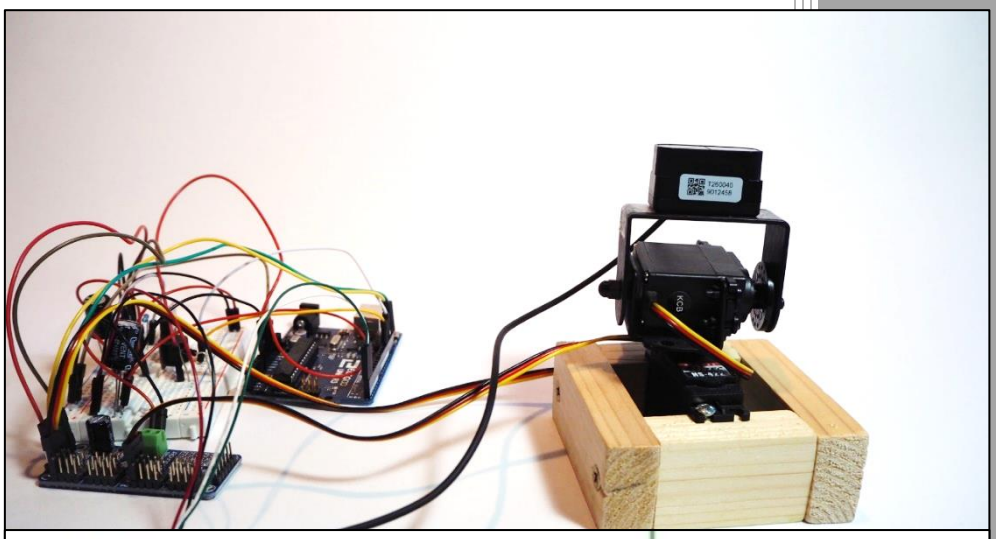
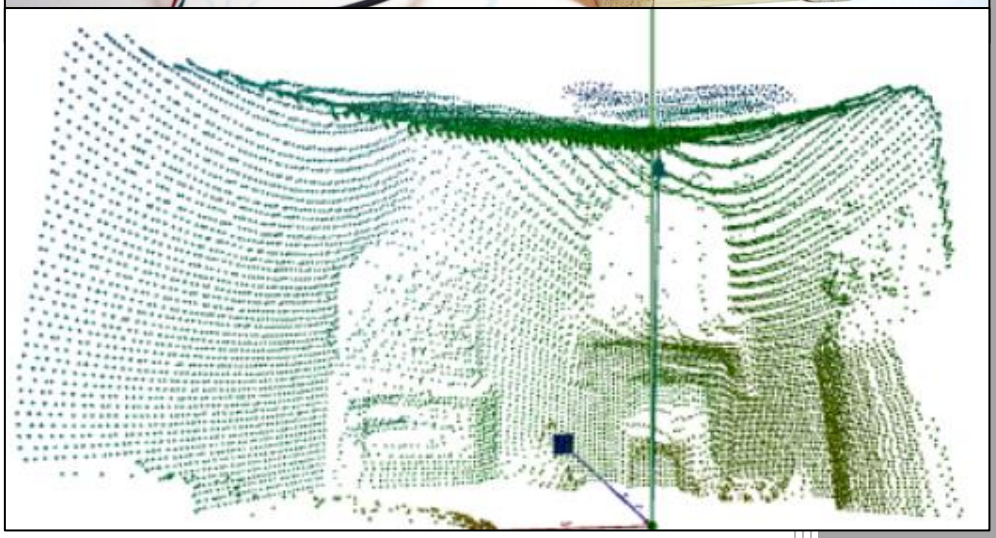
2020

3D Room Scanner





Jamie Holroyde

1 October 2020

1 Project Plan

1.1 Aim

The Topcon GLS2000 (Figure 1 – left) is a prime example of a commercial 3D LiDAR scanner; being a versatile tool it is capable of scanning almost all environments. With a maximum scan range of 500m and a minimum resolution of 6.3mm, it is able to scan as far, and as detailed as is realistically required, for example, a bridge (as shown in Figure 1 – right). Its sturdy, robust frame allows for work in even the most extreme conditions. One of the downsides to the scanner is that it costs between £20000 to £30000, far from affordable for the average hobbyist. Due to this great expense, I am going to attempt to design and build my own 3D scanner, but at a much more affordable cost.

Figure 1 - Example of commercial LiDAR Scanner (left) and Topcon GLS2000 scan (right)

The aim of this project, therefore, is to design and build an affordable 3D room scanner capable of scanning spaces within a 180-degree field of view.

The project aim will be achieved if the scanned dataset is able to provide the user with an accurate representation of the scanned space.

1.2 Objectives

The objectives for this project are to:

a. achieve required range of movement with pan and tilt servos
b. obtain raw data from a distance sensor (such as an infrared sensor) and have it display the data
c. combine the pan and tilt motion with the distance sensor
d. record the raw data in a text file
e. process and display the scanned data.

I have identified the following research topics to help achieve these objectives:

- The effects of reflective surfaces on the reliability of the scan

- Impacts of scan speed and distance to the material surface on data
- Accuracy of readings by the distance sensor and pan and tilt unit

1.3 Wider purpose

The findings of this project could be applied to security (the ability to calculate the difference between two scans shows if anything has moved), spatial mapping (mapping the terrain of an area of land), and mobile positioning (perhaps the most common use: to steer vehicles and drones around obstacles). A more common application is that it could also be used by estate agents to show their potential customers the internal layout of properties.

1.4 Budget

Considering the cost of components and that key part of the aim was creating an **affordable** room scanner, I decided that a reasonable budget was £100. By the end of this project I had spent £96.35 (which is within the budget). This comprised:

- 1 x TFmini Plus – £39.24
- 1 x Lynxmotion Pan and Tilt Kit - £30.65
- 1 x Arduino Uno - £19.14
- 1 x PCA9865 Servo Driver – £2.36
- 1 x selection of basic electronic devices – approx. £5

1.5 Possible approaches

Through research, I have identified a range of approaches to construct the 3D Room Scanner that I can implement including:

- different ways of measuring the distance such as infrared, ultrasonic and LiDAR sensors
- 2D or 3D scans
- connecting the pan and tilt unit directly to the board or via a servo driver, like the PCA9685 Servo Driver
- using different types of micro-controller/micro-computer, for example Arduino or Raspberry Pi
- different display methods including Depth Map and Point Cloud.

1.6 Chosen approach and reasoning

Out of the above approaches, I decided to:

- use an infrared sensor – this was chosen over an ultrasonic sensor because light travels faster than sound; so, an infrared sensor can measure the distance faster than an ultrasonic sensor
- create a 3D model of the mapped environment – selected so the user can more accurately judge distances
- connect the pan/tilt unit via a servo driver – this was done to allow a separate power supply to power the motors as opposed to using the micro-controller's power supply
- use the micro-controller "Arduino Uno" – this was chosen due to my existing familiarity with the micro-controller
- display the data using a Point Cloud – this option was selected as then the user could have the option of colouring the points either by distance or by strength.

1.7 Planning

To help complete the main mission, it was broken down into smaller, more achievable concepts, which were then combined nearer completion. These identified tasks comprised:

1. Servo Movement
    a. to achieve a regulated power supply
    b. to control motor movement using PWM or PPM
2. Pan/Tilt Unit
    a. to assemble servos into pan/tilt unit
3. Distance Sensor
    a. to set up a distance sensor
    b. to obtain reliable results from the distance sensor
4. Distance Sensor and Pan/Tilt Unit
    a. to operate a distance sensor and pan/tilt unit concurrently
5. Serial Port
    a. to enable communication between the computer and the micro-controller via the serial port
    b. to transfer results from the micro-controller to the computer via the Serial Port
6. Data Logging
    a. to record the data in Python
7. Data Display
    a. to display a simulation of the received data
8. GUI
    a. to construct a GUI to connect all the programs.

1.8 Scheduling

A Gantt chart was constructed to show the estimated time each development phase (see Section 2.5) would take.

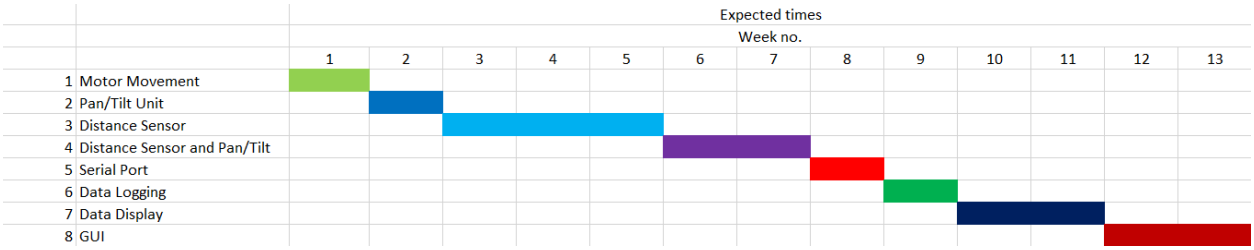


Figure 2 - Gantt chart showing the estimated time for completion of development phases

Once the project was completed, another Gantt chart was constructed to reflect on the length of time the stages took.

