UNIVERSITY OF LEICESTER

Final Report – Group F

Module code:	**EG4006**
Module name:	**Fourth Year Project**

04/05/2018

Author(s):	**Mohamed Khaled, Daniel Thomson, Scott Baker, Hongfei Chen, Jingyu Chen**
Student ID(s):	**149015086, 149018317, 139020585, 169038736, 169038734**
Degree:	**MEng Mechanical Engineering, MEng Mechanical Engineering, MEng Aerospace Engineering with Industry, MEng Electrical Engineering, MEng Electrical Engineering**
Tutor/Project supervisor:	**Dr. Alistair McEwan**

SUPERVISOR'S COPY/EXAMINER'S COPY

By submitting this report for assessment I confirm that this assignment is my own work, is not copied from any other person's work (published or unpublished), and that it has not previously been submitted for assessment on any other module or course.

I am aware of the University of Leicester's policy on plagiarism, and have taken the online tutorial on avoiding plagiarism. I am aware that plagiarism in this project report may result in the application of severe penalties up to and including expulsion from the University without a degree.

Executive Summary

Autonomous control has advanced very heavily over the years and is currently being introduced by many leading vehicle manufacturers in the automotive industry. Autonomous travel brings about many advantages such as for example the removal of human error which therefore reduces accidents and in turn reduces insurance costs. This technology is also readily used in aircraft and has proved successful. The LUTZ pathfinder pods are self-driving vehicles owned by the Transport Systems Catapult. These pods are drive-by-wire vehicles that rely on the input of many expensive sensors. CavLab is a facility built to provide a suitable environment for the testing of these electric vehicles. However, during testing there must always be an engineer present within the vehicle to take over, via manual control, should it be necessary. The vehicles are quite large and expensive therefore testing these vehicles, on a regular basis, can prove quite costly.

Consequently, Catapult have approached us to develop a µCavLab which will require the development of a much smaller vehicle, at about 1:6 scale, which is compatible to the interface used in the original pod. This vehicle should also contain a manual control mode, such as an Xbox controller, to take over control should it be necessary. The µPod will also need to be designed such that it uses a drive train with a control unit that can be programmed in the same way as the original pod. One of the objectives is that the µPod will be used as a representative initial testing model for testing the code and algorithms before testing is conducted on the original pod. The µPod will also be used as a demonstrator in STEM activities. This vehicle should be designed in such a way that its interface and aesthetics are as closely comparable to the original pod as possible.

By having a clear set of base requirements for the vehicle – the overall size, Ackermann steering and capabilities- the project team designed around this framework, whilst still being able incorporate individual choices in the overall design. The mechanical design was subject to a number of different iterations before the detailed design was finalised. The Ackermann steering design was finalised at an early stage along with the overall dimensions of the µPod to fit the 1:6 scale. As the project progressed the majority of developments were associated with the integration of the purchased components to the design, i.e the servo motor, drive motor and differential. A notable late design change was the flipping of the chassis, meaning the drive components were now attached to its underside. This was done to improve the ground clearance and dramatically reduce the risk of it being stuck on uneven surfaces. The Ackermann steering system functioned properly and allowed for an average steering angle between both wheels of 30° in each direction, successfully matching the values seen in commercial automotive vehicles. The motor is capable of giving a speed equal to the 24km/h top speed of the full LUTZ Pathfinder pod, ensuring the usefulness of the µPod for the purpose of performing tests in place of the original vehicle. When manufacturing and assembly was completed some adjustments were made to the final design of the µPod in order for it to function, or to save time. Even so the final assembled vehicle matches with the final design showing its suitability. The completed µPod has the mechanical capabilities to perform its required objectives and provides an effective base for the implementation of autonomous control.

The current system works on self-designed 1:6 scale µPod equipped with the control units of raspberry pi and Arduino. Compass connected on Arduino cooperates with an android phone's GPS

to provide location information for the navigation. In the meanwhile, motor and servo connected on another Arduino achieve the manoeuvrability and steering of the vehicle respectively.

The system will receive a route file containing GPS coordinates including latitude, longitude and heading as an input. Through the path planner, the vehicle will move to a goal point which has been given if a point on the route is within the range. The route is supposed to be a loop and will continue to drive following the loop until the vehicle returns to the start point.

All the software control is based on the architecture of robot operating system (ROS) and is written as a series of ROS nodes. These nodes can be classified into three parts. The first part is the sensors collecting and collating location data. Then, the second part is about the algorithm of path planner which is followed by the speed and steering demand control node. The final one is the control of actuators. The location information and vehicle motion control nodes are built by our university team and have been integrated very well with the intermediate nodes built by Transport Systems Catapult (TSC).

After several months, many tests have been implemented to calibrate the sensors and achieve the accurate control of actuators. The goal of completing a drive-by-wire test has been achieved. However, the vehicle in autonomous mode does not always accurately follow the given command and can't finish the route navigation completely. Work needs to be done to diagnose the problems in the control node and optimise the algorithm to make it work in the future. Although all the goals set in the proposal have not been completed, this project has allowed the team to learn a great deal and broaden its understanding of the whole industry of autonomous vehicle through the cooperation with the industrial company.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

The advancement of technology in the automotive sector has seen the introduction of autonomous control. This has been introduced by many leading vehicle manufacturers in their recently developed cars. This has led to increased research and development in sensors applicable to vehicles on the ground. Autonomous control is already used in air travel, making vast improvements in air travel, so it makes sense that it should be applied to vehicles on the ground. There are many reasons and motivators for the development of autonomous vehicle control. The most obvious being the removal of the human input and therefore human error such as thinking time, reaction time and so on. The autonomous vehicle is able to sense the environment and navigate without any human input. This addition of many sensors to the vehicle means there is much more information that can be used for navigation reducing accidents and therefore improving transport at ground level.

This improvement in travel safety also means increased customer satisfaction and lower insurance rates. These vehicles may also be able to relieve navigation stress and providing travel for and mobility for people that haven't been able to use these forms of transport before such as elderly and disabled people. There are, however, some limitations, as with any new technology, with autonomous vehicles. These include safety as it isn't at safe level yet for full autonomous travel, there must always be a human present behind the wheel. Technology issues and disputes regarding liability for any accident or error that may occur. Risks of hacking and loss of transport related jobs etc. These limitations must be considered when looking at autonomous vehicle transport. However, the advantages far outweigh the disadvantages therefore the development of autonomous vehicle transport can only improve the quality and safety of current forms of ground level transport.

1.2 Background

Transport Systems Catapult (TSC) is a technology and innovation company based in Milton Keynes undertaking applied research projects in collaboration with universities and industry with the aim of making the UK a world leader in Transport Innovation. One of the projects recently completed, in October 2016, at TSC is one that involves a self-driving vehicle. In fact, the company was responsible for putting a self-driving vehicle on UK public streets for the first time in October 2016. This self-driving vehicle, known as the LUTZ Pathfinder, was tested with a trained engineer present in the vehicle for the duration of the trial, to take back manual control of the vehicle should that be required. In the months following this test, the vehicle was tested, in manual driving mode, whilst mapping the route to test the autonomous control systems ability to follow the desired route. The main objectives of this project were to understand and see how the technology will work within the environment, understand how suitable autonomous vehicles are for urban use and look at, and understand, both the passengers and pedestrians that will encounter the vehicle and how they will react under the circumstances.

The LUTZ Pathfinder Pods are drive-by-wire vehicles that have both manual drive control functions as well as that of an autonomy control system. The autonomous control system these vehicles use relies on the input of a wide range of sensors, including stereo cameras, LIDAR (Laser-scanners) and radar-based obstacle detectors. These very expensive sensors are used to collect information about the local environment surrounding the vehicle such that the vehicles autonomous control can respond to it effectively and appropriately. The vehicle is able to do this due to the presence of computers that process the incoming information and steer the pods. The company is currently developing a basic autonomous control system to allow the development and testing demonstrations of a driverless vehicle.

CavLab is a facility that provides a suitable environment for the testing of these electrical vehicles and is the centre for use of these Pods. Software engineers are constantly working on the code, used by the autonomous control system, to try and improve the way the responds to its surrounding environment. However, during these testing trials there must always be an engineer present within the vehicle to take over, via manual control, should it be necessary. The vehicles are quite large and therefore quite expensive, due to the wide range of sensors surrounding the vehicle, therefore testing these vehicles on a regular basis can prove to be quite costly for the company. However, the company needs to constantly test the code and algorithm on the vehicles interface and the vehicle in action.

1.3 Project aims and objectives

Consequently, Catapult have decided on the development of a µCavLab. This µCavLab will require the development of a much smaller drive-by-wire vehicle, a µPod, at about 1:6 scale. This µCavLab facility is exactly the same as the CavLab facility in that it will be provide a suitable environment used to test the µPod. The main objectives for the development of this µPod is for it to be used as a representative initial testing model for the testing of the code and algorithms before testing is conducted on the full-size pod. The company also plans to use the µPod as a demonstrator in STEM activities. The µPod will therefore ideally be designed as similar as possible and with a compatible interface to that of the original model. This is necessary to achieve the objectives. Catapult have decided to approach us, a group of University of Leicester MEng students with this project, developing both the µCavLab and therefore the µPod. The project will also be supported by a technologist, Dave Barnett, to provide advice and access to necessary project requirements and pod specifications.

From a mechanical point of view the µPod should be designed such that it uses an Ackermann steering mechanism. The vehicle should also maintain the same or a similar interface and aesthetic to that used by the original pod such, that it will allow for accurate testing and comparisons to the original pod. The chassis will be at 1:6 to scale whilst keeping the wheelbase proportionate. Also, the vehicle chassis will be designed in such a way to allow for the required components and their mechanical linkages to be attached appropriately. This includes motors, servo, Ackermann steering mechanism and space for the micro control unit and any required sensors. The team have decided to use a thick aluminium sheet for the chassis and a wooden counterpart. This is because the chassis must be strong enough to handle any stresses it may be subjected to as well as the weight of the

components that will be used. The drive train will use one motor with a differential on the rear axle to drive the vehicle. This is seen as the best option to prevent slip.

As regards to the control part, the principal aim is to complete a system that can run the µPod to achieve two modes, one is the drive-by-wire mode and the other can process autonomous motion. The structure should be based on core code provided by TSC and expanded by the project team through adding appropriate nodes to achieve completed operation framework on our miniature µPod. Following this requirement, the micro control unit will mainly lie on the Raspberry Pi3 and other expanding boards such as Arduino will be used for more clear and concise software structure requirement based on ROS. The sensors will be chosen according to related information required by the node. The way to achieve these two modes together on the µPod begins with the GPS route recording for a certain circle within campus through the Xbox remote control which is the drive-by-wire mode. After a completed GPS route recording is collected, the autonomous mode can work by following the recorded GPS file with latitude, longitude and heading as reference, heading to the destination along with the self-adjustment. The final control structure will also allow further expanding work like more sensors added or to realize more sophisticated autonomous motion in the future.

2 Project Management

2.1 Group structure

The management framework is based on the composition of our five members. Due to the various technical disciplines, with three of the team majoring in Mechanical Engineering and two majoring in Electrical Engineering, two subsystem teams are divided. Daniel Thomson, Scott Baker and Mohamed Khaled consists of our mechanical team and Hongfei Chen and Jingyu Chen make up the control team. However, absolute division is not our aim but effective task-oriented subsystems with appropriate combination is the ideal organisation. The detailed task allocations are demonstrated below and each individual's work is shown in the following chapter.

- Daniel Thomson: The overall mechanical design
- Scott Baker: Differential part design and power section
- Mohamed Khaled: Chassis design and optimization
- Hongfei Chen: Motion node build and algorithm optimization
- Jingyu Chen: GPS and compass node build, final integration and test

Generally speaking, our mechanical team take charges of the mechanical design, such as chassis, drive shafts, and the Ackermann steering system, while at the same time the control team is responsible for the code to run and control the vehicle. In addition, the wiring and installation of the sensors are collaborative task completed by all teammates due to the simplicity of the wiring and necessary integrated work from both subsystems.

2.2 Project development

On the basis of such team organisation, the mechanical subsystem and control subsystem parallel their tasks to ensure the smooth process of the project without wasting any time. This paralleled train of thought continues through the whole project and is reflected in the initial Gantt Chart in Appendix 2. When simplified, at the beginning phase of the project the control team works with the ROS system and begins the program framework construction during the implementation of the mechanical design. Once the mechanical design goes into the manufacture phase, the control system head to tailored code for our mechanical design. At the later manufacture phase, the code is tested by means of other available model vehicles borrowed from the department to modify or optimise. Then at the final phase, the manufacture and the control part are linked up together to achieve the final test.

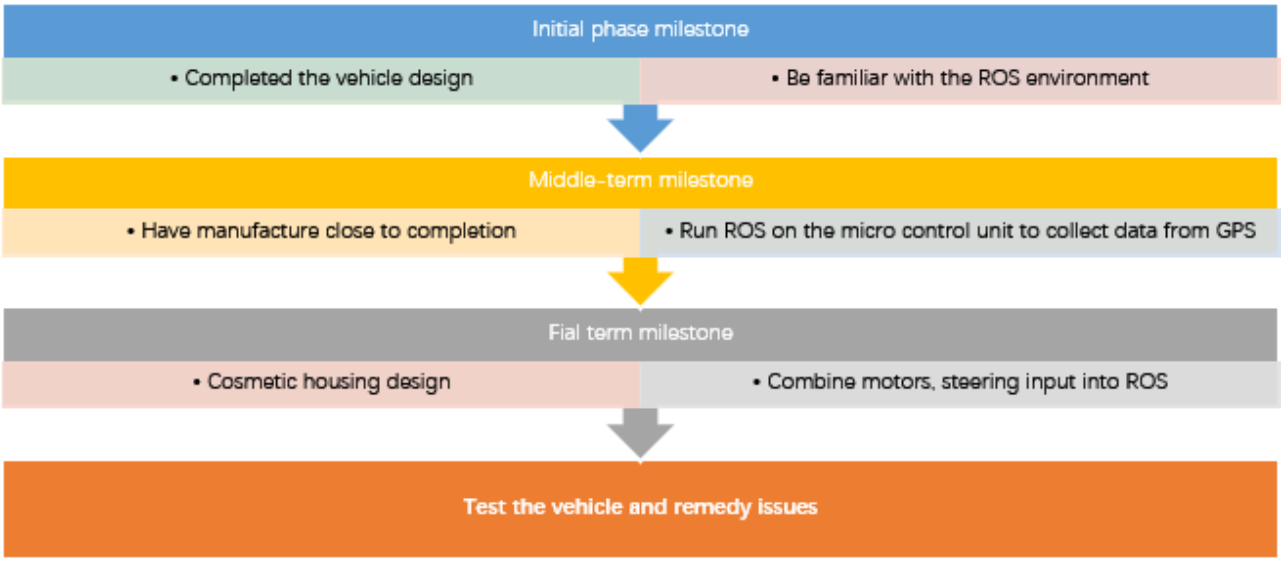


Figure 1- Project milestones

2.3 Development of Mechanical Plan

The plan for the mechanical side of the project was relatively simple due to the nature of the mechanical objectives, which was to design and manufacture a 1:6 scale model of the LUTZ Pathfinder Pod with suitability for autonomous driving. As shown in the Gantt chart in Appendix 2 the plan consisted of completing the design of the µPod for the first assessed meeting in order for the required components to be manufactured in a timely fashion. The vehicle would then be assembled, and testing could begin at an early stage in the second semester. Although due to the issues encountered during the project this timeline was delayed significantly.

The initial design was completed on time and this allowed components and materials to be sourced and purchased. All components for which the design was completed, and the materials sourced were submitted on time to the engineering workshop. Due to the delay in the delivery of the differential, the design of parts which were dependent on the size of this component could not be completed until its arrival. As a result, the plan was updated to aim to complete the manufacture of

all other possible components whilst waiting for the arrival of the differential, which is when the remaining detailed design elements would be completed.

At the start of the second semester all of the submitted designs had been successfully manufactured and the differential delivered. The next task in the plan was to complete the remaining detailed design elements and submit the subsequent drawings to the workshop. The minor manufacturing tasks and assembly would also be completed alongside the manufacture of the remaining parts by the workshop in order to try and mitigate this delay. Due to further unexpected delays a number of components related to the drive system weren't completed prior to the Easter holidays, although discussion with a technician in the workshop indicated a clear understanding of the remaining work required and the likelihood of completion during the holidays.

Once the technician had completed the required manufacturing all the components were available and ready for assembly. As the only member of the mechanical team available over the Easter holidays, Scott assembled the vehicle whilst making the necessary changes in order for it function effectively.

2.4 Development of Control Plan

The control part evolves from the basic system design, component selection, wiring mapping then to the pivotal ROS system node build and finally to the final test phase. At the very beginning, the control project Gantt chart is demonstrated below in Figure 3. From our initial plan for the control part, the control system design and be familiar with ROS operating environment was the principal task during the first semester. Along with the finish of the design, the task of the second semester focused on the programming on ROS and modification or optimization based on the test.

CavLab Project

ID	Task Description	Start Date	Planned End Date
3	**Electrical & Software Design**	**11-Oct-17**	**25-Nov-17**
3.1	Source electronic components	11-Oct-17	20-Oct-17
3.2	Design control system	16-Oct-17	25-Nov-17
5	**Control Implementation**	**15-Jan-18**	**12-Feb-18**
5.1	Install ROS(platform)	15-Jan-18	22-Jan-18
5.2	Implement GPS related ros modules(navagation)	20-Jan-18	12-Feb-18
5.3	Implement speed and steer API(motion)	20-Jan-18	12-Feb-18
5.4	Implement udp communication modules(network)	01-Feb-18	12-Feb-18
6	**Testing**	**12-Feb-18**	**24-Mar-18**
6.1	Whole System Test	12-Feb-18	19-Feb-18
6.2	Bug fixes/improvements	19-Feb-18	12-Mar-18
6.3	Add cosmetic improvements	12-Mar-18	19-Mar-18
6.4	Final testing	19-Mar-18	24-Mar-18

Figure 2- Gantt chart of control sub-system

The initial control part design was modified repeatedly but decided to follow the agenda of the Gantt chart well. However, due to the unexpected damage of battery, wrong Electrical speed

controller (ESC) purchased and the delay of the delivery, the basic check and test for all the components were delayed to the early days of the second semester.

With the critical motion control components: servo and ESC arrived early in semester two, the basic test for the motion control part was completed and later the relative algorithm and programming were developed. The extra joystick node to publish the command from the Xbox remote control was also included into the whole system during the early period of semester two. At the same time, due to the accuracy problems of the GPS and the requirement for the Inertial measurement unit (IMU) information, the GPS information output part was completed by the supplementary components and the node was successfully built. As a result, the final vehicle motion node and GPS node were both completed during this period. The whole system framework was accomplished and waiting for further modification after running on the assembled vehicle.

During the Easter holidays, the control team followed the final assembling from the mechanical team and finished the final wiring and component mapping. Afterwards, several tests are run on the entire pod to help us find if the control part was compatible with the mechanical design. Through many tests, there were some subtle adjustments made by control team such as loosening screws to reduce friction. In the end, the smooth running test was finalized by the control team.

2.5 Budget & Expenditure

The maximum allowed budget for the project was £750.00, of which we spent a total of £508.70. This was broken down into four sub-sections, materials, powertrain and steering, control and assembly. Materials came in at a total spend of £79.00 which covered the raw materials, which was the aluminium sheet, block and rod used for the chassis and mechanical links. The powertrain and steering cost was £217.76 and covered the motor, batteries and chargers, servos and ESCs. This was the largest area of wastage, with re-orders being placed on the battery, servo, charger and ESC. The orders were placed at the end of November, just before the Christmas holidays and were not collected until January. As such, the return date had passed on the battery and charger, which despite being advertised as compatible, were not. The ESC, which was for a brushed motor rather than brushless also could not be returned, which cost £13.10. When a new charger had been ordered (£8.79 waste for the first one), it was discovered that the battery was not working, either from when it arrived or through a short-circuit. This was a wastage of £55.00, and a new battery was ordered at a cheaper rate from a non-preferred supplier. The servo was burnt out during testing and required replacing at a wasted cost of £31.75. All of this came to a total waste cost of £108.64. The control units, such as the Raspberry Pi and sensors came to a total cost of £102.63. The Arduino control board was received as a donation from a group member who no longer had a need for it. The assembly cost, which covered cable adapters and fixtures and fittings etc., came to a total cost of £109.31.

Overall, the cost of reproducing the vehicle from scratch would be £400.

Sub Section	Cost
Materials	79
Powertrain + Steering	217.76

Control	102.63
Assembly	109.31
Re-Purchase Cost (Waste)	108.64

Table 1- Budget

Item	Quantity P1 (P12RE273)	Cost /unit	Total Cost (P12RE273)	Waste Quantity	Waste Cost
Aluminium Block	1	45	45		
Aluminium Sheet (360x250x6mm)	1	28	28		
Aluminium Rod (420x2.7mm)	1	6	6		
Raspberry Pi 3 Bare bones kit	1	37.5	37.5		
Servo wire extender	1	3.89	3.89		
Adafruit GPS board	1	33.1	33.1		
M12 threaded rod (1m) x2	1	1.83	1.83		
Honeywell hall effect sensor	1	1.05	1.05		
Modelcraft M3 threaded rod (500mm) x2	1	1.93	1.93		
Plastic swivel ball joint M3	1	2.08	2.08		
Modelcraft 35A ESC	1	13.1	13.1	1	13.1
Roxxy Robel 90W brushless motor	1	25.31	25.31		
Etronics PowerPal Li-Po charger	1	8.79	8.79	1	8.79
Abisma Diff Housing	1	12.3	12.3		
Abisma Diff Gear	1	5.23	5.23		
Abisma Diff Input Gear	1	13.25	13.25		
Aukey Gaming Controller	1	18.99	18.99		
GP Li-Po Battery 4300mAh	1	55	55	1	55
HiTec Servo	1	31.75	31.75	1	31.75
Kingston 32GB MicroSD	1	11.99	11.99		
Seriel Adaptor cable	1	5	5		
M2.5 x 35mm Cap Head Screw	30	0.27	8.1		
Brushless ESC 110W	1	17.09	17.09		
Rubber Castor Wheels	4	3.9	15.6		
3mm Neodynium Magnets x50	1	5.51	5.51		
M6 Spring Washer x100	1	2.6	2.6		
Velcro Hook Tape	1	7.34	7.34		
Velcro Loop Tape	1	7.34	7.34		
M12 Nylock Nut x100	1	9.13	9.13		
M2.5 Grub Screw	1	8.18	8.18		
Centro 5000mAh Shorty Li-Po Battery	1	37.99	37.99		
Etronix Li-Po charger	1	23.74	23.74		
Core RC Li-Po Balance Lead	1	4.99	4.99		
			508.7		108.64

2.6 Risk Log

Table 2- Risk log

No.	Date Identified	Risk	Probability (L,M,H)	Impact (L,M,H)	Effect on Project	Risk Reduction Actions Proposed & Actual	If it happens: Triggers & Actions Proposed & Actual
1	10/17	Loss to project of a group member	Low	High	Unable to complete key tasks on schedule due to decreased productivity. Increased workload for team members. Loss of overall team knowledge.	Ask that all members email ahead of meetings if they are going to be absent. Ensure all members are kept in the loop.	Triggers Not turning up to group meetings. Actions There are 5 current group members, down from the original 7. This has been reported to the supervisor and course coordinator. The remaining workload needs to be allocated to remaining team members.
2	10/17	Over budget	Low	High	Might be unable to purchase all the components. Unable to fix any unforeseen circumstances.	Require justification for purchases. Prepare and update a budget. Have a contingency fund for any unforeseen circumstances (£100/750).	Triggers Expenditure is high without checking funds or components require replacing. Actions Review purchases. Return unnecessary purchases. Cut spending. Use contingency money.
3	10/17	Delayed decision making	Low	Medium	The schedule will be delayed, missing key deadlines. Decreased testing time. Increased group tension.	Give deadlines for making decisions and feeding back in weekly group meetings. Arrange to meet to discuss important things outside of meetings	Triggers Too much time spent searching for and weighing up options. Actions Open the topic up to group discussion to aid the decision-making process and set a deadline for the decision to be made.
4	10/17	Poor intra-role communications	Low	Medium	Disjointed work, lack of cohesion. Delays work if ideas are not being shared and discussed. Could delay project if roles aren't integrating and thus designs don't integrate.	Monitor project activities. Encourage communications in sessions, have a designated time each week to update progress of individual tasks and about the overall project. Group members have created a group chat to discuss ideas outside of the meetings.	Triggers Evidence at Group meetings that members are 'out of the loop'. Quiet group members at meetings. Incompatible sub systems. Actions Ask members to discuss their ideas and progress with the rest of the group regularly.

5	10/17	Infeasible design	Low	High	Over complication will require extra work from members and will slow down the progress. Could force the cost too high by having more components/sensors. The design might not meet the requirements.	Thorough design process and integration of all roles through communication. Read specification document thoroughly and ensure all team members know the sections relevant to them.	Triggers Complexity of design. Design not within requirement or cost. Actions Ensure team members are regularly checking the specification document against their design. Seek technical guidance if necessary.
6	10/17	Loss of data/access to data	Low	High	Work will have to be re-done, delaying the project.	Back up all data to group OneDrive and external HDDs.	Triggers Unforeseeable IT problems. Negligence. Left USB in computer. Actions Try to recover lost data. Re-do work with full team involvement.
7	10/17	Supplier not delivering component on time	Mediu m	High	If components aren't delivered on time, there will be less time for manufacturing and critically, testing of the vehicle.	Order in stock items. Call suppliers to check stock before making a purchase.	Triggers Unforeseen postal delays Actions Contact supplier urgently to acquire an explanation and expected arrival date. Seek alternative supplier/component.
8	10/17	University workshop being unable to complete work on time	Mediu m	High	If components aren't delivered on time, there will be less time for manufacturing and critically, testing of the vehicle.	Get the designs in ASAP to be at the top of the queue. Make sure all drawings are technically perfect to avoid delays.	Triggers Poor dimensioning of drawings. Overworked/overloaded technicians. Actions Speak with technicians to resolve issues. Go into workshop and manufacture parts ourselves.
9	10/17	Vehicle breaks during testing	Low	High	The vehicle will have to be fixed, costing time and money	Test the electrical components before fitting. Ensure all nuts and bolts are tight	Triggers Overloading the electronics. Poor human control leads to the vehicle failing or crashing Actions Replace broken components. Improvise a fix if there isn't enough time for suppliers to ship replacements.

11	01/18	Sensors too inaccurate	Medium	High	The vehicle won't be able to accurately follow a route	Enquire about accuracy/check specs	Triggers GPS data is too inaccurate Actions Implement a more accurate GPS sensor (Phone) Use multiple sensors
12	02/18	Vehicle too low to ground	Medium	High	Unable to meet specification of driving outdoors	Design accordingly	Triggers Raised at meeting Actions Chassis can be flipped to increase clearance
13	02/18	Steering has too much play	Medium	Medium	The vehicle will not accurately follow the route	Design carefully Use low tolerances	Triggers Raised at meeting Actions Implement spring washers to ensure steering doesn't work loose Use two nuts to 'lock' the spring washer in place

3 Individual Technical Achievement

3.1 Daniel Thomson

In accordance with one of the objectives set in the initial meeting with the representative from Transport Systems Catapult it was decided that an Ackermann steering system should be used in the vehicle to prevent slip and mimic the steering mechanism used in commercial vehicles. Consequently, the first activity assigned to me was to design the joints for the front axle that will allow the inner wheel to turn more than the outer wheel to meet this objective. As it was also decided that the vehicle will be rear wheel drive this simplified the design process for the Ackermann joints.

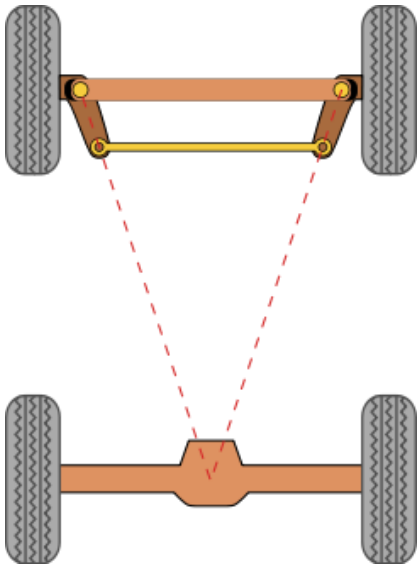


Figure 3- Ackermann theory

The design was based upon the simple approximation of Ackermann steering which links the front and rear axle as shown in figure 3. Using the data provided for the Lutz Pathfinder wheelbase and track a 1:6 scale pod dimension could be created, therefore providing the data needed to design my own Ackermann system.

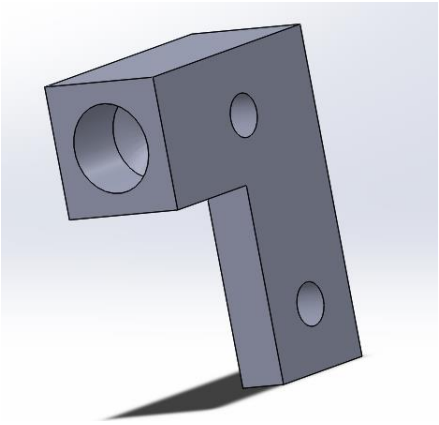


Figure 4- Ackermann joint

Figure 4 shows the left side Ackermann joint as designed, with the two M6 clearance holes that allow the stationary support and tie rod to be connected. There is an M12 threaded hole that allows the wheel rod's to be attached and in turn the front wheels which are free to rotate.

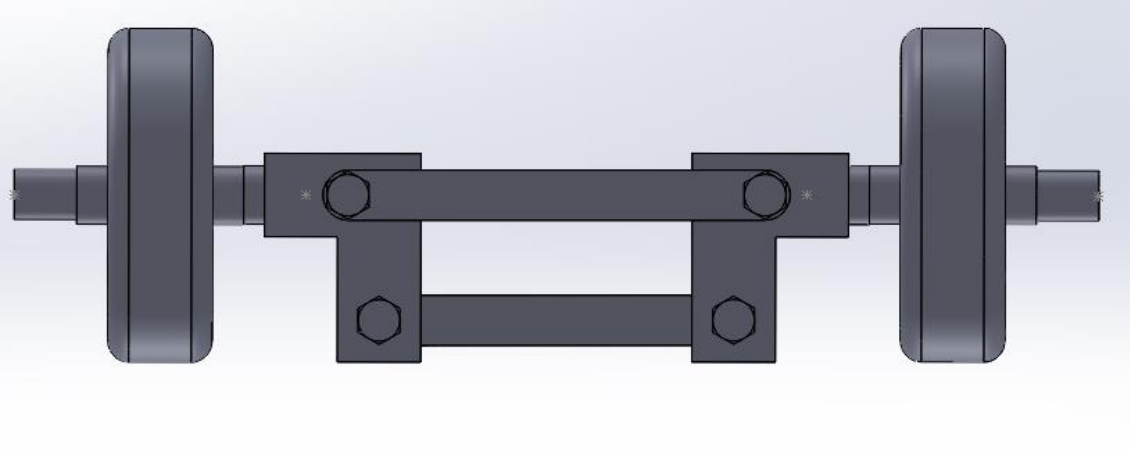


Figure 5- Steering assembly

Once I had designed the stationary support, tie rod and wheel rod's an assembly of the front axle could be created as shown in figure 5. I attached the wheels and modelled an M6 screw to act as the axis of rotation for the system. A 5mm spacer was added separating the Ackermann joints and the wheels so that the correct track was obtained.

In preparation for the first assessed meeting a complete initial design was needed. Upon receiving the initial chassis design I made the changes necessary to allow for the steering mechanism to function, such as moving the front supports and cutting a shape into the chassis that allows for rotation of the wheels. The holes for the front axle screws and the screws to attach the bearing housing was also added. Figure 6 shows the chassis design used for the first assessed meeting.

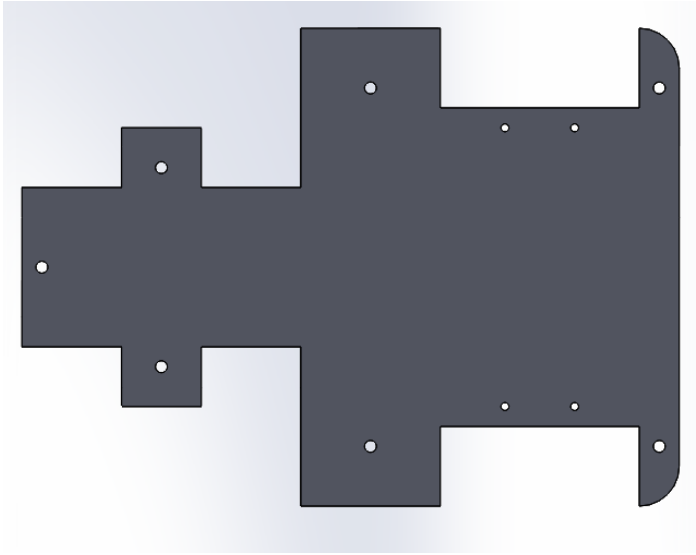


Figure 6- 1st chassis

With the initial design for the components my next task was to assemble the initial design of the vehicle, making any necessary adjustments as necessary. Figure 7 shows the completed assembly

without the top plate, with the simple approximation of a motor and differential at the centre of the rear axle. Connecting the servo motor and the steering system I modelled two simple ball joints attached by a rod to provide a demonstration of the system that would be used, with more work on adding detail to this to come.

