MANCHESTER 1824
The University of Manchester

School of Electrical and Electronic Engineering

Embedded Systems Project

LINE FOLLOWING BUGGY

Title: Proposal Document

Group Number: 18

Group members name:	ID Number	I confirm that this is the group's own work.
Charlie Shelbourne	9725297	☒
Marcell Tóth	9747325	☒
Thomas Hollis	9563426	☒
Jianli Gao	10079470	☒
Jason Brown	9582307	☒

Tutor: Dr Laith Danoon

Date: 25/11/2016

Table of contents

1. Introduction

The line-following buggy project is an embedded systems design project aimed at challenging multiple teams to construct the best possible buggy to race around a track. The teams must compete in designing the most optimal combination of software and hardware in order to outperform their peers. Extra recognition is given to innovative ideas, clear code documentation and efficient performance. The teams will face each other in a live race around a track with the fastest total time being awarded first prize (risk assessment for this is provided in appendix 8.1).

The main aim of the project is to build an electronic line following buggy that will be capable of navigating itself around a track, dealing with various constraints and obstacles along the way, in order to complete it in the fastest possible time.

To achieve this goal the work was divided in smaller objectives. These include, understanding what the buggy must do and overcome, designing the buggy's hardware and software with relation to what is required of it, building the designed buggy within the budget of £40, testing and adjusting the buggies design and finally racing the buggy successfully on race day.

Two design report documents have already been produced outlining the research and laboratory work carried out to ascertain motor & sensor characterisation as well as a software overview. This allowed our team to produce, among others, a full characterisation of the DC motor used to drive the buggy, a gear box design (appendix 8.2.5), a schematic diagram of the sensor array (appendix 8.2.1) and a layout diagram of the PIC interface connections (appendix 8.2.2). This also led to the development of a full software control algorithm and a high-level architecture image of the buggy, shown in figure 1.1.

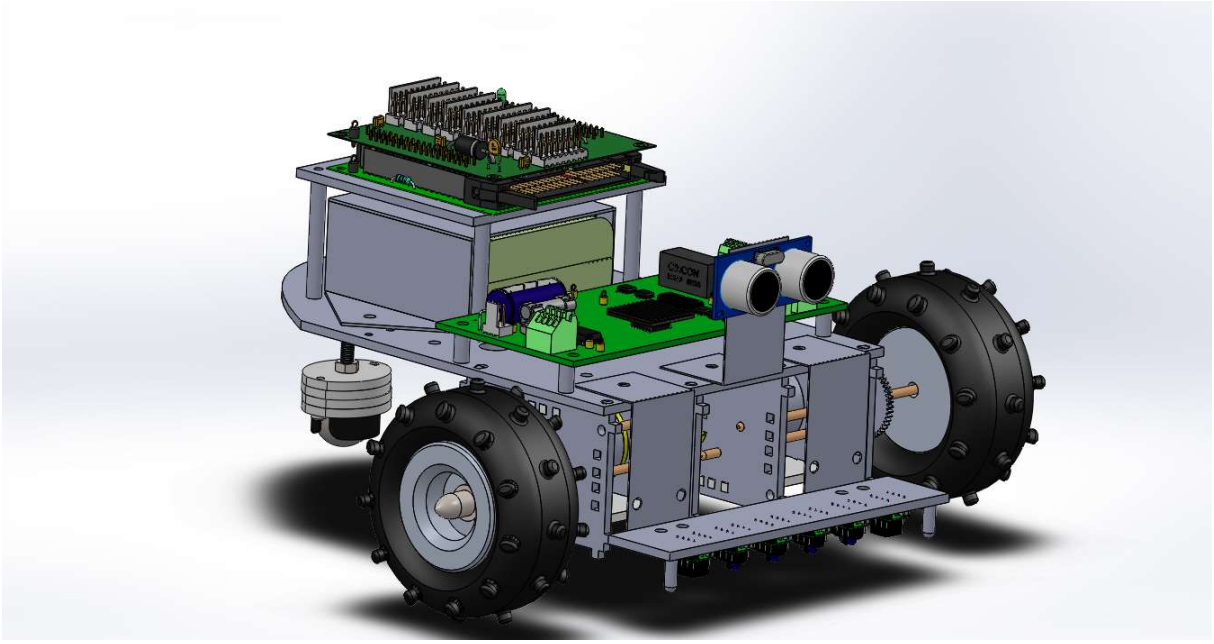


Figure 1.1 - High-level architecture image of buggy

The buggy moves by applying differential PWM[1] for front-wheel steering. The tail of the buggy is reliant on a ball bearing and the entire Acetal chassis base plate supports the wheels, gearbox and other circuit boards. The main data processing is done by a PIC18F8722 located on top of the battery pack at the back of the chassis base plate. It processes all the input signals from the line sensor array, proximity sensor and encoder-disk photo-interrupter device to determine how the buggy is to move. It is programmed in C with a sophisticated dual redundant control algorithm. The PIC has spare IO and comes with a multipurpose IO board that will be used for debugging only. On top of the PIC control board is its corresponding breakout board and in front of it is the motor drive

[1] Pulse Width Modulation

board. Finally, a sensor array board, underneath the proximity sensor, is placed at the front on a hinge to allow it to stay as close to the track as possible and provide reliable data back to the PIC all while remaining flexible for ramp situations.

Design reports #1 and #2 also allowed the team to make lab and simulation based decisions on many different hardware and software features. These are summarised in table 1.1.

Requirements	Implementation & design choices
High top speed	Theoretical maximum speed of 1.6 m/s (but may be limited in practice). Wheels are also to be shaved for optimal performance.
High gearbox efficiency	High mechanical efficiency calculated to be of approx. 72.25%
Follow the white line accurately & avoid line loss	Dual redundant line sensor array chosen composed of 4xTCRT5000, 2xCNY70. Dual redundant algorithm to complement this via a combination of BB[2] and PID[3] control algorithms. Schematic shown in figure 8.2.1 of appendix.
Deal with going up and down slopes	Gearbox designed to enhance torque provided by motor during times where the slope is detected (via encoder disk and photo-interrupter). Gearbox ratio of 15:1 using 16 tooth pinion, 50/10 free running gear and 48/12 press fit gear was chosen as it provides optimal performance given the weight and specifications of our buggy. See appendix 8.2.3.
Deal with breaks in the line	Software implemented with exception coded in to ignore the case where the central TCRT sensor are activated and deactivated rapidly without passing through the side CNY sensors.
Deal with bends of 45°	PID algorithm is used most the time (except in case of errors or emergencies) to provide tightest and most reliable curve fit
Stop 30cm before half way, rotate and return	A front proximity sensor (1x HC-SR04) is used here to detect the end wall and provide a software interrupt to halt the buggy. The interrupt then calls the 180 degree turn function before returning to PID control.
Stay within a given track width	Additional hardware side proximity sensors (2x HC-SR04) are used to provide main function breaks and steer back toward the line when a side wall is detected.
Deal with fast line oscillations	This is implemented purely programmatically by creating another exception in the case where sensors are activated on alternating sides within a fast time period. In this case the PIC instructs constant power to be provided to both wheels until oscillations end. This is also addressed by the positioning of the line following sensors (oriented with the LEDs front-most and the phototransistors rear-most decreasing any lag from detection to the motor's steering).
Stop within 20cm of the end of the start line	This is also implemented purely programmatically: IF buggy on its return journey AND middle sensors are turned off without going past the other sensors AND buggy can no longer find white line by scanning THEN the end function is called and the program is stopped.
Process sensor data appropriately	Sensor interface circuit with the PIC18F8722 shown in appendix 8.2.2. Hardware (ULN)/software buffer used to cope for differences in speed of data acquisition.

