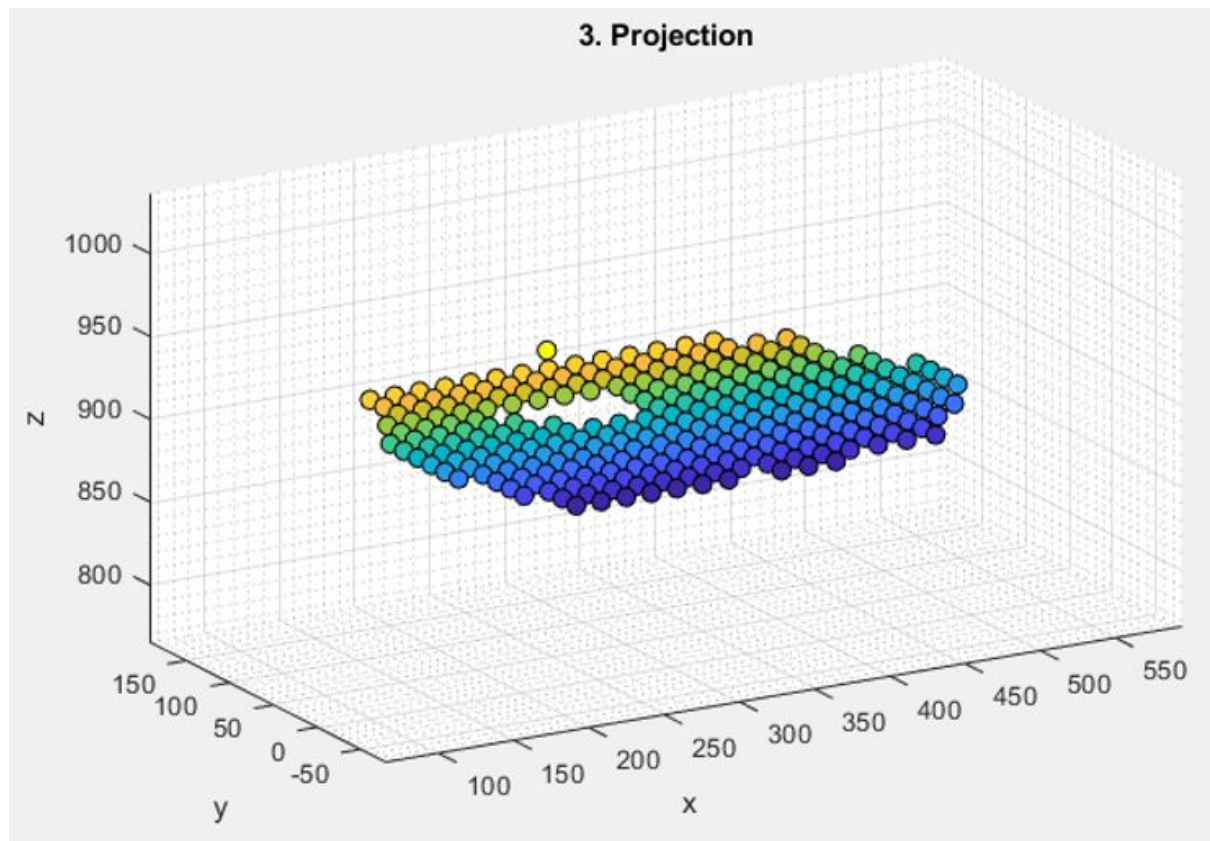


The second filter removes the noise which frequently occurs near the edges of the shape as evident in the image above. This is achieved by filtering out all points more than 40mm away from the median distance (z-axis). The result is shown below.