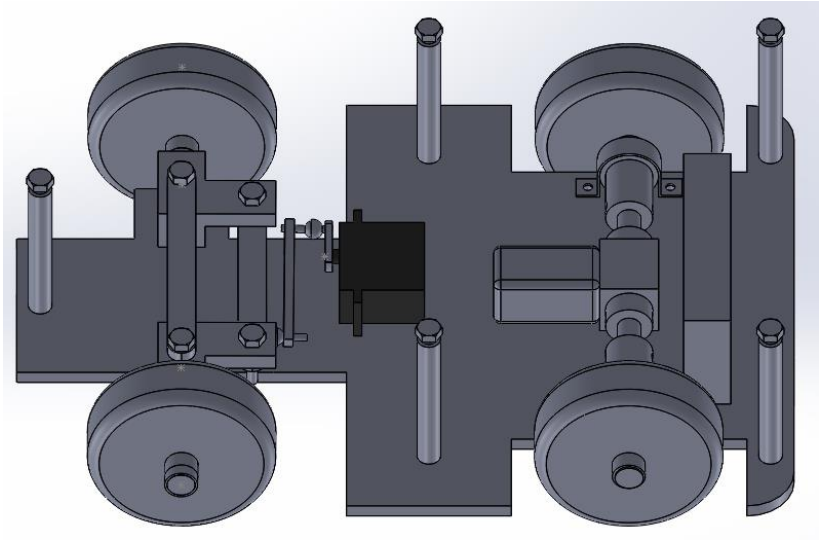


Figure 7- Initial design assembly

My next step was to produce the engineering drawings for the parts that were ready to be manufactured. These were the Ackermann joints, front axle stationary rod and tie rod and the chassis supports, all of which can be seen in Appendix 2. With these parts in the manufacturing stage and while the ordered differential was in the process of being delivered, no further significant design work could be completed. Figure 8 shows the cosmetic attachment for the top of our vehicle that I designed to meet an optional additional objective. As detailed later on this could be used in conjunction with a 3-D printer to make our completed vehicle more recognisable as a miniature Lutz Pathfinder pod.

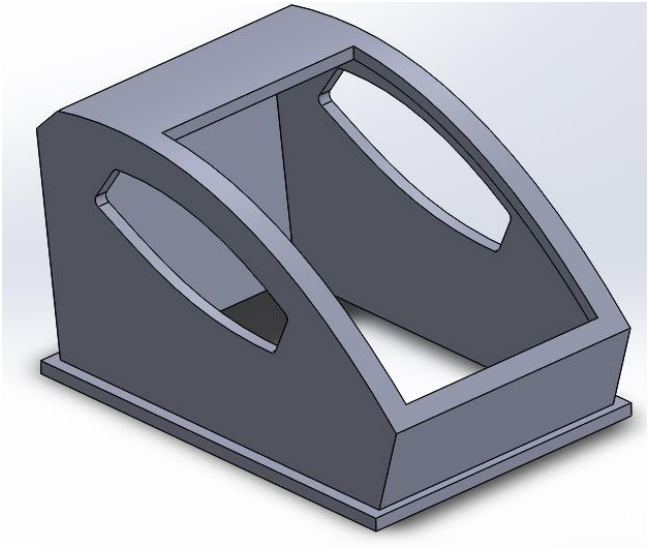


Figure 8- Cosmetic attachment

Once the various purchased components arrived, including the differential, ball joints and bearings, the detailed design for the rear axle components and servo steering system could move forward.

Consequently, the detailed design could be completed, and the remaining parts submitted to the workshop for manufacture.

For the servo motor to be securely attached to the chassis fastenings were designed. These are either side of the servo seen in figure 9 and have screws through the servo and chassis to ensure that the motor is secured when in operation. I also modelled the ball joints that were purchased to give an accurate representation of the components that are being used. By also modelling the servo attachment I was then able to design a threaded rod connecting the ball joints and giving a complete design of the steering system.

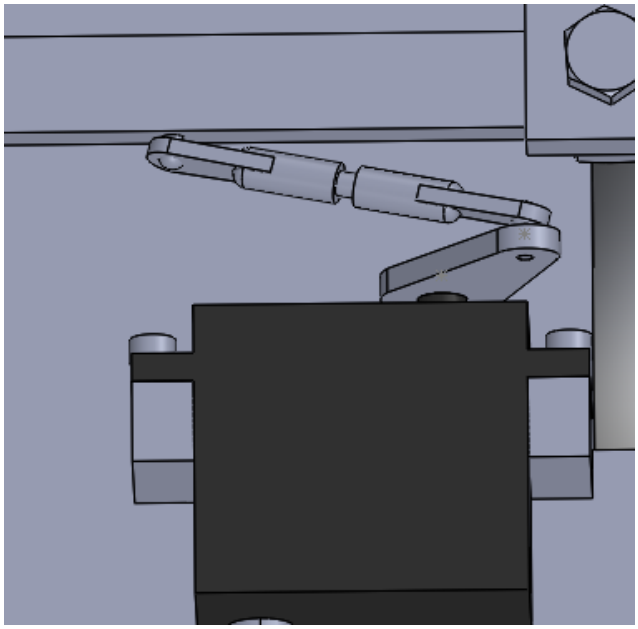


Figure 9- Servo design

As the motor to drive the vehicle had been chosen I was able to accurately model this, along with the mount that is used to attach the motor to the chassis. Due to the size of the differential a platform was required to raise the height of the motor and allow the shat that connects the two to function effectively. As can be seen in figure 10 the motor sub-system was then assembled for use in the overall design. There are holes in the mount and platform that allow for the secure attachment to the chassis using M3 screws.

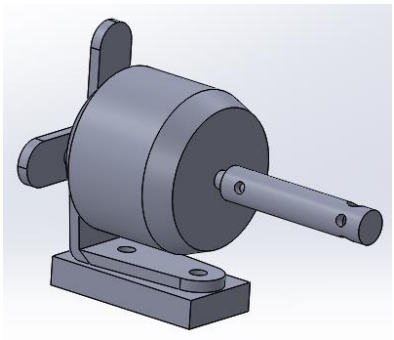


Figure 10- Motor assembly

The differential was modelled to give an accurate representation of the component, which allowed for the detailed design of the rear axle components. When the design was completed I was able to

assemble the components of the rear axle, including the simplified threaded driveshaft, bearing housings with attached bearings and the rear wheels. Additionally, the connection between the motor and the differential could be made, resulting in the completion of the detailed design of the rear axle and drive system.

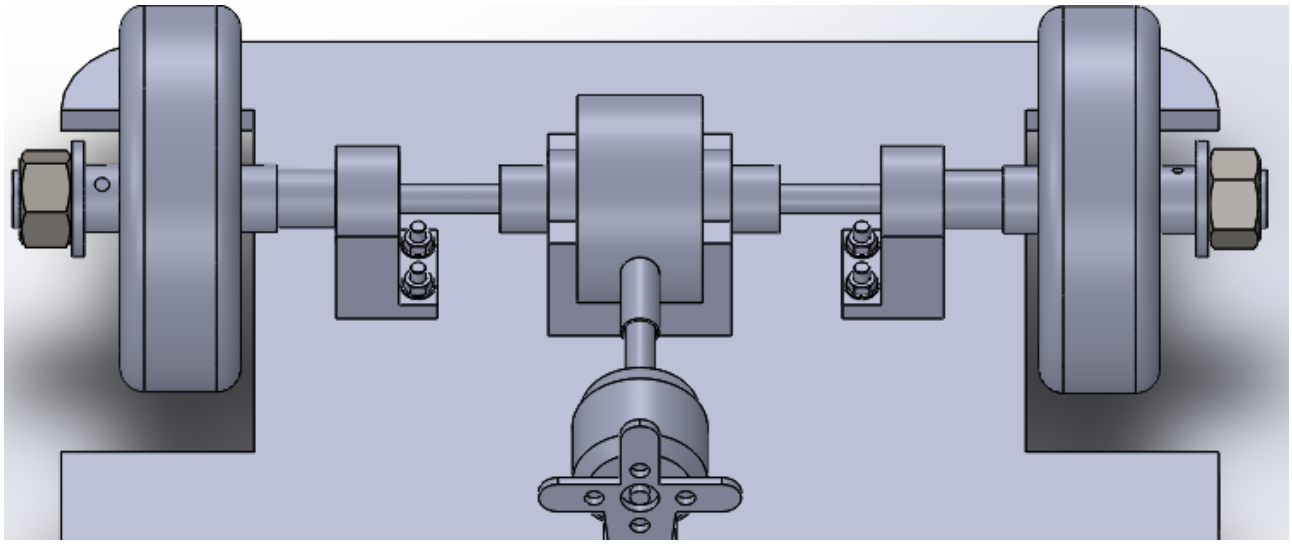


Figure 11- Rear axle assembly

Figure 12 shows the chassis with the additional holes that are required for the attachment of the servo fastenings, bearing housings and motor sub-system. This was done as the positioning of these components had been finalised and therefore the remaining manufacturing processes could be performed on the vehicle's chassis.

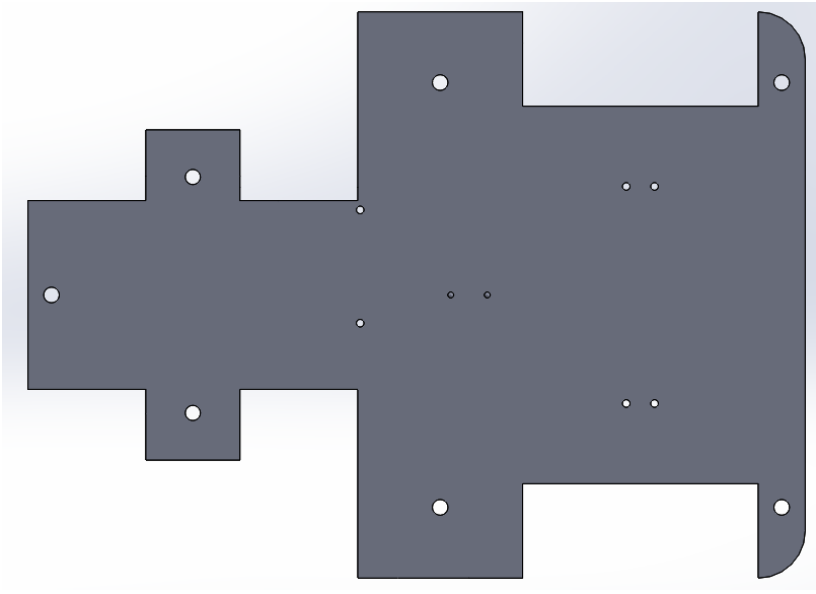


Figure 12- Updated chassis

Once the modelling of said components was completed and the modifications to the chassis made, a higher level of detail could be added to the design. Figure 13 shows the mechanical components of the design all accurately modelled and in their finalised position on the chassis. My next step was to add the screws, nuts, and washers required to the Solidworks model to allow for an accurate representation of the further components required to assemble the vehicle. This meant that when

the assembly stage began no time was wasted having to wait for a component that wasn't available in the university workshop to be delivered leading to an efficient use of time.

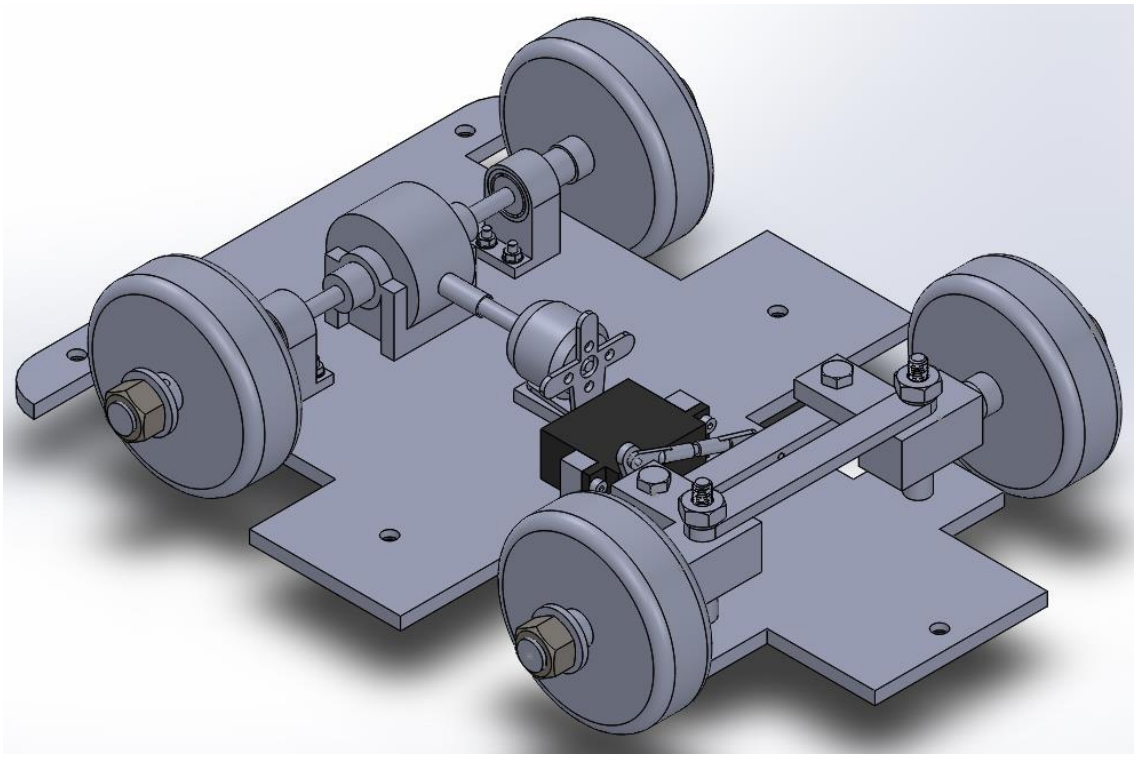


Figure 13- Completed design without top plate

Figure 14 shows the completed design with the top plate attached, which was made to ensure that all components connect properly without causing a potential issue, and that the overall design is still viable. As can be seen due to the size of the differential the ground clearance of the vehicle is low which could potentially lead to problems when in operation. This was how the design remained until the third assessed meeting when the ground clearance issue was raised and an attempt made to remedy the problem.

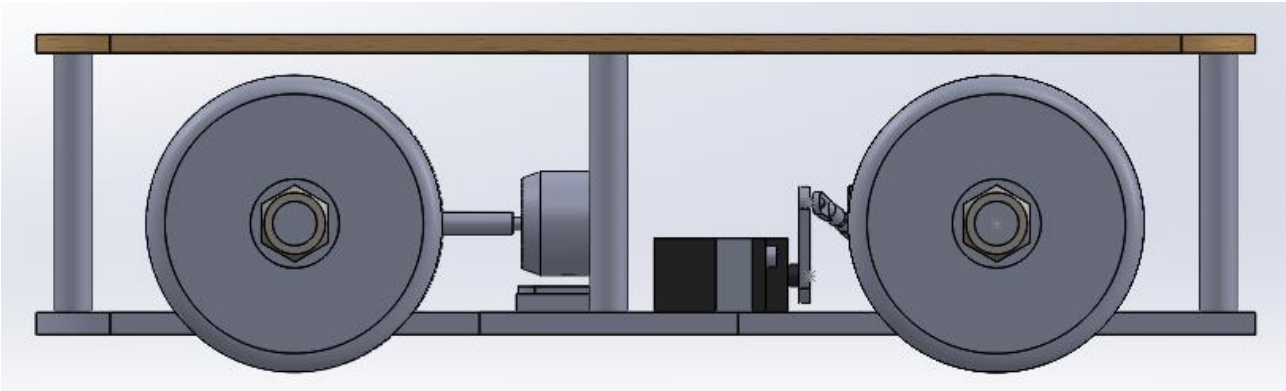


Figure 14- Completed design

After the completion of the design of the remaining components and the modifications made to existing components I produced several engineering drawings. The drawings made were of the threaded driveshaft, bearing housing, motor platform, motor shaft, left and right servo fastenings, front axle wheel rod, base chassis (with the added screw holes), and the front axle spacer. These

drawings can all be seen in Appendix 2, which allowed the final components for manufacturing to be submitted and for the project group to make the adjustments to the chassis ourselves.

To provide autonomous functionality, the relationship between the rotation of the servo motor and the resulting angles of the wheels must be known. This can only be accurately calculated by performing tests on the finished prototype, but an estimation can be provided. As shown in figure 15 servo attachment that is connected to the ball joint steering arm starts off-centre. Ideally the servo will be attached to the centre of the tie rod to provide an even maximum steering in each direction. Due to the small amount of space afforded to us by a vehicle of this size a restriction was applied by the components that could be purchased. As such the servo is attached to the most central part of the tie rod allowable by the ball joints, leading to this off-centre starting position. My first task was to calculate the angle from vertically upright that the servo attachment will start at. By using the tool available on the Solidworks programme measuring from the point at which the attachment is connected to the servo to where the ball joint is connected, the vertical and horizontal components are provided. Through simple trigonometry shown below, the servo attachment was found to start at angle of roughly 28° clockwise when facing in the direction of forward motion.

$$\theta = sin^{-1}\left({}^{19.44}/_{22}\right) = 27.92° \approx 28° \quad (1)$$

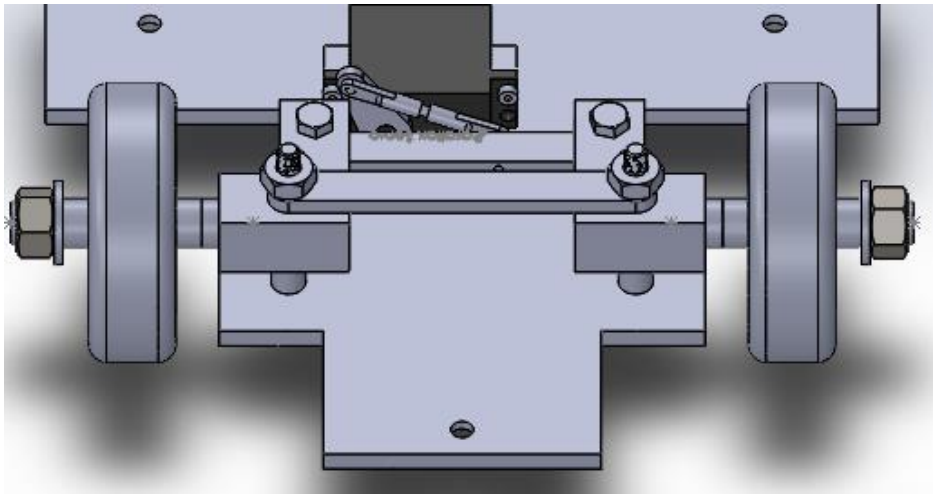


Figure 15- Steering system front view

The next step was to calculate the steering angles for a range of servo rations in order to provide a rough understanding. By applying a rotation to the servo attachment using the move component tool on Solidworks, the resulting steering angle of the wheel could be calculated. Figure 16 shows an example of when the servo was rotated 20°anticlockwise. Again, through trigonometry the resulting angle of rotation of each wheel can be to provide the steering angles for a given servo rotation.

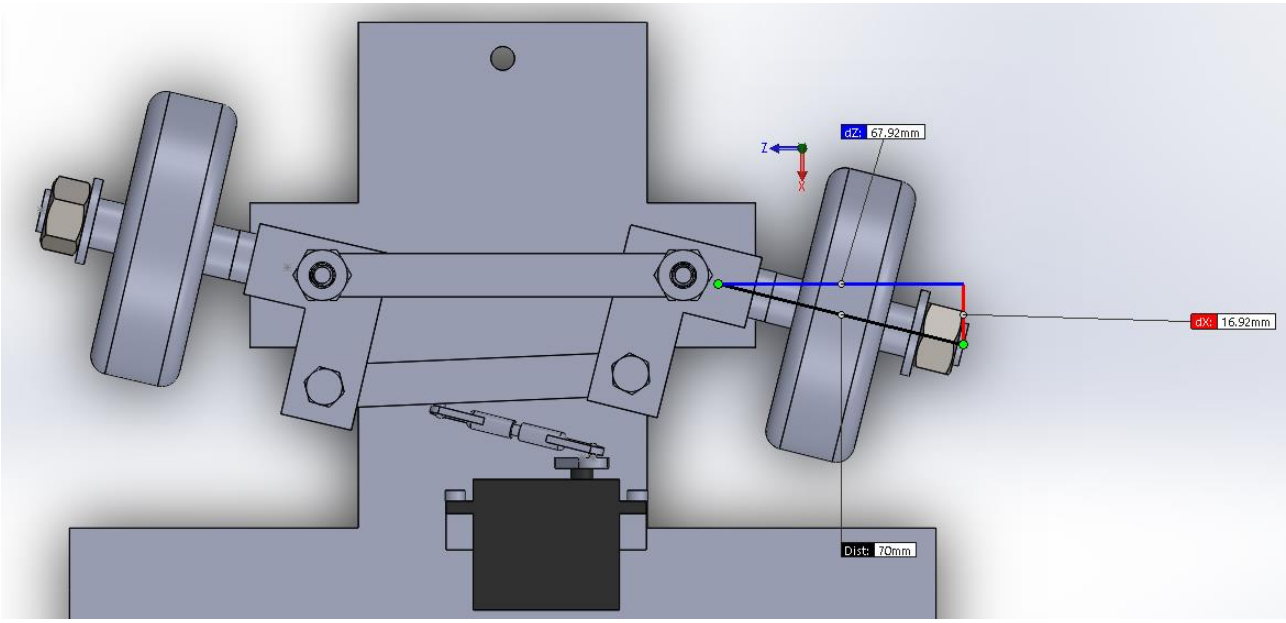


Figure 16- 20° anticlockwise servo rotation

This process was repeated for several servo rotations to give the inner and outer wheel steering angles, as summarised in table 3. As the inner wheel rotates more than the outer wheel due to the Ackermann steering system the average rotation of both wheels was taken. By dividing the average wheel rotation by the servo rotation, the steering ratio was calculated. This number is used to represent the degree of rotation of the wheels per degree of servo rotation. For a commercial automotive this number would be a constant, although due to the off-centre positioning as mentioned previously this is not the case here. The steering ratio is greater when the servo moves anticlockwise as a direct result of the positioning of the connection between the servo and the tie rod. Additionally, in each direction the steering ratio increases as the servo is rotated a larger amount, meaning there is a non-linear relation between the two.

Table 3- Steering calculations

Servo Rotation	Steering Angles			Steering Ratio
	Inner wheel	Outer wheel	Average	
20° anticlockwise	14.0°	13.0°	13.5°	0.68
20° clockwise	8.95°	8.54°	8.75°	0.44
40° anticlockwise	32.6°	27.1°	29.9°	0.75
62° clockwise (maximum)	32.7°	27.2°	30.0°	0.48
50° anticlockwise (maximum)	43.9°	33.4°	38.6°	0.77

Figures 17 and 18 shows the steering angles at maximum clockwise rotation and 40° anticlockwise rotation respectively. As the average steering angle provided by these servo rotations are roughly 30° an estimate of the desired range of servo movement was found. As the maximum wheel rotation of many commercial vehicles is around this 30° value, the suitability of the turning circle that the vehicle will be capable of proportional to its size is acceptable. It was possible for the vehicle to steer further when the servo rotates anticlockwise but it was decided to keep the maximum steering angle the same in both directions.

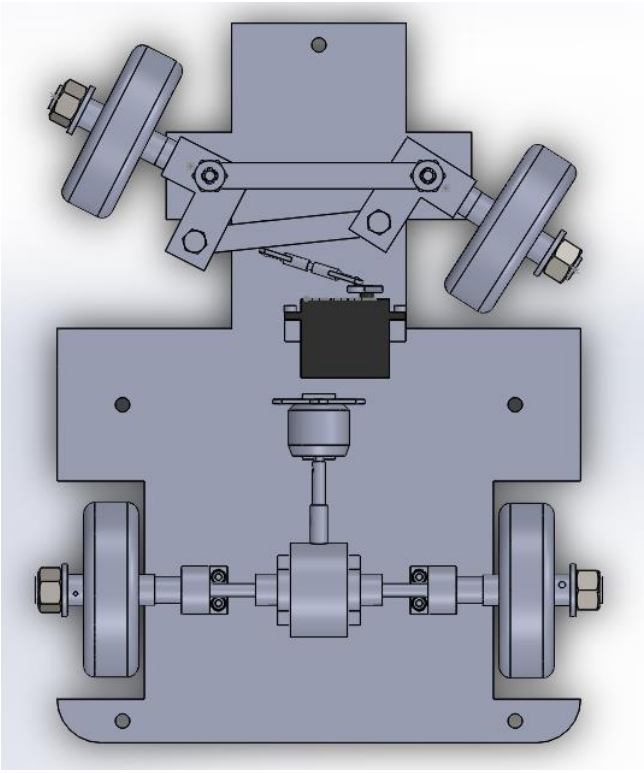


Figure 18- 40° anticlockwise rotation

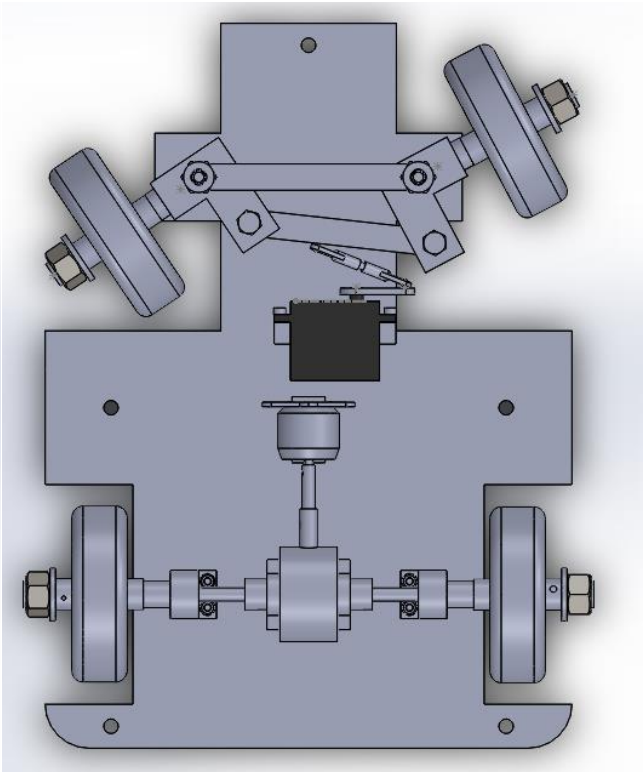


Figure 17- Maximum clockwise rotation

To attempt to remedy the ground clearance issue mentioned previously, the idea was raised to mount the drive components on the underside of the chassis. To investigate whether this solution was viable with the components that had already been manufactured I re-modelled the entire design. The resulting final design as shown in figure 19 was possible with only some minor adjustments required in the chassis. An outcome of this change the steering values of the steering angles remain the same but are now reversed. Therefore, where previously the vehicle could steer right to a greater degree it can now steer left further.

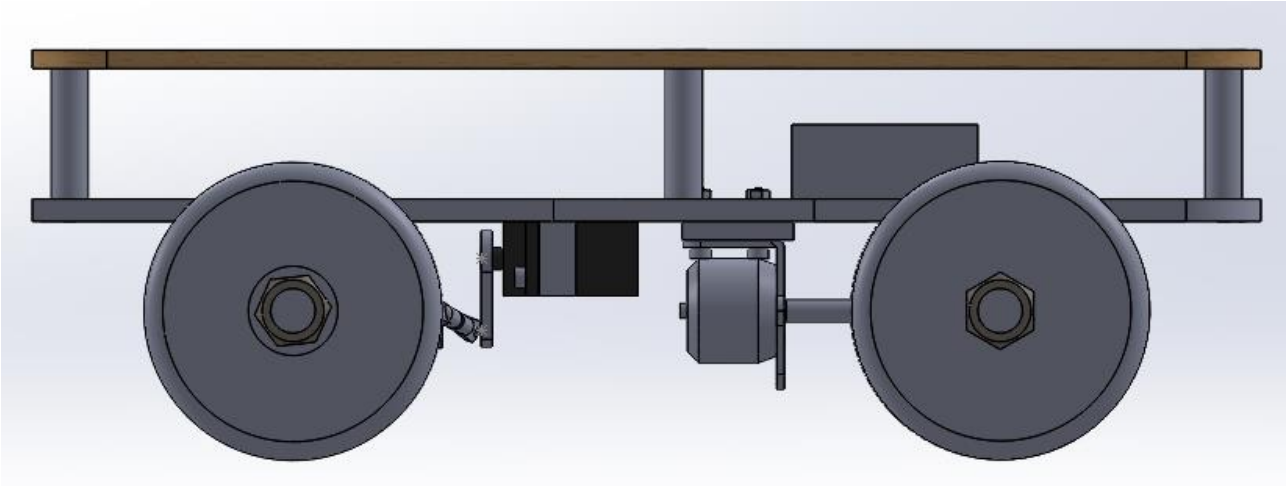


Figure 19- Re-assembled design

3.2 Scott Baker

There were three main areas which I covered for the project, designing the drivetrain, drive component selection and assembly.

During the initial phases of the designing, I drew on previous experience from racing remote-controlled (RC) cars, to influence my designs. The three concepts I put forward differed mostly in the drivetrain. There was an all-wheel drive system, which had a motor on each wheel, and two rear-wheel drive systems, one of which had a motor on each rear wheel and one with a motor attached to a central differential. After discussing the pros and cons of each method, the differential and single motor was chosen due to the cost saving from using one control unit and motor and also less programming because the differential will passively control the speeds of the inside and outside wheels during a turn, rather than having to program each motor to turn at a different speed depending on the angle of turn.

Once the initial concepts had been discussed, we moved on to producing a model in Solidworks, I met with Daniel who was leading this task to give exact dimensions and further explain how the drivetrain linked together. This was to be updated when the differential arrived. The first detailed design of the drivetrain consisted of the motor connected to the differential via a shaft. The output of the differential then went into a drive shaft and hub arrangement before connecting to the wheel. This design consisted of three components and is shown in figure 20 as an assembled view and figure 21 as individual parts.

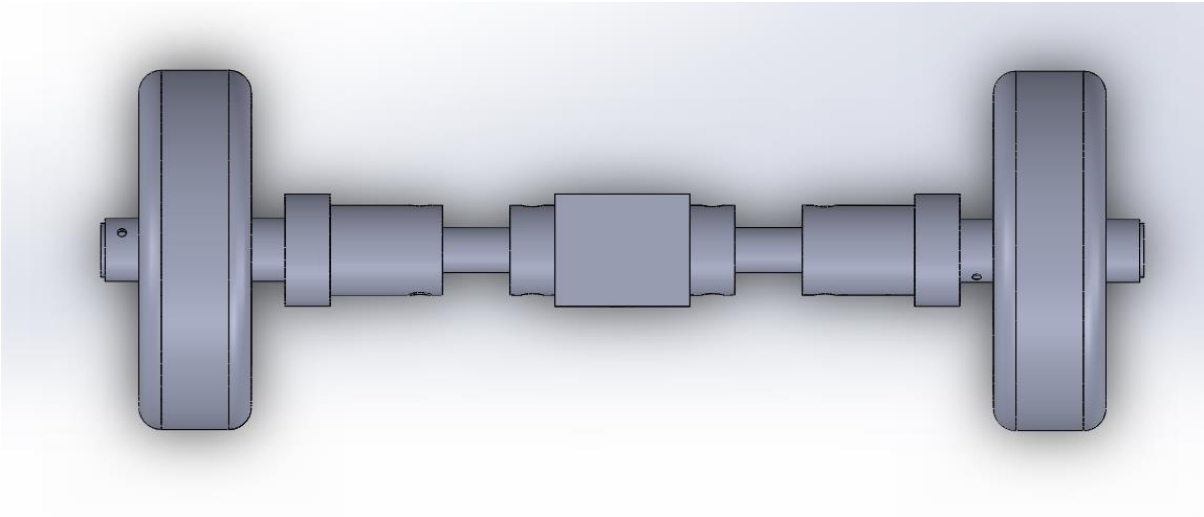


Figure 20 - Assembled Rear Axle

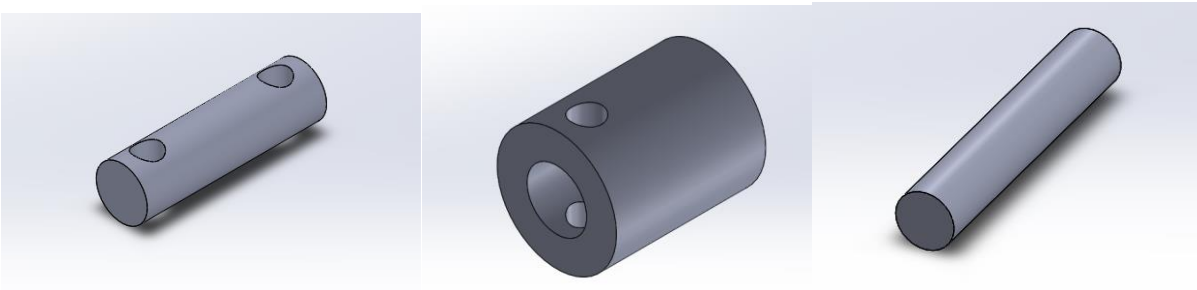


Figure 21 - Individual Components, Driveshaft, Hub and Wheel Rod respectively

The differential was the first component to be ordered as the dimensions of the drive shafts were dependent on the size of the differential. The decision to purchase a differential was made because they are cheap to buy despite being very intricate by design, having a lot of small components meaning that it would be infeasible to produce one within the time of the project without risking serious delays.

The rear wheels had to link to the drive shaft to make them rotate with it and provide the motion, this was initially designed by using a pin and grub screw arrangement, like the differential end of the drive shaft. This was later simplified to using two nylock nuts, one either side of the wheel and tightening them oppositely with two spanners. This design meant that the system was simpler, requiring less cutting and drilling and was also much less likely to fail.

This design was updated following the first assessed meeting as the initial hub and shaft arrangement could be simplified to just a driveshaft which ran through a bearing housing as shown in figure 22 and the drive shaft shown in figure 23.