Table 1.1 - Summary of requirements and their hardware and software solutions

[2] Bang Bang
[3] Proportional Integral Derivative

The following sections describe the full specification of software and hardware features, team organisation as well as a detailed plan for how and when to assemble the buggy. This will provide a comprehensive self-contained document for any engineer to be able to replicate our work. A full system build, based on this report, is set to occur in the next 12 working weeks before the final race begins. The software is first identified as a functional summary and through multiple diagrams before the hardware overview is presented leading to a conclusion with the project's main innovations and constraints.

2. Software specification

In order for the buggy to operate efficiently and have an edge over its opponents, an efficient, performing and hardware-adapted software must be developed. This is first presented as a functional summary from a statement of its structured requirements.

The buggy will first be switched on via a hardware switch. This will initiate the line following program and power up all subsystems of the buggy. The buggy will proceed to follow the white line and move along the track as dictated to by the control algorithm written to the PIC microcontroller board. The sensor array at the front of the buggy will provide information on the current horizontal position of the buggy in terms of the white line to provide a feedback loop. The buggy will use a combined proportional, integral and differential control system that will bring it to a central position over the line via the application of a power differential between both wheel motors.

This main PID control loop will execute at a maximum speed of 1KHz. If the front sensor array fails to read the white line due any discrepancy in sensors' readings, the PID control system will switch to a secondary failsafe BB control system within the program's dual loop structure. This BB control will use the two outer sensors of the sensor array to feed back the position of the buggy in terms of the white line in a more reliable but less efficient way. If the line is completely lost the buggy will stop and scan for the line, pivoting on the spot. Once the line has been found the program will switch back to the PID control and proceed.

Simultaneously to the line following feedback system, as soon as the buggy is switched on, the motors will drive the buggy forward at a constant speed measured by the use of a speed control subsystem which monitors wheel speeds. This third feedback loop will work to keep the buggy at a constant speed when the buggy is on the move, including on inclines and declines, by adjusting the torque provided.

Once the buggy has reached the end wall of the track, its front proximity sensors will activate a set of instructions within the main program to stop the buggy so that its whole entity is within 30cm of the end wall. The buggy will then be instructed to spin by 180 degrees on the spot and will then be guided by the primary control algorithm mentioned before to make its way back to the beginning of the track. Once back to the beginning of the track, the line following algorithm will detect the final line break and bring the buggy to a halt within 20 cm of the end of the line.

The aforementioned control algorithms are implemented in the buggy's PIC microcontroller software. The software takes the form of a C file (.c) with the corresponding header files (.h) that it relies on. The main line code is written to manage the coordination of the tasks and takes the responsibility of a scheduler. The two main control loops are programmed to be completely dual redundant to allow for a failsafe backup in case one loop fails. This level of software based reliability is clearly shown in the flowchart shown in the following diagram (see figure 2.1).

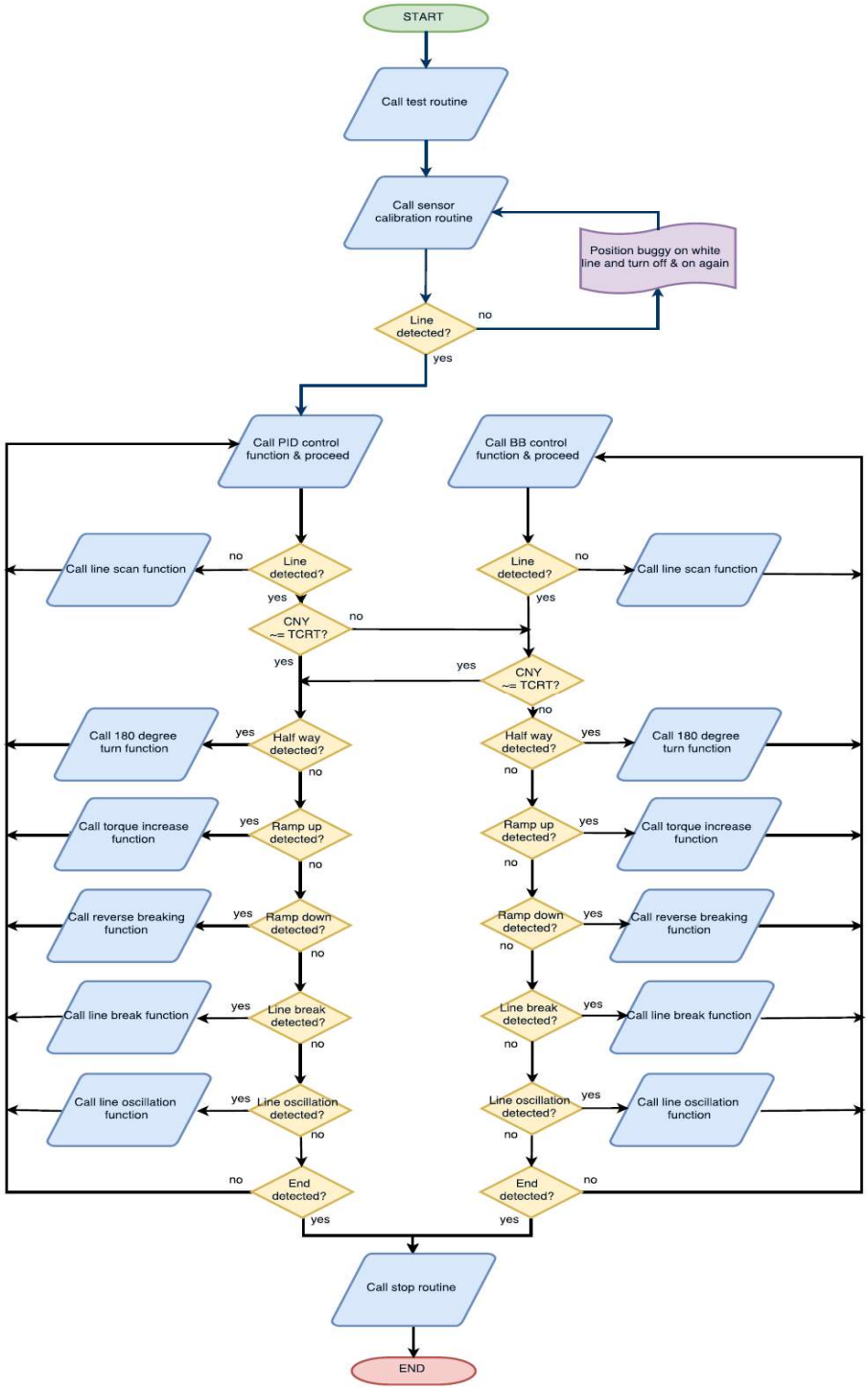


Figure 2.1 - Flowchart of buggy software

The functions and data structures used by the controller are summarised in function prototype form in appendix 8.4. Although it is not explicitly stated in figure 2.1, the end of the line is detected via an interrupt rather than sequentially. This means that whenever the end is detected the main line code is immediately interrupted to execute the stop routine.

However, completing the track also involves a smooth collaboration of hardware with the above software as without it the feedback loop is broken. One approach to analyse how the buggy's software is to interact with hardware is through a context diagram (appendix 8.3.1). This leads to a message table being produced (appendix 8.3.2) as well as a use case diagram that describes the use cases that the buggy will face during the race (as shown in figure 2.2 below).