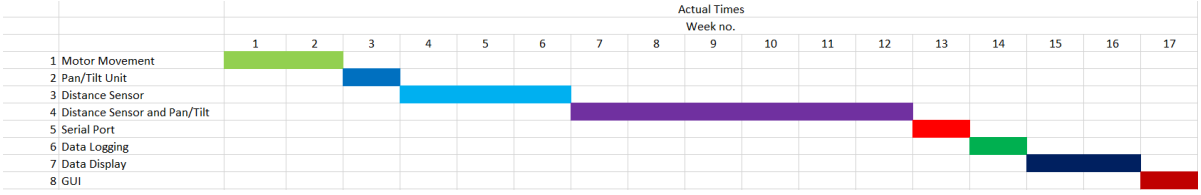


Figure 3 - Gantt chart showing the actual time for completion of development phases

The above Gantt charts only show the development phases and do not account for the planning and write up phases. These were considered relatively small in comparison to the main project.

In Figure 2 and Figure 3 there is a difference in time for the "Distance Sensor and Pan/Tilt" section. The actual time took three weeks longer than expected because there was an issue with obtaining the information from the distance sensor (see section 2.8.6 for more details).

1.9 Resources

There are multiple resources I had available to me, the best of which include: my tutor, who helped me with the logic as well as some of the more complicated C programming for the Arduino; websites such as GitHub and stackoverflow; and finally, I have the Python/C documentation and the documentation for the infrared unit itself.

1.10 Safety

In the interest of safety, a risk assessment was carried out for this project as shown in Table 1.

What are the hazards?	Who might be harmed and how?	What are you already doing?	Do you need to do anything else to control this risk?
Laser	Anyone in the room could be blinded	Telling people not to go in the room and keeping the power equal to or less than 5mW (as per the legal guidelines)	Create a warning sign
Short Circuit	Anyone in the building if a fire is started	Keeping the power off until everything is set up	No, as the risk is so minuscule
High Power	Anyone in the room could be electrocuted	Warning people to stay away from the plug	No, as the risk is so minuscule

Table 1 - Risk assessment

2 The Project

2.1 Units and abbreviations

Unit	Definition	Unit	Definition
g	Grams	**cm**	Centimetres
m	Metres	**ms^{-1}**	Metres per second

Table 2 - Units

Abbreviation	Definition
2D	Two Dimensions
3D	Three Dimensions
FOV	Field of View
GUI	Graphical User Interface
HUD	Heads Up Display
LED	Light Emitting Diode
LiDAR	Light Detection And Ranging
PanValue	The position the pan servo is currently at (in degrees)
PPM	Pulse Positioning Modulation
PW	Pulse Width
PWM	Pulse Width Modulation
TiltValue	The position the tilt servo is currently at (in degrees)

Table 3 - Abbreviations

2.2 Background

I was looking for a project which combined my interests in the fields of physics and computing. This project was chosen because it integrates the physics behind remote sensing with physical computing. I have already done initial work through computer simulations of planetary orbits using Python, some basic work with micro-controllers such as the Arduino, and servo control by using the EV3 and the Marty robot. At the time of selecting my project, I was focusing on wave properties in my physics class and chose a project which reflected this. I then decided to take what I had learned from class and apply it with my other interests: maths and computing. Another reason I took on this challenge was because I could work on it by myself and proceed at a speed I was happy with, yet I still knew people who would be willing and able to help me if required.

2.3 Theory

2.3.1 Infrared sensors

Infrared sensors detect infrared light (which has a frequency between 430 THz and 300 GHz, as seen in Figure 4). One of the properties of the infrared LEDs is that not only do they emit light when they receive

an electric current, they also produce an electric current when they receive light of a specific wavelength.

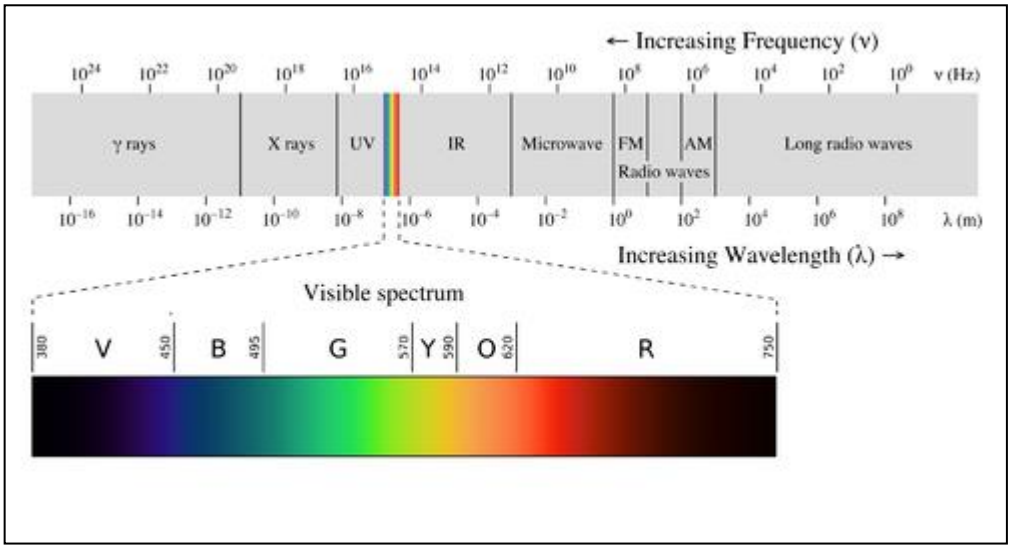


Figure 4 - The frequency of infrared light

A pair of infrared LEDs can be used to measure distance. One of the LEDs is set to output light and the other is set to transmit a signal when it receives light. When there is an object in front of the infrared sensor, the light from the light-emitting diode is reflected off the object and bounced back to the light-receiving diode. In other words, when the light-receiving diode receives light, there is an object in front of the infrared Sensor.

Using the formula:

$$\text{Distance} = \text{Speed} \times \text{Time}$$

the distance can be calculated. In the above formula, **Distance** is the distance the infrared beam travels, **Speed** is the speed of the beam and **Time** is the time it takes for the beam to return to the sensor. All that is needed to use the formula is the speed of light ($3x10^8 ms^{-1}$) and the time it takes for the infrared light to hit the light-receiving diode (this must be timed). The distance must then be divided by two as the calculated distance is the distance to and from the object, and it is only the distance to the object that is required. This is called the Time of Flight principle. In Figure 5 φ1 is the LED which outputs infrared light and φ2 is the LED which transmits an electrical signal when it receives infrared light.

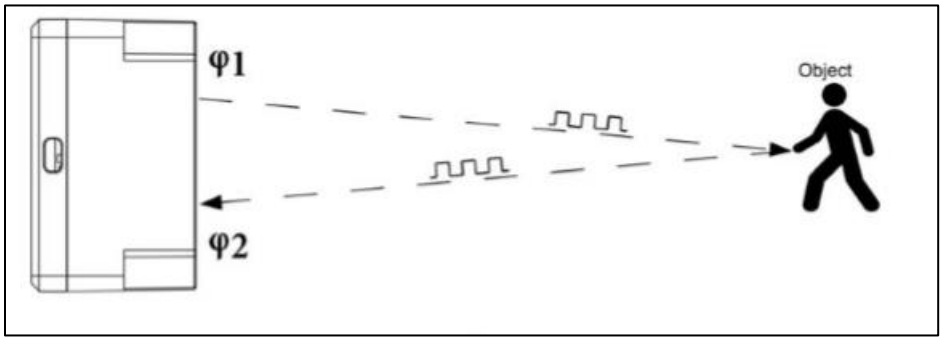


Figure 5 - Illustration of the Time of Flight principle

The accuracy of the sensor depends on the strength of the returning beam. When the beam hits an object, some of the beam will diverge and become background infrared; the rest will return to the sensor. The amount of the beam which will become background infrared depends on the distance to the object, the object's size, and the object's reflectivity (AudioVOX, 2019). For example, if the object has a diffuse (rough) surface, it is very likely that at least part of the beam will be reflected back at the sensor, thus providing more accurate results (as seen in Figure 6 - left). If the object has a specular (smooth) surface there is a chance that none of the beam will reflect back at the sensor, thus providing less accurate, or even no, results (as seen in Figure 6 - right).

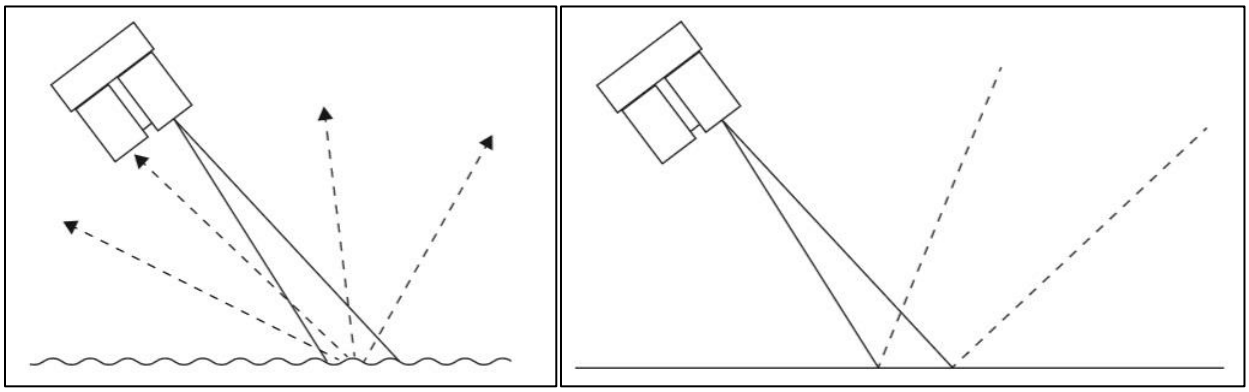


Figure 6 - Diagram of infrared light hitting a diffuse surface (left) and specular surface (right)

Examples of diffuse surfaces include paper, matte walls, and granite. Specular surfaces include mirrors and glass viewed off-axis.

