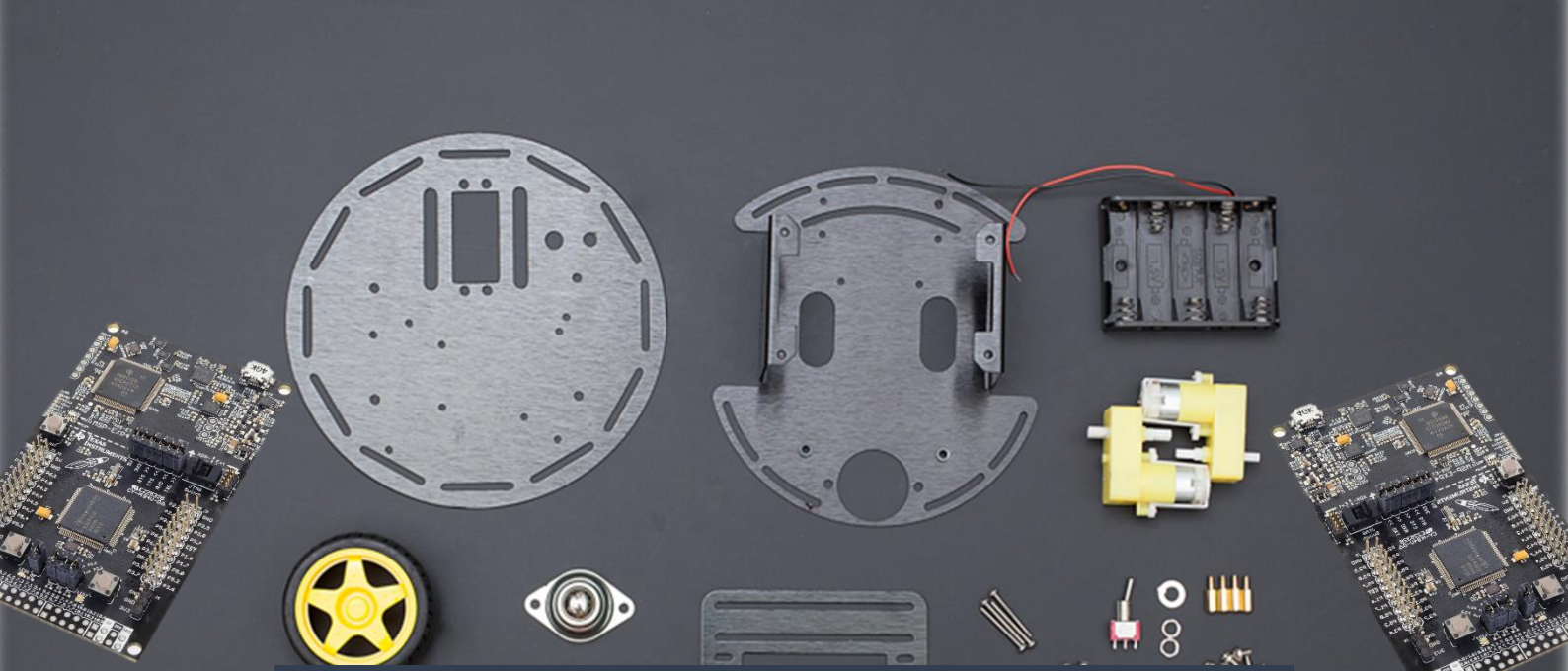

Mobile Robotics

EE303 Final Report

Team 9: Ciarán O'Donnell, Alexander McConnell and Angela Prior
Student Numbers: 15414048, 15352011 and 15309256
Lecturer: Kevin McGuinness

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying is a grave and serious offence in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references.

I have not copied or paraphrased an extract of any length from any source without identifying the source and using quotation marks as appropriate. Any images, audio recordings, video or other materials have likewise been originated and produced by me or are fully acknowledged and identified.

This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study. I have read and understood the referencing guidelines found at http://www.library.dcu.ie/citing&refguide08.pdf and/or recommended in the assignment guidelines.

I understand that I may be required to discuss with the module lecturer/s the contents of this submission.