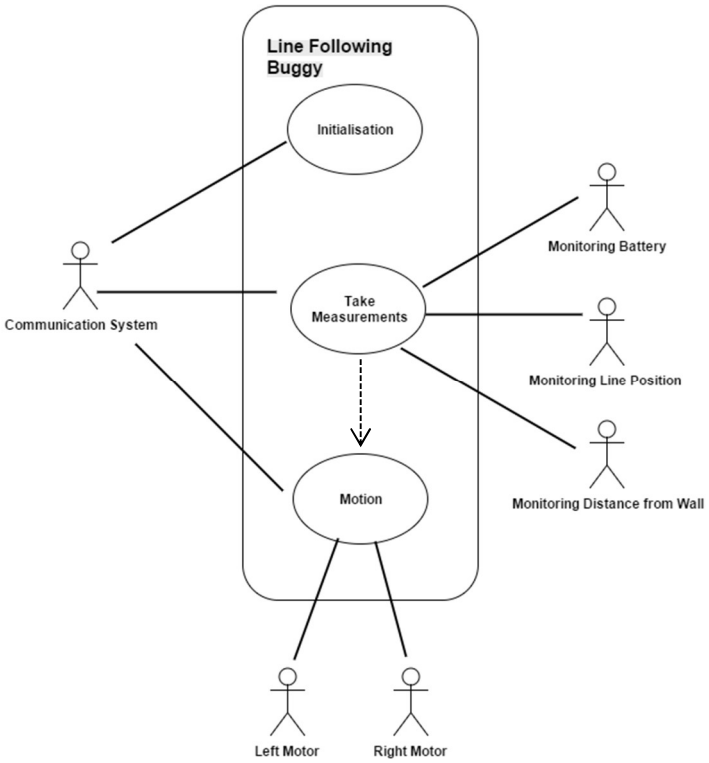


Figure 2.2 - Main Use Case Diagram

Further use case descriptions and a composite diagram are shown in detail in appendix 8.3.3, 8.3.4, 8.3.5, & 8.3.6 respectively.

Each case involves a software and hardware interaction to determine the next case for the buggy to operate within. In the initialisation case, the mechanical switch turns on the buggy, as the PIC and drive boards are connected to the battery powering them. Once the buggy is turned on, the PIC's software starts to interact immediately with its connected devices. The buggy's control algorithm driven by the PIC is responsible for making sure each task executes in a controlled way. First, the PIC starts to collect measurements starting off with the battery power level, line sensors and wall proximity sensors. Once the navigation system is provided with sufficient data to ensure it is calibrated and ready to go, it begins the PID line following algorithm, while the proximity sensors are responsible for keeping a safe distance from the side walls and end wall. Throughout this control, data is fed back to the communication system for further analysis and processing to refine the path taken about the white line. The result of the analysis is then sent to the motor board that drives both the wheels at different speeds to provide steering and navigation about the course and its obstacles.

Another way to examine the software is through the use of an object diagram (see figure 2.3 below).

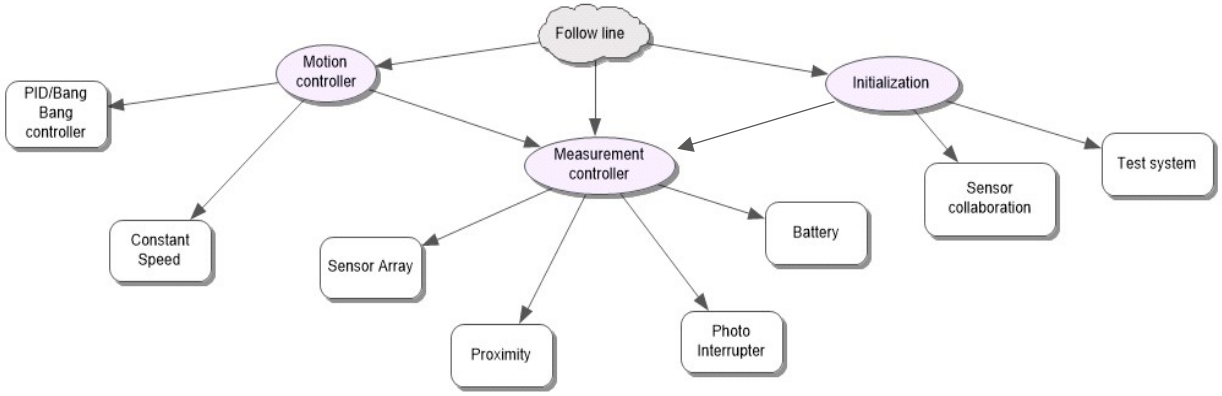


Figure 2.3 - Object diagram

Although C is not an object-oriented language, here an object diagram is used to implement objects as “problem domain and solution domain ‘things’ ”. [4]

Here the object diagram allows us to see that for the line following problem, three solutions are implemented to cope with the challenge: initialisation, motion control and measurement. This matches the use case diagram developed previously but shows all the different objects that can be affected by each solution.

By looking at all the diagrams mentioned previously we can clearly see the interaction between hardware and software. The actors of the system seen in the context diagram will be connected through a breakout board, directly and indirectly, to the I/O pins of the PIC (see appendix 8.2.6). The connections made will allow the software to interact with the hardware of the system. The sensor array connects to the PIC analogue input peripherals, giving the software control over the sensors. A buffer will be implemented within the software to allow for delays caused by the analogue to digital conversion of the data received from the sensors. The data will then be computed and analysed by the software’s control algorithm.

The two motors (actors) are indirectly connected to the PIC through a motor dive board. A bipolar PWM signal is generated in the system’s software via the PIC’s Capture/Compare/PWM modules. The motor control board will also be used as an interface for battery monitoring. For this PortH Pin 3 of the PIC board will be wired to the motor control board which is connected to the battery. A Dallas DS2781 integrated circuit is used to collect data on the voltage and accumulated current of the battery [5]. The data is fed back to the PIC and code within the software of the system will perform the following computation to yield energy used:

$$E = \int v(t)i(t)dt \approx Vb \sum_{k} ik\Delta T \qquad (1)$$

where E is Energy, V_b is voltage across battery, i_k is current samples, T is time. [1]

The photo-interrupter will be linked in to the PIC in the same way as the sensor array. However, it is connected to a digital I/O pin as the signal it gives is digital. A loop within the software will constantly monitor the incoming data and compute the wheel speed. This is implemented within the software by creating Timer0 and Timer1 interrupts and measuring elapsed time.

Finally the proximity sensor actor will be wired to the RB0 pin of the PIC[4]. A RB0 hardware interrupt is then programmed onto the PIC to trigger when the proximity sensor is activated. The interrupt will halt the main line code at any stage to deal with the urgent situation of wall detection.

Throughout the entire software development process, the school’s IO board will be connected to the PIC and used to display debugging data. This is to be removed during race day.

However, the software has some fundamental constraints and limitations:

- The speed of the buggy will be constrained by the maximum frequency of the control loop (approx. 1KHz). The faster the control program can correct error, the faster speeds the buggy will be able to deal with while following the white line.
- Program memory of the PIC will constrain the size of the main line code to 128Kbytes in size allowing 65,536 single-word instructions. The memory type on board is flash memory limiting the amount of programs that can be uploaded at any one time.
- The design of the PID algorithm will be constrained by the stack register memory size. This limits the amount of times functions can be called within eachother to 32.
- Analogue input from sensor constrains the ADC to a 10-bit digital conversion which is further limited to 8 bits due to the 8-bit buses used in the PIC architecture.
- As the microcontroller is an 8-bit MCU with no dedicated floating-point unit, executing floating point arithmetic for the PID control algorithm may be resource intensive and should be minimised where possible.

3. Hardware overview

From DR#1 and DR#2 we have identified the sensors required, their characteristics, a general layout of the buggy, sensor-PIC interaction and the sensor array schematic. This can be seen in appendix 8.2.1 and 8.2.2. The gear ratio has also been defined and a full gearbox design was chosen based on our torque value of 0.231Nm. This is shown in appendix 8.2.5. The navigation strategy was set to a BB/PID dual control algorithm relying on two different types of sensors that are to be placed on a hinged platform at the front of the buggy with the driving wheels.