All surfaces emit infrared light, and this can result in interference with the scan. Usually this is so marginal it can be ignored, however, if the scanned area is in direct sunlight or contains a heat source, the results are less reliable. This is because all heat sources emit greater amounts of infrared light than other objects.

2.3.2 PWM (Pulse Width Modulation)

PWM is an energy efficient way of controlling the amount of electrical current supplied to a component. It works, not by restricting some of the current getting through with resistors, transistors etc, but by stopping the supply of current to the component. For example, if the duty power of a PWM power supply is set to 60%, then the pulse is only on for approximately 60% of the time. The ratio between duty cycle and brightness is not exactly 1:1, as it depends on the efficiency of the LED.

Figure 7 shows pulses which represent the time it is "on" and the depressed areas show the time it is "off". As the length of the "on" and "off" lines are the same in the above figure, the component is being supplied power approximately 50% of the time (Brown, 2017).

Figure 7 - PWM illustration

2.3.3 Servo

Many modern servos operate using PWM; certainly, the ones that are being used for the Room Scanner. Usually when servos receive a 1.5ms pulse, this will reset the motor to its neutral (middle) position. The longer the pulse is, the further clockwise the motor attempts to turn (it cannot exceed its limit). Similarly, the shorter the pulse is, the further anti-clockwise the motor attempts to turn. Normally servos accept a range of pulses between 1-2ms.

Continuous rotation servos (which are not being used in this project) operate slightly differently. Instead of the pulse width instructing the servo which position to turn to, it simply instructs the servo how fast to move either clockwise or anti-clockwise. For example a pulse width of approximately 1.5ms will instruct the motor to stop moving, a pulse width of greater than 1.5ms instructs the motor to turn clockwise, likewise a pulse width of less than 1.5 instructs the motor to spin anti-clockwise. The further the value is from the motor's rest position, the faster the motor turns.

2.4 Concept

The concept behind the scanner is relatively simple: the servos move to the required position and the distance sensor "pings". The computer then sends this information, as well as the current servo positions to Python. With this information, Python then converts the information from spherical coordinates (distance, pan, tilt) to cartesian coordinates (x, y, z) before plotting it live. Next, the moving, "pinging", converting, and plotting process is then repeated until the full scan is complete.

2.5 Development

2.5.1 Motor movement

My first objective was controlling the servos. Originally connecting the servos directly to the micro-controller (Section 2.6.1.5), I created a simple script which moved the servo to the position I had predefined in the script. This phase was completed without any unforeseen obstacles.

2.5.2 Pan/tilt unit

My next aim was to assemble the pan/tilt unit. Once this was achieved, I realised the unit was a bit unstable, as it had a narrow base with no supports. To solve this problem, I constructed a small frame for the servos using wood as shown in Figure 8.

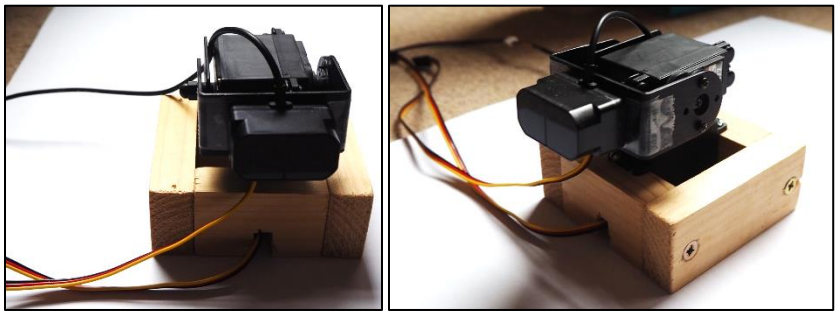


Figure 8 - Pan/tilt unit's frame

Once the frame was finished, I progressed to having the pan servo go back and forth on its own, and then the tilt servo. Finally, I had the pan servo do one sweep, move the tilt servo up one degree and repeat. This created the basic infrastructure for the project.

Consequently, I decided to control the servos using the Servo Controller (Section 2.6.1.4). This resulted in some minor changes in the program, but the basic logic remained the same.

2.5.3 Distance sensor initialisation

Commencing my next phase, I connected the infrared sensor (section 2.6.1.3) to the micro-controller. Using the "TFMiniPlus.h" library provided, I set the distance printed on the screen. Once I had added a continuous loop, the sensor continuously printed the current distance it was measuring; however an error was identified with this method, as further discussed in section 2.8.6.

2.5.4 Distance sensor and pan/tilt unit

The next stage was programming the distance sensor and the pan and tilt unit to work simultaneously. I added the "get distance" function from the distance sensor script into servo control script, which merged the two scripts together.

Physically attaching the sensor to the pan/tilt unit was a small problem as there was no easy way to connect the two. A solution to this was gluing them together; however instead I chose to use double-sided sticky tape as it was easily removable and later made the sensor interchangeable with the laser pointer.

2.5.5 Serial port

To enable processing and plotting of the data, I would first need to gather the information I had produced in Python. To do this, I needed to communicate with the computer from the micro-controller via the serial port. Initially, I attempted to send random words as a test, to verify the connection between the devices. Next, I started sending numbers before finally trying the distance sensor and the pan and tilt unit separately, and then together.

2.5.6 Data logging

Processing the data was the next target. Initially, I had to convert the data from spherical coordinates to cartesian coordinates. This involved using mathematics (see Section 2.8.4 for more information). In addition to this I had to apply a mathematical correction of 5cm to the distance as the distance sensor was 5cm offset from the centre of rotation. This helped produce more accurate results.

2.5.7 Data display

Next, I had to display the data. This was accomplished using the Python module "Visual Python". I decided to plot all the points as small spheres with a variable radius. Using a simple class, I had functions to calculate the colour and the X, Y and Z position of the point. There are also arrows pointing in the X, Y and Z directions to help the user orient themselves.

2.5.8 GUI and HUD

Then, to tie what I had done together, I constructed a GUI using the Python "TKinter" module. The GUI was constructed with the functionality to create a new scan, upload a scan from RAW data, and upload a scan from processed data. This functionality allows users to scan rooms and export the data or import data from somewhere else and have it displayed on the screen. The GUI also makes the scanner more user friendly and makes it easier to adjust the scanning options. The RAW data is recorded in two separate lines, one which contains the distance sensor readings and one which contains the information about the pan and tilt unit. Once the scans have finished, or when importing data from text files, the user can choose to display the point's colour as distance or strength.

Finally, I added some functionality to the program to allow the user to control whether the point's colour was according to distance or strength, and to show the user what position and reading the servos and distance sensor were giving. During the scan, the "Show Distance", "Show Strength" and "Show Filtered" options are disabled as the points are only coloured after the scan is complete.

In Figure 9 you can see the HUD the user is presented with during the scan.

Figure 9 - The HUD the user is presented with

2.6 Design detail

2.6.1 Hardware

Using a combination of two circuit diagrams, a circuit diagram was constructed (see Figure 10) which connected all of the electrical equipment.

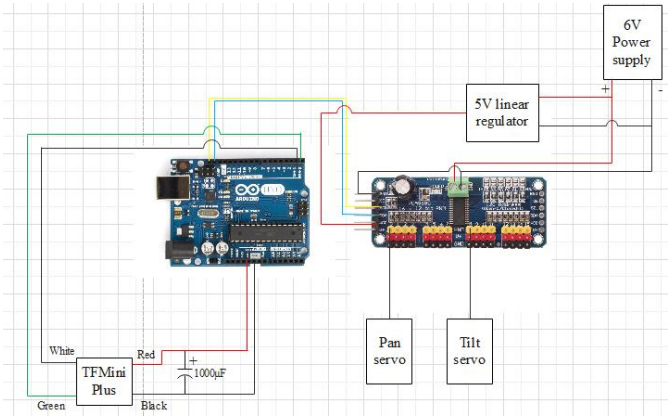


Figure 10 - Circuit diagram for the project

2.6.1.1 Pan/tilt unit

The Lynx B-Pan/Tilt Kit (BPT-KT) is an affordable, robust pan and tilt unit which is easy to build. Sold by Lynxmotion, the kit contains:

- 2 x HS-422 Standard Size Servo (see Section 2.6.1.2 for more information)
- 1 x Aluminium Short "C" Servo Bracket
- 1 x Aluminium Multi-Purpose Servo Bracket
- 1 x 4-40 x .250" Nylon Washer
- 1 x 4-40 x .250 Nylon Acorn Locking Nut
- 5 x #2 x .250" Steel Phillips Head Tapping Screw
- 1 x 4-40 x .375" Steel Hex Button Head Screw
- 9 x .172" x (.187"-.250") Nylon Snap Rivet Fasteners.

Able to be easily de-constructed, the BPT-KT is an ideal choice for hobbyists.