I/me/my incorporates we/us/our in the case of group work, which is signed by all of us.

Signed: *Ciarán O'Donnell, Alexander McConnell, Angela Prior*

Contents

1. Introduction

The objective of this project was to work in a small team in order to create a line following robot which could interact with a server made for the competition and pass between a number of designated positions which were in the form of white tape crossing the main track at 90°. The robot was to be completely autonomous requiring no user input following its connection to the server and the receival of the route it had to drive.

This mobile robot or “mobot” was to be built using a combination of many components, which in turn would require a combination of skills from the team to ensure they all worked correctly and in conjunction with each other. The chassis was built using the DFRobot Turtle Mobile Platform, which is a kit containing a chassis, driving motors and all of the mounting brackets required for the rest of the parts [1]. A breadboard along with several electronic components such as resistors, wires, a voltage regulator and a DRV8835 chip were all used for the creation of the wiring and internal connections within the robot that allowed signals to pass from the inputs made of optical sensors, to the control centres of the PCB’s and out to the outputs of the driving motors. Finally, two types of PCB’s were used to act as the controller for the robot, they were the MSP432 LaunchPad board and the Wi-Fi C3100 module. These were what controlled the robot’s actions in response to the inputs received and were controlled by the code that was written and uploaded to them.

The coding for this project was all done within the Energia software package, which allowed for code to be written that would be compatible with the LaunchPad board. The language used for Energia is based off of the Arduino language and it is implemented using C, meaning no additional learning of a new language would be required [2].

The purpose of this project was to create a team of individuals who had different skill sets and academic knowledge since teams were drawn from both Electronic Engineering and Mechatronic Engineering courses. Meaning that some team members would be able to work better in some areas that in others. The key to success was to find a balance in the team so that each member could be productive and contribute to the overall success of the project. This meant that throughout good communication and teamwork were vital.

2. Development

Objective 1: Forwards and Backwards

Breadboard

In order to have the LaunchPad board control the motors they had to be connected using a breadboard and powered correctly. The entire system was run off a 9.6 Volt battery connected through a DC to DC converter which brought the voltage down to 5v, with a diode in series to prevent current draw when a USB device is connected. A DRV8835 was used between the connections for the motors and the LaunchPad, this chip allowed us to control a voltage pass through from the 5v rail to the motors from the Launchpad. As this wiring was quite simple no problems were encountered.

MSP432

The MSP432-LaunchPad acted as the main control centre for all of the functions of the robot, it had to be programmed in an ide called Energia. The first task that it had to accomplish was to have the mobot be able to go both forward and reverse. This was simple enough as all the LaunchPad had to do was to send two signals to each of the two motors, which were fixed to the wheels. One a PWM signal that controlled the voltage going to the motor and one that controlled the polarity which in turn controlled the direction. As this was easy enough there were no problems encountered in getting the mobot to do this task.

Objective 2: Follow Line

Breadboard

In order to get the mobot to detect and then be able to follow the white line that made up the track a set of optical sensors had to be connected to the LaunchPad so that they could send their signals to it. The signals sent form the sensors had to be run through a 1K Ohm resistor chip so that they would not damage any of the parts of the LaunchPad board. The sensor bank consisted of five separate optical sensors each with their own coloured wire, however once passed through the resistor each sensor was given the same coloured wires. This lead to a problem where a pair of the sensors inputs got mixed up at the input pins, resulting in the car responding incorrectly however it was easily solved.

MSP432

The LaunchPad had already been given the correct code to effectively drive the mobot backwards and forwards, now the code had to be made to have the mobot be able to detect the white line and follow it around the track. This involved taking the readings from all of the optical sensors and responding to them correctly, so if the line turned one way that would be detected and the mobot would respond in turn. The first issue was with effectively calibrating the sensors so that their values could be used. The sensors themselves worked off of projecting an IR light and measuring the amount of light reflected back at them, this signal was interpreted by the LaunchPad as an integer value. In the code for the LaunchPad the output from the sensors would only be acted upon if it was above a certain range, so the correct value for this range had to be found through trial and error testing. This was initially set to a value halfway between a sensor over the white track vs over the blask table-top. The LaunchPad was connected to a laptop and the serial monitor was used to view the exact values from each of the sensors as they ran over the line and the black base of the track. Although a suitable range was found for the sensors this value was still subject to some adjustments later on in the project when lighting and environments changed.