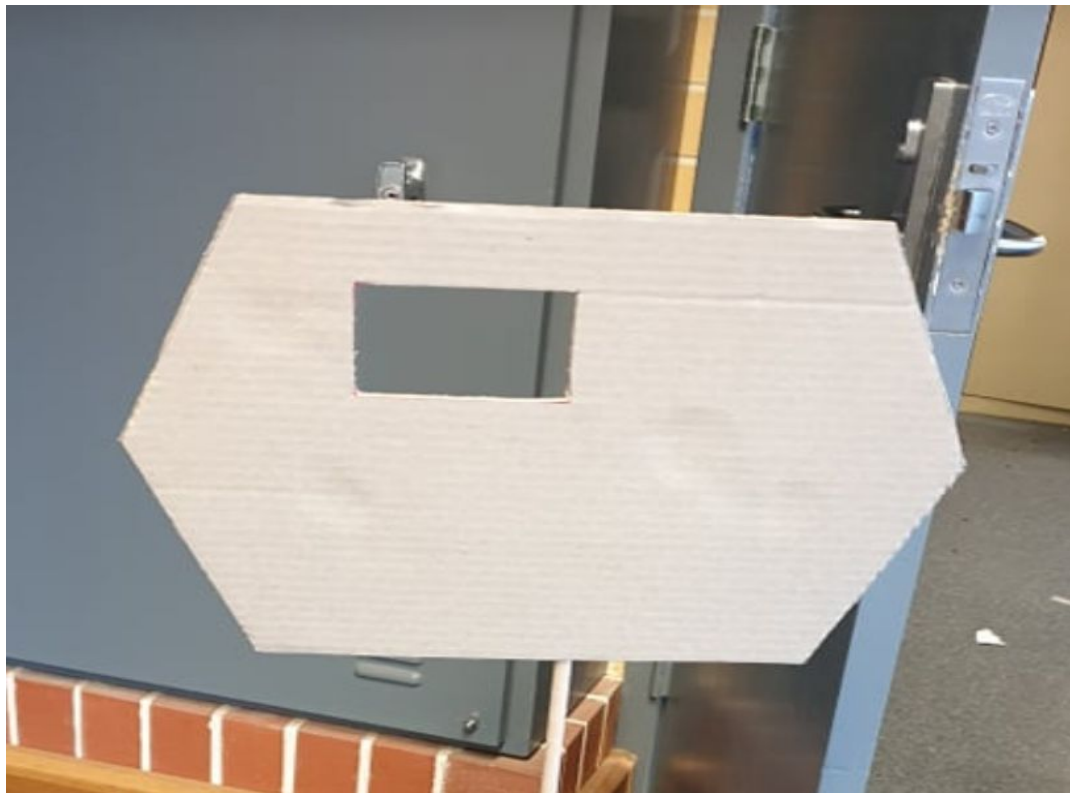


Projection

To calculate surface area, it is optimal to isolate the data's two-dimensional features in the x-y plane, as area is a two dimensional property. To reduce the dimension of the dataset to 2D, a polar projection is performed. This involves finding the median distance (z value), converting the data back to a polar coordinate system, then assigning each pan/tilt value with an r value equivalent to the median z distance in polar form before converting back to Cartesian coordinates. This projects all the data towards the polar origin on to a single plane in the Cartesian system. As the data was originally measured in a polar coordinate system, using a polar projection ensures that each projected point ends up periodically spaced from other points. If a Cartesian projection on to the x-y plane was used instead, many points would be randomly and oddly scattered, reducing the quality of the final plotted shape and making area evaluation more difficult. The result of the projection is shown below.



The actual shape that was scanned is pictured below.