Figure 11 - The BPT-KT pan/tilt unit

2.6.1.2 Servos

The HS-422, which are the servos I used throughout this project, have the following attributes (Hitec, n.d.):

Description	Parameter value	
Control system	+pulse width control 1500usec neutral	
Operating voltage range	4.8v to 6.0v	
Operating temperature range	-20 t0 +60°c	
Test voltage	At 4.8v	At 6.0v
Operating speed	0.21sec/60 at no load	0.16sec/60 at no load
Stall torque	3.3kg.cm(45.82oz.in)	4.1kg.cm(56.93oz.in)
Operating angle	45 /one side pulse traveling 400usec	
Direction	Clockwise/pulse traveling 1500 to 1900usec	
Current drain	8ma/idle and 150ma/no load running	
Dead band width	8usec	
Connector wire length	300mm(11.81in)	
Dimensions	40.6x19.8x36.6mm(1.59x0.77x1.44in)	
Weight	45.5g(1.6oz)	

Table 4 - Specifications for the HS-422 Servo

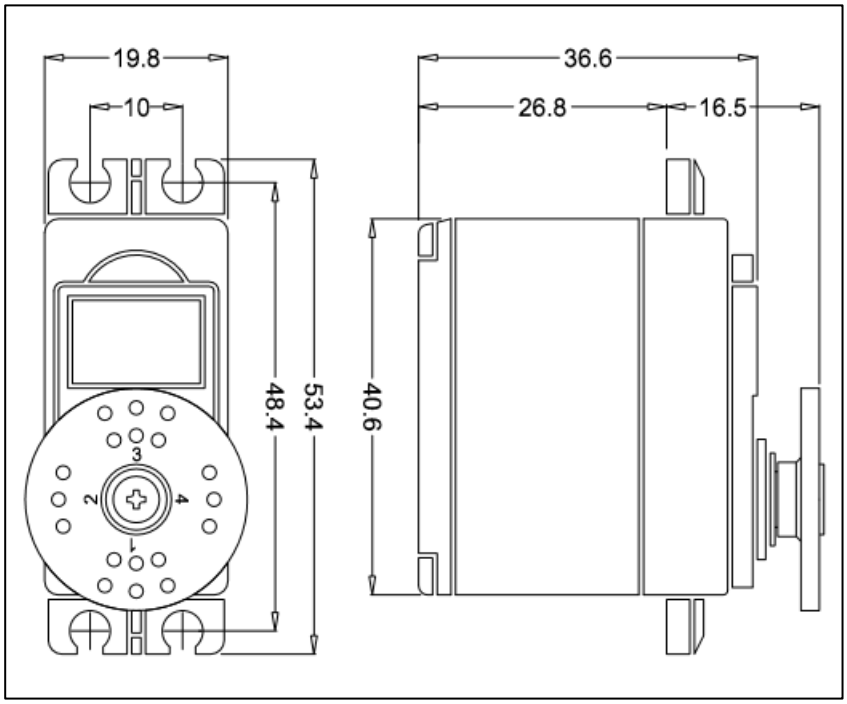


Figure 12 - Dimensions of the HS-422 Servo (cm)

Figure 13 - The HS-422 Servo

2.6.1.3 Infrared sensor

The TFMini Plus is a small, compact infrared sensor which was used for this project. It has the following characteristics:

Description	Parameter value
Operating range	0.1m~12m
Accuracy	±5cm@（0.1-6m）
	±1%@（6m-12m）
Measurement unit	cm
Range resolution	5mm
FOV	3.6°
Frame rate	1~1000Hz（adjustable）

Table 5 - Characteristics for the TFMini Plus

Figure 14 - The TFMini Plus

2.6.1.4 Servo driver

The PCA9685 is an I²C-bus controlled 16-channel LED and Servo controller. With the ability to have 62 chained together, it is possible to connect 992 servos operating off two of the Arduino Pins. It also allows for a separate power supply to power the servos, rather than the servo using power from the micro-controller (Adafruit, 2020).

Description	Parameter Value
Dimensions (no headers or terminal block)	62.5mm x 25.4mm x 3mm
Weight (no headers or terminal block)	5.5g
Weight (with 3x4 headers & terminal block)	9g
Number of address pins	6
Resolution	12-bit
Maximum number of servos/LEDs	16

Table 6 – PCA9685 servo controller specifications

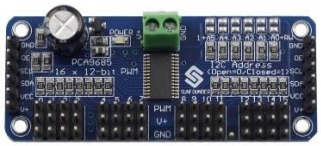


Figure 15 - The PCA9865

2.6.1.5 Micro-controller

The Arduino Uno is a small, compact micro-controller with one serial port, fourteen digital pins and six analogue pins (Components101, 2018).

More about the micro-controller:

Description	Parameter value
Micro-controller	ATmega328P – 8-bit AVR family micro-controller
Operating voltage	5V
Recommended input voltage	7-12V
Input voltage limits	6-20V
Analog input pins	6 (A0 – A5)
Digital I/O pins	14 (Out of which 6 provide PWM output)
DC on I/O pins	40 mA
DC on 3.3V pin	50 mA
Flash memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (clock speed)	16 MHz

Table 7 - Arduino Uno Attributes

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power micro-controller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: Ground pins.
Reset	Reset	Resets the micro-controller.
Analog pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/output pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Table 8 - Pin layout and description for the Arduino Uno

Figure 16 - The Arduino Uno

2.6.1.6 Ancillary items

The other, smaller components include:

- 1 x 5V linear regulator
- 2 x 1000µF capacitors
- 1 x diode
- 1 x 220Ω resistor
- 1 x LED (white)

2.6.2 Software

There are two programs which run simultaneously, that are used to produce the scan. One program is run on the micro-controller, and the other is run on the computer.

2.6.2.1 The micro-controller

Written in the Arduino IDE (which uses a language based on C++) the micro-controller executes the following processes:

1. it moves the motors to the next position – at the very beginning this step sets the motors to their starting position instead
2. it measures the current distance
3. it writes the distance, strength and the pan and tilt values to the serial port.

The micro-controller acquires all the raw data.

2.6.2.2 The computer

Written in Python the computer carries out the following steps:

1. it reads from the serial port
2. it formats the data
3. it converts the position from spherical data to cartesian coordinates
4. it creates a point with the colour corresponding to the distance or strength value
5. it moves the point to the correct position.

The computer processes the raw data and plots it on the computer screen.

2.7 Initial testing

Having completed the development phases, I decided to run some tests. These included:

1. simply keeping the pan and tilt motors still and varying the distance the sensor was sensing (as shown in Figure 17).
2. ensuring that the closer the object was to the distance sensor; the closer the plotted dot was to the origin, and vice versa.
3. having the pan and tilt unit panning.

It was during the third test that I noticed something was incorrect. While the unit was panning clockwise, the dots were how they were expected to be. However, on the return sweep, the dots were not in the same places as shown in Figure 18 as all the points were on the same plane, whereas they should have all been in the same place.

I soon realised this was due to a time lag as mentioned in further detail in Section 2.8.5.

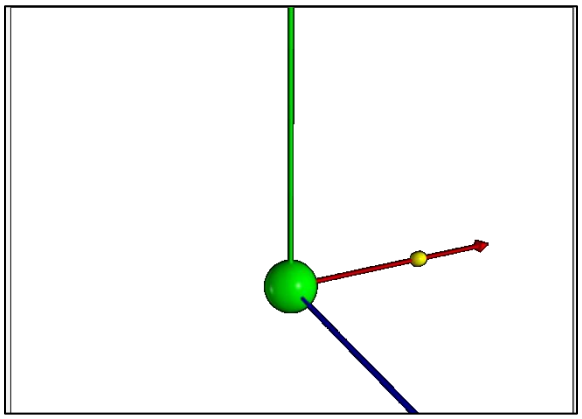


Figure 17 - Checking distances match up with a pan and tilt value of 0

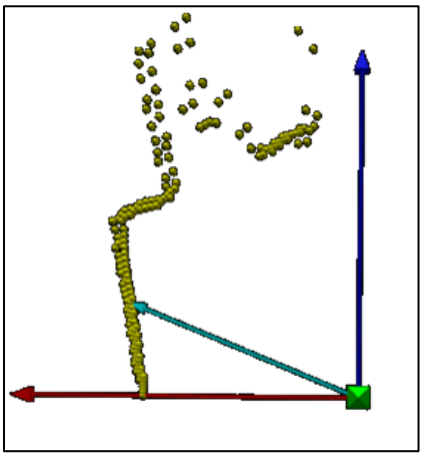


Figure 18 - Time lag error

2.8 Challenges

2.8.1 Version control

Near the beginning of the project, I had written the code to transfer the data from the micro-controller to the computer. During this time, I was overwriting my previous copies with more up-to-date ones. This system worked well until a mistake was made which I could not identify, as I did not have the previous copy to compare. This resulted in a short interruption which was resolved reasonably quickly but resulted in a loss of progress.

This issue was solved by starting a process of version control by simply renaming the files with the day's date and time, and by making a new copy at least once every three hours of work and at the end of the working session. Eventually this method was replaced by using GitHub repositories which allow online source control.

2.8.2 Regulated power supply

The power supply was coming straight from the mains before I had regulated it, this meant it was not an exact voltage. This resulted in the servos being less reliable and on occasion moving "jerkily". To remedy this, I used a 5V linear regulator and added it to the circuit which solved the problem as the servos were now receiving a steady voltage.

2.8.3 Pulse width limits

I realised quickly that the motors were not turning the 180° that they were designed to. I believe this was because the minimum and maximum pulse widths were incorrect for the specific servos which were being used, as I had selected the default minimum and maximum. To rectify this, I added a low-voltage laser pointer (5mW), as shown in Figure 19, so I could more easily see where the distance sensor was pointing. This helped because I could make sure they were both turning a full 180°. I adjusted the minimum and maximum values accordingly.