It is now important to combine all of the information ascertained previously to create a full high level hardware overview for mechanical assembly. Each section of the buggy is merged and combined into a fully buggy architecture design as shown in figure 3.1

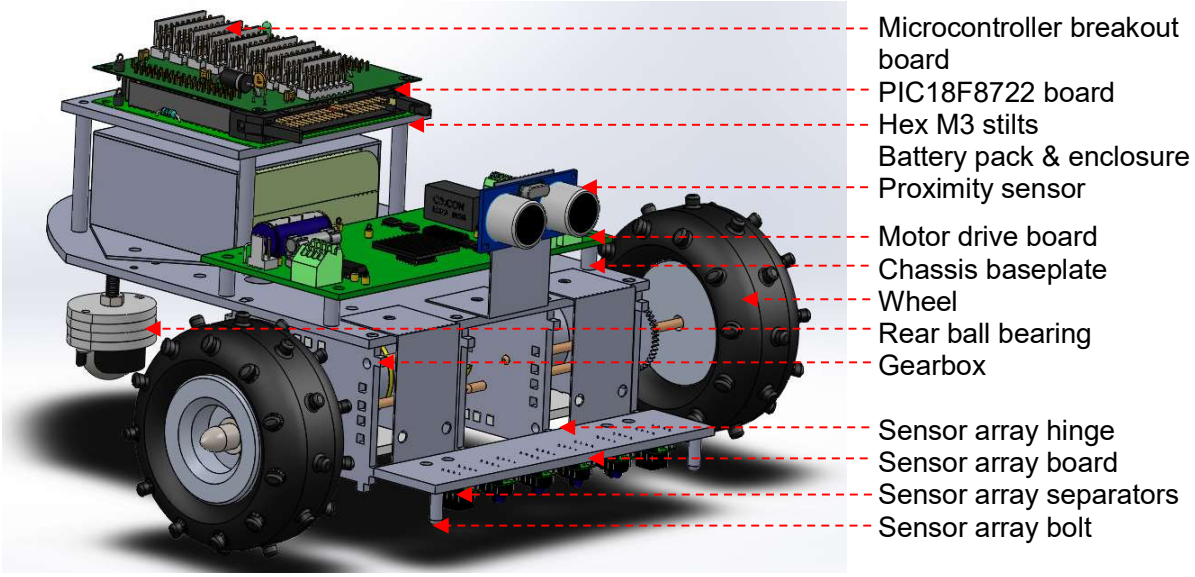


Figure 3.1 - Buggy high-level architecture mechanical model

A panel of acetal is used for the chassis and to hold the PIC microcontroller board. This board is placed in an advantageous position to be able to receive wires from all directions allowing it to communicate to all parts of the buggy in an ordered manner. The microcontroller breakout board will lie directly above it and below both of these boards, the battery pack is fastened by its enclosure to the chassis to use space efficiently.

For the cased CNY70 sensors to be used in the most effective way possible the displacement from the track to the sensor needs to be as close to zero as possible. To implement this strategy a steel L plate hinged component will be used to connect the chassis with the hinge holding the sensor strip. This hinge is used so that the sensors are as close to the surface of the track as possible even during a ramp situation. One of the main design concerns was that the sensors may get damaged in such a layout especially if there is a sudden incline in the track. To prevent this issue a bolt has been fastened to the sensor strip on either side, this bolt is slightly larger than the height of the sensors so that it will mitigate the risks of sensor damage. When completed the buggy will use a front wheel drive transmission system with a ball bearing attached to the back of the chassis. This has been chosen as the line sensors connected to the front of the chassis will allow for the buggy to follow line more effectively minimising sensor lag.

The chassis is a key section of the buggy that must be able to support all the above parts while remaining at an optimal weight and minimal cost. The following chassis base plate design has been developed based on figure 3.1 and is shown in figure 3.2.

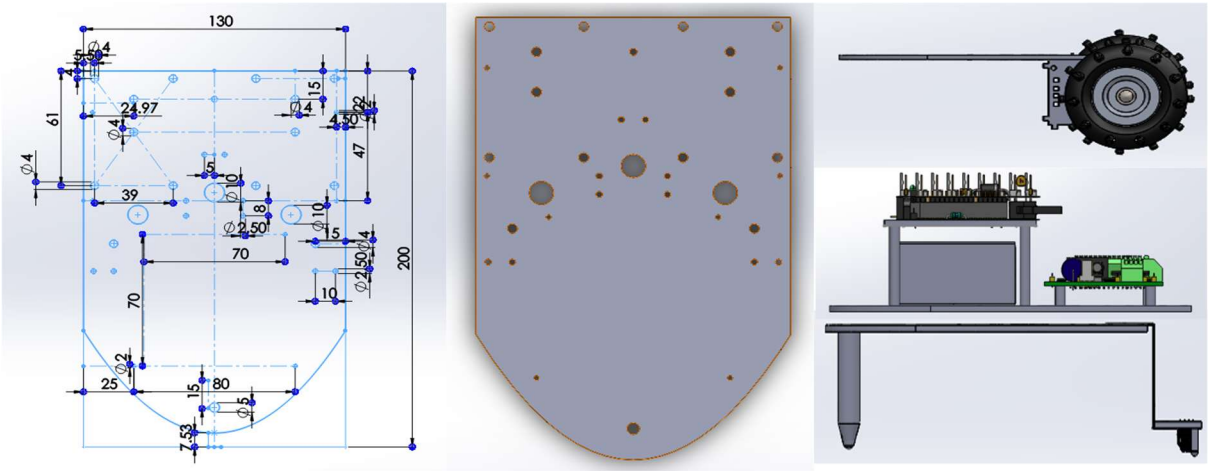


Figure 3.2 - Fully dimensioned layout drawing for chassis base plate

Figure 3.2 shows the layout drawings of the chassis base plate along with measurements and dimensions. The placement of the holes has been chosen so that the surface area of the chassis is used efficiently as well as to prevent the material from bending due to excessive stress. This was done based on data produced via mechanical stress analysis shown in figure 3.3 below.

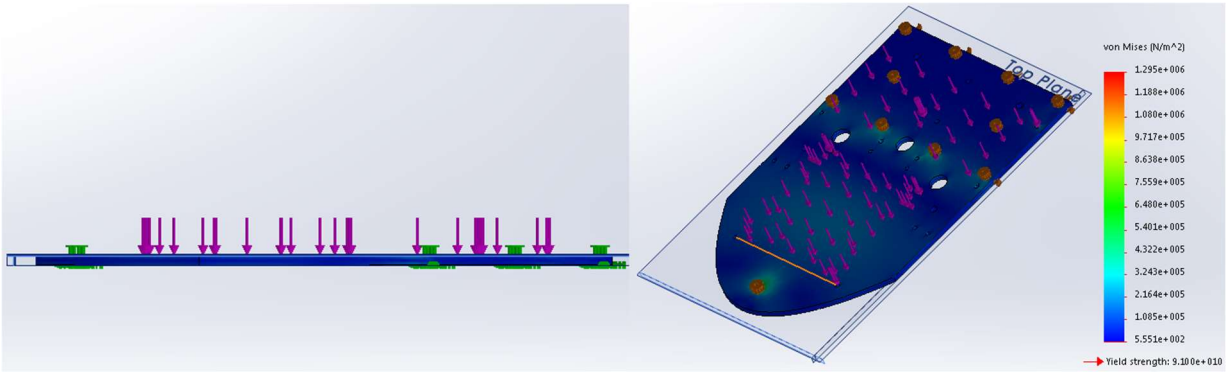


Figure 3.3 - Stress analysis of chassis base plate

As is shown above the force has been applied in the appropriate areas and has been calculated using the mass of the motor driver board, breakout board and battery pack. Acetal is an ideal material in the construction of a buggy as it is lightweight and can effortlessly deal with the force that is acting on it (even for max loads in the green areas of $6.4\mathrm{Nm}^2$). This is aided by the efficient component positioning that distributes weight across the entire board. The weight is slightly shifted to the back ball bearing to minimise stress on the motor wheels.