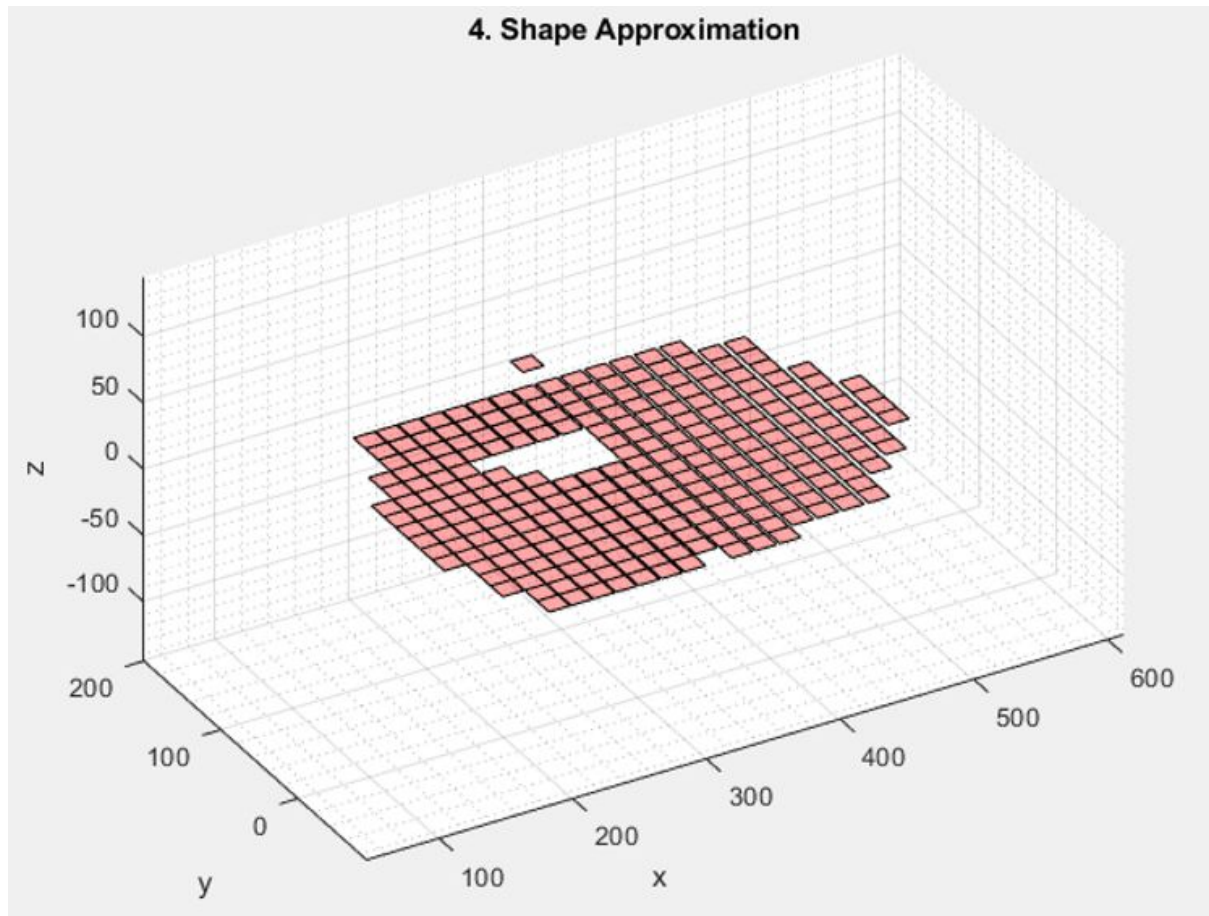
Additional Filtering

Two additional filtering algorithms were developed to be used situationally after the projection step. The first is a filter to remove outliers along the x direction. This was developed as for certain scan settings data points would be detected slightly to the left of the shape at the initial servo position. This was implemented by filtering out all points with an x value outside 1.3 times the interquartile range of the x data. This consistently filtered out the unwanted points while keeping the shape data.

The final filtering algorithm is to remove the stand that the object is fixed to, as this was occasionally present in the scans, depending on the thickness of the scan and the resolution used. This was performed by isolating the y-axis data, then sorting it to iterate through it from the lowest values. The median occurrence of points for each y-value was determined. In the iteration from the lowest values, if the number of points for each value was less than the median occurrence divided by 5, all those values were removed entirely. If a row was found that had more than this threshold, the algorithm would stop and not check the rows above it. For example, if a shape had a median occurrence of 20 points for each row (y-value), the loop will iterate from the lowest y-values of the data. For each of those y-values, if there were less than 4 points with those y-values, they would be removed. As soon as a row is detected with 4 or more points, the algorithm would stop. This effectively removes the stand pole below the object.