In addition to the calibration of the sensors, code had to be written to have the mobot be able to turn with the line around the track. As the car left the track the sensor to one side would pick up the white

line and we could tell whether the mobot was on the left or right hand side of the line. The goal was to have the mobot be moving with the middle sensor on the line always, this was done by creating two types of turns that it could perform. A large turn was made for when the outermost sensors were triggered and a slight turn for when the sensors just off of the middle one were triggered. The large turn had one of the motors turn off while it was turning, and the slight turn just changed the speeds of the motors thus causing it to turn. This method had us test many different speeds for the turns in order to balance speed reliability and a smooth ride, because if the turns were too drastic the car would jerk over and back along the track, too shallow and the car could derail from the track on sharper bends. The build of the mobot and all the connections being made at the start can be seen in Figure 1 below.

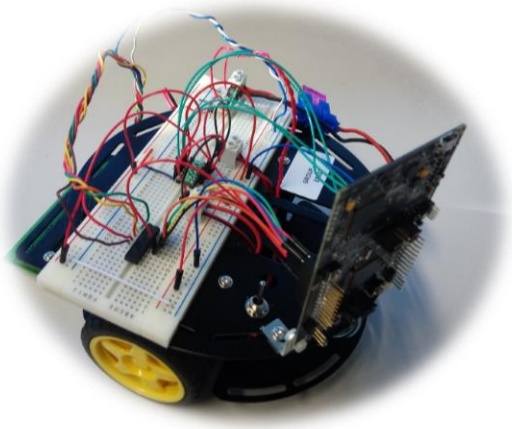

Figure 1

Objective 3: Connect to Server & play melody

Breadboard

The only addition to the breadboard and the wiring for this objective was the small Piezo electric buzzer that was implemented to play a sound. This buzzer was simply powered by the LaunchPad board and that is how its sound is controlled. The signal from the LaunchPad was run through a resistor so as not to damage the speaker, as it was a relatively simple component.

MSP432

The speaker was implemented so that it could play a melody. Using the example code in Energia and editing it we ran into an issue where we specified the pin in the code as 09 instead of pin A9. With this issue solved the melody played as it was supposed to. It was designed so that the speaker would play a fanfare each time the car reached a destination.

The big addition in this objective was the introduction of the Wi-Fi board to the LaunchPad, it was how the connection to the server was made. The wi-Fi board itself was fixed onto the pins on the back of the LaunchPad, meaning no additional connections were required. To begin with the mobot had to respond to only one form of input from this board, being able to start and stop driving on command sent to the car. The car acted as Launchpad itself acted as the server for this objective. The server was accessible from a mobile phone and that was how the control of the mobot was demonstrated. This meant that the signal given from the Wi-Fi board had to be able to trigger this response in the motors, which meant running two separate threads, one to monitor the server's status and one to control the car. These threads are only virtually running separately as the Launchpad itself has only a single core, this means that synchronisation can cause instabilities, luckily later we reworked the code to work as a single thread.

Wi-Fi

We began with the Energia example code for a Wi-Fi client and edited it to work as required. This first step was to get it to connect to the network using the ssid and network wpa password. Then the server would wait for a client to connect to its ip address (printed to serial monitor during setup) it replies with a http site. The client can press a button on this site to start and stop the car. This then on server side changes a global variable that allows the other thread to continue or stop. Figure 2 below displays our mobot at this stage in development.