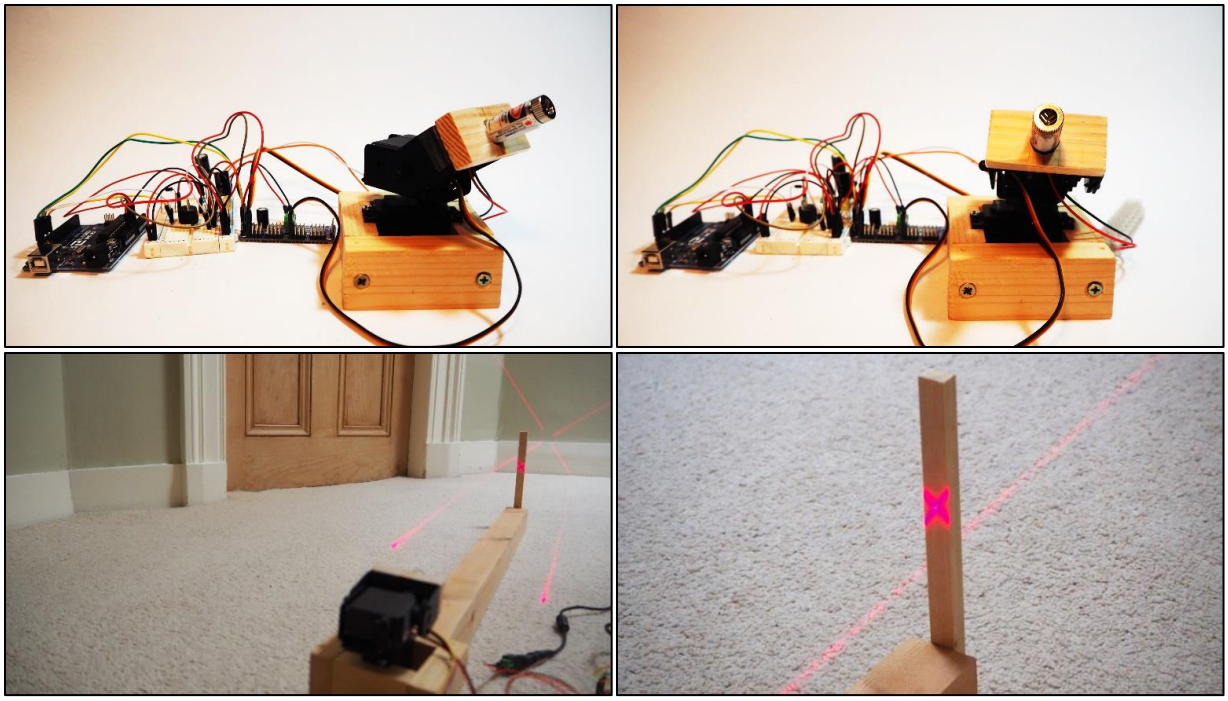


Figure 19 - Using the laser pointer to calibrate the servos

Once I had introduced the lasers, the safety was reviewed as discussed in Section 1.10.

2.8.4 Positioning

Originally, I used trigonometry to convert from spherical to cartesian coordinates. This method produced some abnormal, and varied results. Following some online research, I quickly realised my mistake: this method worked only in two dimensions. Doing further research, I identified a formula for a 3D version of this. The formula for converting spherical to cartesian coordinates is:

$$X = \text{Distance} \times \sin(\text{TiltValue}) \times \sin(\text{PanValue})$$

$$Y = \text{Distance} \times \cos(\text{TiltValue})$$

$$Z = \text{Distance} \times \sin(\text{TiltValue}) \times \cos(\text{PanValue})$$

2.8.5 Time lag

Perhaps the biggest problem I encountered was with the time lag. The distance sensor was scanning at a frequency of 155200Hz; this resulted in the readings it was printing to the Serial Port being queued up and read at a much slower rate. This had a negative impact on the results as the computer would be reading old results and printing the current pan and tilt value. In other words, the results the Python script was reading had accurate pan and tilt values, but the distance would have been the accurate distance approximately 7 iterations ago.

The lag was solved by switching the mode of the distance sensor from continuous mode to single “ping” mode. The command had to be sent in hexadecimal form as discussed in Section 2.8.6. This meant that the sensor would only “trigger” if the computer instructed it to do so. So, to get any readings from the sensor, I had to send it a command to measure, it would measure once, and then the computer would

read that measurement. By doing this, the lag problem was eradicated as the distance sensor sensed at a much more controlled rate.

2.8.6 Incorrectly acquiring information from the distance sensor

Arguably the most complicated part or the project, both mathematically and logically, was reading the information from the infrared sensor. The string the computer received, from the distance sensor via the Arduino, consisted of nine different bytes (see Figure 20), all of which were in hexadecimal form. First, I had to obtain the distance; this comprised a low and high byte which I utilised to calculate the distance. In addition, I had a "checksum" byte which was the total of all the other bytes. This was an issue, as it had to be converted from hexadecimal to decimal values. The distance had to then be calculated before using the "Checksum" value to check the values of the other bytes.

Byte1-2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
0x59 59	Dist_L	Dist_H	Strength_ L	Strength_ H	Reserved	Raw.Qual	CheckSum_ L

Data encoding interpretation	
Byte1	0x59, frame header, all frames are the same
Byte2	0x59, frame header, all frames are the same
Byte3	Dist_L distance value is a low 8-bit. Note: The distance value is a hexadecimal value, for example, Distance 1,000cm = 03 E8 (HEX)
Byte4	Dist_H distance value is a high 8-bit.
Byte5	Strength_L is a low 8-bit.
Byte6	Strength_H is a high 8-bit.
Byte7	Reserved bytes.
Byte8	Original signal quality degree.
Byte9	Checksum parity bit is a low 8-bit, Checksum = Byte1 + Byte2 + ... + Byte8, Checksum is the sum of the first 8 bytes of actual data; here is only a low 8-bit.

Figure 20 - Explanation of received data

2.8.7 Baud rate

One of the last problems I encountered was the motor "juddering" every so often. This was happening because the servos in the pan and tilt unit had a baud rate of 9600, whilst the distance sensor had a baud of 155200. Throughout the project, I had used the Serial Port on a baud rate of 155200 because it had a higher sampling frequency. To solve this problem, I simply changed the distance sensor's default baud rate to 9600 and used the Serial Port on a baud rate of 9600; this lower baud rate did not affect the performance of the sensor because it had been changed to single "ping" mode as discussed in Section 2.8.5. After this alteration, the juddering stopped because the baud rates of the infrared sensor and servos matched.

2.8.8 Reading from the Serial Port

As I now had a GUI, I had to find a way to send information from the computer to the micro-controller. I knew it had to be via the Serial Port, but I was not sure how to send it. Using the PySerial module, I attempted to send the parameters to the micro-controller. I had managed to send the information to the micro-controller successfully, but the difficulty was recognising the information as a list of integers and setting them as variables. After doing some research, I discovered the "atof" and the "atoi" functions in C++. Using these functions, I successfully converted the information to floats and integers

from a string. The last step was simply to use logic to instruct the micro-controller not to scan the room until it had received all the parameters it was expecting.

2.9 Final testing

To make sure everything was working properly, I repeated the steps outlined in Section 2.7, this time without the time lag error (Section 2.8.5) and the testing was considered to have been successfully completed. Subsequently to testing, I scanned a room with the scanner. As the scan performed as expected, I progressed on to scanning multiple rooms one after another.

2.10 Results

With the point's colour corresponding to the strength of the returning beam for that point, final scans were made and simulations plotted. Note – The following images are in 2D, the Python program displays the scan in a 3D virtual environment so the results may seem slightly unclear.