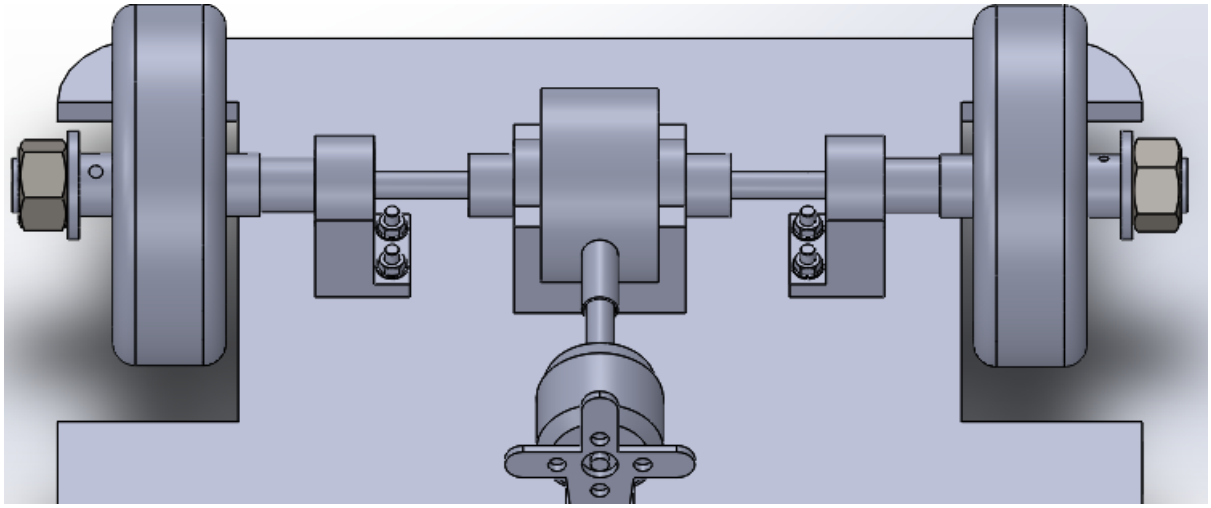


Figure 22 - Assembled Rear Axle Inc. Differential and Drive Shaft

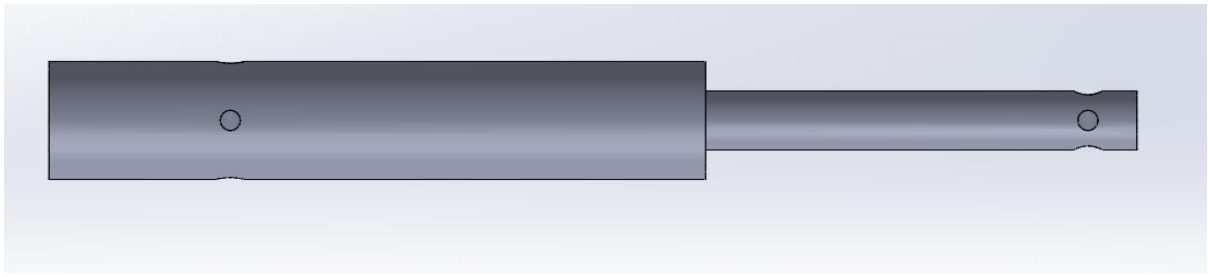


Figure 23 - Drive Shaft

The driveshaft goes through the bearing housing, which holds two bearings. The decision to use two bearings was made because if one is used and the driveshaft is not exactly straight, this introduces a lateral load on the bearing which would cause it to fail much sooner. Using two bearings forces the driveshaft to be straight at all times and significantly decreases the lateral load and thus increases the life of the bearings. This allowed for simpler manufacture and also meant that less materials were required as the same diameter was used for the front axle. The previous design had been influenced by an RC car which used suspension and required the driveshaft be able to move with the hub. Suspension was ruled out of the project due to the complexity of the design and the cost, which could run the risk of the project going over budget.

The process of deciding the battery, motor and speed controller was very closely linked as the requirements for each component depended partly on the others.

The motor chosen for the vehicle had to be able to drive the vehicle at 24kph, as set out in the specification. As the wheels of the vehicle had already been chosen and the differential gear ratio known, it was calculated that the motor would need to be capable of 8018rpm to achieve the speed of 24kph, assuming no losses. Also, due to the size of the vehicle and the space available between the servo and differential, a small motor was required. For this application, a brushless motor was chosen over a brushed motor. There were several reasons for this, they offer higher output speed

and torque for the same voltage, are roughly the same price and also last longer due to having no wearing components. There were two choices available for the speed controller, using an electronic speed controller (ESC) or a motor driver board. An ESC was chosen over the driver board because it offers a plug-and-play solution and is cheaper. The chosen motor had a rating of 1380rpm/V and as such required at least a ~6V battery to achieve the desired RPM of 8018.

The servo chosen for the vehicle was a high torque but relatively inexpensive which easily fits the requirements as there is almost no loading on the servo if the steering mechanism has no resistance (as designed). To better estimate the battery life of the vehicle, the current draw of the servo was measured. Under no load, the servo draws 0.11A which rises to 0.5A when first rotating and then reduces quickly to 0.3A mid-rotation. For the battery calculations an assumed average and constant draw of 0.2A is used.

Both the servo and the ESC require a pulse width modulated signal input which after liaising with the control engineers, was confirmed to be a function of the control boards the vehicle uses, and they would both be compatible.

Again drawing on knowledge from racing RC cars, the standard voltage for batteries is 7.2V for nickel based batteries and 7.4V for lithium based batteries. These standards were chosen as they are manufactured for a competitive market and as such are relatively cheap. Nickel batteries tend to fade through their life and also do not hold their charge as well as lithium. However, lithium batteries are more expensive and require a more expensive charger, both of these were able to be purchased well within the budget. Before purchasing the components some calculations were carried out to ensure their suitability.

Parameters:

Battery capacity = 5Ah
Motor current draw @max rpm = 9A
Motor current draw @min rpm = 5A
Speed required = 24km/h
Wheel diameter = 66mm (66x10^-3m)
Wheel circumference = 0.207m
Differential gear ratio = 54:13 = 4.15:1
Wheel revs/h @ 24km/h = $24000 \div 0.207 = 115942.03$ (2)
Wheel rpm @ 24km/h = 1932
Motor rpm @ 24km/h = 8018
Max motor rpm = 7.4 x 1380 = 10212
Motor rpm as % of max = 78.5
Motor current draw @ 24km/h = 5+(4x.785)=8.14A (3)
Servo continuous current draw = 0.2A (Measured and averaged)

Calculation:

Battery life @ 24km/h = Battery capacity/Current draw@24km/h = $\frac{5}{(8.14+.2)} = 36mins$ (4)

Once all of the chosen components had had their suitability checked through calculations, the orders were placed. The battery life of thirty-six minutes should provide ample time to record a GPS route using the gaming controller and re-drive it using the autonomous function.

During the manufacturing of the vehicle, the were many delays in the workshop producing the components. These were unavoidable delays due to the sheer workload of second year design compared to the number of technicians but with the rest the mechanical sub-group, we went into the workshop to do some of the manufacturing which did not require the use of machinery we were unauthorised to use. This involved drilling and countersinking all the holes in the chassis and cutting components to size. Also whilst in the workshop I liaised with the technicians about the submitted drawings, going through what each part was for and if there was any potential for simplifying the design or the manufacture process. For example, the bearing housing for the rear drive shafts had a rounded top which followed the curvature of the bearings. While this would look better aesthetically, a square design requires less manufacturing and thus saves valuable time.

When all of the components had been produced by the workshop I attended the workshop alone to assemble the vehicle due to the delays forcing this into the Easter holidays. This task was not as simple as first thought as lots of components didn't quite fit and adjustments had to be made. Several issues occurred during assembly such as the steering mechanism coming loose when operated repeatedly. To combat this, I added in spring washers which load the nuts to help stop them coming undone. However, this was not enough and a second 'locking' nut was added to all four bolts in the Ackermann mechanism. The fully assembled chassis with all drive and steering components is shown in figure 24.

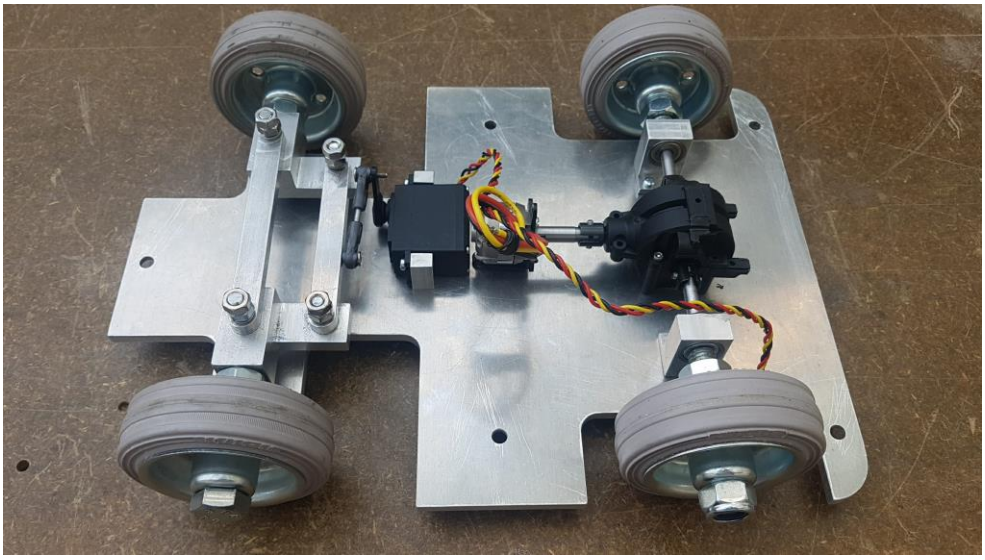


Figure 24 - Assembled Chassis

During testing, the grub screws securing the motor to the driveshaft and the two rear drive shafts to the differential were loosening. This meant that the shafts were not spinning and the vehicle

would not move. To remedy this, I added flat-spots on the motor shaft and the pins securing the drive shaft into the differential as shown in Figure 25 25.

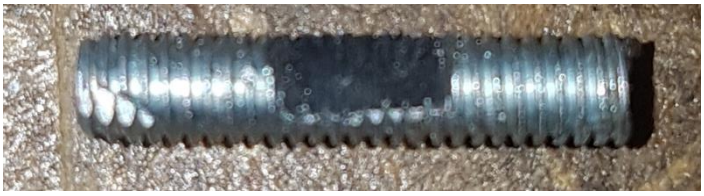


Figure 25 - Drive Shaft Pin with Flat Spot

After assembly, whilst testing the steering mechanism, a servo was burnt out. The engineers present at the time concluded that this was because the steering mechanism was too tight and the servo was having to work too hard to turn the wheels and burnt out. However, I noticed that if the steering turned too far, the servo horn would pass a point of no return and become stuck under the steering joints. Whilst loosening all of the bolts had fixed this, the play in the system meant that this was an infeasible fix and the steering angles were no longer corresponding accurately to servo position. Instead, an electrical limit should be introduced to limit the servo rotation to stop the servo becoming stuck and fix the issue.

3.3 Mohamed Khaled

In this project, the team was split into two sub-teams one tasked with working on the mechanical makeup of the vehicle. This involved the vehicles design, sourcing and manufacture. I was one of the mechanical engineer tasked to work on the mechanical structure of the vehicle. When Catapult approached us, they requested a miniature drive-by-wire vehicle that was at 1:6 the specification of the original pod. The first task was then to calculate the dimensions that were required for the smaller pod. After dimensions were calculated the values achieved are as follows:

Table 4- Pod dimensions

Size (Dimensions)	
Width	24cm
Height	28.5cm
Length	33cm
Wheelbase	19.2cm
Track	19.6cm
Wheel Diameter	8cm

The company also requested that the design was manufactured in such a way that it also be as comparable to the original pod as possible. From a mechanical standpoint, this meant that the vehicles wheelbase and dimensions should be comparable if upscaled. This also meant that the design should be as aesthetically comparable to that of the original pod as possible. However, the

most important thing was that the vehicle was mechanically comparable to the original pod such that the code can be tested on the vehicle in a way that it simulates the larger pod and comparisons can later be made. The cosmetics of the design were left to be dealt with at a later date.

Initially, my role was to design the chassis of the vehicle, parts such as the wheels and mounting rods and some electronic components that will be used such as the servo and Arduino board. After completing the calculations, a chassis design was selected that resembled a similar design to most RC vehicles at that size. In approaching this design, it was clear that the vehicle would contain several electronic components and mechanical components used for the drive train for example. Therefore, the design had to be one that contained a large surface area whilst being strong enough to carry the weight of these components. The chassis design also had to consider the Ackermann steering mechanism as the wheels required a large amount of space for movement. Therefore, the wheel diameter was calculated beforehand, and the appropriate wheels were selected. The vehicles chassis was designed on SolidWorks according to the dimensions calculated. In addition to this lower chassis plate it seemed beneficial to add an upper part to the chassis. This was done to add a larger surface area to the vehicle for the attachment of the electronic components and sensors. This was necessary as it appeared the mechanical components and mechanisms would take up a large amount of space on the lower part of the chassis.

The material that was chosen for the lower part of the chassis was a thick aluminium plate, 6mm thick to be exact. This was chosen due to its many advantages such as strength, ductility, low density and light weight. The material selected for the upper part of the chassis was wood. This was designed to be attached at a higher level to the lower part of the chassis. This wooden plate would be able to use the full extent of the vehicles dimensions as no holes had to be cut out for wheels or any mechanisms. There were also some cylindrical aluminium pieces that were designed for the attachment of this wooden plate at an elevated level, at approximately 60 mm. In addition to these parts some electronic components were also designed on SolidWorks to provide an illustration and visual representation of the components sizes and dimensioning. These were later attached to the vehicle on SolidWorks, before manufacturing, to provide an illustration of what the components designed would look like as a whole vehicle. This was done in case any changes were needed before beginning the manufacturing process. Both bottom and top parts of the chassis can be seen in the figures below.

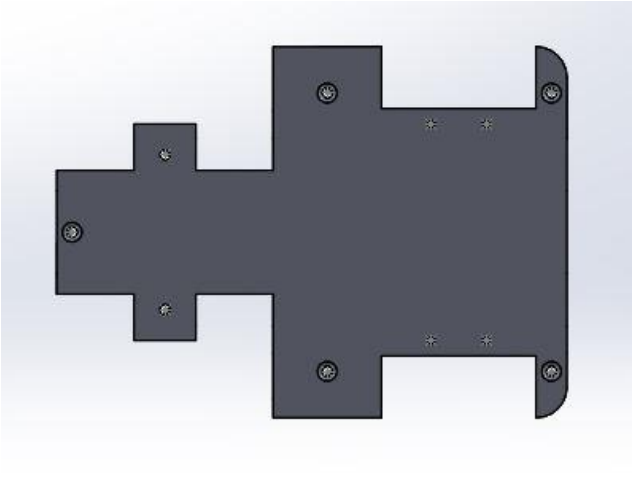


Figure 26- Aluminium plate, Lower part of chassis

Figure 27- Wooden plate, top part of chassis

After these parts were designed and were ready for the manufacturing phase they had to be submitted to the mechanical workshop. However, for these to be submitted for manufacturing, however, they had to be converted from 3D parts into 2D mechanical drawings. This was done using the SolidWorks application, with the appropriate dimensioning provided to be read by the workshop. After these drawings were ready, they had to be submitted into the mechanical workshop for manufacturing. It was essential for these to be submitted as soon as possible so that they are manufactured quickly. However, for these parts to be manufactured the appropriate materials had to purchased ahead of time such that they were available before manufacturing was to begin. It was therefore part of my role to source these materials. This involved contacting companies such as Aluminium Leicester, on several occasions, requesting quotes for some aluminium blocks and sheets that were needed for some parts and filling out the appropriate purchase forms. My role also involved getting the component purchase list forms signed and the work request forms completed and signed so that manufacturing was able to proceed. These had to be signed as quickly as possible as any delay could result in delayed components or a delayed manufacturing which could mitigate or hinder progress in accordance to the set-out Gantt chart.

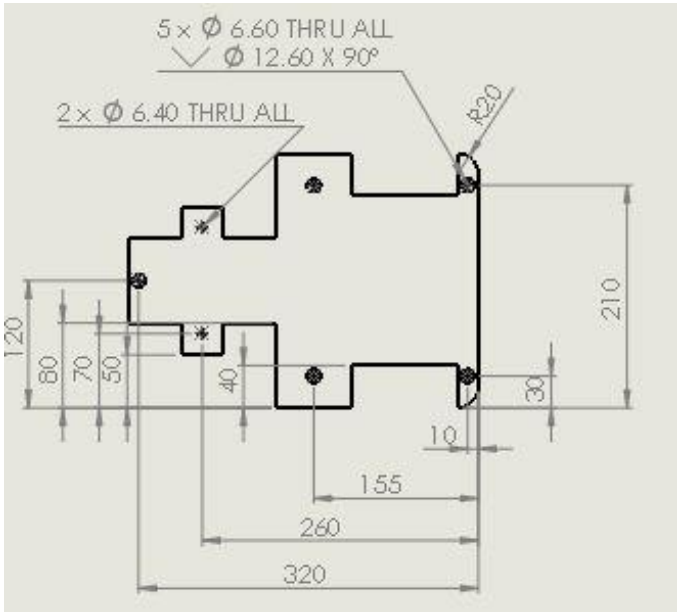


Figure 28- Drawing of chassis bottom part

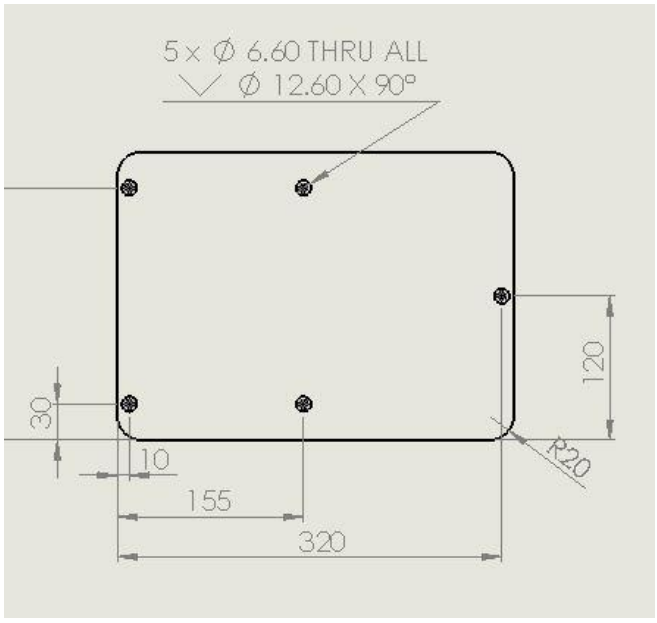


Figure 29- Drawing of chassis top part

After these parts were completely designed, with all adjustments and improvements made, the chassis aluminium plate was updated to include some holes for the mechanical mechanisms that were to be drilled onto the plate, they were submitted into the workshop for manufacturing. These were submitted alongside a signed work request form and the parts were discussed in detail, with the technicians, to explain what was required of the parts and how we plan on using the parts, respectively. These were always submitted ahead of time to avoid any delays. However, there were some delays around February due to some parts arriving late, namely the differential, this was required for some calculations and critical design changes that had to be made before submission. There were also some delays due to the strikes and the consequent difficulties in attaining a signature, for the purchasing of some components, and work request forms, subsequently delaying manufacturing. The workshop was also requested to prioritise the work of the second-year

engineering students. This meant that there would be further delays to the project. This subsequently affected our project and therefore mitigated and hindered progress. The manufacturing stage was necessary at this point such that the vehicle can be tested appropriately both mechanically and electronically to identify any risks at this stage and mitigate them before project completion.

To combat this, I decided to contact the physics lab with a request to ask if some of manufacturing work may be transferred to their manufacturing work shop to speed up manufacturing time. This involved emailing Mr Andrew Underhill and sending several drawings for evaluation. This request was accepted; however, the completion time that was provided wasn't that much different than that of the Engineering workshop. Therefore, it was decided that, since the drawings were already in the que at the mechanical workshop, it would be better to submit the all the drawings to the Engineering workshop and wait for manufacturing to take place. Though, this would take a couple of weeks for the completion of manufacturing therefore project progress was hindered.

To speed up progress, and to insure the project progress isn't affected that much, the mechanical sub-team decided to go in to the manufacturing workshop for a couple of hours, on several occasions, and work on the manufacturing of the more simpler parts that weren't too complicated and didn't require the help of the technicians or heavy machinery. This involved going in during breaks and working on things such as drilling the holes necessary for the chassis plates, manufacturing the motor mounts and other parts and beginning to assemble the parts that have already been manufactured completely. Although, we tried to manufacture as many of the parts required as possible, there remained some important parts that were too complicated to be manufactured by the mechanical sub-team. Also, as it was at a later stage in the project it wasn't ideal to try and manufacture these complicated components in case anything went wrong, which seemed likely. These parts drawings were therefore submitted to the workshop, alongside the other drawings, for manufacturing by the technicians.

After the previous assessment, it was noticed that the vehicle had a very low clearance, and this had to be addressed. Therefore, the mechanical sub-team got together and decided on what the best course of action would be to increase this clearance and improve the vehicles drive features. One thing the team looked at was changing the size of the motor mounts and trying to move some components on the vehicle, such that the aluminium plate could be raised. This only added a couple of "mm" to the clearance which wasn't enough. The other solution was to flip the aluminium plate such that the components and mechanical mechanisms are attached to the bottom side of the chassis. This allowed for a much greater clearance and gave more room for the electrical components. The only change that had to be made was to the size of the cylindrical aluminium parts used to raise the wooden platform. These were cut down to half, from 60mm to about 30mm. Images of this can be seen in the figures found in the final design section below

3.4 Hongfei Chen

At the very beginning phase, I had the task to choose the GPS module and decide micro-control unit structure due to another control team member taking charge of route planning and choice of

sensors. However, after the first meeting the whole control part was simplified so that only GPS module and IMU module were required, leaving other sensors for further optional expansion. Thus, the main emphasize of the control team moved to the node building and compatibility adjustment between the nodes we built and the existing core code. Since there were two nodes required to build with one focusing on the GPS data collecting and the other controlling the vehicle motion, I mainly took charge of the vehicle motion node build.

At the first stage, the initial control idea was to integrate an ultrasonic sensor, three infrared sensors, motor, servo, and a servo for ultrasonic sensor. Therefore, I chose the RoboHAT as the expanding board to mount on the Raspberry Pi3 because of its well-integrated expansion ports for many sensors with standard signal output port. The GPS module I chose was the Adafruit Ultimate GPS module for its prevalence in DIY electronically controlled and because it is highly cost effective.

The first design diagram for the board mapping is shown below in figure 30. However, as the Adafruit GPS module that I chose required the Inter-Integrated Circuit (I2C) communication method with 4 ports to be connected while the available I2C port on Raspberry Pi3 was covered by the RoboHAT and there was no space for the access of the GPS. Although connection through Bluetooth is another option, the Bluetooth would conflict with the GPIO 14 connection port that has already been integrated in the input area. Thus, I had the expanding board changed and delivered another micro-control unit structure.

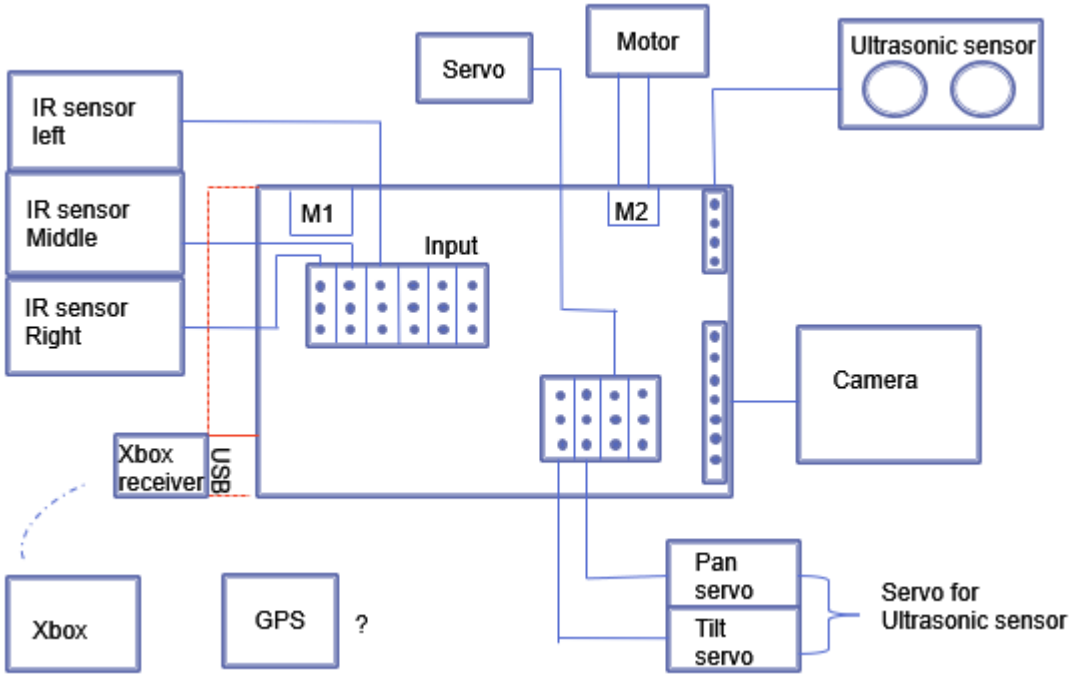


Figure 30- Initial control microunit structure

The idea of the micro-control unit structure second version arose because of sufficient ports on the Raspberry Pi itself. Abandoning any auxiliary extending board mounted on the Raspberry Pi3, the ultrasonic sensor and infrared sensors were connected to the port on the Pi directly while the GPS connected to the Pi through the USB port. I chose the independent speed control unit with the control core on the Arduino Uno board. There was a different motor driver board L298N expanded

to control the motor. At the same time, another servo accompanied by this motor driver board were connected to the Arduino board which worked as a complete node to function the speed control. To connect the Arduino and Pi3, the USB connection was adopted. The wiring design for Pi3 part is shown in figure 31 and the wiring for speed control unit is shown in figure 32.

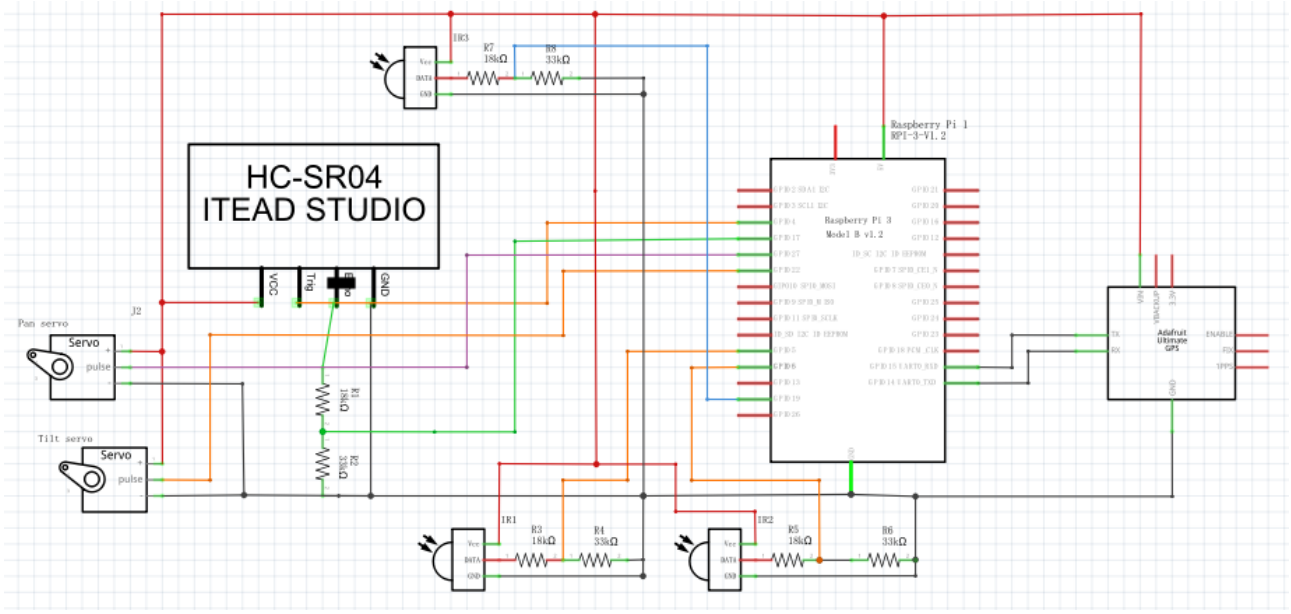


Figure 31- The wiring design for Pi3 part.

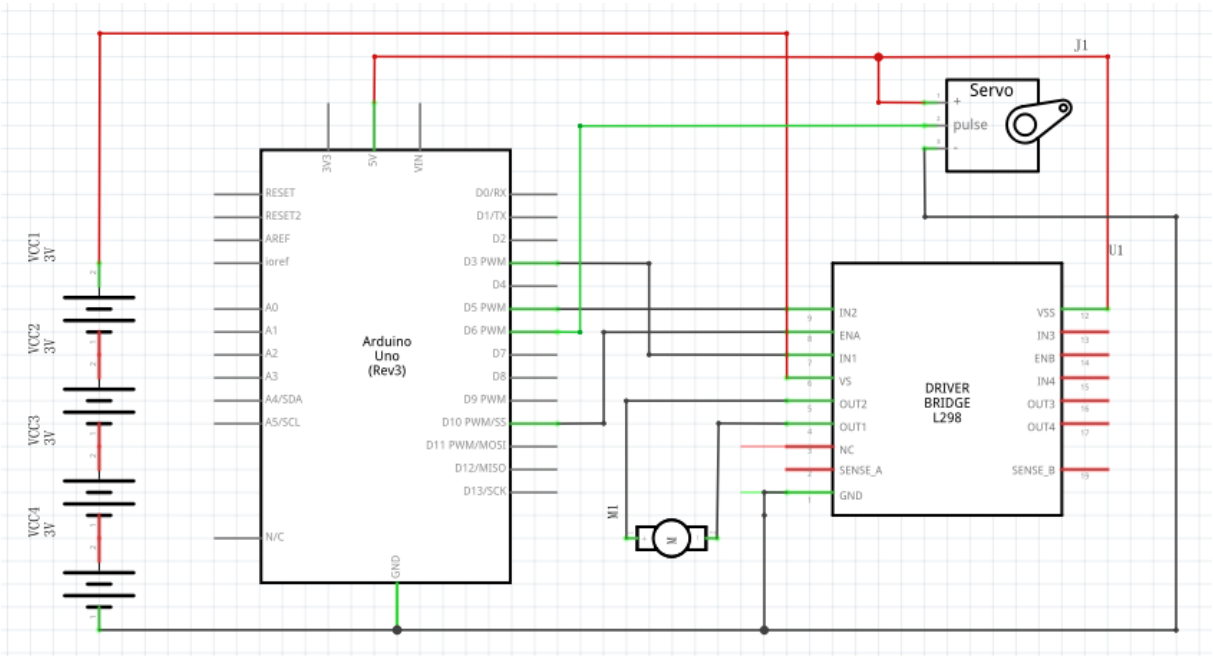


Figure 32- The wiring design for speed control unit.

After the 1st assessed meeting, the GPS pathfinder was confirmed as the principal task while other sensors were cancelled. The rear axle differential design held the different working principle with the motor driving board. Following the mechanical change, the speed control part was changed into a new form where a servo is used to control the direction turning while the Electrical Speed Controller (ESC) is used to control the motor to deliver the speed on the rear wheels. Therefore, the

speed control part concentrated on the Arduino and the wiring is demonstrated in figure 33. As a whole, the 5000 mAH battery provided power for the speed control part. As regards to the software, this is the unit that undertook the action to implement the vehicle motion node. In the end, the components required to be considered for the wiring only consisted of a servo motor, ESC and GPS. The final completed wiring plan is shown section 3.5.

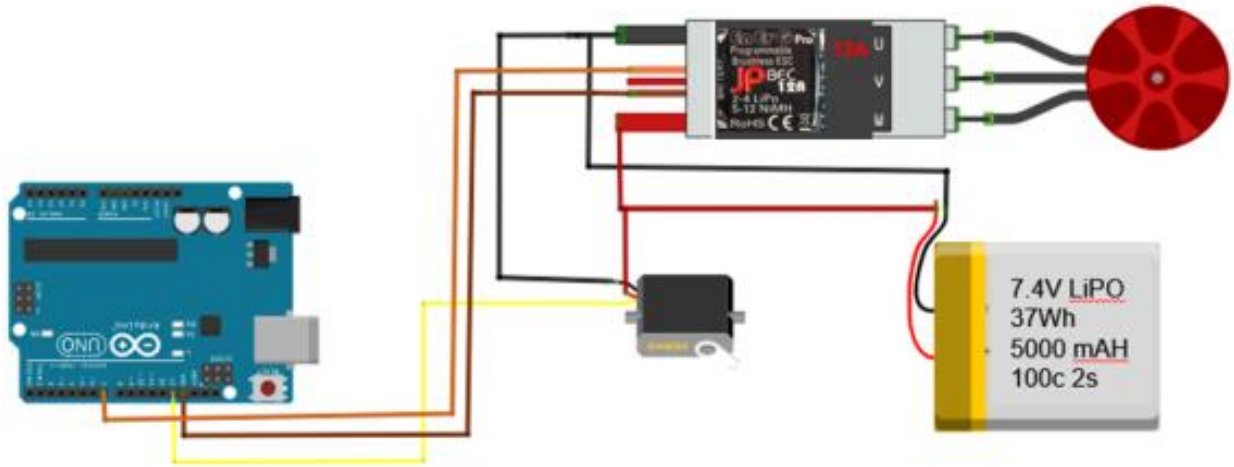


Figure 33- The updated speed control unit wiring.