An important parameter that needed to be determined when designing the chassis was material choice. The main shortlist produced from preliminary research was the following four materials: acetal (GRP[4]), glass-reinforced laminate, aluminium and mild steel. Acetal was chosen for the following reasons as compared with aluminium and mild steel, acetal is much lighter, which means it can keep the buggy weight lower and maximise battery life and vehicle speed. Although acetal is almost as light as glass-reinforced laminate, it is much easier to be cut by the laser, therefore easing the manufacture process. Assuming the chassis supports a distributed load with a mass ($m = 0.408\mathrm{kg}$), length ($l = 192.47\mathrm{mm}$), width ($b = 130\mathrm{mm}$) and thickness ($h = 3mm$), the area moment of inertia of acetal (I) is given by:

$$I = \frac{bh^3}{12} = 2.925\times10^{-10}m^4 \quad (2)$$

[4] Glass Reinforced Plastic

And the deflection per unit mass is given by:

$$x = \frac{5wl^4}{384\,EI} = 1.11\ \text{mm} \tag{3}$$

Equations (2) and (3) are used to calculate the characteristics of all the other materials and the following table is produced:

Characteristic	Acetal GRP	Glass-reinforced laminate	Aluminium	Mild steel
Density (mg/m^3)	1.8	1.9	2.7	7.8
Young's Modulus (GPa)	2.8	11.5	69	207
Deflection per unit mass (mm/kg)	1.110	0.270	0.045	0.015
Price difference (ratio to acetal)	1	3	94	87

Table 3.1 - Mechanical comparison of chassis material shortlist

Simulations (figure 3.3) and calculations (table 3.1) are consistent with each other and show that although acetal is not the strongest material, it is by far strong enough to cope with loads the required, all in the cheapest and most lightweight fashion.

Another important aspect to consider when designing the chassis baseplate is wiring. Wiring of the boards has been implemented such that wires can be passed through three large central holes in the chassis. This allows for a well-organised cable management strategy minimising the risk of parts becoming loose and falling off (one of the major flaws occurring in last year's competitors' designs). Increasing reliability is crucial for the race to remain in the competition and for this reason a full wiring diagram is therefore presented in appendix 8.2.6.

The next section to consider is the gearbox. The following fully dimensioned drawings were produced:

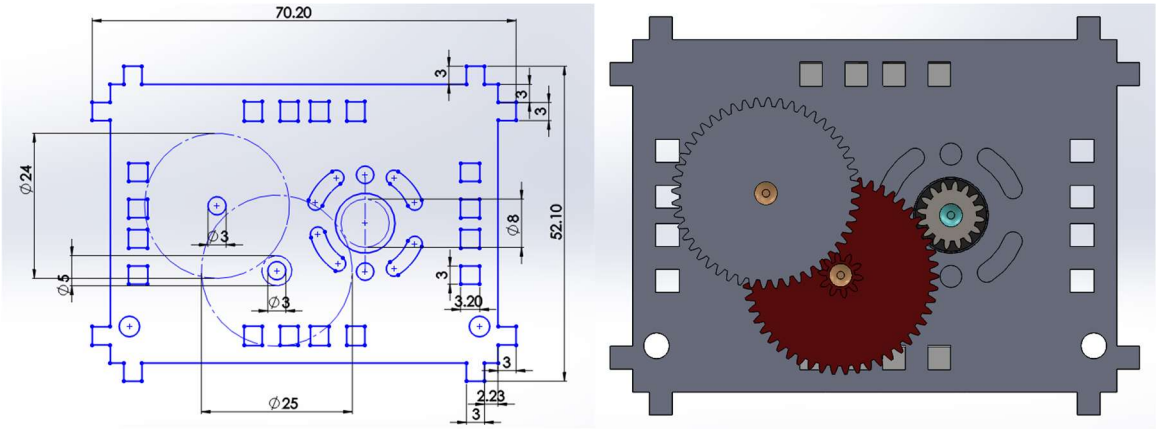


Figure 3.4 - Fully dimensioned drilling diagram of the gearbox plates

The main priority whilst designing the drilling diagram of the gearbox plates was to ensure that the gears would fit within the gearbox as shown in figure 3.3. Whilst designing the gearbox, holes distance measurements are crucial to the overall performance as, if the distances between the holes are slightly incorrect, this could prevent the gears from meshing and mean that the entire gearbox would not function correctly.

Distances used have been taken from Design Report 1, where the pitch circular diameter was calculated between the pinion gear, the free running gear and the press fit gear. As figure 3.4 shows, the sketch function of SolidWorks was used to ensure that the holes were placed at the correct distances so that the gears would mesh successfully before the gearbox is to be assembled.

Finally, in order to assemble the entire system onto the chassis, the following image is produced:

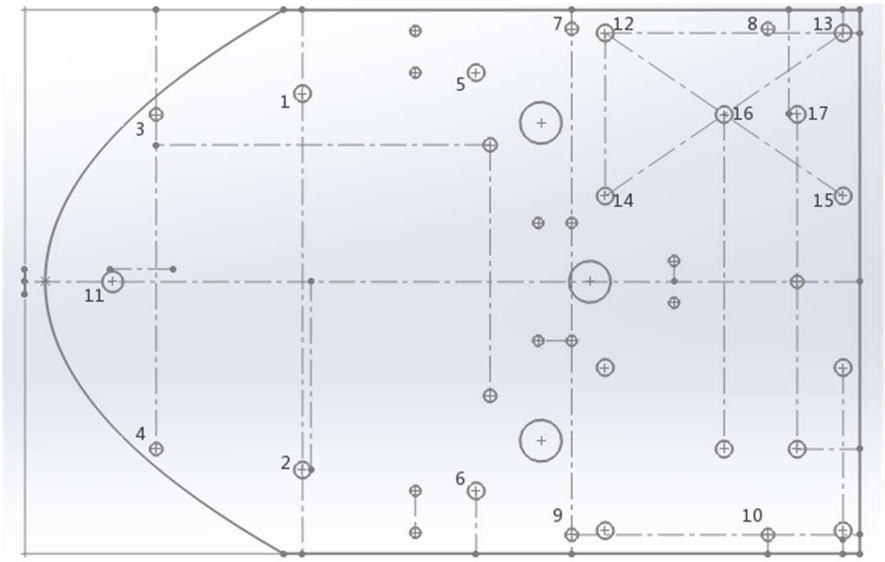


Figure 3.5 - Numbered diagram of chassis mounting holes

In order to assemble this embedded system, the chassis is taken as the fundamental base. The components must first be assembled to the top most part of the chassis plate. Then this must be placed onto the gearbox, wheels and sensor array and finally all the boards must be wired together. The detailed steps are listed below:

1. Put battery holder on back of chassis along dotted lines. Fix with two screws through holes 1 and 2. Insert battery.

2. Insert 4 cylindrical pillars (3cm) into holes 3, 4, 5 and 6. Put and fix microcontroller board on pillars. Screw in breakout board on top.

3. Insert 4 pillars (1cm) into holes 7, 8, 9 and 10. Place the motor drive board on the pillars through the holes and screw in place.

4. Insert the axle into hole 11 and screw it in. Fix a wheel (small) to its end.

5. Place the motor into the bracket box and fix it in place. Insert the gear (16) to the shaft of motor. Fit gear (50/10) and gear (50) to each other. Put the wheel (big) and tyre together and then insert to the shaft through the hole of gear (50). Then insert the entire part to the chassis through holes 12, 13, 14, 15, 16.

6. Repeat step 5 and finish the opposite side of steering section.

7. Connect the sensor base to the bracket boxes and fix it as tightly as possible. Place all the sensors uniformly on their hinged platform. Make sure they are in the right direction. Insert the protection legs to holes on the sensor base.