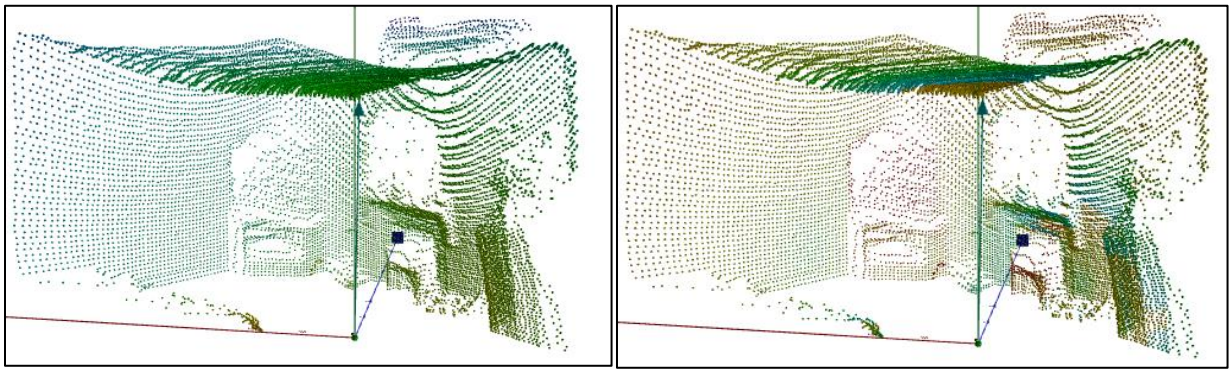


Figure 21 - Distance (left) and strength (right) scan of a living room

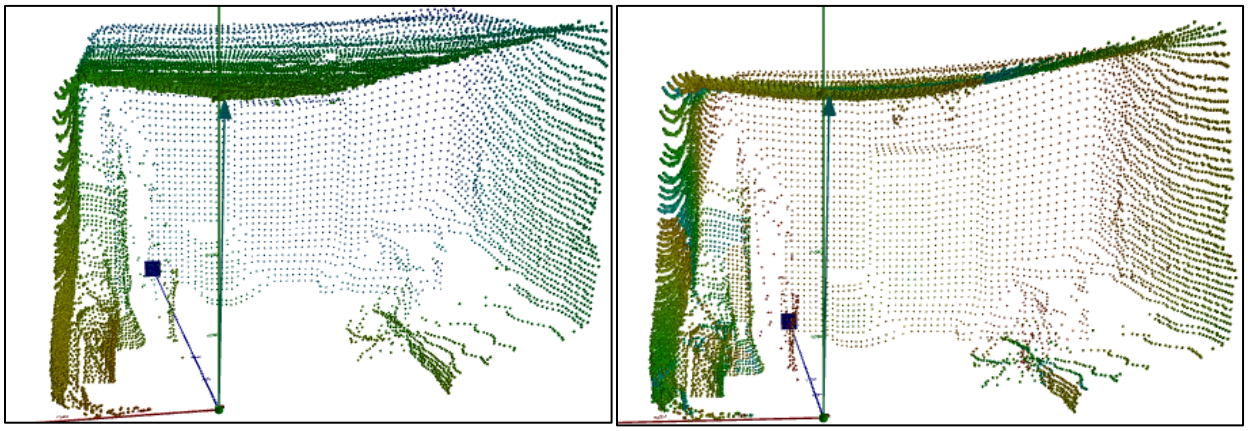


Figure 22 - Distance (left) and strength (right) scan of a study

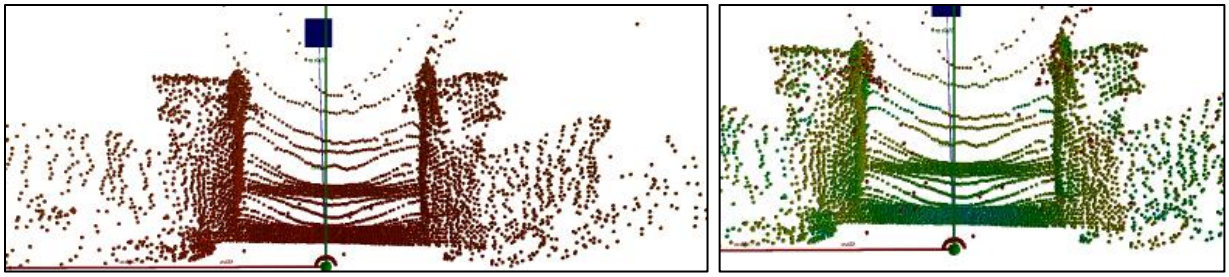


Figure 23 - Distance (left) and strength (right) scan of outside steps

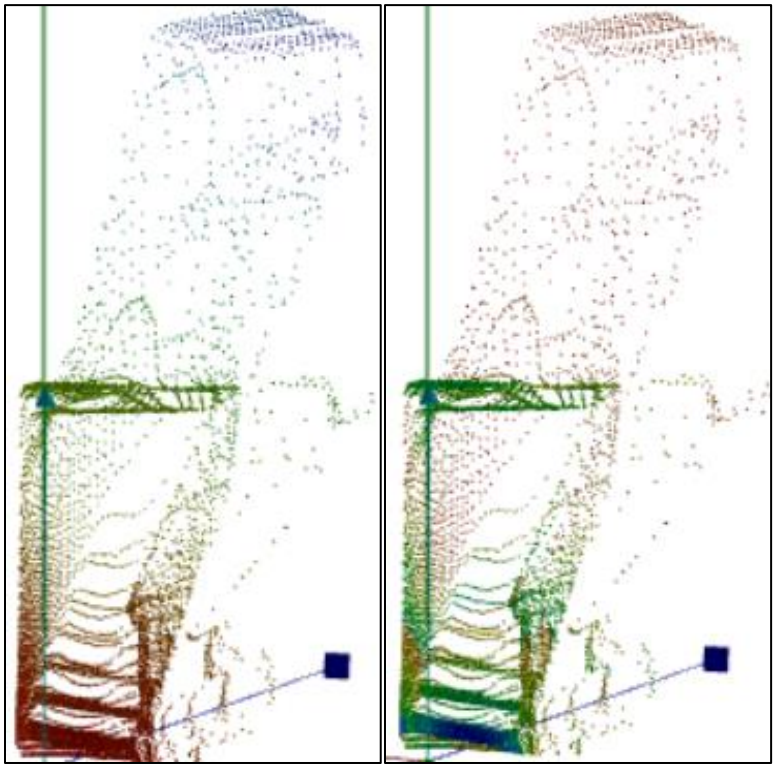


Figure 24 - Distance (left) and strength (right) scan of a staircase

In the above distance scans, the points closest to the origin are red and the points furthest from the origin are blue, with the points in between ranging from yellow to green. In the above strength scans, the points are being coloured as red if the sensor returns a low strength value, and blue if it returns a high strength value. It was found that materials which were harder, like wood, gave higher strength readings, and materials which were soft, like fabrics gave lower strength values. This agrees with the theory introduced in section 2.3.1

2.11 Research points

2.11.1 The effects of reflective surfaces on the reliability of the scan

Mirrors were found to influence the scan reliability. Instead of the infrared beams hitting the mirror and being reflected back at the sensor, the beams were reflected off the mirror and kept on going until they hit an object; in which case they bounced back towards the sensor via the mirror and created artefacts in the data.

This artefact (discrepancy in the data) was first observed in a scan of a room with a large mirror as shown in Figure 25 where the mirror can clearly be seen in the left-hand scan.

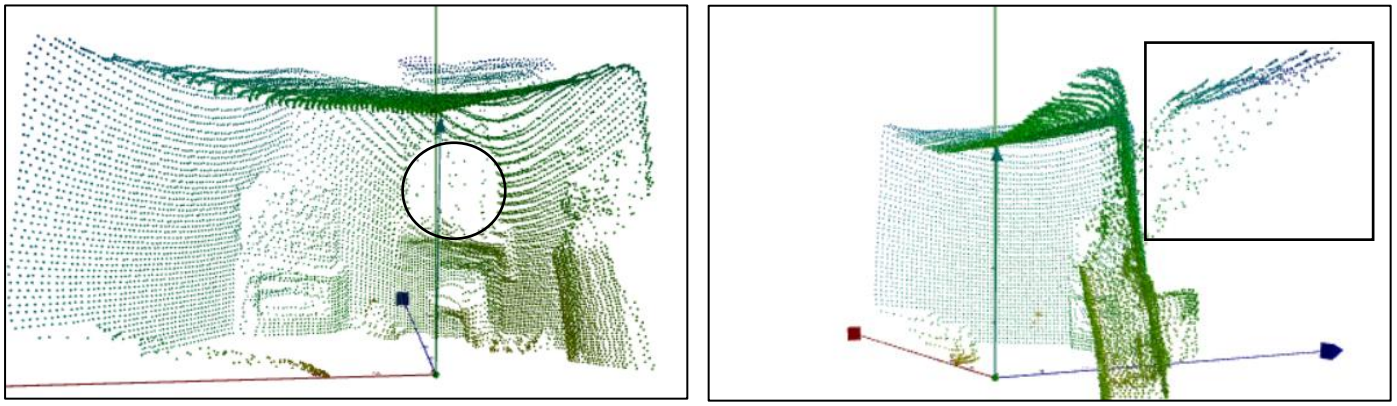


Figure 25 - Scan of room with mirror

In the left-hand scan, the location of the mirror is circled. In the right-hand scan, the square encompasses the artefact which can clearly be seen appearing to emerge from behind the mirror.

Research showed that infrared light is affected by mirrors with the degree of reflectance varying on the backing material of the mirror. Since infrared light has a wavelength between 700nm and 1mm (1µm) Figure 26 shows all three common backing materials for mirrors (Aluminium - Al, Silver - Ag and Gold - Au) have a high reflectivity (greater than 85%) for infrared light.