Calculating Area and Minimum Distance

To output the minimum distance, the median z value previously calculated is used, as this value corresponds to the perpendicular distance from the plane of the shape face to the LIDAR. To calculate the area, each point is approximated as a rectangle and the sum of the area of all the rectangles is taken. As points further away from the origin are more spaced out as the measurements were taken in polar coordinates with an angular step size, each of the rectangles have slightly different dimensions. To get the dimensions of the rectangles, each point is converted to a polar coordinate, and the rectangle edges are stored as the 4 edges 0.5 of the resolution above, below, to the right and to the left of that point. The angles of the edges are then converted back to Cartesian coordinates and the area is calculated using the difference between the x-axis edges multiplied by the difference between the y-axis edges. A plot visualising the approximation is shown below.



MATLAB Interface

The interface to MATLAB from the board was developed through the MATLAB serial module, which provides many high-level features for communication through serial. After creating a serial port object, a read command is used by the `fscanf()` function and a write command is executed using the `fprintf()` function.

The approach to designing the MATLAB interface was to generalise and simplify the PC side as much as possible, such that it only needed to perform input/output operations. Our original PC interface program was quite long, and mirrored the interface structure in the C code running on the microcontroller. Upon realising it could be simplified to a general input/output interface, this was changed.

Upon initialisation of the program, the message in the MATLAB command window is printed "Press Enter to send PC Mode Request..." Pressing the Enter key will send a request to the board. If the board is still in the beginning menu screen, it will detect the serial interrupt, and enter PC mode. As serial interrupts are disabled if the board has already been set to Board mode, the PC mode request will not work at other points in the program.

The interface functions through the board sending command requests to the PC. Each command request is a character or short string. When the PC receives the character or string that matches a defined command, the PC executes the corresponding operation. The

main program is simply an indefinite while loop polling the serial input and checking for commands.

The list of commands are outlined below.

'a' - This character indicates a data request from PC. When an 'a' is detected, the PC reads in another string. In the C code, a data request is always followed by a message in string format. The PC prints the received message, then prompts the user for input. This input is then sent to the board. If the input is invalid, the board will ignore the value and send another data request.

'b' - This character indicates to the PC that the board has initialised in Board mode, and that the PC functionality is not required. Nothing will happen on the PC side after this.

'p' - This character indicates to the PC that the board has initialised in PC mode. The MATLAB interface will then begin polling serial input for commands from the board.

'd' - This character tells the PC that scan data will be sent. After sending a 'd', the board also sends the resolution of the scan and the scan type ("rectangle" or "full"), both as strings.

'-1' - This string tells the PC that all the scan data has been sent. The PC will then perform the filtering, processing and plotting depending on the type of scan specified earlier.

'l' - This command indicates LIDAR test mode. When detected, the MATLAB program will begin polling serial input, and printing all incoming data as distances.

Preconditions for Software

Preconditions for System Startup:

In order for the assumptions made in the software and algorithms to be met, several preconditions must be satisfied. Firstly, the shape must be perpendicular to the line that would be made in the direction that the lidar points to in the centre of its scanning range.

Additionally, the shape must be within 1.2m and 0.8m from the lidar. This is so that the filtering algorithm will not omit points that it should actually plot.

Furthermore, the shape must be in the same horizontal plane as the LIDAR. This is because the initial edge detection algorithm depends on the lidar finding the right and left edges of the shape on its centered horizontal scan.

Also, the shape should not be exceptionally concave, or have extremely steep slopes, as otherwise, the smaller region for scanning that is defined by the algorithm during the Full scan method will not actually be big enough to scan the whole shape.

Finally, when reading in magnetometer values, it is required that the servo array be parallel to the horizon, such that the sinusoidal function produced by the magnetometer can be interpreted to range between -90 and 90 degrees.

Preconditions for System Shutdown

For a clean system shutdown, the system should be allowed to complete its current scan, otherwise the serial communication file opened in MATLAB will not be closed properly, and as such MATLAB will have to be restarted before another scan can be undertaken.

System Performance

Performance Testing

Servo

The servo motors were calibrated manually through the software to a high degree of accuracy by mapping the generated duty cycle to the resultant angle. As the servos were much more accurate than the IMU sensors, there was no way to test the angle error. Additionally, it was able to increment at an angular step size of 0.1 degrees, which was the highest resolution supported by the software, and much higher than needed for a typical scan.