8. Wire up the microcontroller board, sensor arrays and motor drive board as shown in appendix 8.2.6.

It is important for the buggy's components to be assembled properly and securely to prevent any components from coming loose and causing issues. Part of this challenge is to have an organised wiring connecting all the relevant boards together which has been clearly outlined previously and for which the chassis has been adapted.

The buggy's main hardware limitations are its inability to change direction by more than 60° every 50mm for an angle bends, its maximum speed of 1.6 m/s and a maximum carrying load of 20 N. Values beyond this are theoretically possible but will significantly hinder on the control algorithm's performance and on the speed of the buggy in ramp situations respectively.

4. Team organisation

The team of five was organised, in the first week, into leadership by a team leader, responsible for the overall smooth execution of tasks required to complete the project. The team meets each week to discuss activities of the past week and set deadlines for activities of the week to come. The responsibilities of each team member are uploaded each week by the team leader in the weekly meeting summary to the project's main informal communication channel: the "ESP Project (18)" Facebook group (see figure 4.1). This is where additional ad-hoc project meetings are called.

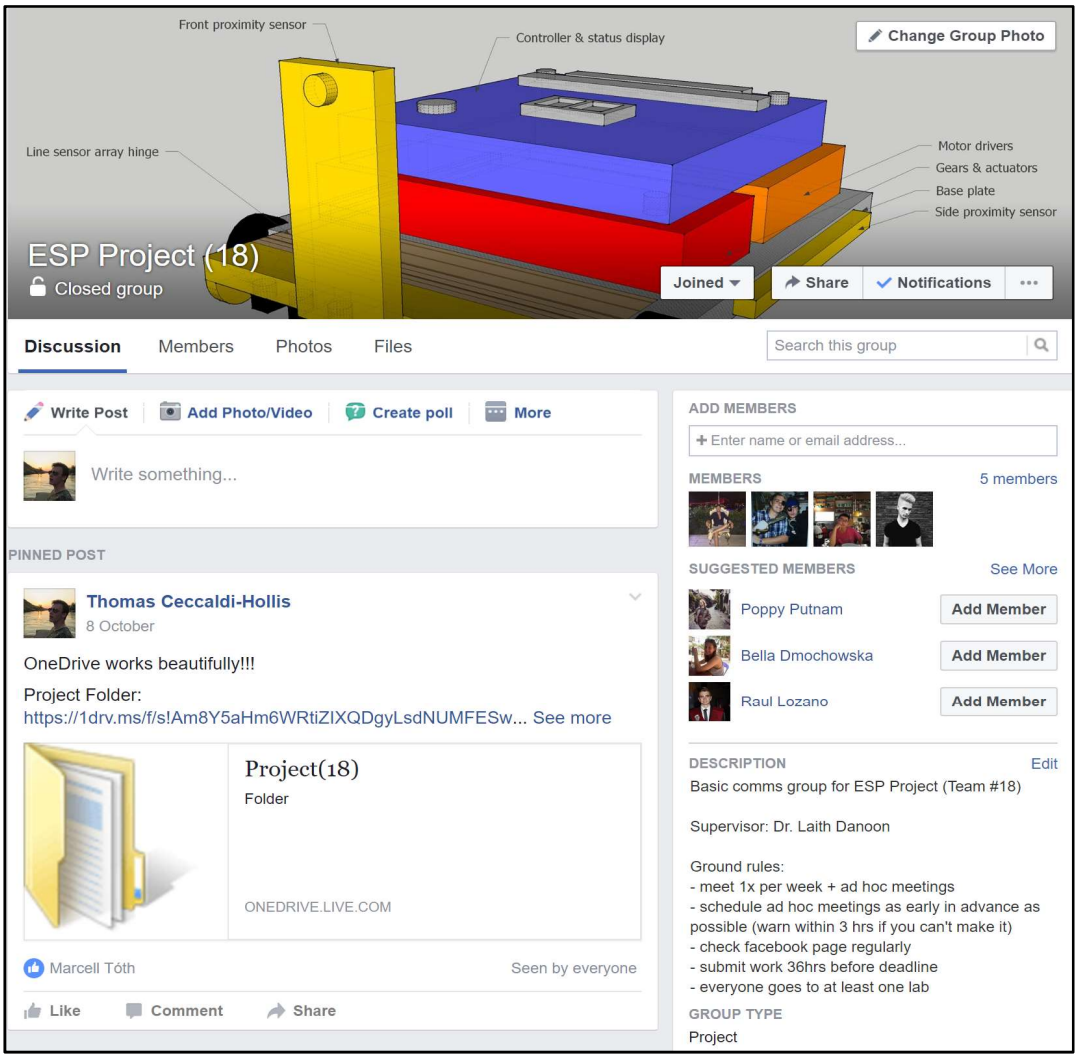


Figure 4.1 - ESP Group (18) Facebook Group

The ground rules are clearly outlined in the group description and team members can easily communicate urgent information to each other via group messages on a network they already use daily. This page is however also used to create polls, for a democratic design decision-making process, and upload certain files for fast feedback (see figure 4.2).

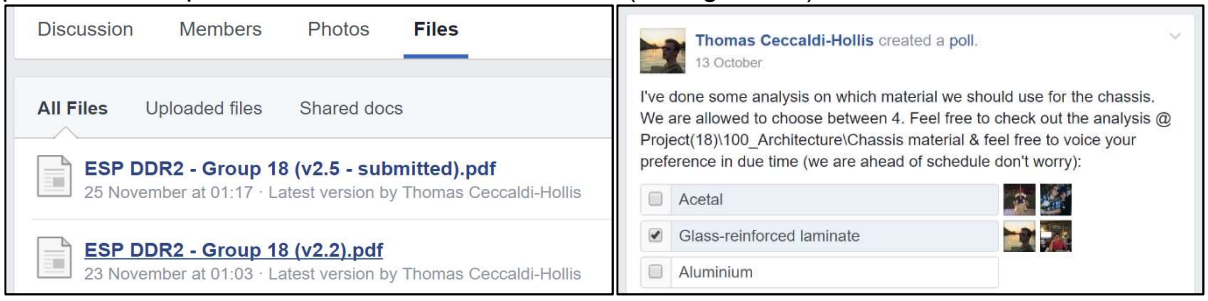


Figure 4.2 - File upload and poll creation on group page

Although some important files, mostly design reports, are uploaded to the group for fast peer review, all files are stored across a multi-platform folder hierarchy accessible natively within the file explorers of all five team members' devices (see figure 4.3 aside).

This system allows for all data to appear on all machines as files are stored both locally and on the cloud. The automatic upload feature allows for full cross-platform synchronisation where team members can populate the project's document portfolio at any time.

Files can be collaborated on in real time and backups are made weekly. Each team member has his own "personal working area" where he can store independent project-related work that he is currently doing, before he moves the completed files to their respective project folder. These project files are also always saved as a new version to prevent accidental deletion or confusion. This personal working area, visible to all, also allows the team leader to verify, easily and fast, that tasks done are of appropriate quality and on time.

In addition, all members keep a log file with a summary of all the work they have done so that a weekly journal can be produced and uploaded by the end of each week for the supervisor to oversee. An example of such a log file is shown in appendix 8.5.1.

Weekly reports are also submitted weekly to the supervisor, as a double-sided page clearly detailing the individual responsibilities and tasks carried out each week (see appendix 8.5.2). This enables him to promptly ensure the work being produced is at a stage he is comfortable with.