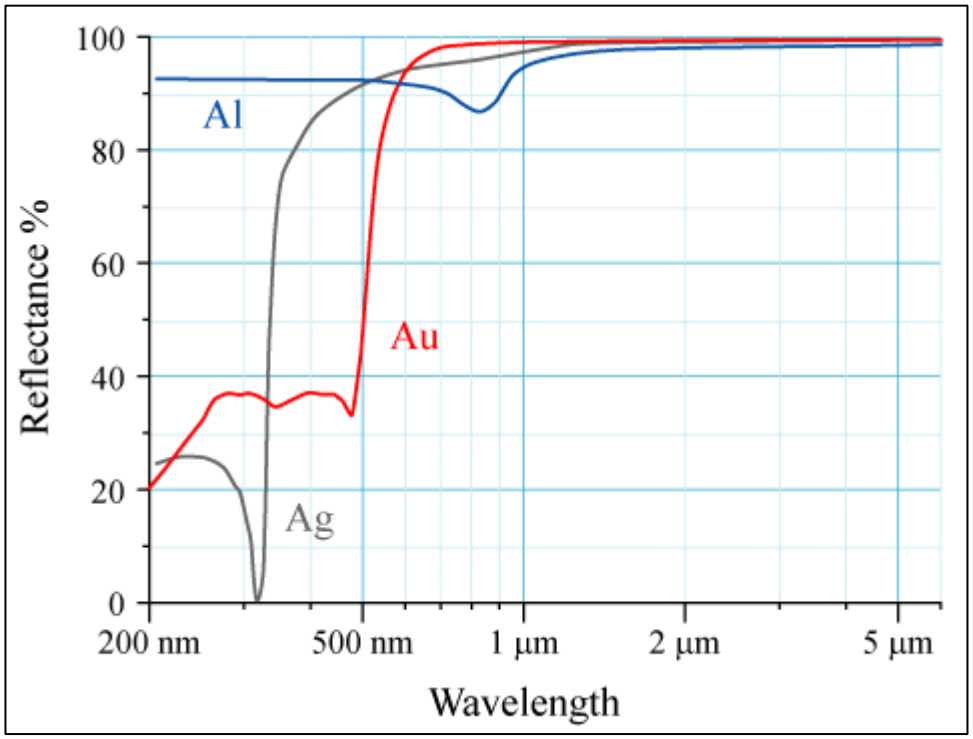


Figure 26 - Graph showing the reflectance versus the wavelength of different backing materials

To investigate the cause of the artefacts found in Figure 25, I devised an experiment to research the effects of infrared light when exposed to reflective materials with the infrared sensor set up at a pre-planned distance and angle offset from a mirror, as seen in Figure 27.

In Figure 27 the distance sensor would measure a distance of 100cm as opposed to the actual 50cm. This is because the infrared beam (from the sensor) is being reflected off the mirror to hit the box; it is only now that the beam is "bounced" off the box to reflect back off the mirror to finally reach the sensor. As the sensor assumes the beam always travels in a straight line, it does not account for the change in direction and thus creates artefacts in the data set. This explains that the artefacts seen in Figure 25 are a result of the reflections of the ceiling in the mirror.

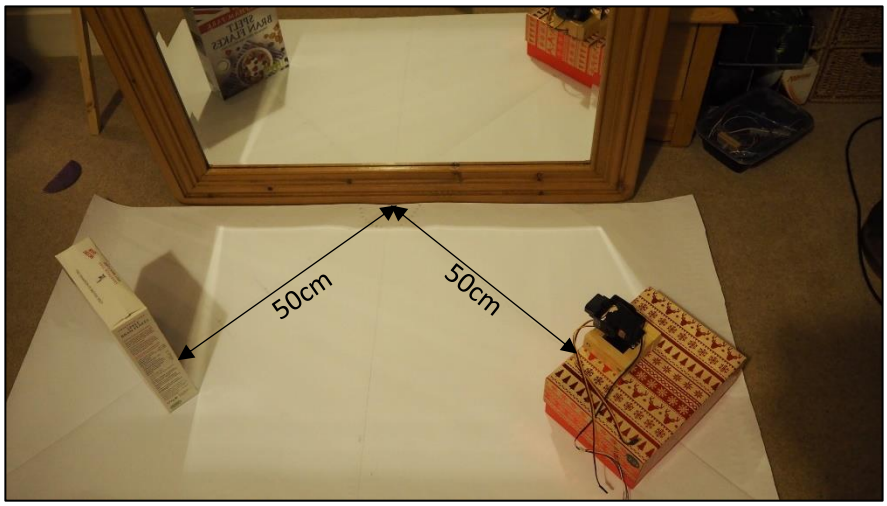


Figure 27 - Layout of the mirror investigation

To further investigate the effects of different materials, glass was tested in a similar method as above. This experiment revealed that infrared light is unaffected by glass. This was expected as infrared waves are part of the electromagnetic spectrum, with a wavelength very similar to visible light. This was shown using the layout below in Figure 28.

Figure 28 - Layout of the glass investigation

The sensor always measured a static distance in this scenario, regardless of whether there was glass in front of the sensor or the angle of rotation of the glass.

2.11.2 Impacts of scan speed and distance to the material surface on data resolution

Scan speed had a moderate impact on the resolution. The slower the scan, the more measurements the distance scanner has taken, resulting in a more reliable average. As well as this, the slower the speed, the finer the movements the pan and tilt unit takes, increasing the resolution. This shows that there is a direct positive correlation between the resolution of the scan, and the time taken for the scan.

The further the material surface is from the distance sensor, the more spaced out the dots will appear on the simulation, giving the appearance of a lower resolution. This is because the calculations in the program use spherical coordinates rather than cartesian coordinates. The program converts between the two just before it plots the point. This could be solved when scanning smaller objects if, instead of using a pan and tilt unit, the whole sensor moved vertically and horizontally.

2.11.3 Accuracy of readings by the distance sensor and pan and tilt unit

The Benewake user's manual for the TFMini Plus (the distance sensor that is being used) states that the sensor's tolerance is ± 0.5 cm, or to the nearest cm. It also states that the minimum range is 10cm. Through testing, it has been found that the tolerance value is correct. It was also discovered that the sensor can detect objects down to a minimum of four centimetres, but this results in the occasional mistake (about one in ten times) proving it to be unreliable. Due to this unreliability of results, I continued to use the recommended minimum for my scans but would implement an option for the user to view "unreliable" data points. The TFMini Plus also returns an output of strength, which depends on the number of returning infrared beams. Any points with a strength of less than 100 will be deemed unreliable.

Angular resolution is an important aspect to take into consideration when measuring distances with the infrared sensor. If part of the beam hits an object closer than the other part of the beam, see Figure 29, a value between the two is reported.

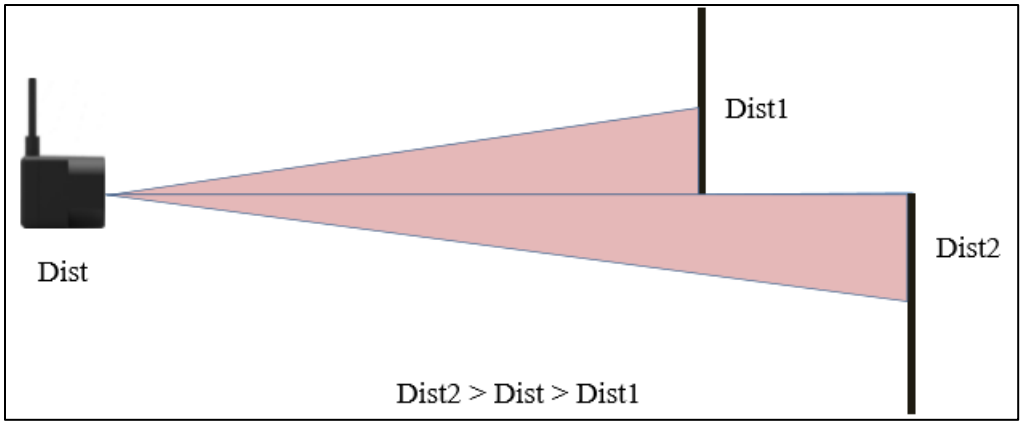


Figure 29 - Distance Measurement in the case of two Objects of Different Distances

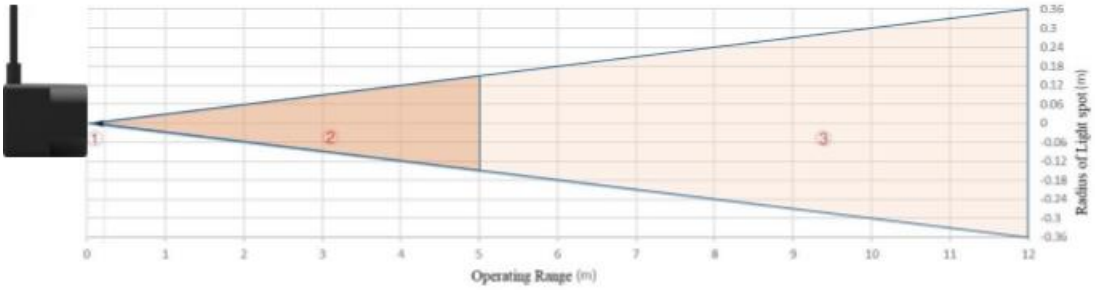


Figure 30 - Schematic diagram of size of light spot

Detecting Range (m)	1	2	3	4	5	6	7	8	9	10	11	12
Minimum side length (cm)	6	12	18	24	30	36	42	48	54	60	66	72

Table 9 - Minimum side length of effective detection corresponding to Detecting Range

Figure 30 and Table 9 show that there is a direct positive correlation between the distance to the object and the width of the beam. For example, according to Benewake - the manufacturers of TFMiniPlus, at 1 metre, the beam is approximately 6cm wide. Likewise, at 12 metres the beam is about 72cm wide. Also, according to the manufactures, the formula to calculate the beam width is:

$$d = D \times \tan \beta$$

where d is the width of the beam; D is the detecting range and β is the value of the receiving angle of TFmini Plus, which is 3.6°.

To investigate how reliable the manufacturer's claims were, I designed and carried out my own experiment.

The concept behind the investigation included using two boxes, one on either side of the beam (as shown in Figure 31), which were slowly moved towards each other. When the reading from the distance sensor change, some of the beam was hitting the boxes and interfering with the results. This was repeated for multiple distances to help obtain a more reliable result.