After the speed control test, the components were finally mapped by me on the upper platform shown in figure 34. Owing to the inversion of the chassis at the final manufacture phase, the spacer layer between the chassis and the upper platform is left empty. But in order to help the wiring go through this layer and plug into the port of components on the upper layer, the upper wood board was handed by me to the workshop to have a 20mm*20mm hole opened. The size of the hole will allow the ESC goes through to the upper layer while the ESC stayed at the spacer layer in the end due to the limited wiring length. The view from the side of the µPod is shown in figure 35 below.

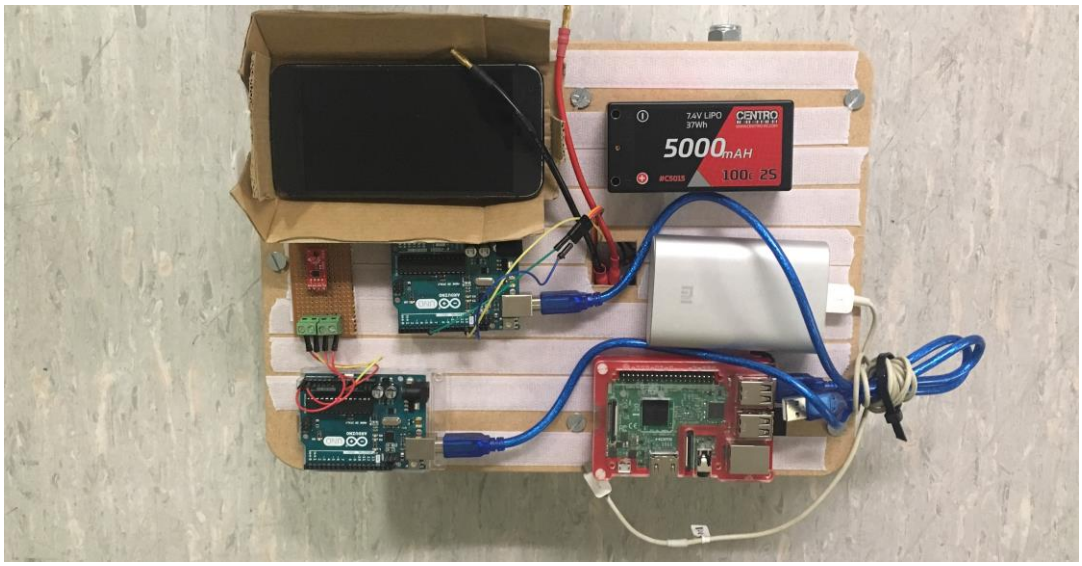


Figure 34- The final upper layer layout.

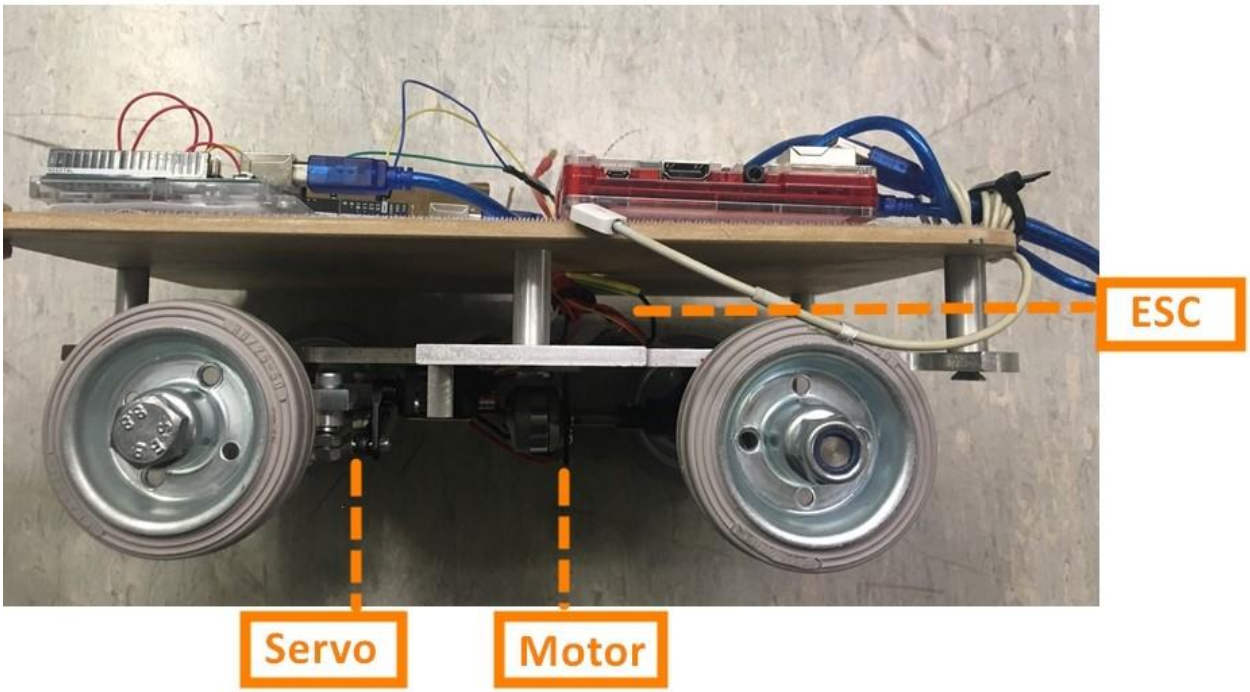


Figure 35- The side view of the µPod.

The vehicle motion node build was another task I took charge of. The function of this node was to receive the speed and steer message from the age checked node and specified them to control the motion of the µPod. Therefore, building the relative function to bridge the initial data and the output to the ESC and servo was the critical mission of this node build.

The idea for the servo function was, as the speed was sent as the form of the curvature K, the first step was to transfer the K to a certain angle of the vehicle body turning angle θ_i. The curvature K is the rate of the heading change and it is what heading angle (radians by convention, rather than degrees) that the vehicle should change by for every metre it moves forward.

After this step, the function between the car body turning angle θ_i and the servo turning angle θ_{servo} can be developed through the basic test. The theory to connect the K and car turning angle θ_i was demonstrated in the figure 36 below. In detail, the first equation was deduced by the mechanical design of the µPod while the K = 1/R came from the definition of the curvature. Therefore, the connection between the θ_i and the K was built, leading to the test between servo turning and real µPod body turning. Owing to the Ackermann steering system design, there was a certain range of the servo turning angle rather than the range marked on the instruction of the servo. To find the appropriate turning range, the µPod was placed upside down to test for better observation to see if the ball joint attaching the steering arm to the servo motor touched the chassis. Although this process could be estimated by the mechanical design, the actual assembled vehicle had added limitations, requiring me to directly implement the test through observation. The test code came from the Arduino library, which was called “Sweep” that swept the shaft of a RC servo motor from side to side across 180 degrees. With the limits changed from 0° to 180°, the ideal turning angle ranged from 48° to 150°.

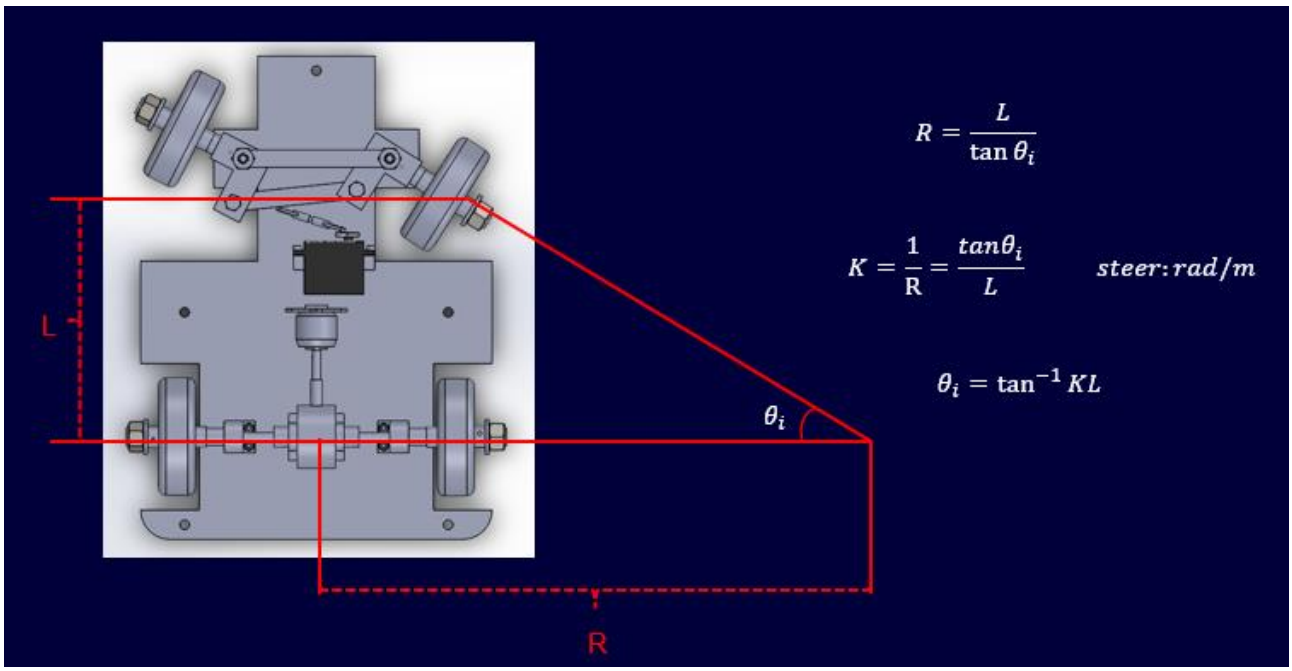


Figure 36- Steering control principles

The K value was set to range from -5 to 5 (the plus-minus sign only represented the turning direction), accordingly

$$\theta_i = \tan^{-1}(\pm 0.5\, rad/m \times 0.19\, m) \times \frac{180}{\pi} = \pm 5.43° \quad (5)$$

Thus, the clockwise turning limits for the servo is 48° which corresponds to the $\theta_i = -5.43°$. Following the same principle, the anticlockwise turning limits located at the servo turning angle 150° while it corresponded the $\theta_i = 5.43°$. In addition, the straight direction of the car was measured between 99° to 103°. These data outlined a linear function and was shown in figure 37. The core representation was written as a statement: angle = (atan(cmd_msg.podDemand.steer * 0.19)* 180/PI)*9.392+99 in the vehicle motion to upload to the Arduino.

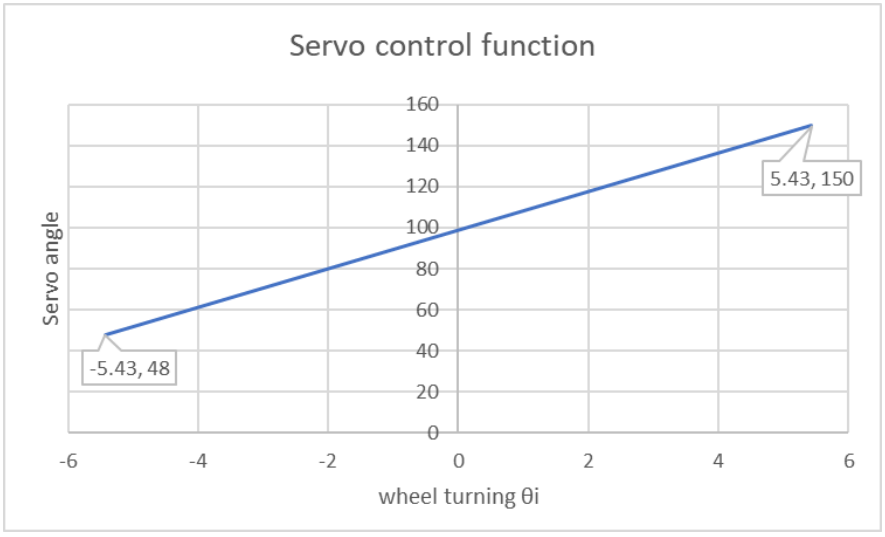


Figure 37- Servo control function

Through the test, there was an optimization for the algorithm. During the first stage, the process was to get the required K and then write directly the corresponding position into the servo, bringing sudden and violent turning. This consideration partly came from the first servo burning accident that is described and analysed in chapter 4.2. Considering such operation would cause the mechanical wear and even lead to more serious consequence, a "for" loop was added to smooth the servo motion under the premise of minimum delay assurance.

The function to develop the motor speed was relatively easy compared with the servo. As the motor was controlled by the ESC which theoretically possessed the same control method as the servo. That being the build-in function in servo library to tell the servo which position it should turn to. Therefore, the idea to deliver the speed control function is to correspond the relative position range of ESC to the required Round(s) Per Minute (RPM).

The diameter of the wheels was given that $d_{wheel} = 8\,cm = 0.08\,m$, and deduced from the definition of the RPM, the speed equals to the product of the perimeter of the wheel and the RPM. Thus, the speed deduction function is

$$\frac{\pi d_{wheel} \times RPM}{60\ s} = V\ m/s \qquad (6)$$

Given that the speed V was the information collected directly form the pod_demand message, the function of required RPM was reformed into

$$\text{RPM} = \frac{60V}{0.08 \times \pi} \qquad (7)$$

The gear ratio from the differential structure also should be multiplied to this original RPM, in detail, bridging the real relation between wheel speed and the motor rotation. The final required RMP function was

$$M_{RPM = \frac{60V}{4.15 \times 0.08 \times \pi}}. \qquad (8)$$

The mapping relations between the position to write to the ESC and the motor RMP is: $180° \rightarrow \text{RMP} = 10212, 60° \rightarrow \text{RMP} = 0.$ This simple linear relation was then written as

$$\text{Angle} = \frac{M_{RPM} \times 15}{851} = \frac{60 \times V \times 15}{4.15 \times 0.08 \times \pi \times 851} = \frac{3735 \times V}{0.08 \times \pi \times 851} \qquad (9)$$

The function representation was written as a statement in the code that: angle1= 24.96*V/(PI*0.08*4.15)+60. However, all these calculations were based on the idling test. The real motion was not as accurate due to the heavy load and limited power supply. These will be discussed by the control team in the test, discussion and further work chapters.

The completed vehicle motion node is included in the Appendix 1 and related explanation follows behind each sentence as a command.

3.5 Jingyu Chen

In the team, I am doing the job as an electronic and software engineer who is responsible for the hardware testing and the establishment of the control system.

Here is the list of the evidence of my contribution and the details will be explained step by step.

- Test the vehicle with Robohat (**shown in the warming-up test**)
- Build the ROS platform on Raspberry Pi
- Analyse the packages provided by the industrial company and revise a plan for building the software codes. (**node specification is shown in the appendix**)
- Test the hardware GPS and IMU and write the related codes to integrate the data into ROS (**part of IMU code is shown in the appendix**)
- Help with the code building of vehicle motion
- Design and draw the layout of the electronic components on the vehicle
- Implement the test of recording the map. (**shown in the further test**)
- Implement the final test of drive by wire and autonomous mode. (**shown in the further test**)

3.5.1 Product analysis

For this project, our duty is to develop a smaller 1:6 scale autonomous driving vehicle, which is compatible to the interface used in the original pod. We are provided with the original software package which contains many codes and documents. In my point of view, this project development is like a black-box test.

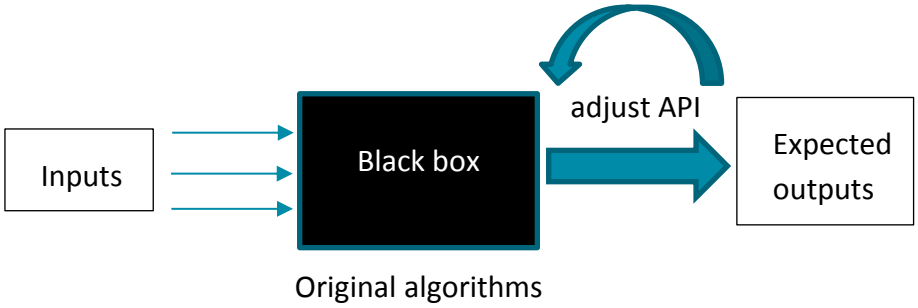


Figure 38- Black box test

We need to feed the inputs and use APIs to achieve each function and adjust the parameters based on the performance of the vehicle motion. Finally, the goal is to fit well with the original algorithms and achieve the function of autonomous driving.

After some researching, I concluded some key points of the construction of the control system.

- Platform construction
- Hardware testing
- Software testing and simulation

The main task is the calibration of input hardware and the control of actuators. In the meanwhile, the platform for further development is also required to be built as soon as possible.

3.5.2 Software and hardware revision

According to the project overview (provided by industrial partner), the structure of their software nodes is shown below,

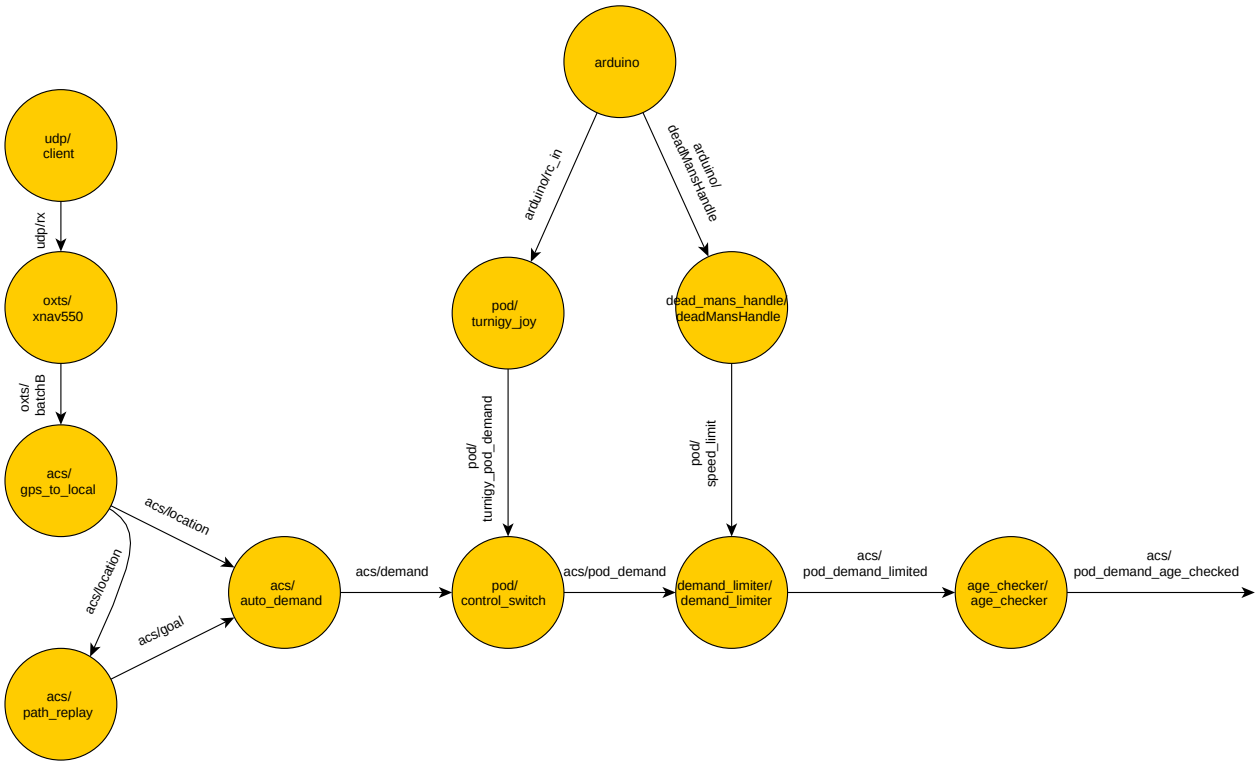


Figure 39- Original structure

From the structure, we can see they are using the following devices.

Figure 40- Original components

- OXTS GPS: receive GPS signal and provide accurate location and heading angle.
- Deadman switch: receive handle position from Arduino and set a speed limit
- Can: an USB to CAN converter is connected to the ACS ECU to provide access to the vehicle's CAN bus.

However, the project budget is not large enough to afford them. Considering the small scale of our project, my solution is to replace the high-price device with some cheap components which is commonly used for STEM activities. Firstly, we choose Raspberry Pi 3 as the main control unit. The reason is that it is suitable for the development of a mobile robot and is compatible with ROS.

Arduino, which has many Pulse Width Modulation (PWM) ports and can easily control motors and servos, is chosen as the auxiliary control unit. Adfruit GPS and Sparkfun IMU (inertial measurement unit) is also considered in our design.

After many discussions with teammates, I made the final design of software nodes and drew the diagram. This update includes deleting some redundant functions and integrating available nodes.

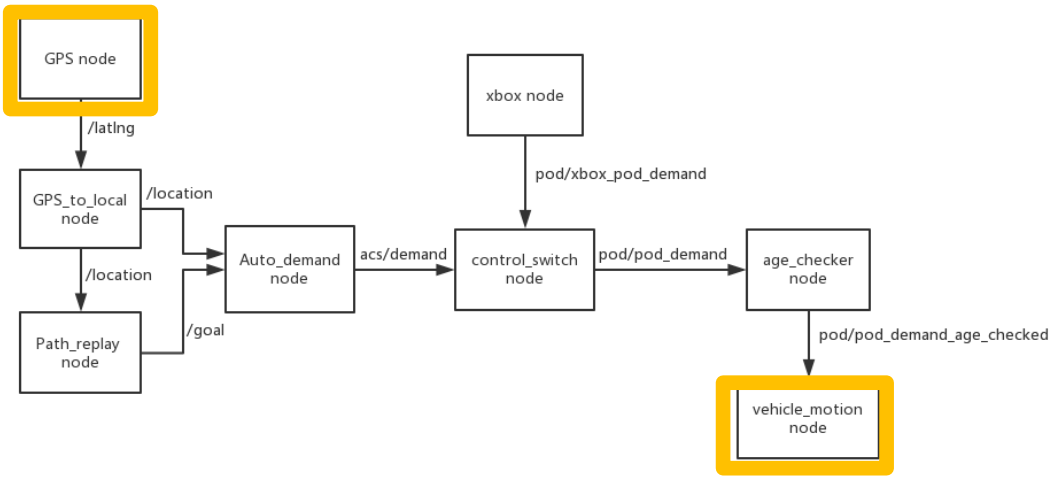


Figure 41- ROS software design

As you can see, these two yellow nodes are the nodes we need to build by ourselves (other nodes are provided). When we launch all the nodes, the vehicle should follow the map (containing many latitudes and longitudes) and move around. The corresponding nodes specification will be shown in the next chapter. Combined with the hardware, the hardware design diagram is shown below.

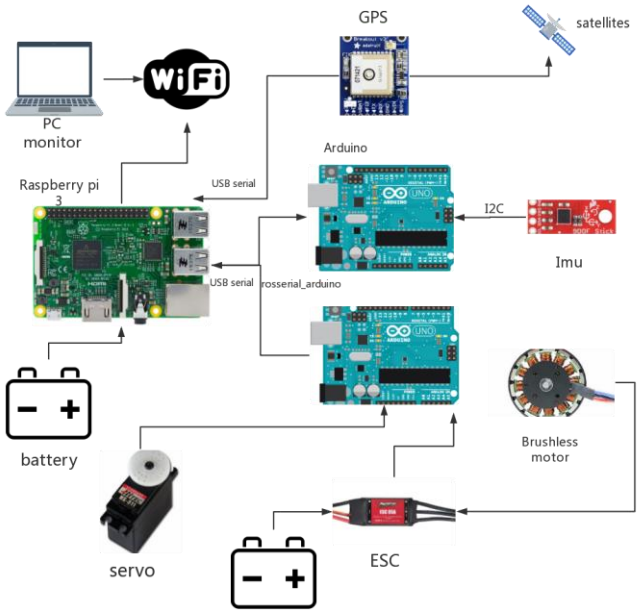


Figure 42- Hardware design

3.5.3 Platform establishment

Our vehicle is based on the ROS framework and we choose Kinectic version of ROS which is compatible with Ubuntu 16.04 on RPI.

Get ROS Indigo Igloo on Ubuntu Linux Get ROS Kinetic Kame on Ubuntu Linux Get ROS Lunar Loggerhead on Ubuntu Linux

Figure 43- Three versions of ROS

After all these steps, a typical workspace is shown below:

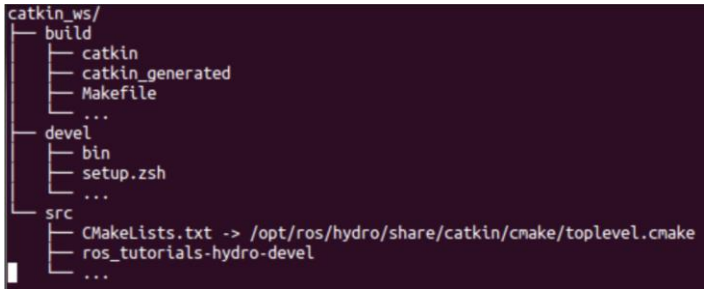


Figure 44- Indigo workspace example

In src folder, we can create or clone packages.

3.5.4 Hardware testing and calibration

- GPS

Adafruit GPS is chosen as the GPS component of the vehicle. Firstly, it has to be connected with control unit RPI. My initial plan is to use UTRA connection and here is the diagram showing the port connections.

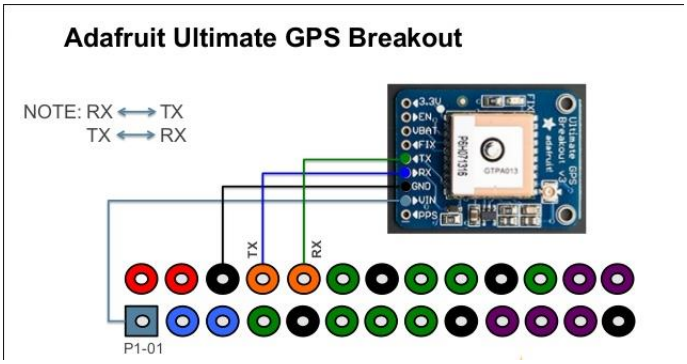


Figure 45- UTRA connection between GPS and RPI

When you finish connecting GPS correctly to the Raspberry Pi (RPI), it is time to do some UART test on RPI to ensure the effective communication between GPS and RPI. There are two UART ports on RPI3. The one is ttyAMA0, the other one is ttyS0. As for RPI3, there are some changes compared with some old versions. The ttyAMA0 is assigned to Bluetooth module and GPIO has the serial port ttyS0. However, ttyS0 has many disadvantages. The most significant one is the lack of independent clock source and it will result in some unstable situations.

Therefore, what we need to do is to shut down the use of BLUETOOTH and assign ttyAMA0 to GPIO (we cannot use Bluetooth and UART in the meanwhile).

However, on RPI3, we met the problem using UART (GPIO 14 & GPIO 15). In other words, GPIO and Bluetooth interfere with each other and GPS data can't be transmitted to RPI correctly. We have to choose another plan using USB to TTL serial cable.

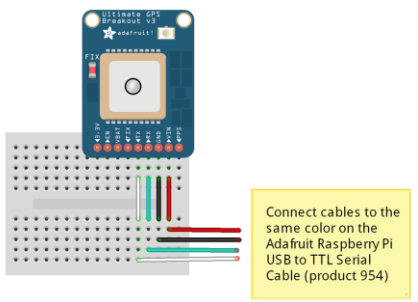


Figure 46- USB to ttl serial connection

After correct connection, make sure can we view the raw NMEA data from /ttyUSB0. Then a package called "gpsd" is used to parse the raw NMEA data. It acts as a layer between application and GPS hardware and also provides a good interface to show the information about longitude and latitude.

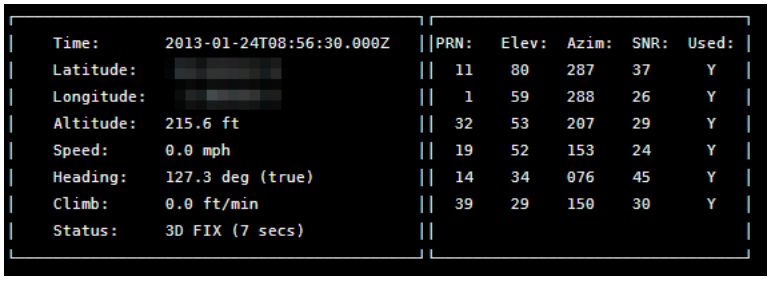


Figure 47- gpsd GUI

However, gpsd client can't achieve the communication with ROS. An approach which is compatible with ROS needs to be found. Finally, we find nmea_navsat_driver package and it can be used to make Adfruit gps communicate with ROS (robot operating system) very well.

Although latitude and longitude can be recorded, we find the collected information is not very accurate. Here is a picture showing the path I recorded near my apartment.

Figure 48- Home path

Obviously, the recorded path seems to drift from the actual path. After many adjustments of parameters, we still couldn't revise this path and decided to buy a more accurate GPS. In order to save the budget, a GPS of mobile phone seems to be a good alternative device.

Android_all_sensors_driver package can build the communication between ROS master and android phone. Through WIFI, it can publish a phone's GPS data in the remote ROS master. And the corresponding diagram for the whole design is also changed.

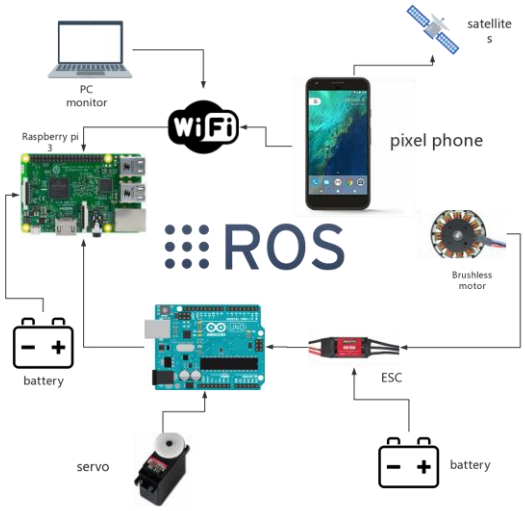


Figure 49- ROS android diagram

Google pixel phone is chosen as the device. For **Android_all_sensors_driver** package**,** it is written in ros java and is packaged to an android app which provided an easy way to use android-based device in ROS.

Firstly, we download the ROS Sensors app in android market.

Then determine IP address for network which ROS master and android phone will share.

```
export ROS_MASTER_URI=http://ip_address:11311/
```

Next start up a roscore

```
roscore
```

Then set up the android phone, enter ROS_MASTER_URI to connect it. Finally, topics containg phone's GPS fix information can show up in RPI.

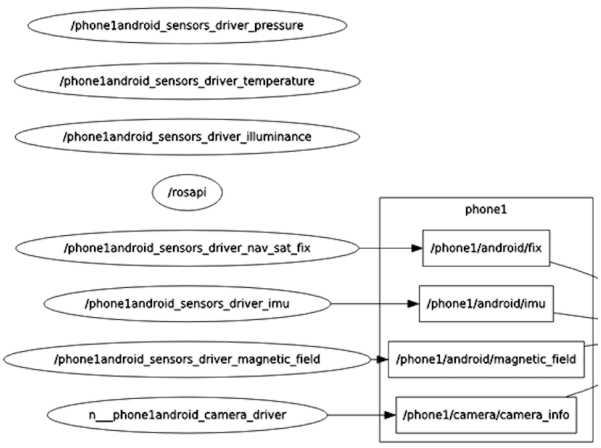


Figure 50- Phone's nodes

As is shown the node graph, it can publish many sensor data of the phone, like pressure, temperature, camera and fix.

In our project, we only use fix data of GPS. Other data can be used for further expanded development.

And we do another outdoor test for GPS. At this time, the path was very close to the one we actually moved. It shows that phone's GPS is more accurate than the previous one. (more details in further tests)

Figure 51- GPS test path 1

- IMU

Actually, we only use magnetometer in IMU to get the axis of magnetic field and use mathematic equations to calculate heading angle. After many attempts, Sparkfun sensor stick IMU is chosen. This IMU utilizes the LSM9DS1 motion-sensing system-in-a-chip. [1]

Here are the components of sensor stick IMU:

- 3-axis magnetometer
- 3-axis accelerometer
- 3-aix gyroscope

I2C communication protocol is used to achieve the data exchange between Arduino.

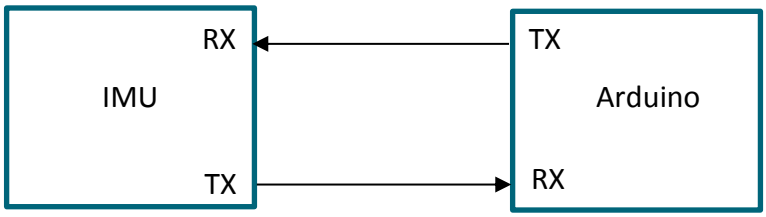


Figure 52- I2C connection

The reason why I2C communication is selected is that I2C can support multi-master system and each master can talk to all slaves connected in the bus. In the meanwhile, rate loss doesn't happen and it has a high communication rate at 100k Hz and 400k Hz. On the contrary, the asynchronous communication of serial port can cause the garbled data and SPI requires many pins. The I2C default address for the magnetometer is 0x1E and the default address for the accelerometer and gyroscope is 0x6B. [2] In our project, only one sensor stick IMU is used and extra address is not required. Therefore, only the default address 0x1E is used.