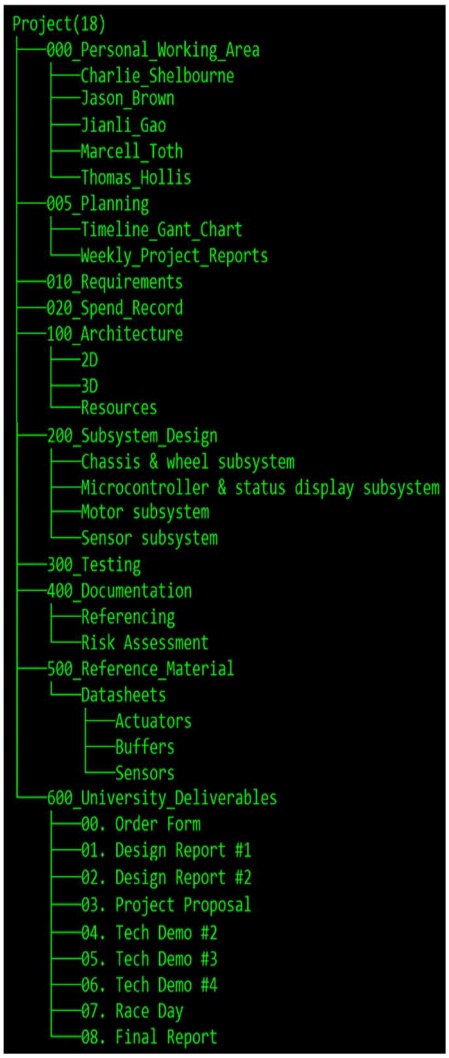


Figure 4.3 - Folder hierarchy

5. Planning and budget

The project was planned based on the its main deliverables and their interdependencies (see figure 5.1). In order to optimise time spent on the project and ensure all team members had a constant flow of work, interdependencies and critical path were analysed to implement maximal pipelining within the plan. Most of the interdependencies are reliant on the project deliverables and ordering components in good time. This analysis was later removed from the Gantt chart for clarity purposes.

It is also important for the plan to be flexible to accommodate changes as all projects are prone to early task completion or delays. In our case, some sections were started earlier than planned (in orange) to ensure a minimum margin of safety for important university deliverables (3rd section).

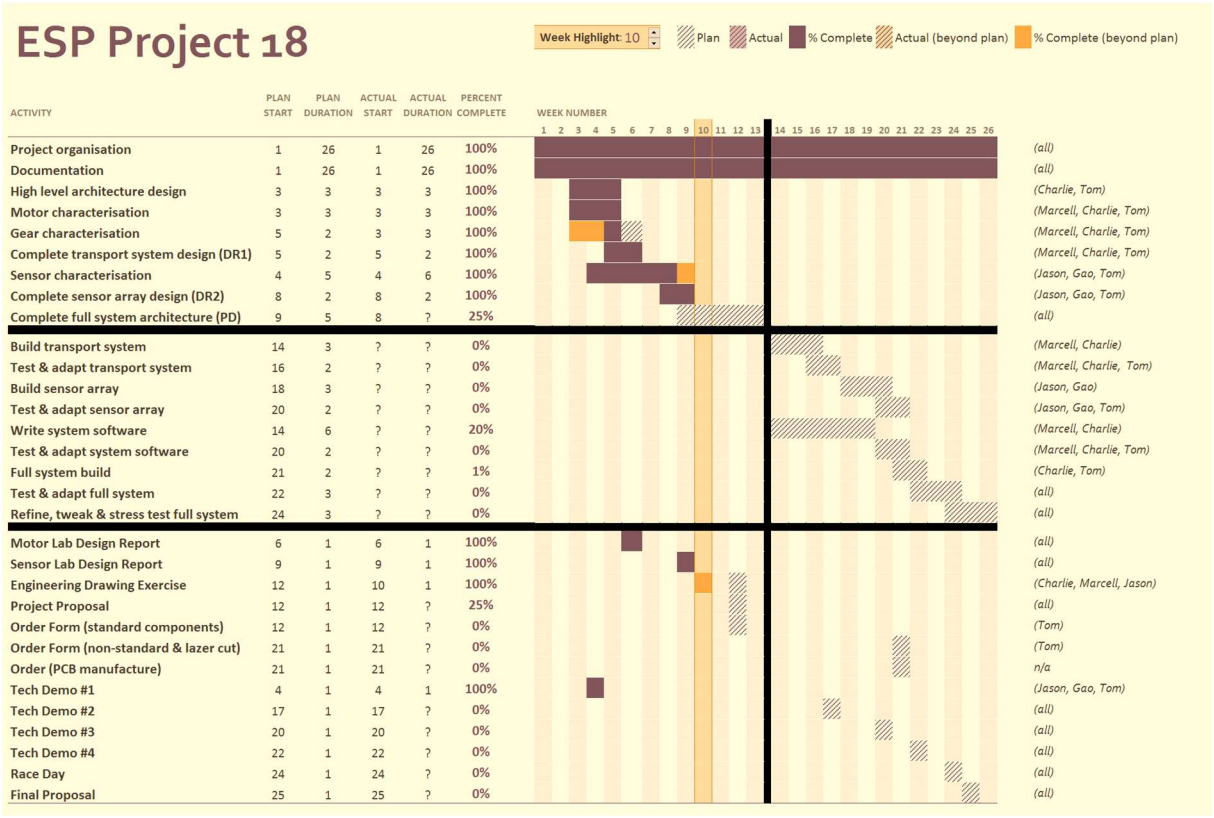


Figure 5.1 - ESP Team Project Plan (Gantt chart)

This plan was used to set strict weekly deadlines for each team member to complete SMART[5] tasks. The amount of manpower allocated to each task is specified and amended as required. Each individual's responsibilities is outlined in their Personalised Project Plan and reviewed by the team leader. These tasks are on occasion amended at the start of the week, during the weekly meetings where the previous week's tasks are also presented. A typical team member's weekly meeting progress presentation, based on his Personalised Project Plan, is shown below (figure 5.2).

Achievements This Week

- Design Report 1 complete and submitted
- Lab 2 Schematic diagram designed
- Lab 2 Layout diagram designed
- Lab 2 preparation excel spreadsheet complete
- Sensor comparison excel spreadsheet (v1.0) written
- Sensor comparison communicated & merged with Jason's work
- Lab 2 schematic & layout diagrams tested with Gao
- Lab 2 complete
- Lab 2 data post processing started & pictures added to folder
- Design report 2 started, marking criteria examined.

Activities Planned for next Week

- Sensor simulations improved with the addition of ULN. Added to DDR2.
- Schematics redrawn to present the full 6 sensor array and added to DDR2.
- Data presentation improved, graphs reformatted, min, max and gradients calculated. Added to DDR2.
- Control algorithm simulations and 3D models produced and added to DDR2.
- Major reworking of the DDR2 report's written analysis and layout. Publication of second draft (v2.0).
- Feedback from all the rest of the team incorporated. Report rewritten to fit the mark scheme even closer (up to v2.5). Final review & submission of DDR2.

Milestones

Milestone	Due Date	Status	Comment
Design Report 1	Week 6	COMPLETE	Final draft complete and submitted
Design Report 2	Week 9	STARTED	Data collected, first draft in writing
Proposal Report	Week 12	IN PROGRESS	High level architecture defined
Testing & documentation phase	n/a	NOT STARTED	Still too early
Final working buggy	n/a	NOT STARTED	Still too early

Red = Predicted Late, Orange = At Risk of delay, Green = On schedule

Short Term Risks / Blockers

- Risk – initial starting inertia
- Blocker – delays in collection of externally sourced sensors
- Risk – ultrasound sensor not part of mark scheme

Figure 5.2 - Weekly meeting progress presentation example

[5] SMART = Specific, Measurable, Attainable, Relevant/Realistic & Time-Bound

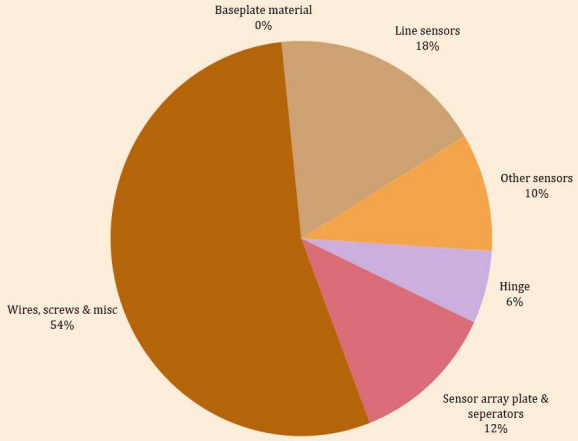
A risk assessment was also developed to ensure the project's race day is undertaken in a safe environment where risks are mitigated as much as possible. This is shown in appendix 8.1.