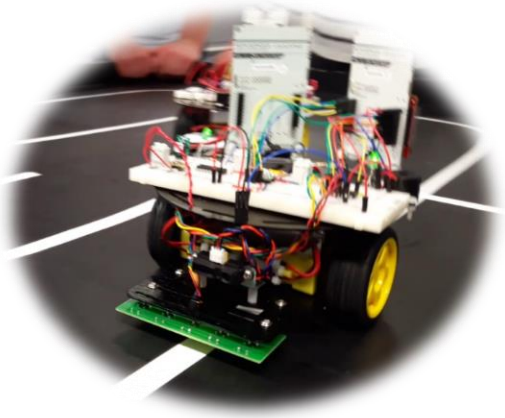

Figure 2

Objective 4: Receive Destinations and go to them

MSP432

The next goal is to have the car receive a destination from a server and travel along the white line to the correct destination. With this in mind we added a 180° turn function. Issues that were encountered with doing this was to ensuring the car would ignore the lines until facing the other way. We used a time delay function to accomplish this turn; however the plan was at some point to implement a gyroscope to perform exact turns.

The work on the actual pathfinding for the mobot was done by creating a 2d array, the index of each variable in the array corresponded to a position and the 2 elements within contained its clockwise and anti-clockwise neighbours. The car would know what position it was at and could then recursively search down through its neighbours on each side to find the destination point. How many jumps it made in the array was equal to the number of stops it had to take and whether the destination was the first or second element in the array corresponded to the direction it had to travel. The testing of the pathfinding code took awhile, as the mobot had to be able to be able to do any combination of destinations and we wanted to test them all. When the testing of this code began the board was not yet connecting to the server, therefore the destinations it travelled between had to be hard coded directly into the program instead of being received.

The 180-turn function that was implemented proved to be problematic as it was overturning itself, causing the mobot to come off the line and not be able to rectify it. This was fixed by altering that functions run time. Another problem that arose with this function was that when the car was powered on whilst sitting on the track it would change its direction prematurely using the 180° turn function and then not go to the incorrect destinations. However if the mobot was held above the track for a few moments while turning it on and then place it on the track it would go to all the points correctly. This was found to be because of the optical sensors were incorrectly triggering functions at the start

of the code based on their initial inputs being high regardless of whether they were above the line, this was fixed by having the code read from the sensors once during setup without using the values.

Once the pathfinding code was fully tested and it was found to work, but only through the use of hardcoded locations we then had to solve for the location 1 that was not on the perimeter of the track. This was achieved by hardcoding the car to get to one by first going to the cross roads on the track and then turning onto the centre line and following it too 1.

This meant that the car however would not always take the shortest route too one (sometimes the shorter route was to travel to the top of the track to get onto the centre line. It was not picking up the turn location here consistently; we decided to move on and return to optimise this later on.

No further adjustments needed to be made to the code, however there appeared to be severe driving problems in the mobot during further tests. After a long time of confusion, code debugging and hardware changing, it was discovered that a pin was moved as to stop one wire interfering with the Wi-Fi board, however this wire which was moved earlier on was placed in a terminal which was found to be broken. Once the wire was moved and location recoded, the driving was fixed and the mobot returned to its previous capabilities.

Wi-Fi

The functionality of starting and stopping the robot over Wi-Fi was already implemented into the code, but this only allowed the robot to drive forward in a straight line. The pathfinding code had to be integrated with this to have the car follow the line when started over Wi-Fi. One problem that came up when beginning to test this was that when the mobot was told to begin driving it wouldn't respond until one of the optical sensors were triggered. This was fixed by having the mobot begin moving forward as soon as it gets the signal to go from the Wi-Fi and then begin to look for the line with the sensors.

Once this was solved work then began on setting up the code to have the LaunchPad to connect itself to the server to receive a destination through the Wi-Fi board. The previous code used to control the stopping and starting of the mobot was used as the groundwork for connecting to the server to receive destinations.

The first major issue encountered with the connection was that while a connection was being made to the server to the server the only response being given was that a bad request had been sent to it. Some time was spent on diagnosing this issue before it was discovered that the connection was being made to the competition server, which was yet to be completely configured correctly for the team. Once the connection was made to the test server instead, destinations were received and the mobot could begin to follow them. With this the client could successfully pull a location from the server, with the buzzer beeping the number of times, equal to the number of its destination for testing. The destination was pulled in the form of an int from the array of characters sent from the server. An issue with this was that the html code sent by the server must be ignored and only the relevant information retained.