Figure 31 - Layout of the beam width experiment

Carrying out my own investigation, and then comparing and contrasting the manufacturer's and my own results, a graph was produced (shown in Figure 32). This graph showed that the manufacturer's claims were approximately at the inflection point of both curves.

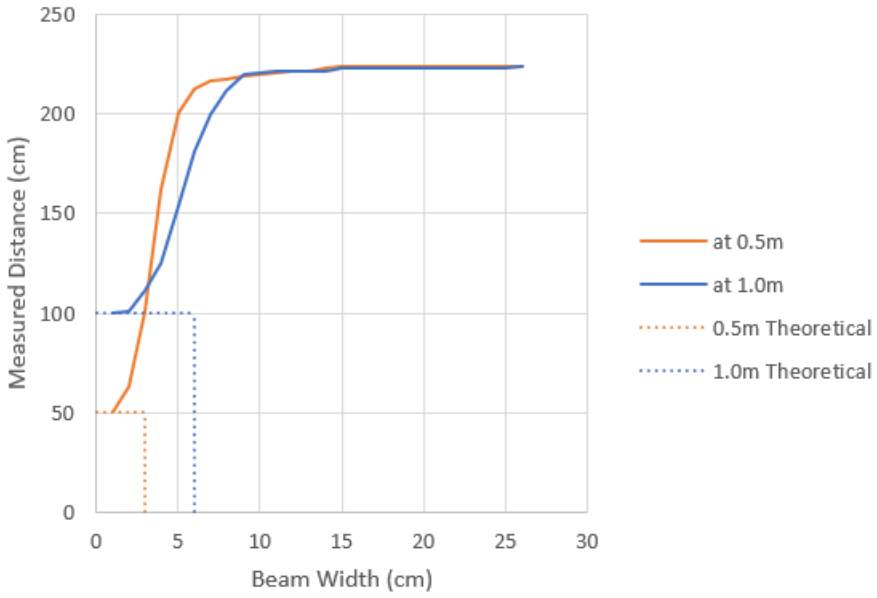


Figure 32 - Graph showing empirical Beam widths for the TFMiniPlus

The HS-422 Servo is a 180° servo; however, it can turn to approximately 200°. The minimum reliable resolution of the servo is 1°. It can unreliably turn in 0.8° intervals, but this results in the occasional pause as it is unable to move to that position.

2.12 Future development

If the project were to be taken further, there are multiple ideas that could be developed. Accuracy could be improved by using a better-quality distance sensor, probably a LiDAR sensor, and this would help create more reliable results. This is because LiDAR units have a higher resolution. For example, the Leddar™ Vu8 which has a minimum scan resolution of 6mm as opposed to the TFMini Plus's minimum resolution of 1cm. This would also reduce the time taken for the scan to be complete.

Another way to increase accuracy, is to decrease the minimum distance the servo can turn. A way to do this would be to use a geared stepper motor, for example the Geared Bipolar Stepper Motor as shown in Figure 33. This has the advantage of having an integrated planetary gearbox with a minimum step angle of 0.067°, which would provide an approximate factor of 20 increase in angular resolution compared to the servos I used. However, the cost of this stepper motor plus the necessary driver boards would be above the project budget.

Figure 33 - Example of a geared stepper motor

If a 360° LiDAR scanner, such as the X2 or G2 from YDLiDAR or the A1 from RPLiDAR, were to be mounted on a 180 degree servo, this would decrease scan time as the 360° LiDAR scanner can spin much faster than a normal servo. This would also allow for a 360° scan rather than just a 180° scan.

The following figures are third party examples of how a 360° LiDAR scanner could be mounted on a 18O° servo.

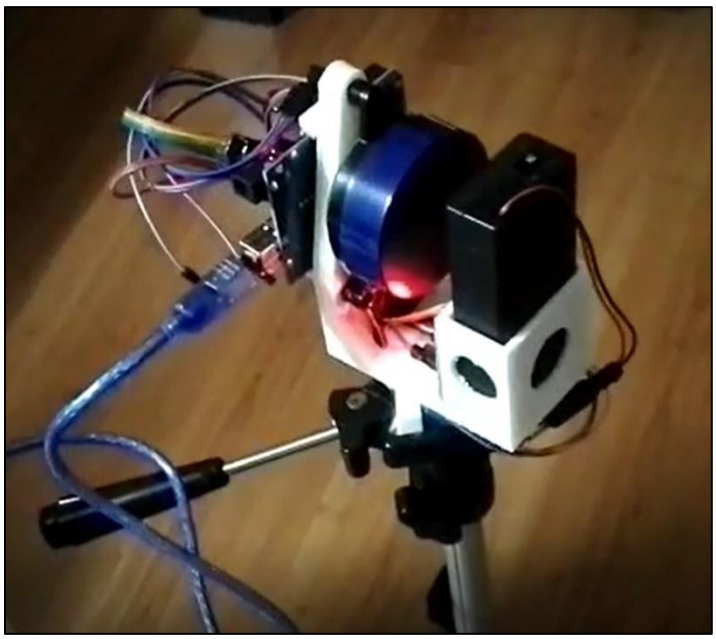


Figure 34 - Example of the X2 by YDLiDAR

Figure 35 - Example of the G2 by YDLiDAR

Another improvement that could be made is if a printed circuit board were used over a breadboard. This would secure the wires thus making the device sturdier and more portable as well as the wires being tidier.

However, these improvements would increase the price of the scanner and since the aim of the project was to produce an affordable room scanner these alterations were not added.

3 Conclusion

3.1 Project outcomes

The overall aim of the project was to design and build an affordable 3D room scanner capable of scanning spaces within a 180-degree field of view. I believe this has been successfully achieved using the constructed scanner, as the user can gain an accurate representation of the scanned space by looking at the constructed image as demonstrated by the 2D figures included throughout the document.

The project objectives were accomplished. These were:

a. to achieve the required range of movement with a pan and tilt – this was completed as discussed in section 2.5.1 to give a range up to 180° on the pan servo and 90° on the tilt servo.
b. to obtain raw data from a distance sensor (such as an infrared sensor) printing on the screen – this was completed as discussed in section 2.5.3 to give a constant output of data being shown on the screen.
c. to combine the pan and tilt motion with the distance sensor – this was completed as discussed in section 2.5.4 to have the pan and tilt unit simultaneously working with the distance sensor.
d. to process and record the raw data in a text file – as shown in section 2.5.6 processing to convert from spherical data to cartesian coordinates.
e. to display the scanned data – as discussed in section 2.5.7 this was completed since the user can easily visualise the scanned dataset.

3.2 Evaluation

Choosing to create a 3D point cloud rather than a 2D depth map resulted in a clearer image which showed the area from all angles as opposed to just one angle. Although it required more planning and programming to plot, I considered the extra time to have added value overall because the programming functionality was more user friendly.

Whilst the data display of the 2D depth map would have been easier to program and thus it would have been completed earlier, it would have been harder to show the distance of the point from the origin (distance sensor) through colour rather than distance.

Another choice which affected the project was using the Arduino Uno as my micro-controller – chosen given my existing familiarity with the unit; this would not have been the case if either a different model of Arduino or a Raspberry Pi had been used. Having completed this project, I believe the Uno was the correct interface to be used, as even though the Pi is easier to program, the Uno has higher compatibility; thus it can be used with cheaper devices which help achieve the project aim: to design and build an affordable 3D room scanner capable of scanning spaces within a 180 degree field of view.

An infrared sensor was used over an ultrasonic sensor, in hindsight, I believe this was the correct choice, however, it was chosen due to poor reasoning. I stated that my reasoning for choosing the infrared sensor over the ultrasonic was due to light travelling faster than sound. While this is true, the speeds will

have virtually no impact on the scan time; the main factor which affects the time taken for the scan to be complete is the servo movement speed. A more justifiable reason could be that light has a smaller wavelength, consequently the infrared sensor can detect smaller objects than the ultrasonic sensor can.

This project has greatly increased my knowledge on:

- both infrared and ultrasonic sensors
- micro-controllers and micro-computers
- basic Python and C++ programming
- constructing circuits with a multitude of components.

Throughout this project I have learned:

- to convert spherical coordinates to cartesian coordinates
- to create circuit diagrams
- to conduct my own research into the reliability of devices
- how different surface textures affect the reflectivity of an object.

This project has drastically increased my scientific skillset and I have become more competent in the theory behind both infrared sensors and servo control. Throughout this project I have greatly developed my time management, organisational and report writing skills. Working on the project has also deepened my understanding of waves, this is particularly useful because many consider this area to be the foundation of Physics.

4 References

Adafruit. (2020, July 13). *Adafruit PCA9685 16-Channel Servo Driver*. Retrieved from https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf

AudioVOX. (2019, January 2). *Lidar lite v3 hp*. Retrieved from RobotShop: https://www.robotshop.com/community/forum/t/lidar-lite-v3-hp/37397

Brown, N. (2017, April 25). *Introduction To PWM: How Pulse Width Modulation Works*. Retrieved from Kompulsa: https://www.kompulsa.com/introduction-pwm-pulse-width-modulation-works/

Components101. (2018, February 28). *Arduino Uno*. Retrieved from Components101: https://components101.com/microcontrollers/arduino-uno

Hitec. (n.d.). *Hitec HS-422 Servo Motor*. Retrieved from RobotShop: https://www.robotshop.com/uk/hitec-hs-422-servo-motor.html