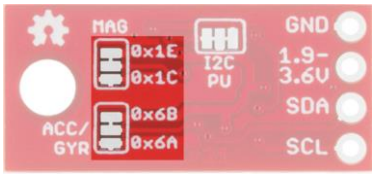


Figure 53- I2C address on IMU

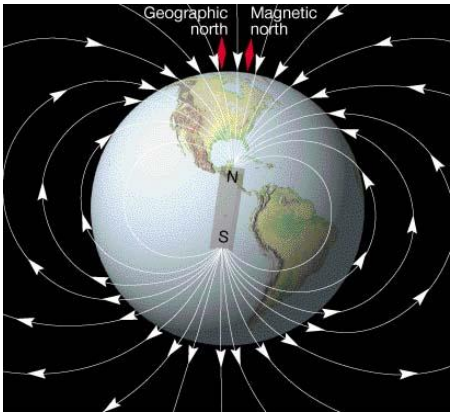
As is shown in our design diagram, IMU is connected to Arduino.

The port connection is shown below:

- VCC → 3.3V
- GND → GND
- SDA → SDA/A4
- SCL → SCL/A5

VCC should be connected to 3.3V rather than 5V. SDA means the data signal and SCL means the clock signal. When connected correctly, open serial monitor in Arduino and should get the following formatted data.

In Arduino code, LSM9DS1 object is created and calcMag() function is used to calculate the magnetic strength.



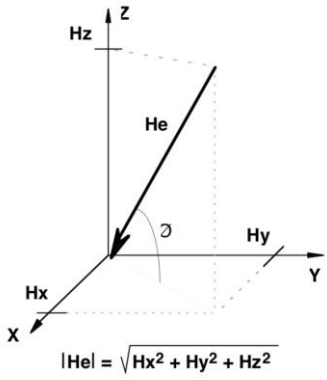


Figure 54- Magnetic field

The magnetic field of earth is very similar to a bar magnet whose field line is pointing from south to North Pole. However, magnetic north is different from the geographic north and the angle between magnetic north and geographic north is called declination angle. When calculating the compass heading, this angle should be considered and varies from different places. In order to get an accurate declination angle, online magnetic field calculator from National Oceanic and Atmospheric Administration (NOAA) is used.

Enter the location of the test field, declination angle can be easily obtained.

Declination

Model Used:	WMM2015
Latitude:	52.61926° N
Longitude:	1.1253° W
Date	**Declination**
2018-04-26	0.88° W ± 0.37° changing by 0.15° E per year

Figure 55- Declination angle

The strength and direction of magnetic field can be divided into three axis, Hx, Hy, Hz. Hx and Hy parallel the surface of the earth and Hz is perpendicular to the ground. A compass heading can be determined from Hx and Hy.

The equations [3] are:

Direction (y>0) = 90 - [arcTAN(x/y)]*180/ Pi

Direction (y<0) = 270 - [arcTAN(x/y)]*180/ Pi

Direction(y=0,x<0)=180.0

Direction (y=0, x>0) = 0.0

To get a true north heading, the declination angle can be added or subtracted.

For the first time, we just normalize the compass heading in the range of 0-360 degree. However, the input heading angle is required to be in the range of [-Pi, Pi] shown in the radians format. We have to make this transform.

When heading > pi, heading = heading -2 * pi

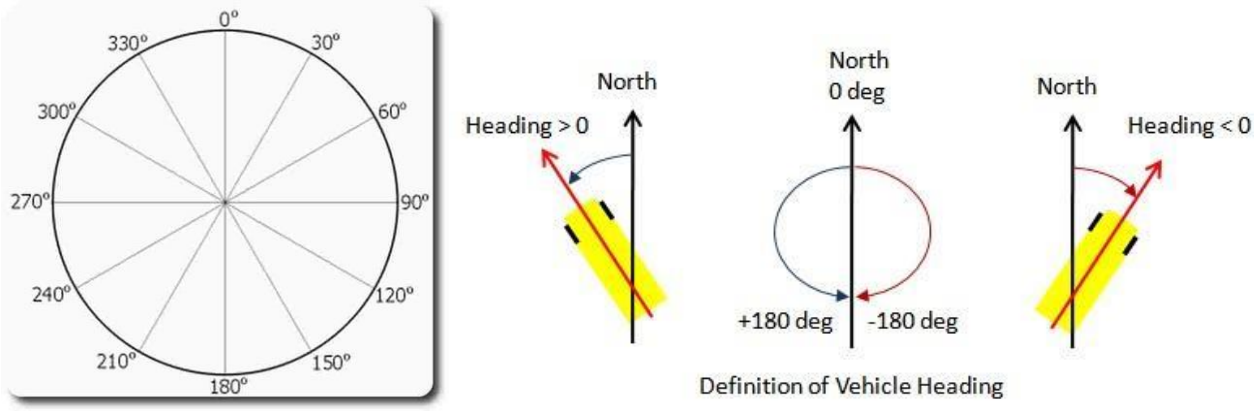


Figure 56- Vehicle heading definition

However, the result may not be very stable and the calibration for compass heading is very essential.

```
for (i=0; i<128; i++)
{
    while (!magAvailable())
        ;
    readMag();
    int16_t magTemp[3] = {0, 0, 0};
    magTemp[0] = mx;
    magTemp[1] = my;
    magTemp[2] = mz;
    for (j = 0; j < 3; j++)
    {
        if (magTemp[j] > magMax[j]) magMax[j] = magTemp[j];
        if (magTemp[j] < magMin[j]) magMin[j] = magTemp[j];
    }
}
for (j = 0; j < 3; j++)
{
    mBiasRaw[j] = (magMax[j] + magMin[j]) / 2;
    mBias[j] = calcMag(mBiasRaw[j]);
    if (loadIn)
        magOffset(j, mBiasRaw[j]);
}
```

Figure 57- Heading code

A function is designed using first in first out (FIFO) to accumulate the magnetic data. Find the maximum value and minimum value of magnetic field. Then average and scale the data and subtract the result of biases to remove the error. This method is the basic step and more sophisticated algorithms can be manipulated to make the heading angle more accurate. For instance, tilt compensation can be executed effectively when the magnetic sensor is not precisely leveled.

For the final step, Arduino needs to be communicated with ROS master by rosserial. Rosserial provides a ROS communication protocol that works over Arduino's UART. [4] It enables Arduino to directly publish and subscribe to topics which is connected to ROS master.

3.5.5 Electronics Layout design

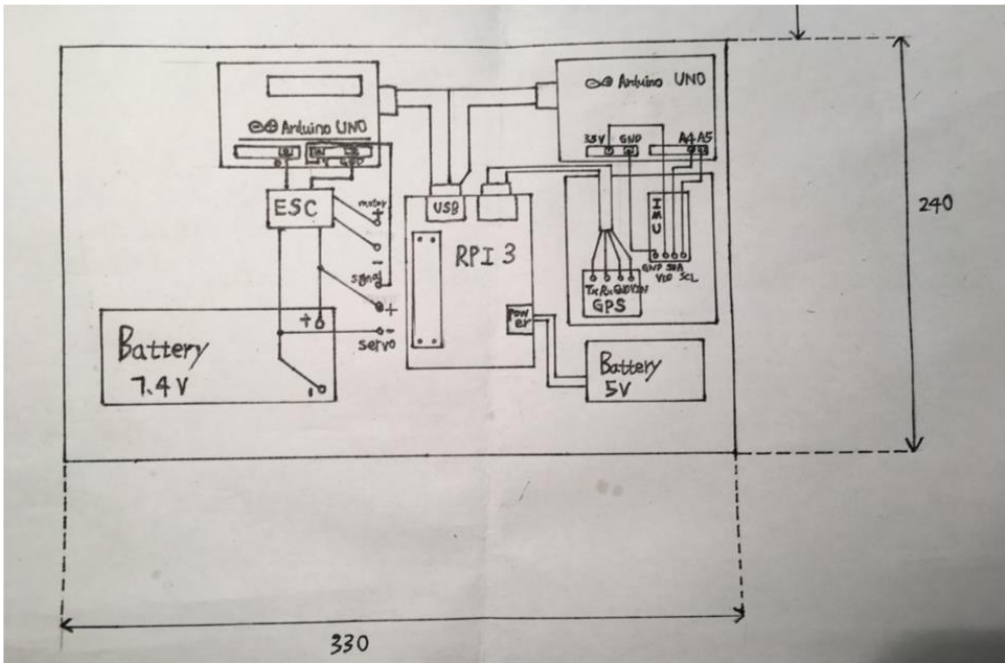


Figure 58- Electronic layout

Based on the mechanical design, the initial layout of electronic components was completed. Actually, some changes have been made after the adjustment of the mechanical structure and the components are distributed in three layers in the new design.

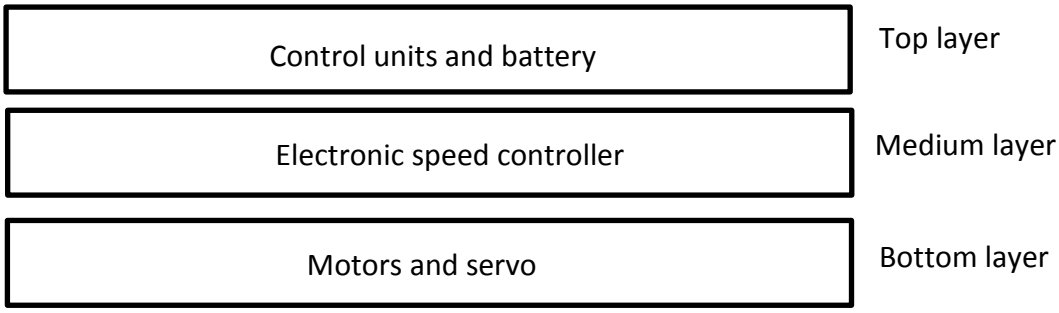


Figure 59- Component layout

In common rules, power units should be separate from the control units. However, in our design, batteries have to be placed in the top board because of the limitation of the mechanical structure. ESC is fixed in the intermediate layer to connect the bottom actuators.

4 Implementation & Testing

4.1 Final Design

The final design of the µPod that was aimed to be manufactured is shown in figure 60. The track and the wheelbase are both 190mm which is a rounded representation of the actual track and wheelbase of the LUTZ Pathfinder Pod. This was done to try and accurately mimic the vehicle that this is based upon in order to ensure the usefulness of the tests that it is built for. Similarly, by purchasing wheels of 80mm diameter an attempt was made to install wheels of 1:6 scale on the µPod. The resulting restrictions applied by the choice of wheelbase, track, and wheels meant that the µPod would inherently have similar chassis dimensions and drive characteristics.

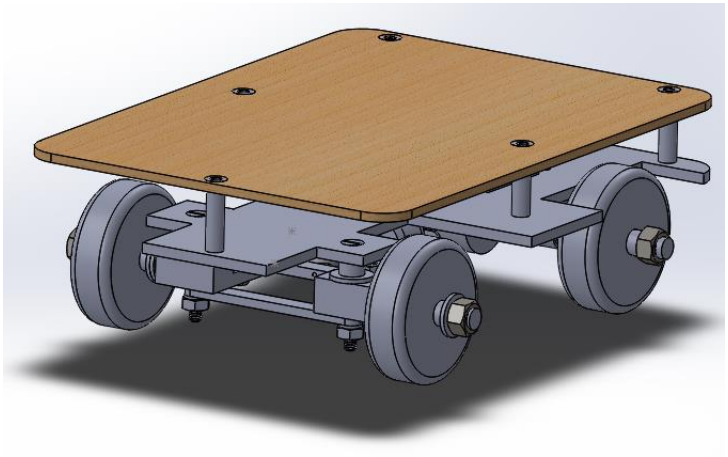


Figure 60- Final design

After analysing the initial concepts it was decided to progress with a two tier chassis system, with the top plate being made of wood. This would allow the mechanical drive components and battery to sit on the lower chassis with the electronic components needed to provide autonomous functionality attached to the wooden chassis. This provides the dual benefit of allowing plenty of room for all components to function and gives space for the addition of further components or sensors if necessary. Even though the µPod was designed with the purpose of functioning on a flat surface due to the low ground clearance of the lower chassis and the expected issues of navigating even the smallest of inclines the decision was made to change the positioning of the mechanical components. With only some slight adjustment the mechanical components are now attached to the underside of the chassis, with only the battery remaining on the top of the chassis bottom plate.

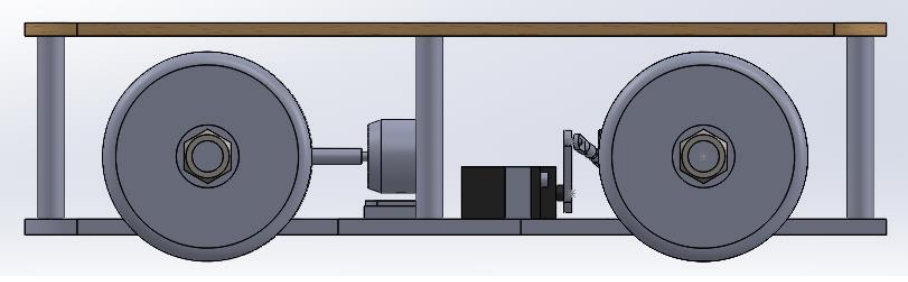


Figure 61- Original design

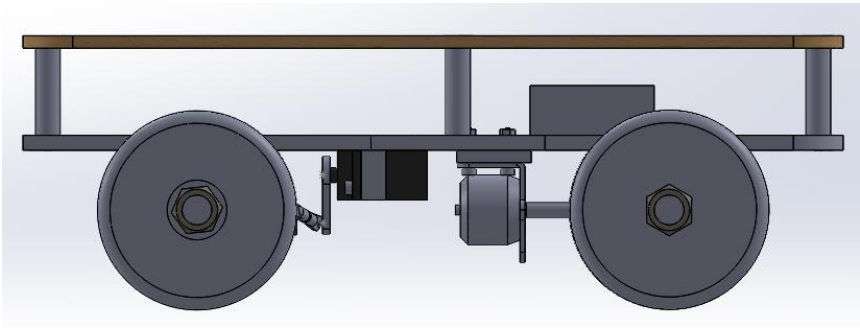


Figure 62- Final design side view

As can be seen clearly in a comparison between figure 61 and 62 the potential issue of the vehicle being unable to navigate even the slightest incline or rough terrain has been removed with more ground clearance being present in the design.

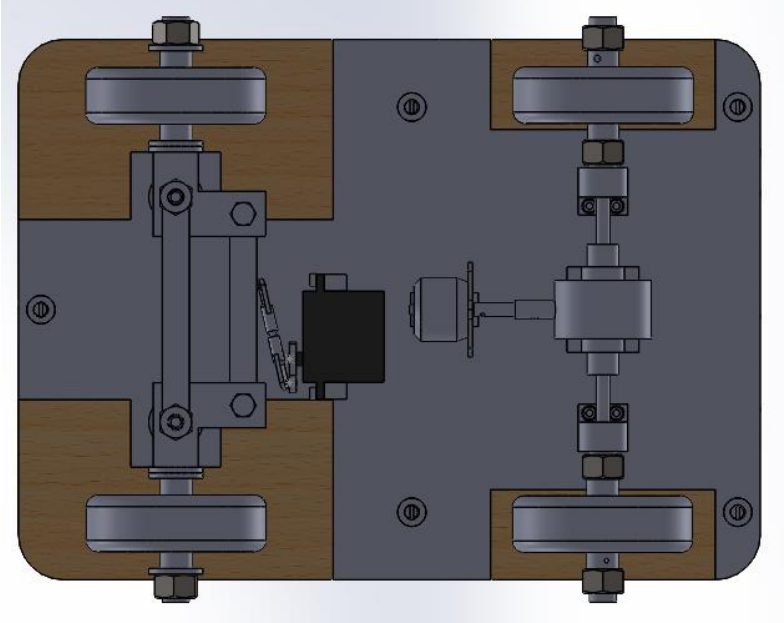


Figure 63- Final design underside view

Figure 63 shows the underside of the vehicle and the mechanical components required to drive the vehicle. As can be seen on the left of the image the steering system is comprised of a tie rod and stationary rod connecting two joints, with the servo attached to the tie rod to generate the steering movement. This allows for Ackermann steering of the vehicle, which was one of the key mechanical deliverables decided upon at the start of the project and is necessary to prevent slip when turning. The front axle wheels are free to rotate about the threaded rod, with M12 washers used to ensure that they rotate freely and maintain the correct track. As detailed previously the servo motor controls the steering and provides a suitable turning radius in each direction. The Ackermann joints are also fitted with a plastic spacer and washer to separate the front axle from the chassis and guarantee that the front and rear axles are at the same height.

The single motor in the centre of the chassis drives the vehicle and is connected to a differential that was purchased rather than designed. Driveshafts are connected to either side of the differential and pass through bearings on each side. The wheels are secured by two locking nuts so that they rotate with the driveshaft in order for motion to be possible. The differential is used so that when

turning the outer drive wheel is rotates faster than the inner drive wheel, which again mimics the operation of a commercial automotive vehicle.

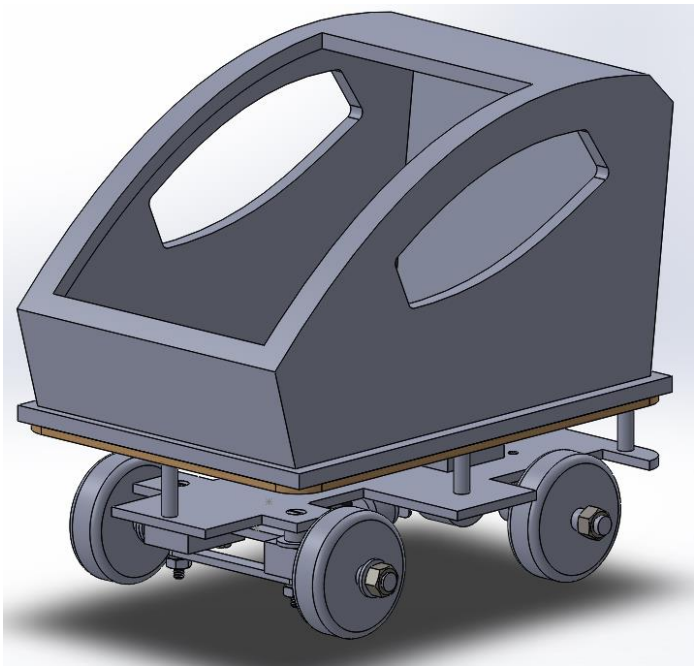


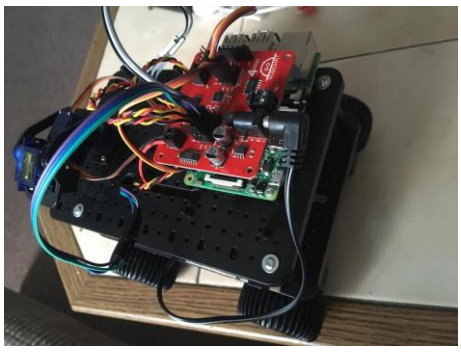
Figure 64- Full design with cosmetic attachment

A cosmetic attachment was designed so that the finished µPod would give a closer visual representation of a miniature version of the full-size LUTZ Pathfinder. The idea to manufacture this component would be using a 3-D printer, with a small lip around the edge so that it could be clipped to the wooden top plate. As the cosmetic piece could be removed and attached easily this would still allow access to the electronic components if required.

4.2 Electrical Testing

4.2.1 Warming-up test

Early on, TSC provided a test vehicle for the project team. The aim is to make us familiar with the vehicle control by the Robohat board (a kind of motor driven board).



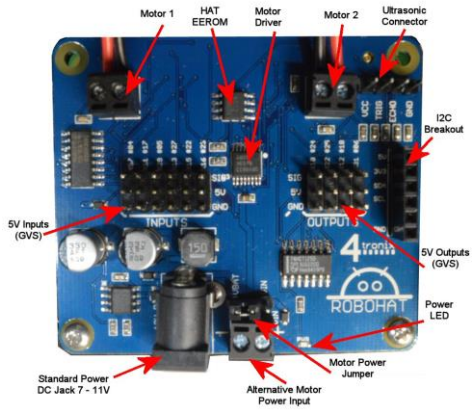


Figure 65- test vehicle and Robohat controller board

This electric vehicle (EV) has four wheels. The left and right-side wheels are controlled by the two different motors respectively to achieve the distributed driving system, which means it can provide independent torques by each separate motor to realize a better manoeuvrability of the vehicle. The master control unit is the latest version of Raspberry Pi 3. Considering the convenience, it doesn't use the on-board GPIO (common purpose input/output) of RPI and chose the Robohat to control the motors.

Robohat is a 40-pin controller shield which completely supports RPI. On the board, there are many interfaces and components specified for different purposes like dual H-bridge motor driver, 4 pin-male header for ultrasonic sensors, etc. Before the vehicle was completed, research was done on the control system and mechanical engineers advised about how to design the structure of a steering system.

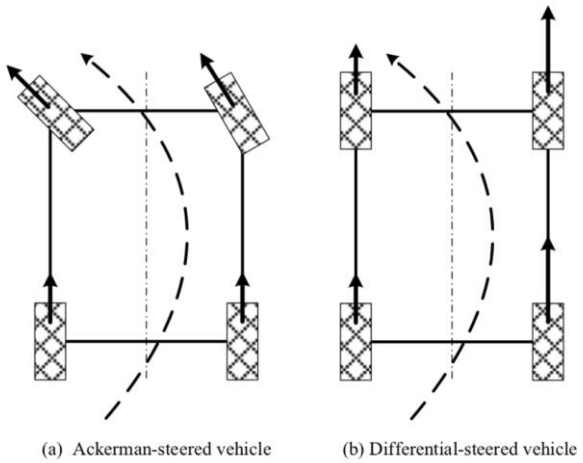


Figure 66- Different steering principles

There are two different methods for vehicle steering. The first one is Ackermann steering and another one is differential speed steering. In this project, the vehicle is required to receive the curvature information from the calculation and implement the algorithm of vehicle control. An Ackermann steering system was chosen at an early stage. One reason is that easy transformation of the curvature into the turning angle of the front wheel and this angle can be controlled by the servo. As for differential steering, Steering Speed Control (SSC) is proposed to keep the turning curve in desired value during acceleration and deceleration driving. [5] However, this speed control is much more complicated.

4.2.2 First integrated entity test

After the completion of the wiring and mechanical assembling, several tests are implemented. Some tests have resulted in failure to some extent. However, by implementing these solutions the µPod has been optimised to a fully functioning state. This is a record of the testing and implementation of necessary changes. The detailed analysis is expanded at the next Chapter.

This was the first test that all the mechanical framework, differential system and control part worked together as a whole. Due to the remote control that allowed the team to visually judge the µPod performance, we run the drive-by-wire mode to test if the control code matches and works

well on the assembled µPod. The process went through with the µPod upside down, which facilitated the observation for the motion.

With the command from the handle, the effective and quick rotation was delivered which showed that the speed control part was successful. However, the servo couldn't turn to the required angle and it seemed to be limited by the Ackermann steering system. We recorded this problem and focussed on the electrical component layout plan on the upper platform in order to test if the vehicle could drive successfully. The next step was to attach the electronic components to the upper platform of the µPod.

After the layout, we began to wire and test the µPod. However, with the power line of the ESC and servo plugged into the battery, the servo immediately turned the front wheels to its right sided limit, which was unexpected. We switched off the power and manually corrected the steering system to the forward direction. A repeated test returned a similar result, although this time led to smoke being emitted by the servo motor. The shell of the servo had been melted partly and the test was ended with the servo broken.

4.2.3 Further testing

4.2.3.1 Steering and power test

The road test show that the motor's torque is not adequate enough and was stuck when encountering some slope. However, in the lab test, motors were running very smoothly at a very fast speed. After some analysis, I find the problems lies in the following points:

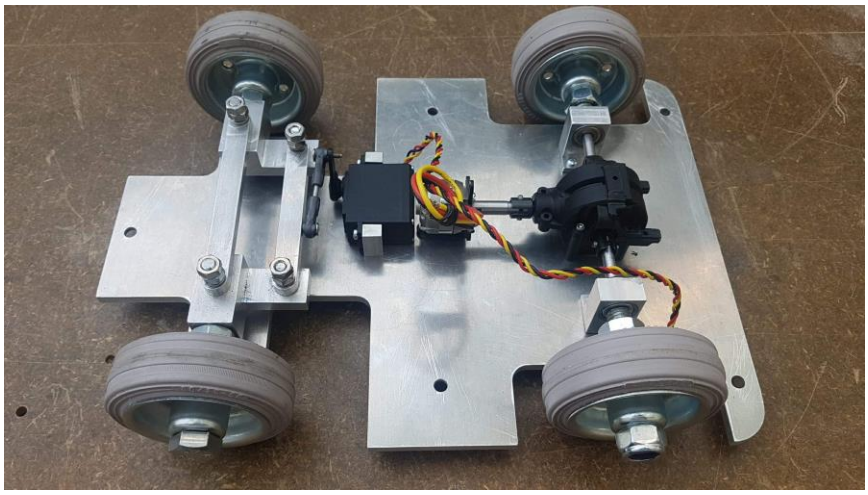


Figure 67- Test without load

- The road friction hasn't been considered.
- The previous calculation of the required motor torque was not accurate.
- Underestimate the weight of the vehicle
- Servo can't rotate the accurate angle

Recalling the selected motor

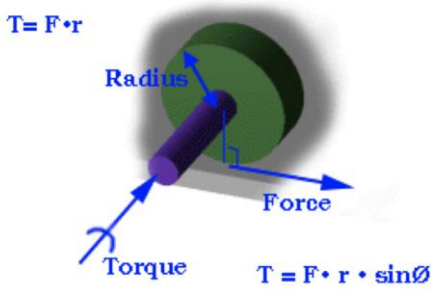


Figure 68- Wheel torque

Table 5- Motor specification

Category	Brushless motor
kv(RPM per volt)	1380
Output power	105w
Operating voltage	7-12 V/DC

The simple way to check the power of the required motor is

$P = T * \omega$ (10)

Where P is power to maintain the motion, T is torque driving the motion, ω is angular velocity of the motion.

Assume vehicle is running at 5m/s, the driving torque is ideally only the friction.

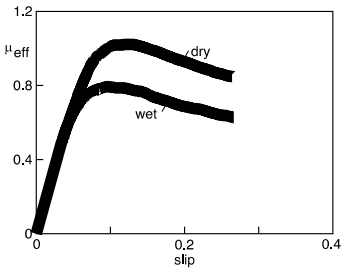


Figure 69- Rubber friction coefficient

Roughly, slip is chosen to be 0.2 and dry surface. The friction coefficient is approximately 0.9. Therefore,

$T = F_{friction} * r = \mu * mg * r = 0.9 * r * mg$ (11)

$\omega = V/r$ (12)

Where V = 5 m/s, r is the radius of the wheel and r = 0.08 m, g = 9.8 N/kg

The whole frame of the vehicle is made of aluminum and its weight is 3.8 kg.

$T = 0.9 * 0.08 * 3.8\ kg * 9.8N/kg$ = 2.68128 (N*m) (13)

$\omega = \frac{5\ m/s}{0.08\ m} = 62.5\ rad/s$ (14)

Therefore,

$P = T * \omega = 2.68128 * 62.5 = 167.58\ (W)$ (15)

The minimum power requirement is 167.58 W (not considering other resistance). When the system is run in reality there would be a power loss between the motor and the connected wheels.

However, our motor's output power is only 105 W which is much lower than the minimum requirement. This was one of the errors made by the team, with an added issue being the aluminium

frame increasing the weight by a large degree. In the future, the estimation of the weight of the vehicle should be calculated more accurately.

As for the control of the motor, ESC is chosen to control the duty cycle of pulse width modulation (PWM).

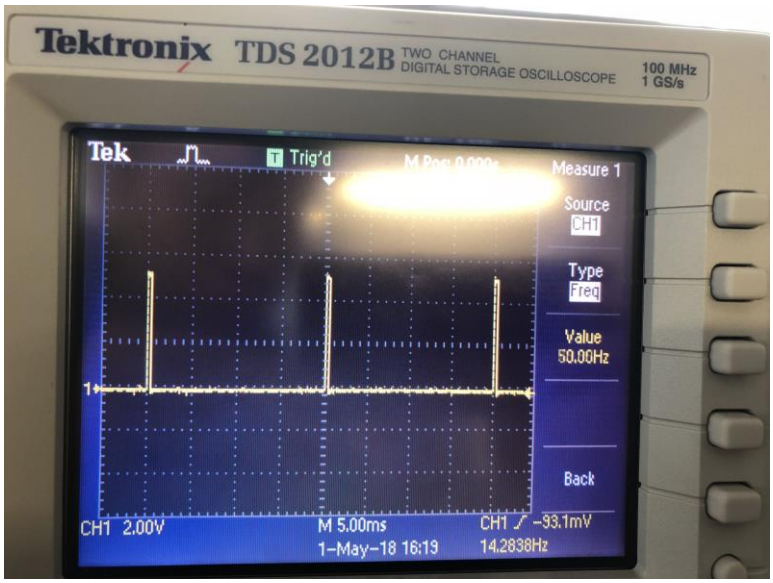


Figure 70- Pulse Width Modulation

The control signal of ESC is the pulse triggered at the frequency of 50.0Hz and the total width of the pulse is 0.02s. From the real test, the maximum rpm is lower than the idea parameter.

$$RPM_{max} = 1380 * 7.4V = 10212 \quad (16)$$

In the Arduino code, motor is controlled by *motor.write(value). where Value ~[0,180].* When the value exceeds the 55, motor is able to drive the vehicle to move. In the whole scale, the corresponding RPM should be RPM_{min} = 10212 * 55/180 = 3120.

The minimum RPM required to drive the vehicle should be calculated in the following steps.

$$\omega = \frac{P}{T} \quad T = F_{friction} * r = \mu * mg * r = 2.68128(N * m) \; P = 105\,W \quad (17)$$

Therefore, $RPM_{ideal} = \omega * \frac{60}{2*\pi} = \frac{P}{T} * \frac{60}{2*\pi} = 374$ (18)

Compared with the real minimum RPM, the ideal RPM is much lower. This is resulted from some other ignored reasons. For example, mechanical resistance of joints.

Figure 71- Servo-steering arm connection

As for servo, in the anticlockwise direction, servo can rotate as has been designed. However, in clockwise direction, servo can't rotate as expected. The reason may lie in the joint of the mechanical structure. As is shown in the picture, this connection design could be improved as it makes this small component require too much load torque. In addition, on the steering side, more bearings should be used to make it move more smoothly and decrease the torque burden for joints.

4.2.3.2 Sensor test

- GPS signal

In this test, we choose to compare two products: pixel phone's GPS and Adafruit GPS.

From the previous part, we have illustrated the performance of these two GPS. Android phone's GPS is more accurate than the other one. This is mainly because of the calibration in the android application. When recording the map, we use rosbag command to record GPS's data.

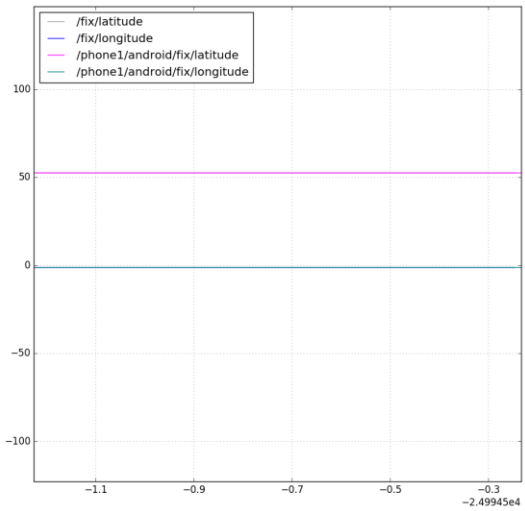


Figure 72- Data comparison 1

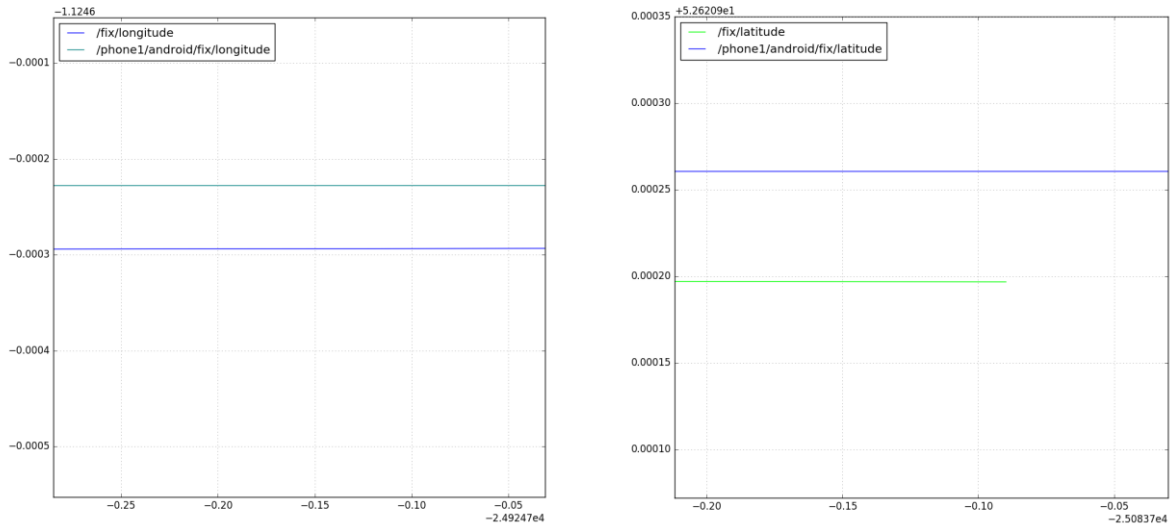


Figure 73- Data comparison 2

In the first picture, we can't see any difference between these two GPS. However, when we narrow the scale, the difference of the latitude and longitude is shown rounded to at least 4 decimals. Theoretically, 4 decimals accuracy is also not very accurate for some sophisticated measurements. In addition, 1 Hz refresh frequency of GPS is not adequate enough and can't ensure a quick response.