The biggest problem was that the mobot would only accept the first destination from the list of destinations that was sent from the server. The mobot appeared to disconnect from the server every time when the first destination was reached. The problem was found to come from a misunderstanding in the server code. Every time the mobot reached a destination an 'if(client.connect)' statement was entered to check to see if the mobot was still connected. However, it was found that this also reinitialised the connection after each destination, effectively disconnecting it from the server, requiring it to be restarted to reconnect the mobot to the server. To solve this, the

'if' statement was adjusted to 'if(client.connected)' as to not reinitialise upon arriving at each destination, and solely check if the mobot was still connected.

This should have solved the problem, however after the mobot continued stopping after its first destination, it was found that in earlier testing of solutions, there was another statement within the code that was closing the connection left at the end of the server connection loop, therefore it could not proceed to loop through again since there was no connection to the server to receive the next destination. Once this was confirmed to be the real problem it was solved quite easily and the mobot could move as intended.

Objective 5: Park at Final destination

Breadboard

To have the mobot perform the final objective of travelling to destination 5 and parking itself as close as possible to the wall without hitting it an optical sensor was required. This sensor was different to the other ones used for following the line as it was much larger and had a greater range for detection. The same principle of reflecting Infrared light is still used however. This sensor was mounted on the front of the mobot to give it an unobstructed view, and its connections were made into the analogue inputs for the LaunchPad board. The sensor had to also be calibrated so that it would tell the LaunchPad to stop driving at the right time, ensuring that the mobot wouldn't hit the wall. This calibration took a long time to complete as the response of the sensor changed greatly for seemingly small changes in the code.

MSP432

The LaunchPad had to now have the location for destination 5 saved into its array so that the mobot could effectively go there when instructed to. As destination 5 was not on the line that made up the track the mobot had to be told to go off the line at the correct location and then proceed straight until the sensor triggered telling it to stop. The main problem encountered in doing this was that the mobot would not always come off the line perfectly, causing it to reach the wall at an angle. The mobot also tended to list to one side while it was supposed to be going straight, this was solved by setting the motors two slightly different PWM values. Another issue was that the end of the track was a distance from the wall greater than the operational range of the sensor, this meant that the sensor gave false readings at this distance. The solution was to delay reading the sensor until it was close enough to get a reliable reading. This solution only ended up working part of the time, meaning that this problem was never completely solved. The final assembly of all the components and the cleaned-up wiring can be seen in Figure 3 as follows.

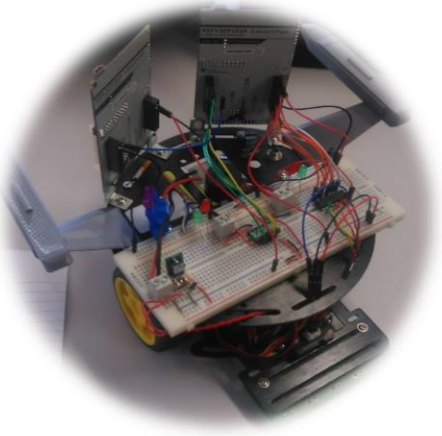

Figure 3

3. Team Dynamics & Individual Contribution

The workload proposed by this project was not evenly distributed, but given based on the individuals' strengths, while having each member learn from the strengths of the others. Ciarán decided to undertake a heavy amount of the programming due to his previous exposure to path finding in previous Electronic Engineering modules, and confidence in using 'C' because of his greater experience in the language. A large majority of programming was completed by Ciarán to fulfil the weekly objectives, all of which were finished on time. Problems were solved through group discussion and analysis. The process for troubleshooting used by team 9 was as follows in Figure 4:

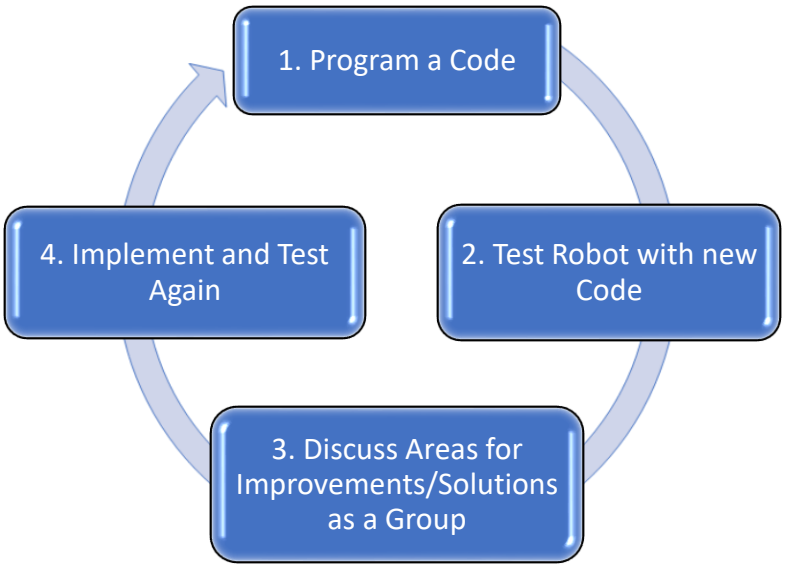

Figure 4

Alex and Angela were heavily involved in the construction portion of the project. Alex built the robot and made many of the wiring connections, implementing the circuits for use in later programming to complete the weekly objectives. Throughout the project, the wires were reconnected, by Alex, for clarity in tracing problems through the circuit and improving aesthetics. When deciding on a final aesthetic for the Robot, Alex conceptualised a Star Trek theme, settling on a name for the Robot. 'USS-Br33zy'. Alex and Angela prepared the use of the SHARP-screen module, by familiarising themselves with the intricacies necessary to understand its functions and limitations, for use in the innovation later.

The Star Trek theme was implemented by Angela, who designed the 'Starship Enterprise's' engines (Nacelles) in Solidworks, which were 3D-printed by Alex. The final innovation on the project was to create a combination locked Robot which wouldn't start until the correct combination was read by the Robot. The design for the shelf and aesthetic was completed by Angela. Due to the lesser involvement in the programming of the Robot, considered the key aspect of this module, Angela and Alex completed a majority of this final report, for fairness amongst all those involved in team 9's Mobile Robotics project. A Gantt chart shown below in Figure 5, depicts the breakdown of all the work, in finer detail, over the project's 6-week span.

Team 9 - Team Contribution Gantt Chart

Figure 5

4. Innovation

During week 1, the team realised that the innovation may be the most important objective to complete, as it gave individuality to every Robot. The innovation also proved itself to be the major factor in deciding the final competition winners. Realising this, brainstorming happened throughout week 1 and 2. Several ideas were made such as a remote-control car, speedometer and mapping out the route taken to name a few. What was finally agreed on was an idea proposed by Angela to have an innovation which calculates the change in power with respect to time, at the beginning and end of the Robot's run, to estimate how much milli-amp hours were expended and when the battery should be recharged.

After multiple group deliberations, it was decided that the innovation would be the screen detailing the efficiency of the battery at different speeds as controlled by the RPM and estimated using the MPU6050 Gyroscope/Accelerometer. This was to be accomplished using 'power' equations within the code, made relevant through wiring and programming a digital ammeter and voltmeter into the circuit. However, due to time constraints, complex implementation and the required overhaul of the circuit, to power two MSP432's in series while maintaining consistent power through both and accurately measuring the voltage and amperage, the initial approved innovation amongst the group had to be replaced with a more achievable innovation.

This was manifested in the form of team 9's Robot becoming a Star Trek-inspired Robot. Engine cores inspired by the 'Nacelles' on the 'Starship Enterprise' became the ship's innovation aesthetic, with its electronic innovation comprising of a combination lock, made from 3 potentiometers. The combination lock was incorporated with relative ease, with its method of connection to the Robot posing the largest issue. However, this was avoided through the use of a combination of brackets and screws to pinch the board in place, where the knobs were easily accessible, and the screen could be seen easily at the same time. The model and constructed version can be seen in Figure 6 and Figure 7 below.