LIDAR & Scanning

The LIDAR readings contained a considerable amount of noise, where occasionally a reading would be completely off. These outliers were interfering with our measurements, even when taking a higher number of samples per point. Upon switching from taking the average of the samples to taking the median, the number of outliers in the distance readings reduced drastically. When aimed at a stationary object, the LIDAR readings generally deviate less than 20mm from the distance. For certain solid, non-reflective objects, the LIDAR was even more precise, deviating less than 5mm from the distance.

To test the scanning and LIDAR performance simultaneously, a sweep test was developed. This involved measuring the time, mean, standard deviation, median and median absolute deviation of a single LIDAR sweep from 65 degrees to 105 degrees with varying resolutions and sampling rates. The results of the test are displayed in the table below.

Resolution	Samples	Time (s)	Mean	SD	Median	MAD
0.1	1	21.02	968.9265	19.4084	968.8873	13.4316
0.1	3	23.43	966.1455	12.7149	966.29	10.5697
0.1	5	26.13	963.5667	12.4502	963.82	9.799
0.5	1	5.1	963.6848	19.5365	967.08	13.4954
0.5	3	5.59	967.9446	13.1046	965.96	11.3469
0.5	5	6.08	960.0076	13.1938	961.32	10.5193
1	1	3.23	959.4482	27.4731	959.77	15.8333
1	3	3.35	964.3218	14.9084	966.85	13.0058
1	5	3.69	960.5626	13.011	961.33	10.4805

As the samples per point was increased by 2, the time taken for the sweep increased by approximately another 10% from a single sample per point. The standard deviation saw a large decrease from 1 sample to 3 samples, but barely any change from 3 samples to 5 samples. Therefore taking 3 samples is the best compromise between minimising scan time while maximising accuracy. Scanning with a higher resolution increased the time taken for the scan to complete. The step size was inversely proportional to the scan time: doubling the step size approximately halved the scan time.

Scanning Algorithm

The full scan algorithm was quite accurate after the various data filtering and processing techniques were applied. A 1 degree resolution scan at 5 samples per point would take approximately 45 seconds, with a maximum error of 15%, with a typical error between 0-10%. A 0.5 degree resolution scan at 5 samples per point would take approximately 3 minutes, with a maximum error of 10% and a typical error below 5%.

The rectangle scanning algorithm was usually more accurate for rectangles than the full scan algorithm, despite taking a much shorter time to execute. A 1 degree resolution scan at 5 samples per point would finish in approximately 7 seconds, and yield a maximum error of 10%. An 0.5 degree resolution scan at 5 samples per point would finish in approximately 15 seconds, with a maximum error of 5%. Note that the maximum error refers to the maximum error range encompassing over 90% of scans conducted; the typical error value was even less.

IMU

The performance of the IMU sensors were all tested through manual evaluation. All of these were fairly accurate. However, they were nowhere near as accurate as the servo's for rotating, hence they were not incorporated in the scanning algorithm. The gyroscope had issues where if the PTU was rotated suddenly or at too high velocity, it would lose track of its orientation. As the raw data from the magnetometer and accelerometer provided absolute orientation, this problem did not occur for those sensors.

State of the System as Delivered

The algorithms used for scanning performed exceptionally well in terms of speed, accuracy and consistency, as outlined in the performance testing above.

The user interface in both board and PC mode are easy to use, simple and error proof, amounting to a convenient user experience.

Overall, each of the criteria in the specifications has been considered and completed to a sufficient standard, hence conforming to the assignment specifications.

Future Improvements

Alternate Scanning Algorithms

As specified in the software section of the report, the initial algorithm evaluated as the optimal generalised solution to this object area evaluation problem is an edge detection algorithm. However, due to time constraints and the heavy reliance of the algorithm on LIDAR accuracy and the high susceptibility to noise, this method was not implemented. As we drastically reduced the noise of our LIDAR unit for higher samples closer to the project due date, the implementation of the edge detection algorithm could be reconsidered.