- Compass signal

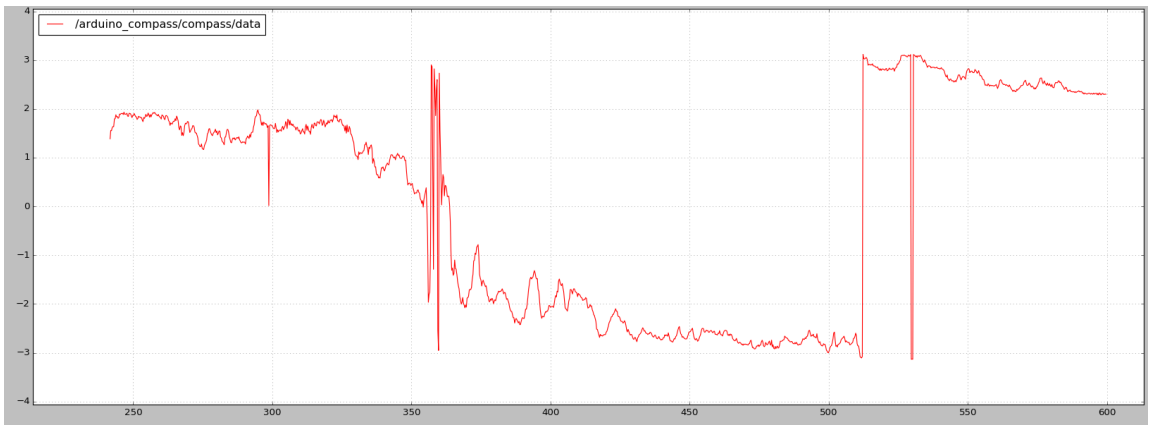


Figure 74- Compass signal

We run the vehicle in a circle and the compass signal should be symmetric. Compass heading is shown in radians and the range is limited to [-PI, PI]. But there are still some significant fluctuations in the graph. This is mainly because that steering at the corner is not very smooth and the calibration of compass doesn't consider the influence of roll and pitch of the vehicle (especially when the road is not very flat).

4.2.3.3 Remote monitor

In order to monitor the topics and nodes when implementing the outdoor test, ROS control center is a good monitor tool which offers an easy way to:

- show nodes, topics and service names.
- subscribe and publish messages.

- call services.
- show and change parameters.

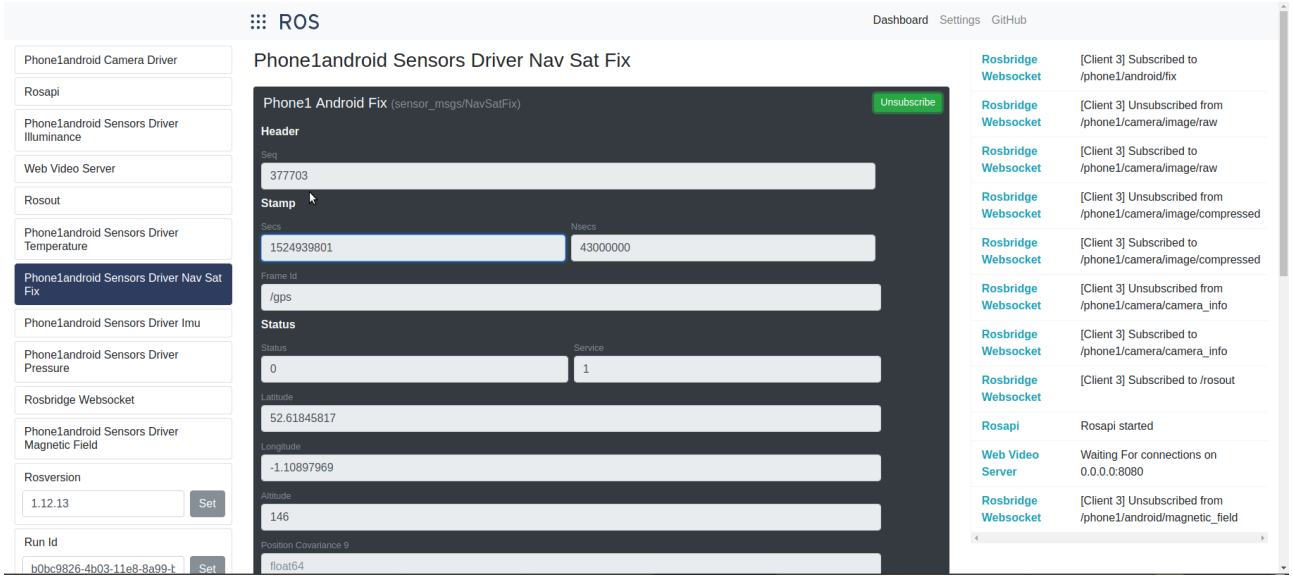


Figure 75- ROS control center

In the website, we can easily subscribe and monitor a specific topic.

Another tool is VNC viewer.

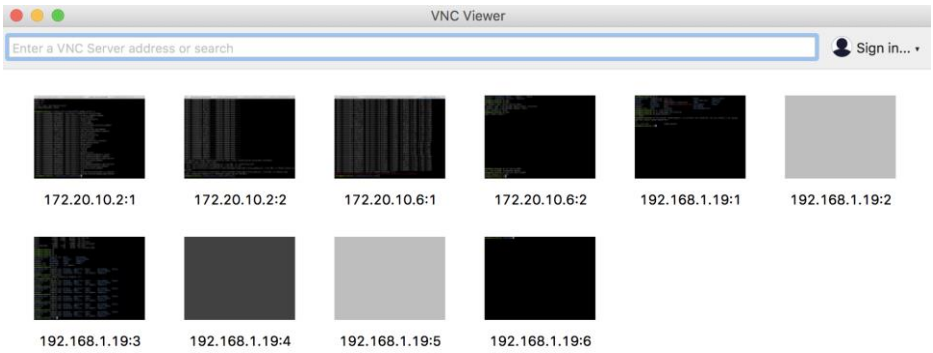
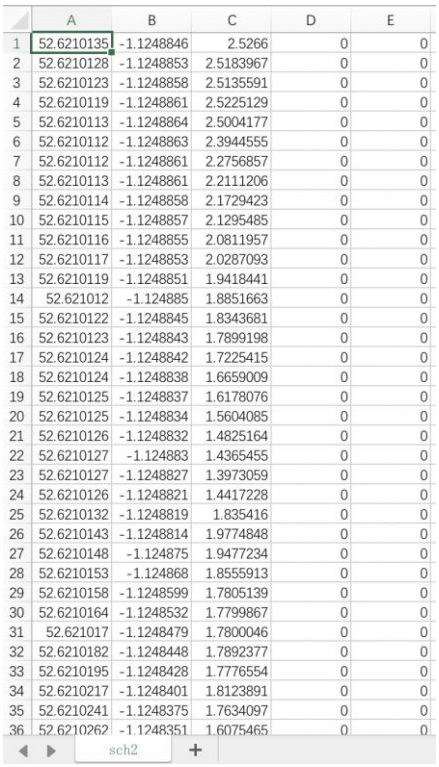


Figure 76- VNC viewer

Users can remotely log into the terminal through connecting IP address in the target computer. It is very convenient for us to log into RPI terminal and do the debugging without the monitor in the outdoor test. In addition, all the computers need to be connected in the same WIFI to achieve the connection.

4.2.3.4 Record map



	A	B	C	D	E
1	52.6210135	-1.1248846	2.5266	0	0
2	52.6210128	-1.1248853	2.5183967	0	0
3	52.6210123	-1.1248858	2.5135591	0	0
4	52.6210119	-1.1248861	2.5225129	0	0
5	52.6210113	-1.1248864	2.5004177	0	0
6	52.6210112	-1.1248863	2.3944555	0	0
7	52.6210112	-1.1248861	2.2756857	0	0
8	52.6210113	-1.1248861	2.2111206	0	0
9	52.6210114	-1.1248858	2.1729423	0	0
10	52.6210115	-1.1248857	2.1295485	0	0
11	52.6210116	-1.1248855	2.0811957	0	0
12	52.6210117	-1.1248853	2.0287093	0	0
13	52.6210119	-1.1248851	1.9418441	0	0
14	52.621012	-1.124885	1.8851663	0	0
15	52.6210122	-1.1248845	1.8343681	0	0
16	52.6210123	-1.1248843	1.7899198	0	0
17	52.6210124	-1.1248842	1.7225415	0	0
18	52.6210124	-1.1248838	1.6659009	0	0
19	52.6210125	-1.1248837	1.6178076	0	0
20	52.6210125	-1.1248834	1.5604085	0	0
21	52.6210126	-1.1248832	1.4825164	0	0
22	52.6210127	-1.124883	1.4365455	0	0
23	52.6210127	-1.1248827	1.3973059	0	0
24	52.6210126	-1.1248821	1.4417228	0	0
25	52.6210132	-1.1248819	1.835416	0	0
26	52.6210143	-1.1248814	1.9774848	0	0
27	52.6210148	-1.124875	1.9477234	0	0
28	52.6210153	-1.124868	1.8555913	0	0
29	52.6210158	-1.1248599	1.7805139	0	0
30	52.6210164	-1.1248532	1.7799867	0	0
31	52.621017	-1.1248479	1.7800046	0	0
32	52.6210182	-1.1248448	1.7892377	0	0
33	52.6210195	-1.1248428	1.7776554	0	0
34	52.6210217	-1.1248401	1.8123891	0	0
35	52.6210241	-1.1248375	1.7634097	0	0
36	52.6210262	-1.1248351	1.6075465	0	0

sch2

Figure 77- CSV file

Launch the record_route node and use its key_entry function to record the whole map (more information is illustrated in the node description). At the end, a CSV (comment separated values) can be obtained from the folder which has been specified before.

To visualize this data, do the following steps:

- Convert the CSV file to KML file (compatible with google earth)
- Use google earth to visualize the location data.

The python script that can automatically transform the CSV file to KML file is shown below. This code refers to the code in the book of “make a raspberry pi-controlled robot” [6] and is modified by us.

```
import string

gps=open('sch1.csv','r')
kml=open('sch1.kml','w')

kml.write('<?xml version="1.0" encoding="UTF-8" ?>\n')
kml.write('<kml xmlns="http://www.opengis.net/kml/2.2">\n')
kml.write('<Document>\n')
kml.write('<name>school8 Path</name>\n')
kml.write('<description>Path taken by catapult</description>\n')
kml.write('<Style id="yellowLineGreenpoly">\n')
kml.write('<LineStyle><color>7f00ffff</color><width>4</width></LineStyle>\n')
kml.write('<PolyStyle><color>7f00ff00</color></PolyStyle>\n')
kml.write('</Style>\n')
kml.write('<Placemark><name>fist100 Path</name>\n')
```

```
kml.write('<styleUrl>#yellowLineGreenPoly</styleUrl>\n')
kml.write('<LineString>\n')
kml.write('<extrude>1</extrude><tesselate>1</tesselate>\n')
kml.write('<altitudeMode>relative</altitudeMode>\n')
kml.write('<coordinates>\n')

for line in gps:
    coordinate=string.split(line,',')
    print coordinate
    longitude = coordinate[1]
    latitude = coordinate[0]
    kml.write(longitude+','+latitude+'\n')
kml.write('</coordinates>\n')
kml.write('</LineString>\n')
kml.write('</Placemark>\n')
kml.write('</Document>\n')
kml.write('</kml>\n')
gps.close()

kml.close()
```

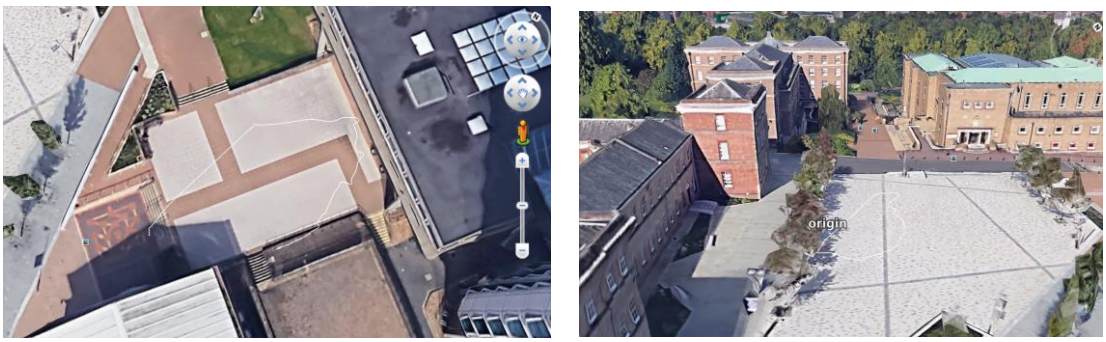


Figure 78- Visualisation in Google Earth

In the first picture, because of the block of the building, the GPS receiver can't receive the signal accurately and therefore drift of the signal results. For the next test, the route is supposed to be away from tall buildings to mitigate this influence.

In the second picture, the vehicle starts in the origin point and run in the circle finally returning to the beginning point. This route is more accurate compared with the real route. However, restricting to the circumstance, the route seems to be not very smooth in the corner. In addition, the start point and returning point is not completely coincided. These two problems may definitely influence the following autonomous test.

4.2.3.5 Drive by wire test

```
podDemand:
  header:
    seq: 0
    stamp:
      secs: 1519752322
      nsecs: 890781461
    frame_id: ''
  steer: 100.0
  speed: 0.0
  source: "xbox"
  driveByWireRequest: True
abdicate: False
```

1

```
---
podDemand:
  header:
    seq: 0
    stamp:
      secs: 1519752246
      nsecs: 590802119
    frame_id: ''
  steer: -100.0
  speed: 0.0
  source: "xbox"
  driveByWireRequest: True
abdicate: False
---
```

2

```
podDemand:
  header:
    seq: 0
    stamp:
      secs: 1519752624
      nsecs: 590795904
    frame_id: ''
  steer: 0.0
  speed: 1.38888883591
  source: "xbox"
  driveByWireRequest: True
abdicate: False
```

3

```
podDemand:
  header:
    seq: 0
    stamp:
      secs: 1519752643
      nsecs: 790788004
    frame_id: ''
  steer: 0.0
  speed: -1.38888883591
  source: "xbox"
  driveByWireRequest: True
abdicate: False
```

4

NO.	command
1	Turn right
2	Turn left
3	forward
4	Backward

Figure 79- Xbox controller demand publish

The drive by wire test is relatively successful and the vehicle can make a correct response when sending the corresponding command. However, the backward function has not been achieved.

4.2.3.6 Drive by autonomous test

Run all_demo.sh to start the autonomous test.

```
#!/bin/bash
# Get script location
SCRIPTPATH="$(dirname "$(readlink -f "$0")")"

# Check path file exists
. $SCRIPTPATH/auxFiles/checkPathFileExists.sh $1

ROUTE_FOLDER="$SCRIPTPATH/../tsc_acs/routes/uniLei"
FENCE_FILE="$SCRIPTPATH/../tsc_acs/fences/empty.csv"

# please revise the original location
ORIGIN="52.6210135,-1.1248846"
MAX_SPEED="5.0"

## Launch nodes
roslaunch $SCRIPTPATH/back.launch &

echo "Sleep start  ********************************************************************************************"
sleep 10
echo "Sleep end    ********************************************************************************************"

## Launch nodes
roslaunch $SCRIPTPATH/front.launch folder:=$ROUTE_FOLDER fencefile:=$FENCE_FILE origin:=$ORIGIN max_speed:=$MAX_SPEED
```

Firstly, specify the location of the route which the vehicle will follow.

Then, address the start point and set the max_speed.

Finally, run the back.launch which is related to motion control and front.launch which is related to path planning.

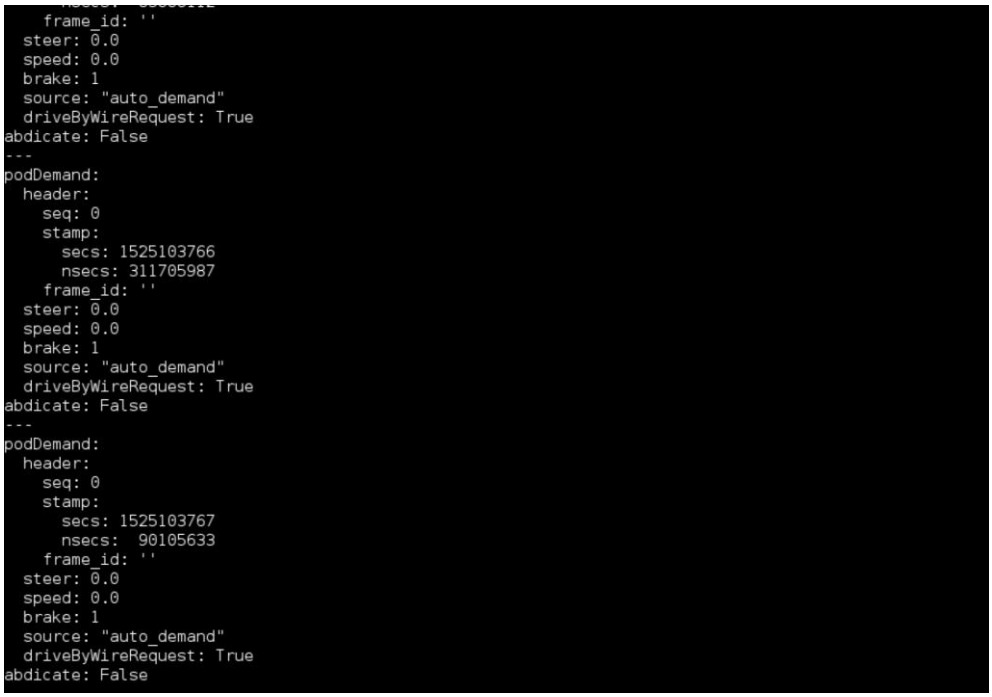


Figure 80- Information received after launch

When all the nodes have been launched it shows that demand by path planning is always zero. It means that path planner can't work correctly and the vehicle can only be controlled by remotely. The problems may result from the accuracy of the sensor. For the next test, it was decided to find a flatter road which is away from the building.

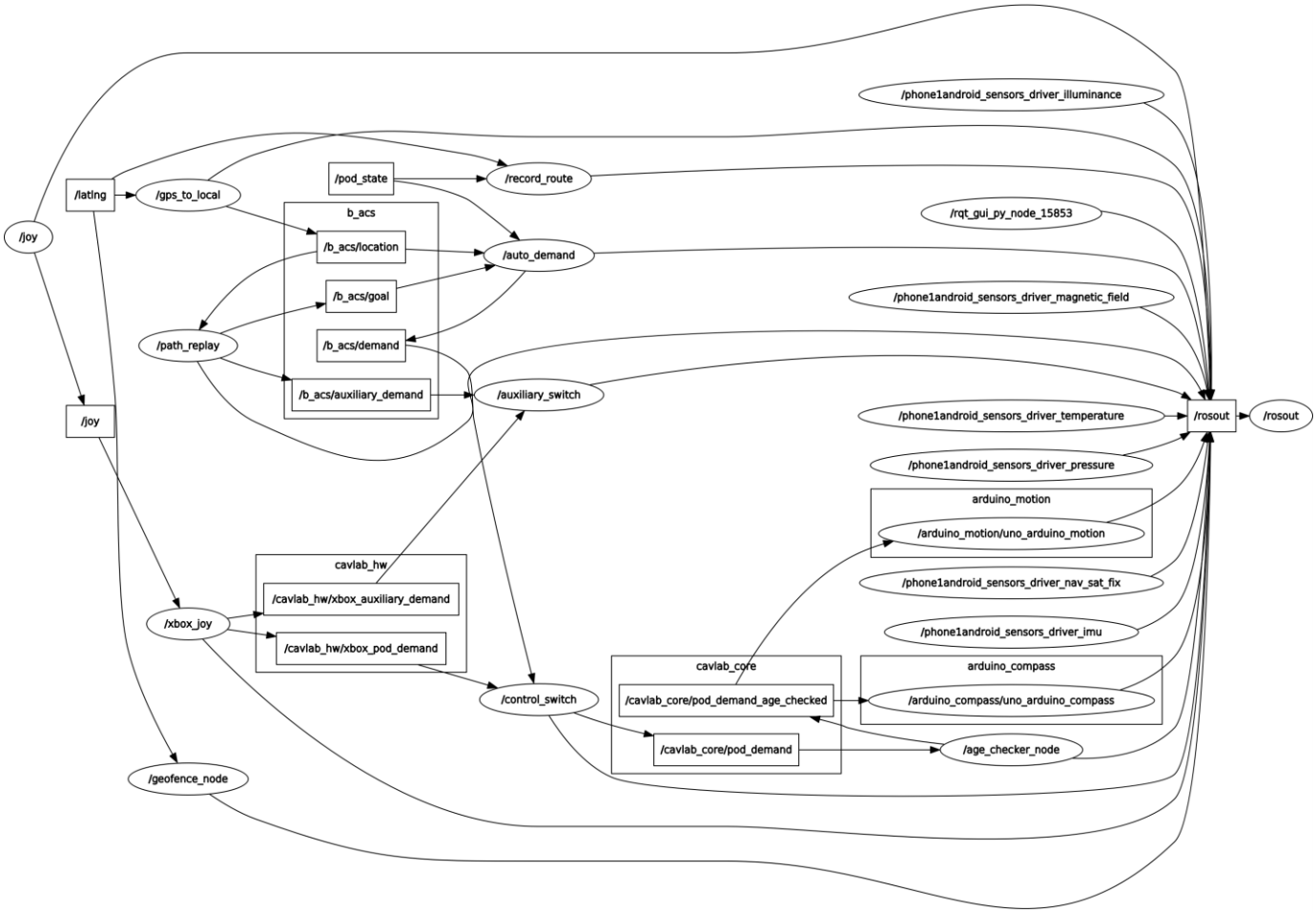


Figure 81- all_demo.sh launch

The autonomous test failed. But we have checked all the relationship between each node and haven't found some problems. Therefore, for the further work, we need to do a thorough diagnosis.

5 Discussion

5.1 Project Management

Upon the completion of the design stage the manufacturing stage was set to begin. However, there had to be some preparation done before this stage. This involved sourcing the necessary materials and attaining and filling out the appropriate forms such as the purchase and work request forms and getting them signed, respectively. The parts also had to be converted into 2D mechanical drawings which are suitable for submission into the manufacturing workshop. All of these had to be completed for manufacturing to begin. The way that the project was initially managed was that the group was split into two sub-teams and a member from each sub-team would fill out their own purchase list forms and try and attain a signature. It was later decided that this isn't ideal, and a team member, Mohamed Khaled, was tasked with attaining a signature, filling out and submitting

the appropriate forms. This made attaining signatures and keeping track of form submissions easier and helped manage delays.

During our project, we were unfortunate in that some strikes took place which meant it would become more difficult to attain a signature for the appropriate forms required for submission. This occurred for both purchase and work request forms causing a delay not only to manufacturing but also to parts arrival time. There were also delays due to some parts arriving very late, namely the differential. This was required for many calculations and design optimisations on other parts that linked together. This therefore also mitigated the projects progress and meant that the project completion date would have to be extended from the initial time thought for project completion.

When these delays were solved, the appropriate forms were filled out, and the appropriate parts and materials were sourced. The mechanical drawings were submitted into the workshop alongside a work request form. This meant that manufacturing would begin. However, it was around this time that most of the fourth-year project groups had also submitted their mechanical drawings for manufacturing, as well as many second and third year students. This meant that there would be further delays to project progress. We were also notified that the second-year project work would be prioritised, and our manufacturing work request would be pushed to the back of the list.

To combat this the physics lab was contacted and a request for manufacturing to be transferred to them was made, this request was accepted. However, the time for completion was like that of the engineering manufacturing workshop. Therefore, it was decided that the drawings would remain in the engineering workshop for manufacturing. This added to the delays that were faced. Some of these delays could have been mitigated by better project management. However, some were unforeseeable events that couldn't have been avoided.

5.2 Mechanical Aspects

In terms of the mechanical design the aim of this project as stated previously was to design and manufacture a 1:6 scale model of the LUTZ Pathfinder pod. As an exact cosmetic replica would be difficult to produce, the initial base requirement was for the µPod to have a track and wheelbase as a scaled down version of the original vehicle. By using the values from the LUTZ Pathfinder specification provided to the team by Transport Systems Catapult, the required dimensions were obtained allowing the team to complete the first key part of the design. By successfully fulfilling this requirement the µPod has the chassis dimensions to act as a model of the desired size. Similarly the wheels of the µPod were selected as close to 1:6 size as possible as this would aid in the attempt to produce a vehicle of inherently similar driving capabilities, albeit on a smaller scale.

The height of the µPod was of less importance to the overall design as it had less impact on the drive capabilities of the vehicle. As such even by utilising the two-tier chassis design the overall height of the completed vehicle is smaller than the 1:6 scale decided. The option of incorporating a cosmetic attachment to imitate the striking design of the LUTZ Pathfinder was raised with a simple design being created but the main priorities of the project laid elsewhere and so this area of the design was left unfinished. In the original plan the requirement to have a completed cosmetic attachment was

always dependent on the remaining budget and time when the more important work was completed so this was not seen as an issue.

The next key design requirement was for the µPod to function with an Ackermann steering system to reflect the steering mechanism of common commercial vehicles. As such a large amount of design time was spent in this area at the beginning of the project. A simple Ackermann approximation was used, where the steering pivot points are moved inwards so as to lie on a line drawn from the steering kingpins to the centre of the rear axle. This meant that from the information gathered about the track and wheelbase the required data to design this system accurately was known. The resulting design for the vehicle therefore has an Ackermann steering system that should ensure the µPod can manoeuvre in a stable manner. The resulting maximum steering angles provided an average of 30° of motion between both wheels, with the inner wheel rotating further than the outer in accordance with Ackermann geometry. This was seen as suitable as the value is consistent with the maximum steering angles majority of commercial automotive vehicles, showing the legitimacy of the design.

A serious issue that was raised in the latter stages of the project was the ground clearance of the chassis, at only 10mm. As the original design consisted of this flat plate being on the underside of all components and hence the component closest to the ground across the whole of the vehicle the chance of problems arising when even the slightest incline was met was high. The change to the design was swift and resulted in a vehicle that is more capable of traversing a wider range of real surfaces, such as being driven on paved surfaces outside. This late significant design change did significantly reduce the need for the two-tier chassis, although there is no a large amount of room on the µPod that could be used for the addition of more sensors if they are needed.

It was decided from an early stage that the µPod should aim to have a top speed at or close to that of the actual LUTZ Pathfinder pod. The calculated theoretical top wheel speed at the maximum motor rpm is 30.5km/h. As the top speed of the original pod is 24km/h the µPod should be capable of reaching this speed, which in turn would allow for a wider range of testing capabilities. The battery life was also calculated, and this should theoretically give 31 minutes, although this is slightly less at 28 minutes when at maximum motor rpm.

When the µPod was tested in the real world a number if problems were revealed. Athough the steering system functions as planned when steering anti-clockwise, in the opposite direction there was an issue with the Ackermann steering system .The mechanical reasons for this are clear, firstly due to the small amount of space available for the steering arm to function the ball joints contact the aluminium tie rod, causing the whole system to lock. This could have been remedied by moving the servo further towards the back of the vehicle, although this would be unable in its current design. If the track was lengthened slightly this would be possible and would significantly reduce the chance of contact between these critical moving parts. An additional improvement would be to add bearings to the steering arm in place of the ball joints to reduce the frictional forces that need to be overcome in order for the steering to function in this direction.

An added issue is related to the drive system. When on a smooth flat surface the vehicle can drive effectively, with the ability to reach a good speed. When faced with even the smallest of incline or step the vehicle is unable to overcome the obstacle. The issue with the ground clearance has been solved, so the issue lays somewhere else. By inspecting the problem the issue seems to stem from the motor not having the required power to drive the vehicle in this situation. Similarly when used on a flat surface with some friction, such as the area outside the David Wilson Library the vehicle cannot reach anywhere near its theoretical top speed and motion is laborious. The root of the problem comes down to the heavy and overdesigned chassis, along with a motor not providing the required power. Moving forward further tests are planned to further inspect this issue and ensure the motor linkage to the differential is functioning effectively. If, as expected, the motor is not sufficient an improved model could be purchased and installed.

5.3 Electrical Aspects

Through the first test, the shell of servo is melted partly as shown with red square in figure 82. On the basis of the servo burning, the backtracking for the problem is mapped below.

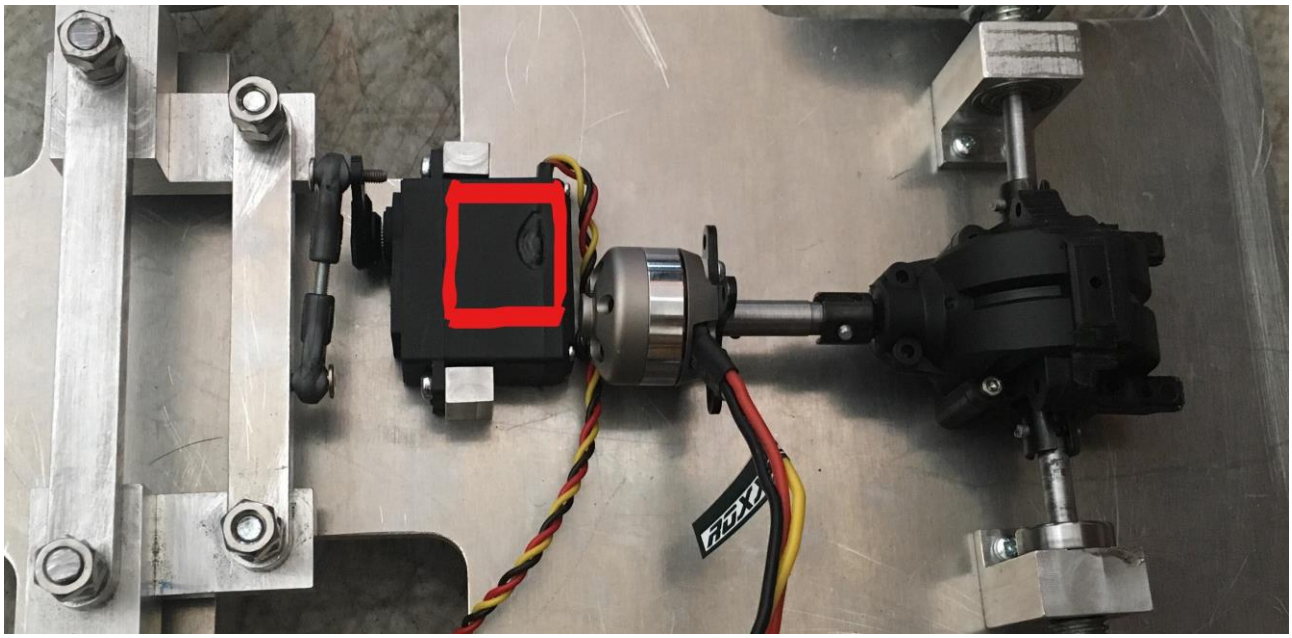


Figure 82- The melted shell of the servo.

After testing another independent set of motion system (ESC and servo), we first check the wiring with the Arduino. By testing three separate conditions, servo signal only, ESC signal only, and both signal lines connected are tested. When both signal lines are connected the servo attachment rotates randomly without any control input.

The final diagnosis for the loss of the servo is the incorrect operation of the power supply. With no power provided to the Arduino, the digital ports are highly likely to output a disordered signal which leads to the oscillation of the servo attachment. Further, a certain oscillating angle will exceed the functioning limits of the steering system. Such angle rotates too far and exceeds the servo rotation range, which leads to the front wheel system becoming jammed. Despite that the servo will be also

called to turn back by the random oscillating signals, the locking of the steering system prevents such motion leading to the damage on the servo motor.

After this motor failure, the possibility of damaging the motion system forces the team to consider how to detect the possible issue timely and how to minimise the consequence for other components. Therefore, the design was adjusted and the battery located on the top platform with the other electrical components. Although the change differs the centre of the gravity, the heavy weight of our vehicle reduces the impact largely. As a result, besides the mechanical drive components, all other components are located at the upper platform.

5.3.1 Backward motion achievement

Until now, our vehicle can't achieve backwards control. There are two solutions:

- Using L298N dual motor driven board

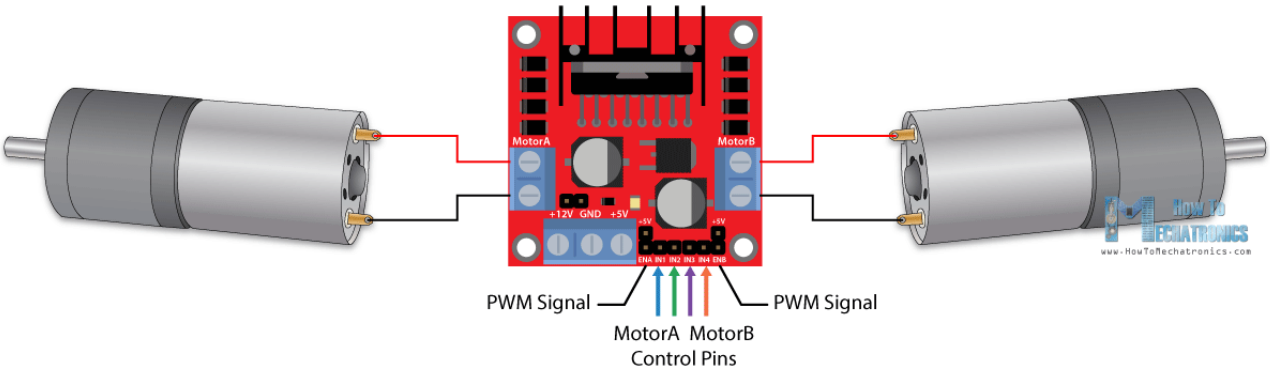


Figure 83- L289N dual motor

INPUT 1	INPUT 2	DIRECTION
Ground	Ground	Motor Off
5 Volts	Ground	Forward
Ground	5 Volts	Reverse
5 Volts	5 Volts	Not Used

It is very easy to achieve the reverse direction of motors by changing the state of the signal input. However, it can only control the brushed motor.