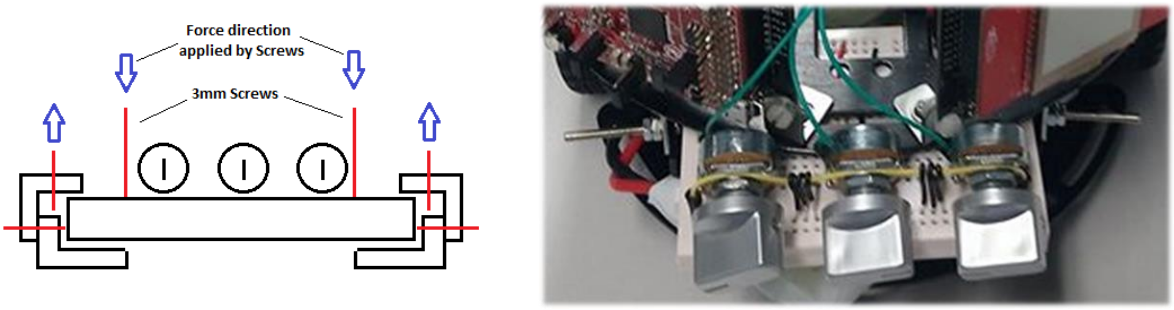

Figure 6 + 7

The pair of Nacelles were 3D printed components that were designed and added to the chassis of the robot. These were purely cosmetic in value as they had no real function or effect on the operation of the car. This was done due to the car being named the "USS BR33ZY" as a refence to other vessels from Star Trek that followed the same naming conventions. The 3D CAD program Solidworks was what was used to design these components and they were printed using a Prusa Mk.2S 3D printer [3].

5. Conclusions and Recommendations

- Locate each team member's strengths and weaknesses:
  To ensure a smooth working process amongst the 3/4 team members, it is important to have everyone work to their strengths as often as possible. This is not always an option but should be done if time and tasks allow it. When an individual works on a task that is not suited to them, it can create angst and a disconnect in the team, discouraging efficient teamwork.
  Working to people's strengths will lead to a more enjoyable experience for each member and in turn reproduce a better result and learning outcome in the team.

- Distribute and agree on division of work:
  The weekly objectives provide a great perspective of the end goal, breaking it down into more digestible targets. Since this is already done, it is not difficult to break the work down slightly further and distribute it amongst the team. This would mean smoother running of the team, with all being involved, rather than one individual taking the lead. It is also important to make sure that everyone is satisfied with the division of work such that the work load is not skewed or playing to someone's weakness.

- Start working on the innovation as soon as possible, and ensure its feasibility:
  As mentioned before, the innovation is what provides individuality to a Robot amongst the rest. The strength of the innovation determines the formidability of the Robot in the competition. Starting this early is a great way to distribute work from the very start (mentioned previously) and would benefit the team later.

- Reinstate the more mechanical portion of the module by creating the chassis using Lego to play to the mechatronic strengths:
  Mechatronic students can bring a lot of skill to this module; however, it doesn't usually show in the form of adept programming. Although programming is an integral part of this module and the module should raise the skill level of all involved, we believe this to be achievable alongside the constructive Lego chassis portion, previously part of this module. It would recreate the opportunity for electronic students to learn from the mechatronic strengths in mechanical matters and vice versa. Meanwhile, many mechatronic students have to step up fast to catch up to the skill level of their peer electronic students, otherwise leading to a significant decrease in their grade.

6. Bibliography

[1] DFROBOT, “Turtle: 2WD Arduino Mobile Robot Platform,” 2018. [Online]. Available: http://image.dfrobot.com/image/data/ROB0005/3PA%20InstructionManual%20V1.1.pdf. [Accessed 30 03 2018].

[2] Energia, “Energia/Arduino/Processing Language Comparison,” 2017. [Online]. Available: http://energia.nu/Comparison.html. [Accessed 30 03 2018].

[3] PRUSA, “ORIGINAL PRUSA I3 MK2S 3D PRINTER KIT,” PRUSA Research, 2018. [Online]. Available: ttps://www.prusa3d.com/prusa-i3-kit. [Accessed 30 03 2018].