As the proposed algorithm functions in terms of distance differences above a certain threshold between each scan step, a single point of noise could throw the algorithm completely off. To further reduce noise, a 'confirmation step' could be added to the program, such that each time a distance difference outside the threshold is detected, extra samples are taken to confirm that the change has actually occurred and that it is not a result of noise.

Advanced User Interface

The user interface via the LCD display on the HCS12 board presents many limitations as the display offers a maximum of two lines of text, each 16 characters in length. This often results in being unable to fit the desired information onto the screen at a single time, and occasionally having to use abbreviations instead of printing out the complete word. This problem is most evident once the user enters the testing mode, where 5 selections are then presented to the user for the separate testing of the servos, LIDAR, gyroscope, magnetometer and accelerometer units. While it would be ideal to have the 5 options on 5 separate lines, this information must instead be condensed to 2.

To address this issue, more advanced features could be introduced to the board user interface, such as scrolling in the vertical and horizontal directions, for both user input and data display. For example, this could involve the user using the A and B keys of the keypad to navigate a scrolling menu, allowing user input selection to span as many lines as necessary.

Safety Implications

The following list identifies and evaluates all possible safety hazards for all common and uncommon uses of the product:

- Laser: This device uses a LIDAR-Lite v2 which contains a class 1 laser. This laser class ensures that the laser is safe in all typical use cases. The definition of safe in terms of lasers is defined (<https://www.safeworkaustralia.gov.au/doc/laser-classification-and-potential-hazards>). But for class 1 lasers they will never exceed the maximum permissible exposure (MPE) under all conditions of exposure.
- Electric shock: This device uses powered devices that could build up a static charge. As the board and related equipment are not completely grounded there is a chance for a build up in static charge which can be discharged when the device is handled. Such as in removing or reconnect cables between the micro controller and the external peripherals.
- Electric shock: It also contains a power supply. As all charge carrying cables are fully insulated it is very unlikely to provide substantial shock to any user. However, after the wear and tear of common use the cable insulation might thin and all for current to flow. This is limited by the power supply that limits the output to 5V 3A and so any shock induced would be limited in effect.
- Servo Jamming: As this device contains a servo () and the related uncovered moving parts, there is a possibility of fingers or similarly small objects to get stuck in the path of the servo. As the moving parts are metal then this could provide damage. However, this is limited by the limited range that the servos move, as they do not have a large catchment area where things could get stuck. This is partly due to the servos moving in rotational changes rather than linear movement. And this is also mitigated by the lack of open internal spaces where fingers would fit and that would also lock them in position into the device that would cause actual damage.

Overall product has minimal safety hazards as it would require significant negligence or severe inappropriate use of the product to encounter any possible hazards. Furthermore the hazards outlined above all have minimal damage, which limits the effect of any hazards to limited short term effects.

Conclusion

Overall, the system was built to a high standard, satisfying all the requirements in the specification. While the finished product satisfies the specifications, it is not perfect and there is room for further improvement, especially if the product is to be distributed for professional usage.

References

- [1] Adafruit Industries, "ADXL Digital Accelerometer," ADXL345 datasheet, <https://learn.adafruit.com/adxl345-digital-accelerometer>, May 2014.
- [2] STMicroelectronics, "L3G4200D MEMS motion sensor: ultra-stable three-axis digital output gyroscope," L3G2300D datasheet, Apr. 2010 [Revised Dec. 2010].
- [3] Honeywell, "3-Axis Digital Compass IC HMC5883L," HMC5883L datasheet, www.magneticsensors.com, Feb. 2013.
- [4] HiTEC, "HiTEC General Servo Information," Servo Datasheet, www.hitecrcd.com.
- [5] PulsedLight LLC, "LIDAR-Lite v2 "blue Label", " LIDAR datasheet, Jul. 2016.
- [6] Safe Work Australia, "Laser Classification and Potential Hazards", Website, <https://www.safeworkaustralia.gov.au/doc/laser-classification-and-potential-hazards>, Apr. 2017.