Although there are many disadvantages of brushed motor compared with the brushed motor, it is suitable for our vehicle which doesn't have strict requirements of speed.

- Dual electric speed controller(ESC)

For the previous purchase, we haven't chosen the dual ESC. As for the control of dual ESC, the signal is also at the frequency of 50Hz and the total period is 20ms. When the duty cycle exceeds the half of the total pulse width, it can achieve the reverse direction of motors.

5.3.2 Reverse connection

In the previous test, we have the experience of reversely connecting the battery powering the ESC and servo. It resulted in the burn of ESC immediately which partly destroyed our design. In order to avoid this problem, a reverse connection protection circuit should be required.

There are two solutions:

- Using a diode

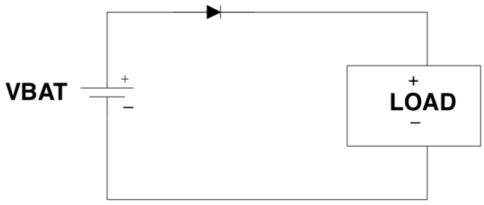


Figure 84- Diode protection circuit

When the battery is connected correctly, the diode becomes forward biased and the load's normal operating current flows through the diode. When the reverse connection happens, the diode will become reverse-biased and no current flows. [7] This application is based on the characteristics of diode.

- Using a MOSFET

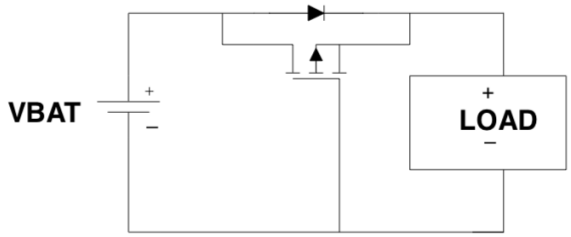


Figure 85- MOSFET protection circuit

In each circuit, the FET's body diode is oriented in the direction of normal current flow. When the battery is reversely connected, the gate voltage of NMOS (PMOS) FET will become low (high) to prevent it from turning on. The resistance of MOSFET is relatively lower but it is more expensive.

6 Conclusions

The µPod was successfully designed and manufactured in accordance with the requirements decided upon at the start of the project. In particular the key requirements of a chassis design to 1:6 scale has been created, with the wheels in use again to the correct scale. An Ackermann steering system has been successfully implemented and functions properly with the inner wheel rotating further than the outer to combat slip when steering. Similarly, on the rear axle a differential has been incorporated effectively and functions as required. Overall the µPod has been designed and

manufactured mechanically as desired, although some components have been over-designed and could have been more streamlined. The design allowed for the attachment of the required sensors and control components, which along with the battery mean a drive-by-wire interface is used to drive the vehicle. Similarly, the mechanical components interact with the sensors and control components successfully to allow for the implementation of autonomous control.

Although the design successfully met the requirements, the performance of the µPod in reality does not at this stage. With the steering system having the tendency to lock when it steers too far in the clockwise direction changes are required in order to permanently solve the issue. Equally, the drive can function well at times but some changes are needed to ensure the vehicle functions more reliably. The addition of a more powerful motor would potentially remedy this issue, although further testing is certainly required to ensure the issue does not lie somewhere else.

With regards to the objective set at the beginning, the completed ROS framework with nodes built by the project team which is compatible with the code provided by TSC, has been successfully structured. The flow line begins with the GPS data collection node written by the team to pass through the processing part provided by Transport Systems Catapult to give the required motion information, and in the end this information has been processed by the vehicle motion node built by us to guide our µPod to move as required. Fairly speaking, the whole system runs smoothly as we expected. It is worth mentioning that the µPod performs well with drive-by-wire mode through tests in campus and the final system can record good quality GPS route. Even though the speed and power are limited, it still proves the node building is quite successful. However, due to the limited time, the final µPod still confronts with the package launch problem with the automotive mode. With positive expectation, the promising automotive GPS path finder µPod is worth of expectation for much more trials will be implemented before the final presentation and also with later optimized power system adopted.

7 Further Work

There remains some work to be done on the mechanical side of the project. This is primarily due to the delays that were faced in the manufacturing stages of the project. There were also some delays due to some important parts arriving at a later than expected date. Further adjustments are required for the drive and steering mechanisms to function correctly. During this time, the team will also look at improving the cosmetics of the vehicle and the completion of the design of the dome that will sit on top of the vehicle. We are also looking at different methods for manufacturing this dome, and their availability, such as 3D printing.

If there was more time the team would have liked to improve the vehicles design such that it allows for an upgradeable chassis that allows for the addition of several other, more expensive, sensors. If we had a larger budget the team would focus more on improving the cosmetics of the design to get it to a more aesthetically comparable level to the original pod. Also, the team would try and simplify the design more and reduce the amount of parts required. The Ackermann steering mechanism for example requires several parts connected, this was complicated in design and produced more risks such as some parts failing resulting in the mechanism not working.

In anticlockwise direction, there exists a collision between the rod and the chassis in the steering part when servo is running. In software algorithm, some changes have been made to avoid the collision. However, these changes have also limited the maximum angle which the servo can turn in the anticlockwise direction. Some changes of the mechanical design should be made in the later time to mitigate this situation.

Although the current motion meets the requirement of the initial objective, the backward motion can be a good point to be developed in the future. The details for these further works are provided in detail in section 5.4. The first car entity test failure also leads the team to believe the protective circuit should be taken into consideration and that the reverse connection protection mechanism should be added later. With such mechanism developed, the outcome from some carelessness operation or accidental attachment of power wires will be minimized.

References

[1] Sparkfun official website. Sparkfun 9Dof sensor stick datasheet [online]. Available from https://www.sparkfun.com/products/13944 [Accessed 30/04/2018]

[2] Sparkfun official website. Alex the giant. 9dof-sensor-stick-hookup-guide [online]. Available from: https://learn.sparkfun.com/tutorials/9dof-sensor-stick-hookup-guide?_ga=2.132081420.206286191.1524680270-239157693.1516142898 [Accessed 30/04/2018]

[3] Honeywell company. COMPASS HEADING USING MAGNETOMETERS [online]. Available from: https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/Magnetic__Literature_Application_notes-documents/AN203_Compass_Heading_Using_Magnetometers.pdf [Accessed 30/04/2018]

[4] Roswiki. Rosserial-2018, Available from: http://wiki.ros.org/rosserial [Accessed 30/04/2018]

[5] Wu, X., Xu, M., & Wang, L. (2013, May). Differential speed steering control for four-wheel independent driving electric vehicle. In Industrial Electronics (ISIE), 2013 IEEE International Symposium on (pp. 1-6). IEEE.

[6] Donat, W. 2014. Make a Raspberry Pi-Controlled Robot: Building a Rover with Python, Linux, Motors, and Sensors. San Francisco, US: Maker Media, Inc.

[7] Falin, J. from PMP Portable Power. 2003. Reverse Current/Battery Protection Circuits. Texas Instrument Application report (SLVA139).

Appendix 1 – Control Data

ROS nodes specification:

1. GPS node

GPS node contains three sub-nodes.

- nmea_navsat_driver(using Adafruit GPS)

This package provides a ROS interface for GPS devices that output compatible National Marine Electronics Association (NMEA) sentences and publishes latitude and longitude taken from NMEA sentences. GPS device should be compatible with NMEA. From the real test, Adafruit ultimate GPS can work well.

	Topic	Type
Published topics	fix	sensor_msgs/NavSatFix
Subscribed topics	None	-

- android_all_sensors_driver

Using android phone to publish its GPS data.

	Topic	Type
Published topics	Phone1_android/fix	sensor_msgs/NavSatFix
Subscribed topics	None	-

- Sparkfun compass

Using sparfun sensor stick IMU to publish compass heading in the range of [-PI, PI].

	Topic	Type
Published topics	Arduino_compass/compass	std_msgs/Float64
Subscribed topics	None	-

- connection

Combine latitude and longitude from GPS and heading from IMU to publish an integrated topic.

	Topic	Type
Published topics	latlng	b_acs::LatLngHeadingFix
Subscribed topics	Arduino_compass/compass	std_msgs/Float64
	Phone1_android/fix	sensor_msgs/NavSatFix

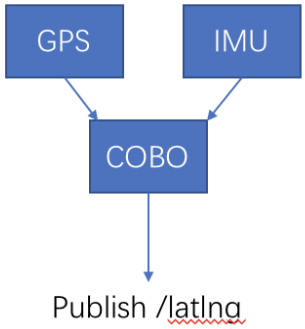


Figure 86- Combinations

When we implement this process, the frequency of GPS and IMU need to be considered. This refresh rate is a very important parameter to evaluate and can directly influence the quality of the decision of a vehicle. The faster refresh rate is, the more accurate navigation we can get. In fact, our GPS can only provide 1Hz refresh rate and IMU is 10Hz. Therefore, in order to synchronize two signals, we publish integrated topic when GPS data is received because IMU data is faster and available.

2. Gps_to_local

Node to translate incoming GPS co-ordinates into a locationin a local frame.

$$x = x_o + R\cos lat_o \sin lng - lng_o$$

$$y = y_o + R\sin lat - lat_o$$

$$\theta = \frac{\pi}{2} - heading$$

where R is the radius of the earth (6371000m), (x_o, y_o) is the origin in local co-ordinates, (lat_o, lng_o) are the GPS co-ordinates of the origin

	Topic	Type
Published topics	b_acs/location	b_acs::Pose2DStamped
Subscribed topics	latlng	b_acs::LatLngHeadingFix

3. Path_replay

Reads a CSV file containing a sequence of GPS co-ordinates (lat, lng, heading). Starting at the start of the list the points are published in turn as a goal as soon as they come into range. Published goals are in the local frame.

	Topic	Type
Published topics	b_acs/goal	b_acs::Goal
Subscribed topics	b_acs/location	b_acs::Pose2DStamped

4. Auto_demand

This is the central control node of the ACS. It publishes steer and speed demand based on current location and current goal.

$$\mathrm{K} = \mathrm{K}_l \frac{\mathrm{e}_l}{r} + \mathrm{K}_\psi e_\psi$$

where e_l is the lateral error to the goal, r is the distance to the goal and e_ψ is the difference between goal heading and current heading.

	Topic	Type
subscribed topics	b_acs/goal	b_acs::Goal
	b_acs/location	b_acs::Pose2DStamped
Published topics	b_acs/demand	PodDemandSource

5. Xbox_joy

A general ROS node for controlling linux xbox controller

	Topic	Type
Published topics	joy	Sensor_msgs/Joy
Subscribed topics	None	-

6. Control switch

The control switch publishes one or more prioritized demand sources from Xbox controller or autonomous demand.

	Topic	Type
Published topics	Cav_core/Pod_demand	Cav_core/PodDemand
Subscribed topics	b_acs/demand	PodDemandSource

7. Age_checker

The age checker republishes all incoming messages provided that they are no older than age_to_start_ramp. Once the most recent demand is older that age_to_start_ramp the speed in the demand is reduced linearly down to zero after age_to_stop

	Topic	Type
Published topics	Cav_core/pod_demand_age_checked	Cav_core/PodDemand
Subscribed topics	Cav_core/Pod_demand	Cav_core/PodDemand

8. Vehicle motion

This node is built in Arduino and used to control the motor and servo to achieve the manoeuvrability. Servo is used for steering and the communication between the master node and Arduino is connected by rosserial package.

	Topic	Type
Published topics	None	-
Subscribed topics	Cav_core/pod_demand_age_checked	Cav_core/PodDemand

Considering the copyright, we will not mention the intermediate nodes which is constructed by catapult.

Programming codes(partly)

motion_node.cpp

```
/***
designed by feifei
created by wawa
From leicester university team
Email:736499127wawa@gmail.com

***/

#if (ARDUINO >= 100)
 #include <Arduino.h>
#else
 #include <WProgram.h>
#endif

#include <ros.h>
#include <Servo.h>
#include <cavlab_core/PodDemand.h>

ros::NodeHandle  nh;
Servo mymotor,servo;

float angle_ini= 99.0;
float i,angle;
int angle1;
float V,Mrmp;


//For rosserial void nh.loginfo() can only publish string data.
//showme_servo will help you check the signal you write to servo
void showme_servo(float value){
  char result[8];
  char log_msg[30];
  dtostrf(value,6,2,result);
  sprintf(log_msg,"servo angle is %s",result);
  nh.loginfo(log_msg);
  }

//showme_motor will help you check the signal you write to motor
void showme_motor(int value){
  char result[8];
  char log_msg[30];
  dtostrf(value,6,2,result);
  sprintf(log_msg,"motor angle is %s",result);
  nh.loginfo(log_msg);
}



void servo_cb( const cavlab_core::PodDemand& cmd_msg){

  V = cmd_msg.speed;

  // according to Ackmansteering, our servo's effective angle range is [48,150]
 //you have to scale it to your max and min angle of your design
  angle = (atan(cmd_msg.steer * 0.19)* 180/PI)*9.392+99;
```

```
  // make sure velocity is positive because our vehicle cannot backward
  if (V>=0)
  {

  angle1= 24.96*V/(PI*0.08*4.15)+60;
  showme_motor(angle1);
  mymotor.write(angle1); //set servo angle, should be from 0-180
  nh.loginfo("--motor is powering on--");
  }

  // this for loop is to make the steering move smoothly and gradually
  // you can change the delay time to meet your requirement
  if(angle >= angle_ini){
  for(i=angle_ini;i<=angle;i++){
      showme_servo(i);
      servo.write(i);
      delay(6);
    }
  } else if(angle <angle_ini){
    for(i=angle_ini;i>=angle;i--){
      showme_servo(i);
      servo.write(i);
      delay(6);
    }
    }
    angle_ini = angle;//store the angle_ini
}

// the subscribed topic should be changed to age_checked_pod_demand
ros::Subscriber<cavlab_core::PodDemand> sub("/cavlab_core/pod_demand_age_checked",
servo_cb);

void setup() {
  mymotor.write(20);
//  servo.write(103);
  nh.initNode();
  nh.subscribe(sub);


 // signal wire connection
  mymotor.attach(7);
  servo.attach(13);

}

void loop() {

  nh.spinOnce();
  delay(1);
}
```

Compass_node.cpp

```
/******************************************************************
Transfrom from arduino serial to rosserial(ROS version)
******************************************************************/

/******************************************************************
LSM9DS1_Basic_I2C.ino
SFE_LSM9DS1 Library Simple Example Code - I2C Interface
Jim Lindblom @ SparkFun Electronics
Original Creation Date: April 30, 2015
https://github.com/sparkfun/LSM9DS1_Breakout

The LSM9DS1 is a versatile 9DOF sensor. It has a built-in
accelerometer, gyroscope, and magnetometer. Very cool! Plus it
functions over either SPI or I2C.

This Arduino sketch is a demo of the simple side of the
SFE_LSM9DS1 library. It'll demo the following:
* How to create a LSM9DS1 object, using a constructor (global
  variables section).
* How to use the begin() function of the LSM9DS1 class.
* How to read the gyroscope, accelerometer, and magnetometer
  using the readGryo(), readAccel(), readMag() functions and
  the gx, gy, gz, ax, ay, az, mx, my, and mz variables.
* How to calculate actual acceleration, rotation speed,
  magnetic field strength using the calcAccel(), calcGyro()
  and calcMag() functions.
* How to use the data from the LSM9DS1 to calculate
  orientation and heading.

Hardware setup: This library supports communicating with the
LSM9DS1 over either I2C or SPI. This example demonstrates how
to use I2C. The pin-out is as follows:
        LSM9DS1 --------- Arduino
         SCL ---------- SCL (A5 on older 'Duinos')
         SDA ---------- SDA (A4 on older 'Duinos')
         VDD ------------- 3.3V
         GND ------------- GND
(CSG, CSXM, SDOG, and SDOXM should all be pulled high.
Jumpers on the breakout board will do this for you.)

The LSM9DS1 has a maximum voltage of 3.6V. Make sure you power it
off the 3.3V rail! I2C pins are open-drain, so you'll be
(mostly) safe connecting the LSM9DS1's SCL and SDA pins
directly to the Arduino.

Development environment specifics:
```

```
        IDE: Arduino 1.6.3
        Hardware Platform: SparkFun Redboard
        LSM9DS1 Breakout Version: 1.0

This code is beerware. If you see me (or any other SparkFun
employee) at the local, and you've found our code helpful,
please buy us a round!

Distributed as-is; no warranty is given.
*****************************************************************/

/*****************************************************************
This file is modified by wawa from university of leicester
Data: sat 28 Apr 2018 21:14
Email: 736499127wawa@gmail.com
*****************************************************************/


// The SFE_LSM9DS1 library requires both Wire and SPI be
// included BEFORE including the 9DS1 library.
#include <ros.h>
#include <Wire.h>
#include <SPI.h>
#include <SparkFunLSM9DS1.h>
#include <std_msgs/Float64.h>
#include <sensor_msgs/Imu.h>
///////////////////////////
// LSM9DS1 Library Init //
///////////////////////////
// Use the LSM9DS1 class to create an object. [imu] can be
// named anything, we'll refer to that throught the sketch.
ros::NodeHandle  nh;
LSM9DS1 imu;

///////////////////////
// Example I2C Setup //
///////////////////////
// SDO_XM and SDO_G are both pulled high, so our addresses are:
#define LSM9DS1_M 0x1E // Would be 0x1C if SDO_M is LOW
#define LSM9DS1_AG          0x6B // Would be 0x6A if SDO_AG is LOW

////////////////////////////
// Sketch Output Settings //
////////////////////////////
#define PRINT_CALCULATED
//#define PRINT_RAW
#define PRINT_SPEED 250 // 250 ms between prints
static unsigned long lastPrint = 0; // Keep track of print time
```

```
// Earth's magnetic field varies by location. Add or subtract
// a declination to get a more accurate heading. Calculate
// your's here:
// http://www.ngdc.noaa.gov/geomag-web/#declination
//#define DECLINATION -8.58 // Declination (degrees) in Boulder, CO.
#define DECLINATION -0.88

float heading_data;
std_msgs::Float64 heading;
//sensor_msgs::Imu imu_data;
ros::Publisher pub("compass",&heading);
//ros::Publisher pub1("/sensor_stick/imu",&imu_data);
float printAttitude(float ax, float ay, float az, float mx, float my, float mz);

void setup()
{
  nh.initNode();
  nh.advertise(pub);
  //nh.advertise(pub1);
  // Before initializing the IMU, there are a few settings
  // we may need to adjust. Use the settings struct to set
  // the device's communication mode and addresses:
  imu.settings.device.commInterface = IMU_MODE_I2C;
  imu.settings.device.mAddress = LSM9DS1_M;
  imu.settings.device.agAddress = LSM9DS1_AG;
  // The above lines will only take effect AFTER calling
  // imu.begin(), which verifies communication with the IMU
  // and turns it on.
  if (!imu.begin())
  {
    nh.loginfo("Failed to communicate with LSM9DS1.");
    nh.loginfo("Double-check wiring.");
    nh.loginfo("Default settings in this sketch will " \
               "work for an out of the box LSM9DS1 " \
               "Breakout, but may need to be modified " \
               "if the board jumpers are.");
    while (1)
      ;
  }
}

void loop()
{
  // Update the sensor values whenever new data is available
  if ( imu.gyroAvailable() )
  {
    // To read from the gyroscope,  first call the
    // readGyro() function. When it exits, it'll update the
    // gx, gy, and gz variables with the most current data.
```

```
    imu.readGyro();
  }
  if ( imu.accelAvailable() )
  {
    // To read from the accelerometer, first call the
    // readAccel() function. When it exits, it'll update the
    // ax, ay, and az variables with the most current data.
    imu.readAccel();
  }
  if ( imu.magAvailable() )
  {
    // To read from the magnetometer, first call the
    // readMag() function. When it exits, it'll update the
    // mx, my, and mz variables with the most current data.
    imu.readMag();
  }

  if ((lastPrint + PRINT_SPEED) < millis())
  {
   // printImu();
   // pub1.publish(&imu_data);

    // Print the heading and orientation for fun!
    // Call print attitude. The LSM9DS1's mag x and y
    // axes are opposite to the accelerometer, so my, mx are
    // substituted for each other.
    heading_data = printAttitude(imu.ax, imu.ay, imu.az,
                -imu.my, -imu.mx, imu.mz);
    heading.data = heading_data;
    pub.publish(&heading);
    nh.spinOnce();
    delay(10);

    lastPrint = millis(); // Update lastPrint time
  }
}

/***void printImu()
{
  imu_data.linear_acceleration.x = imu.calcAccel(imu.ax);
  imu_data.linear_acceleration.y = imu.calcAccel(imu.ay);
  imu_data.linear_acceleration.z = imu.calcAccel(imu.az);

  imu_data.angular_velocity.x = imu.calcGyro(imu.gx);
  imu_data.angular_velocity.y = imu.calcGyro(imu.gy);
  imu_data.angular_velocity.z = imu.calcGyro(imu.gz);

  imu_data.orientation.x = imu.calcMag(imu.mx);
  imu_data.orientation.y = imu.calcMag(imu.my);
```

```
  imu_data.orientation.z = imu.calcMag(imu.mz);

  imu_data.header.stamp = nh.now();
}
***/

// Calculate pitch, roll, and heading.
// Pitch/roll calculations take from this app note:
// http://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf?fpsp=1
// Heading calculations taken from this app note:
// http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-
documents/Defense_Brochures-documents/Magnetic__Literature_Application_notes-
documents/AN203_Compass_Heading_Using_Magnetometers.pdf
float printAttitude(float ax, float ay, float az, float mx, float my, float mz)
{
  float roll = atan2(ay, az);
  float pitch = atan2(-ax, sqrt(ay * ay + az * az));

  float heading;
  if (my == 0)
    heading = (mx < 0) ? PI : 0;
  else
    heading = atan2(mx, my);

  heading -= DECLINATION * PI / 180;

  if (heading > PI) heading -= (2 * PI);
  else if (heading < -PI) heading += (2 * PI);
  else if (heading < 0) heading += 2 * PI;

  // Convert everything from radians to degrees:
  if (heading > PI) heading -= (2 * PI);

  return heading;
}
```

Connection.py

```
#! /usr/bin/env python

import rospy
from sensor_msgs.msg import NavSatFix
from std_msgs.msg import Float64
from cavlab_core.msg import LatLngHeadingFix

data = LatLngHeadingFix()
```

```
pub = rospy.Publisher('latlng', LatLngHeadingFix, queue_size = 10)

def onGps(gps_msg):
        data.latitude = gps_msg.latitude
        data.longitude = gps_msg.longitude
       data.header =  gps_msg.header
        rospy.loginfo("----reading gps data------")

        pub.publish(data)

def onCompass(compass_msg):
        data.heading = compass_msg.data
       rospy.loginfo("----reading compass data-----")

def connection():
       rospy.Subscriber('phone1/android/fix', NavSatFix, onGps)
        rospy.Subscriber('arduino_compass/compass', Float64, onCompass)
        rospy.init_node('gps_compass')
        rospy.spin()

if __name__ == '__main__':
        connection()
```

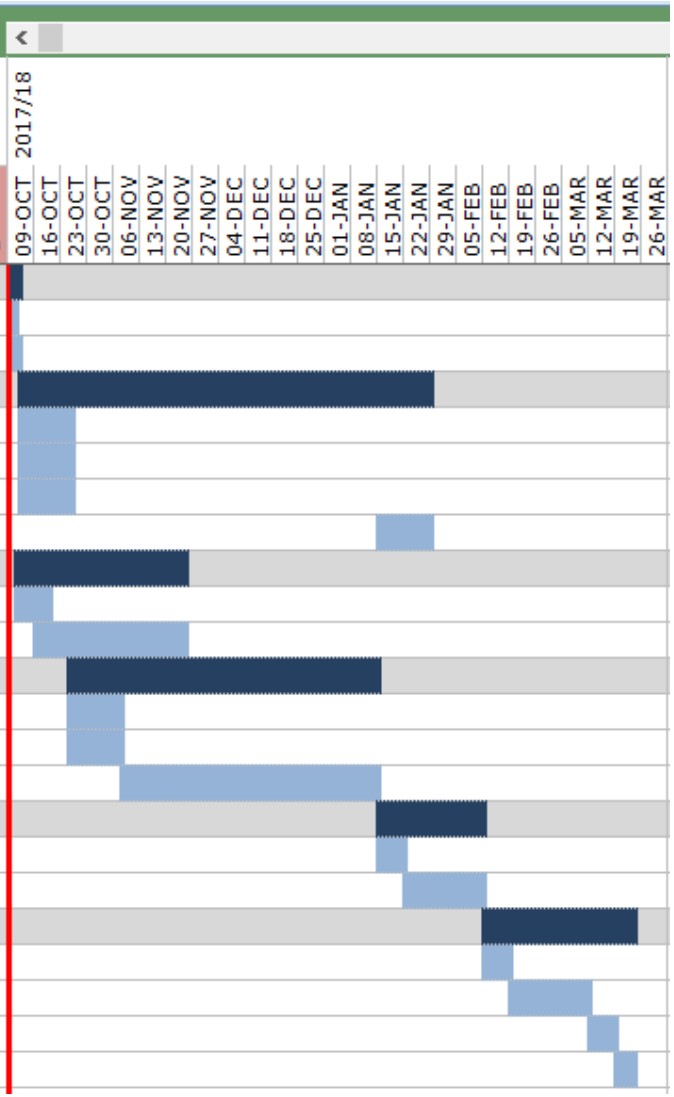
Appendix 2 – Gantt Chart

CavLab Project — Version 1.0

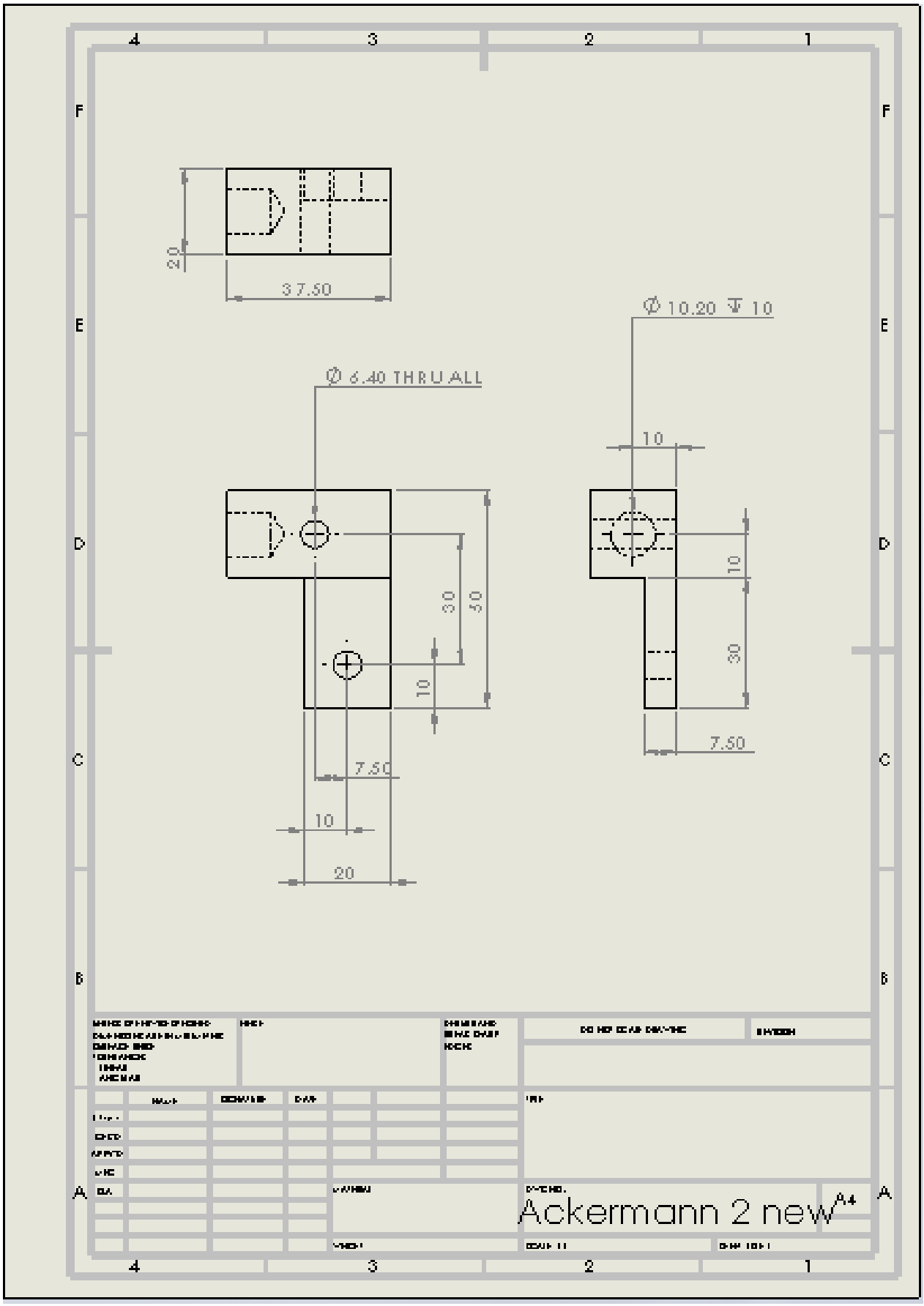
PM Hongfei Chen
Start 09-Oct-17
Today 09-Oct-17

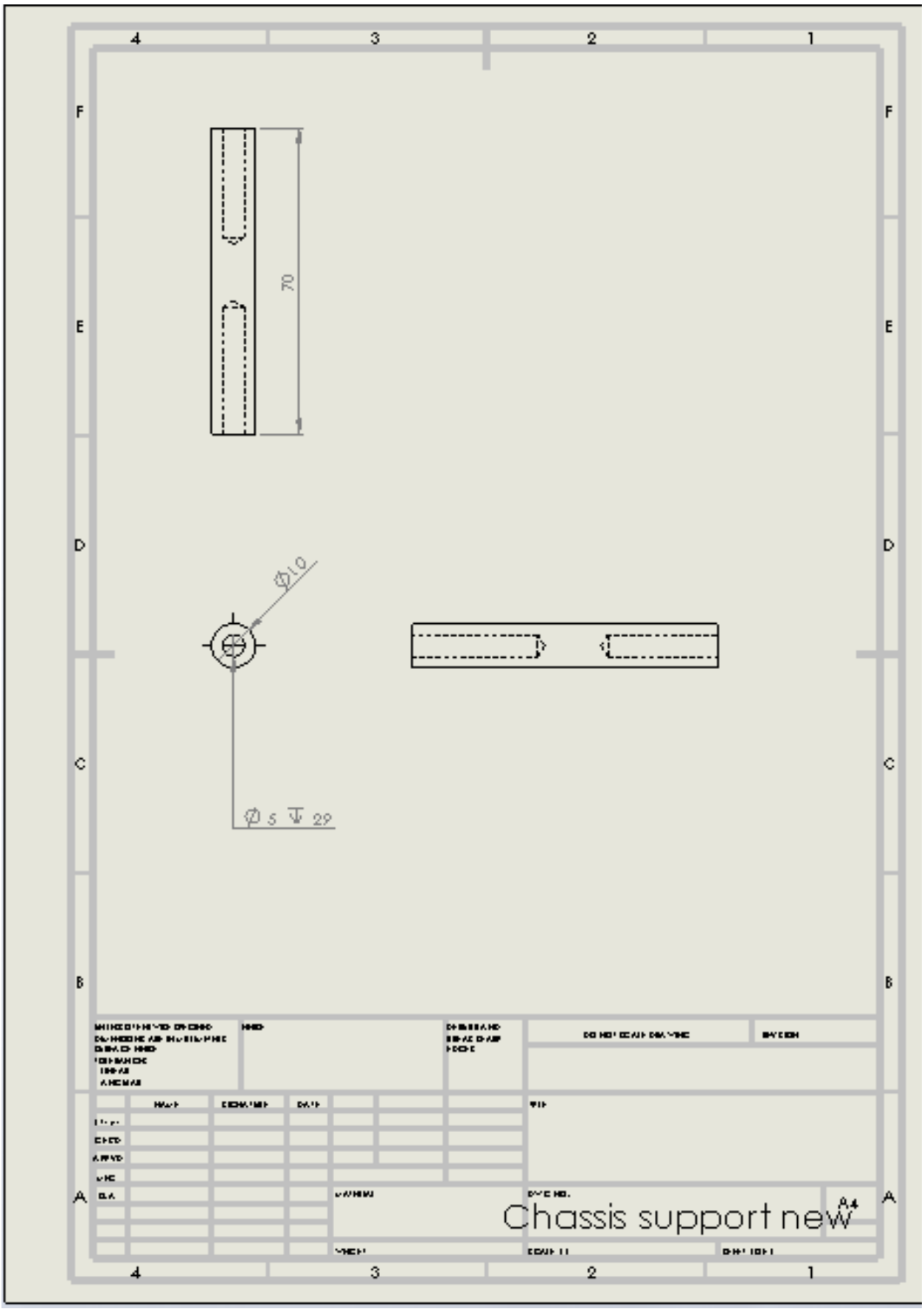
Professional Excel

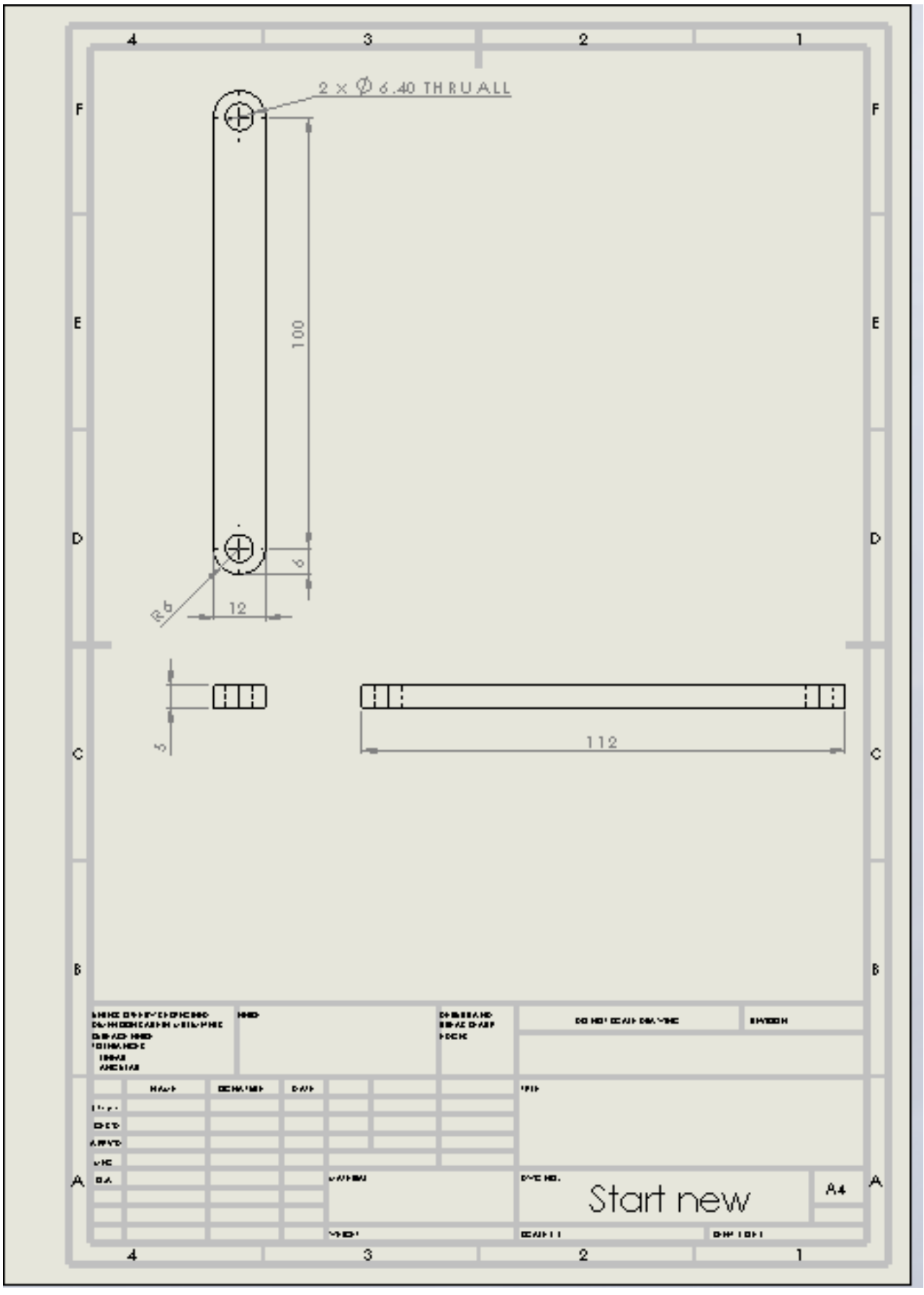
ID	Task Description	Start Date	Planned End Date	Actual End Date	Work Days Planned	Work Days Elapsed	Work Days Remain	Work Days Overrun
1	**Initial Documentation**	**10-Oct-17**	**12-Oct-17**		**3**	**0**	**3**	**0**
1.1	Gather requirements	10-Oct-17	11-Oct-17		2	0	2	0
1.2	Study LUTZ Pathfinder specification	10-Oct-17	12-Oct-17		3	0	3	0
2	**Mechanical Design**	**12-Oct-17**	**29-Jan-18**		**78**	**0**	**78**	**0**
2.1	Design chassis	12-Oct-17	26-Oct-17		11	0	11	0
2.2	Design drivetrain	12-Oct-17	26-Oct-17		11	0	11	0
2.3	Design Ackermann steering	12-Oct-17	26-Oct-17		11	0	11	0
2.4	Design body work for the vehicle	15-Jan-18	29-Jan-18		11	0	11	0
3	**Electrical & Software Design**	**11-Oct-17**	**25-Nov-17**		**33**	**0**	**33**	**0**
3.1	Source electronic components	11-Oct-17	20-Oct-17		8	0	8	0
3.2	Design control system	16-Oct-17	25-Nov-17		30	0	30	0
4	**Manufacture**	**25-Oct-17**	**15-Jan-18**		**59**	**0**	**59**	**0**
4.1	Source required materials	25-Oct-17	08-Nov-17		11	0	11	0
4.2	Source necessary mechnical parts	25-Oct-17	08-Nov-17		11	0	11	0
4.3	Manufacture base vehicle	08-Nov-17	15-Jan-18		49	0	49	0
5	**Control Implementation**	**15-Jan-18**	**12-Feb-18**		**21**	**0**	**21**	**0**
5.1	Install ROS	15-Jan-18	22-Jan-18		6	0	6	0
5.2	Implement speed and steer API	22-Jan-18	12-Feb-18		16	0	16	0
6	**Testing**	**12-Feb-18**	**24-Mar-18**		**30**	**0**	**30**	**0**
6.1	System Test	12-Feb-18	19-Feb-18		6	0	6	0
6.2	Bug fixes/improvements	19-Feb-18	12-Mar-18		16	0	16	0
6.3	Add cosmetic improvements	12-Mar-18	19-Mar-18		6	0	6	0
6.4	Final testing	19-Mar-18	24-Mar-18		5	0	5	0

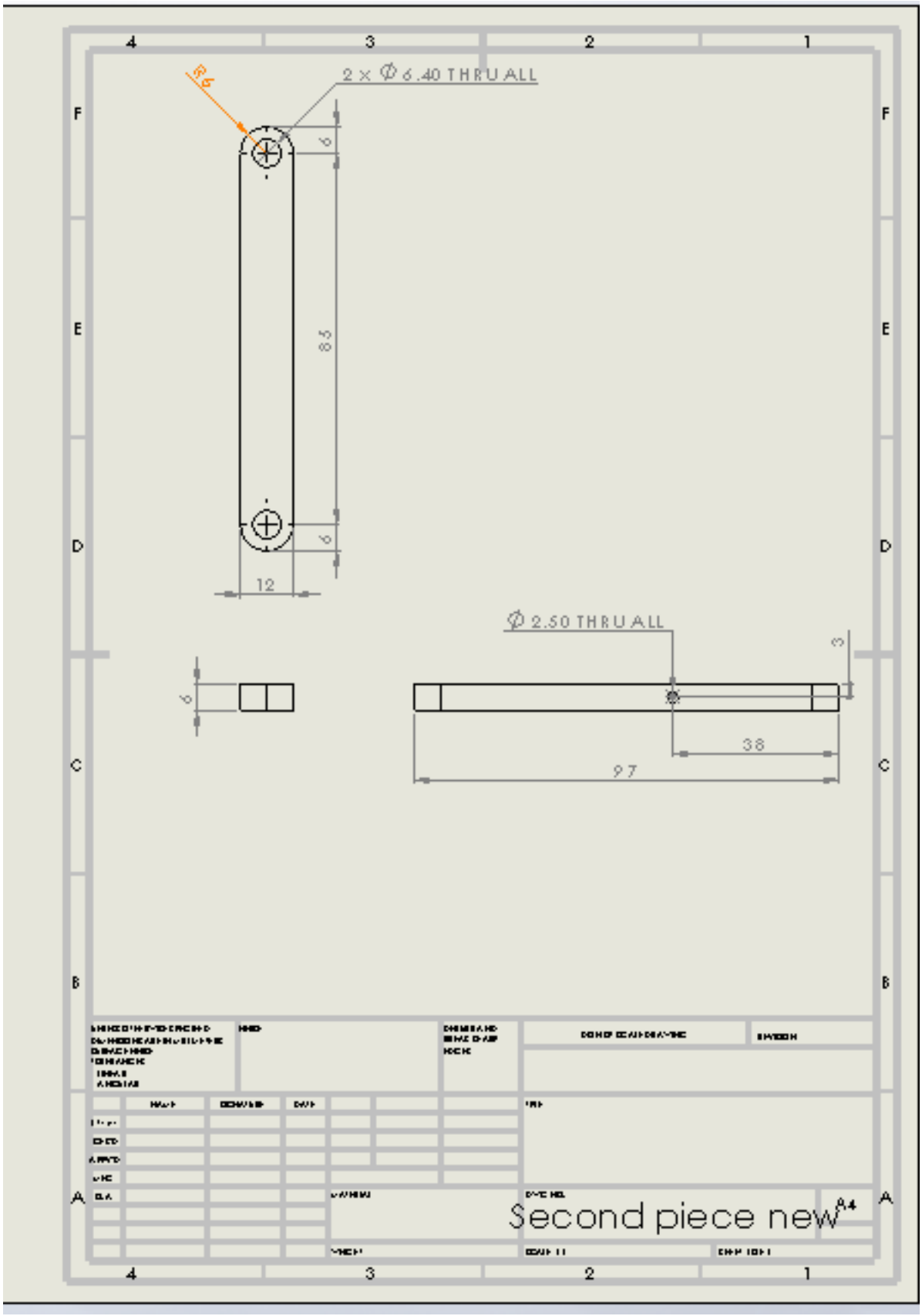


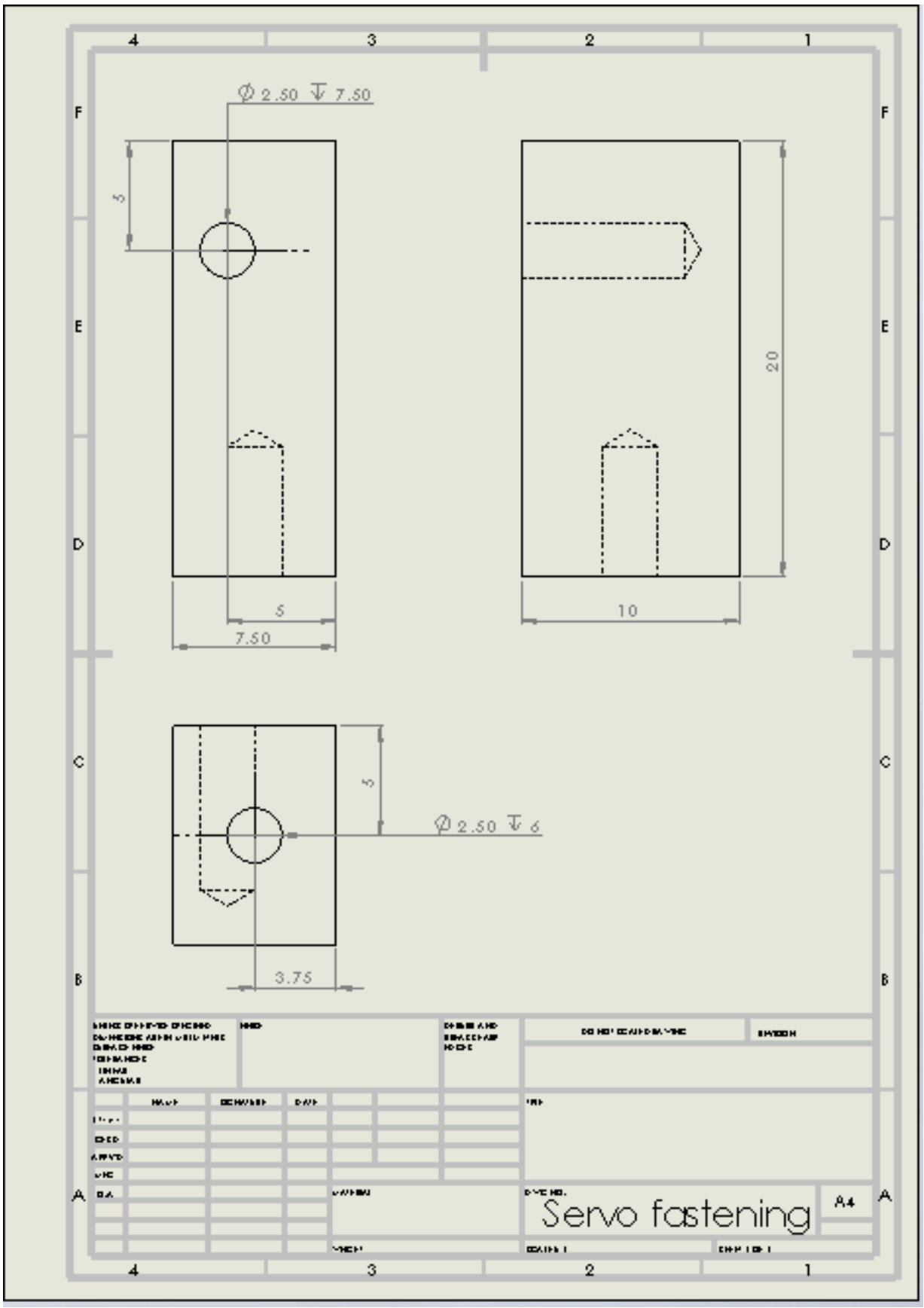
Appendix 3 – Engineering Drawing Examples

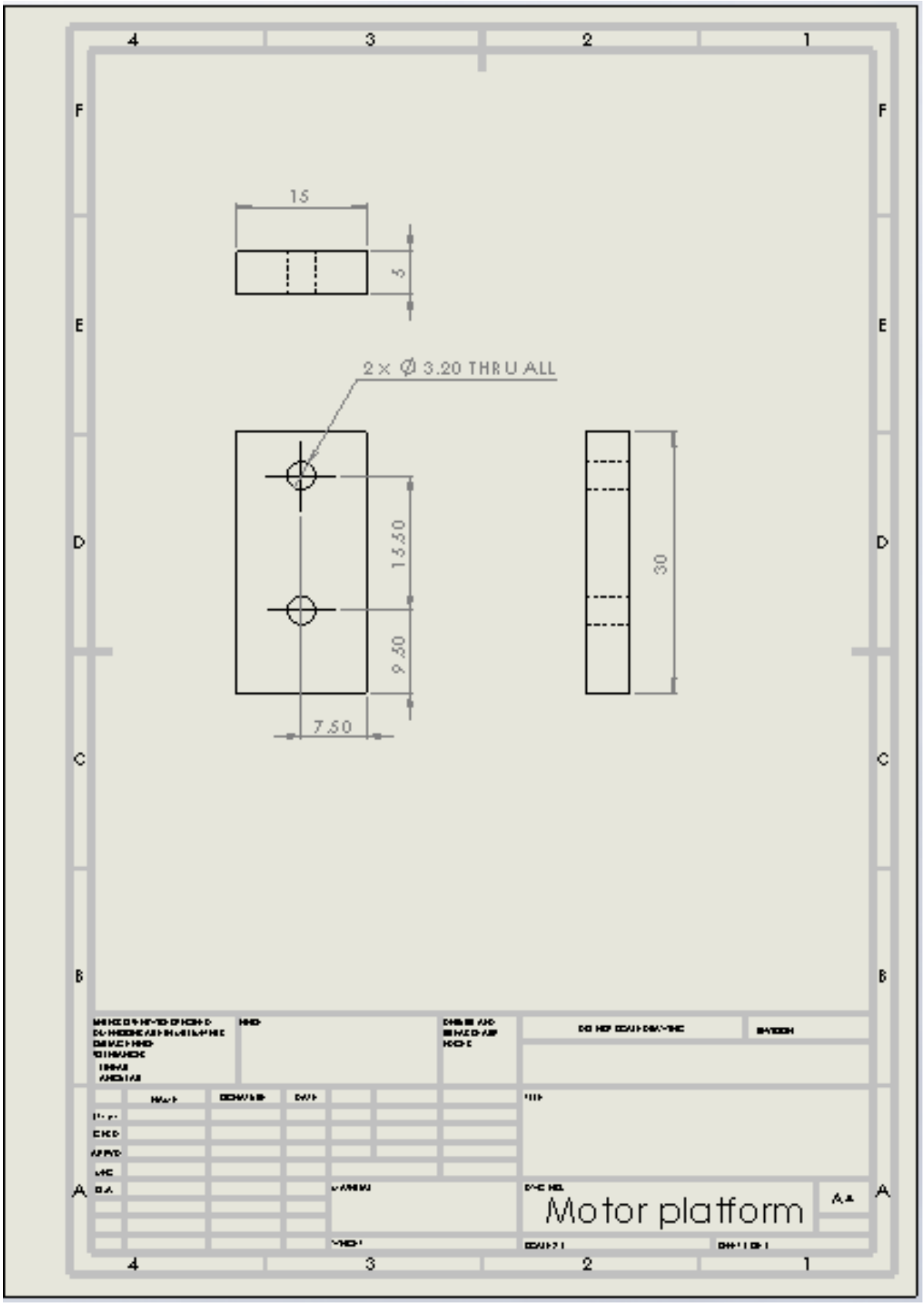


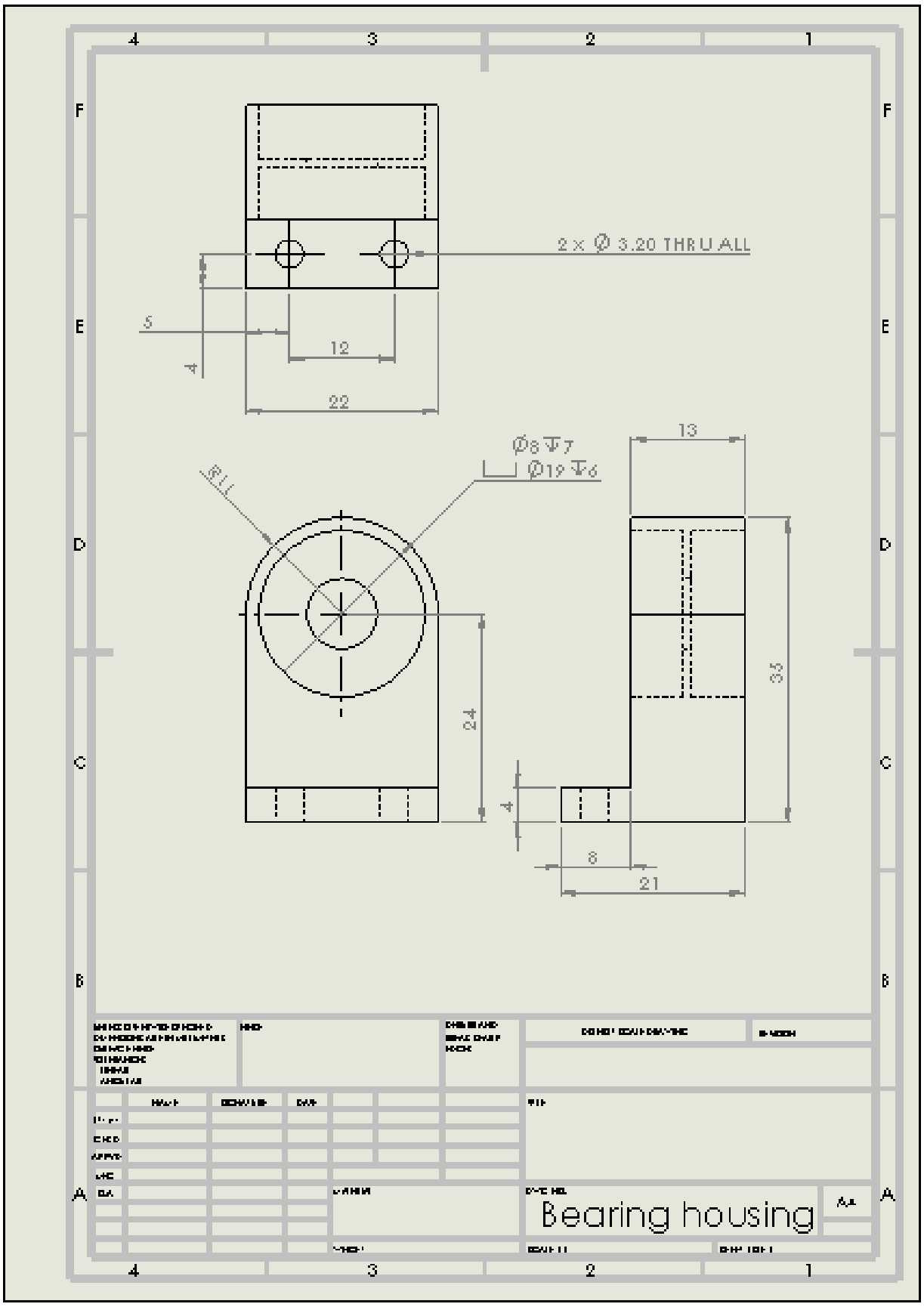

70
Ø10
Ø 5 ↧ 29
Chassis support new
A4

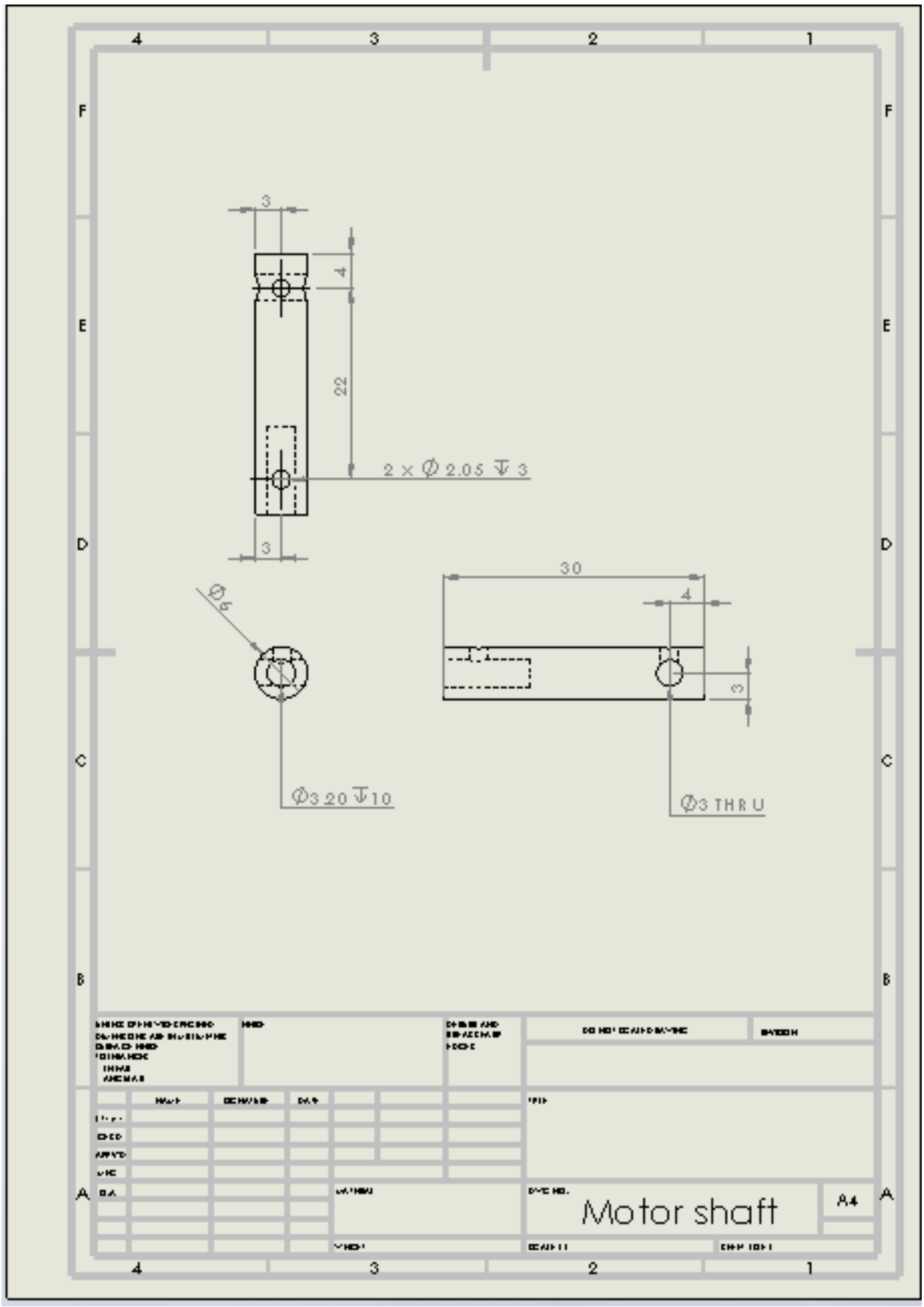

2 × Ø 6.40 THRU ALL
100
6
12
3
112
Start new
A4

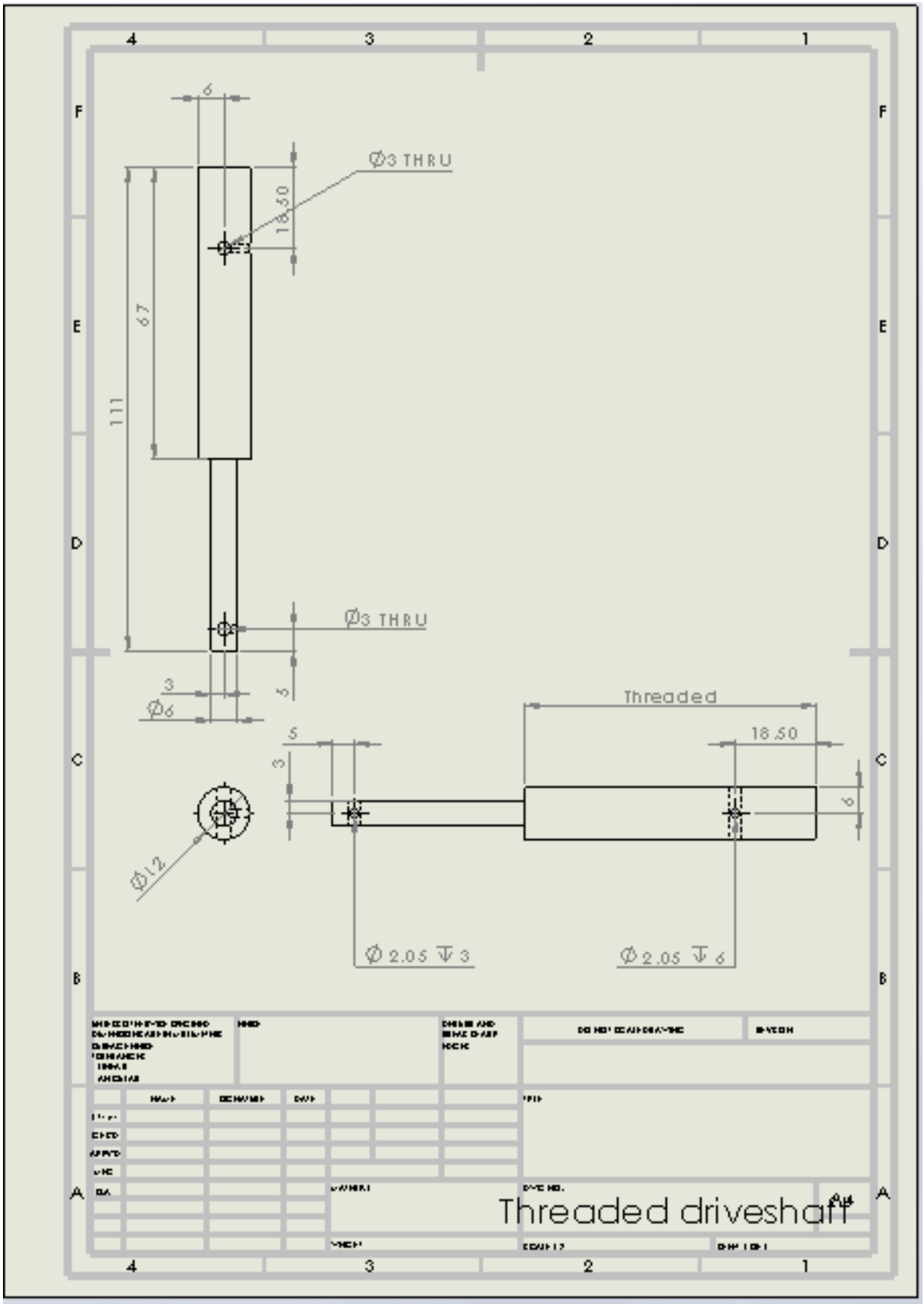

R6
2 × Ø 6.40 THRU ALL
6
85
6
12
Ø 2.50 THRU ALL
6
3
38
97
Second piece new
A4

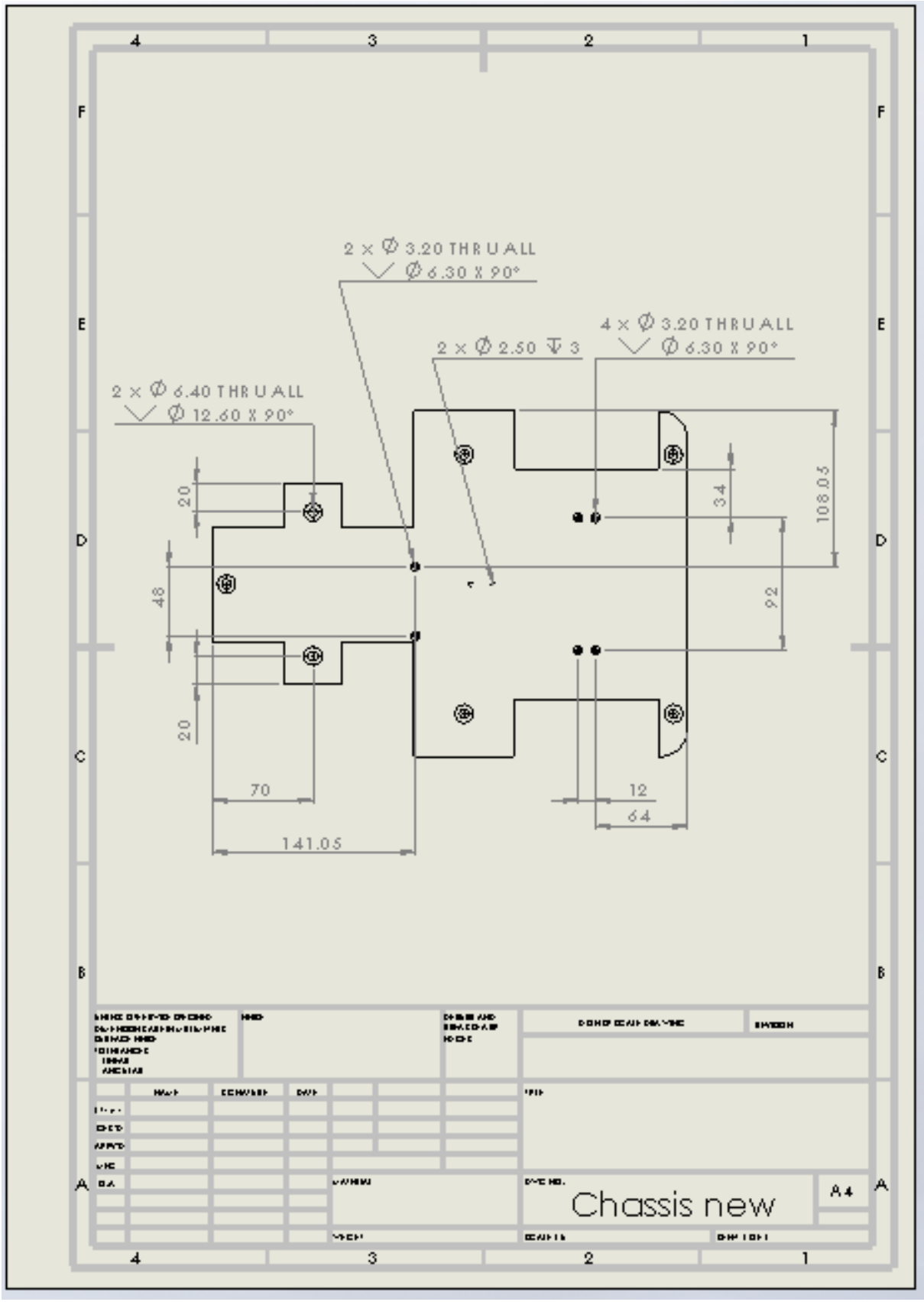

Ø 2.50 ↧ 7.50
5
7.50
20
10
3
Ø 2.50 ↧ 6
3.75
Servo fastening
A4


15
3
2 × Ø 3.20 THRU ALL
13.30
9.50
7.50
30
Motor platform


2 x Ø 3.20 THRU ALL
5
12
4
22
Ø8 ↧7
⌴ Ø19 ↧6
R11
13
24
33
4
8
21
Bearing housing


3
4
22
2 × Ø 2.05 ↧ 3
3
30
4
3
Ø6
Ø3 20 ↧10
Ø3 THRU
Motor shaft
A4


6
Ø3 THRU
18.50
67
111
Ø3 THRU
3
Ø6
3
5
3
Threaded
18.50
6
Ø12
Ø 2.05 ↧ 3
Ø 2.05 ↧ 6
Threaded driveshaft


2 × Ø 3.20 THRU ALL
Ø 6.30 X 90°
4 × Ø 3.20 THRU ALL
Ø 6.30 X 90°
2 × Ø 2.50 ↧ 3
2 × Ø 6.40 THRU ALL
Ø 12.60 X 90°
20
48
20
34
92
108.05
70
141.05
12
64
Chassis new
A4