The budget was managed using spreadsheets and a summary pie chart. The full cost breakdown is first calculated in a table and automatically linked to update the summary pie chart to ensure strategic design choices are realistic and feasible. This also allows budget optimisation.

Embedded Systems Project (Team 18)
December 15, 2017 Budget Available: £40.00

ESP Budget Summary

ITEM	ESTIMATED	ACTUAL	OVER/UNDER
Baseplate material	5.00	0.00	5.00
Baseplace machining	2.50	0.00	2.50
Line sensors	5.00	5.53	-0.53
Other sensors	10.00	2.97	7.03
Hinge	2.50	1.83	0.67
Sensor array plate & seperators	2.50	3.74	-1.24
Wheels	2.50	0.00	2.50
Gearbox	5.00	0.00	5.00
Motors	5.00	0.00	5.00
Wires, screws & misc	0.00	16.64	-16.64
Total Expenses	**40.00**	**30.71**	**9.29**



Baseplate material

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Aluminium (15x20cm)	1.98	0.00	1.98
Acetal (500x250x3mm)	12.02	0.00	12.02
Baseplate Total	**14.00**	**0.00**	**14.00**

Baseplate machining

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Machining costs	2.50	0.00	2.50
Reception Total	**2.50**	**0.00**	**2.50**

Line sensors

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Vishay TCRT5000 (6pcs)	5.90	3.56	2.34
Vishay CNY70 (3pcs)	3.05	1.97	1.08
Line Sensor Total	**8.95**	**5.53**	**3.42**

Other sensors

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
HC-SR04 (3 pcs)	1.30	1.30	0.00
TEKT5400S NPN phototransistor	0.00	0.72	-0.72
TSK5400S IR emiter	0.00	0.95	-0.95
Proximity Sensor Total	**1.30**	**2.97**	**-1.67**

Hinge

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Hinge part (2 pcs)	2.50	1.83	0.67
Hinge tape	0.00	0.00	0.00
Hinge Total	**2.50**	**1.83**	**0.67**

Sensor array plate

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Hinges (2pcs)	2.50	1.83	0.67
Seperators (5 pcs)	0.00	0.00	0.00
Caster ball	0.00	1.91	-1.91
Decorations Total	**2.50**	**3.74**	**-1.24**

Wheels

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Wheels (2 pcs)	2.50	0.00	2.50
Flowers Total	**2.50**	**0.00**	**2.50**

Gearbox

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Gearbox + plate + gears + brackets	5.00	0.00	5.00
Gifts Total	**5.00**	**0.00**	**5.00**

Motors

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
Motor (2 pcs)	5.00	0.00	5.00
Axle	0.00	0.00	0.00
Battery	0.00	0.00	0.00
Travel/Transportation Total	**5.00**	**0.00**	**5.00**

Other Expenses

CATEGORY	ESTIMATED	ACTUAL	OVER/UNDER
M3 Shake proof washers (100 pcs)	0.00	0.73	-0.73
M4 Shakeproof washers (100 pcs)	0.00	0.51	-0.51
M4 x10 Screw Rapid (100 pcs)	0.00	1.04	-1.04
M3 x10 Screw Rapid 33-2305 (100 pcs)	0.00	0.89	-0.89
Carriage bolts (2 pcs)	0.00	0.48	-0.48
Spacers 18mm 3mm (10 pcs)	0.00	2.49	-2.49
Steel threaded rod (d=6mm, l=120mm)	0.00	0.94	-0.94
M3 Nuts Rapid 33-1710 (100 pcs)	0.00	0.57	-0.57
Resistor multipack	0.00	0.34	-0.34
8 core Cable 5m	0.00	2.66	-2.66
Cable ties	0.00	0.57	-0.57
Heat shrink	0.00	3.60	-3.60
Rocker switches	0.00	1.82	-1.82
Other Expenses Total	**0.00**	**16.64**	**-16.64**

Figure 5.3 - Budget & cost breakdown

Other than standard components, the main non-standard parts chosen were line sensors, proximity sensors, speed sensors and an entire hinge section that the dual-redundant design heavily relies upon. These additional parts cost £30.71 and are well within our £40 budget with £10 to spare in case redesign is required. Spares of certain sensors were also purchased in case of emergency replacements being required. The order forms for mechanical and electronic parts have already been placed and construction is the next task to overcome.

6. Conclusions

Having developed a full hardware, software and team overview of the ESP buggy project, this proposal aims to reveal that the building phase of the project is now ready to commence. It also allows for the verification that all the risks have been carefully planned for to avoid any time loss, design error and health and safety hazards that may occur (see appendix 8.1).

Having already determined sensor positioning, material choice, stress analysis, gear ratio, sensor selection and many other design features from previous design reports, this document details the full specifications of hardware and software of our embedded systems project.

The main bulk of the budget has been spent on the two different types of sensors: one shielded and one unshielded mounted on a front-facing hinge. This allows for one of the projects main innovations: a dual redundant line sensor detection system that sticks closely to the floor. If one type of sensor fails due to interference or any other error the buggy can still continue the race preventing itself from being knocked out of the competition. This extra element of reliability is important in a race where other teams are faced with reliability issues as the buggy's capabilities are tested through various obstacles.

This hardware innovation is directly linked to the main software innovation which implements a failsafe bi-stable software loop that prevents the PIC's control algorithm from failing completely in the case of errors being detected. While this helps reliability it also allows the PIC's control algorithm to push the buggy's performance to its limits without the fear that faster speeds could open the buggy up to unpredictable behaviour.

The software system will comprise eleven objects, of which three will act as controllers coordinating the other objects and their functions. The main software limitations of the buggy is that the PIC has a finite data acquisition and processing rate when it comes to floating point arithmetic and although the sensors may be capable of reading at a very high frequency, the software must be designed concisely to be able to match this speed. The program must also be written efficiently for memory conservation purposes as the PIC also has a limited memory to store a finite number of instructions.

From a hardware perspective, the buggy is limited by its mass and power. While a larger battery could provide more power, it would mean a significant size increase making the buggy more difficult to navigate about the track's obstacles. The chassis is not prone to failure via stress however it is important during assembly all the parts are bolted down securely to avoid any uncertainty in robustness of the system.

7. References

[1] Podd, F, Apsley, J, *Embedded Systems Project: Technical and Management Manual*, School of Electrical and Electronic Engineering, The University of Manchester, Sept 2016.

[2] Podd, F, Apsley, J, *Embedded Systems Project: Project Handbook*, School of Electrical and Electronic Engineering, The University of Manchester, Sept 2016.

[3] J, Apsley, Embedded systems Project 2013 – Updated current calibration. School of Electrical and Electronic Engineering, The University of Manchester, Sept 2016.

[4] P, Green, *Microcontroller Engineering 2 Lecture notes*. School of Electrical and Electronic Engineering, The University of Manchester, Sept 2016.

[5] Microchip, *PIC18F8722 Family Data Sheet*. pg 8. Available at: www.microchip.com/downloads/en/DeviceDoc/39646c.pdf (Accessed: 14.12.2016)

8. Appendices

8.1 Risk assessment

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUPS)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	WITH EXISTING CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE?
Any activity during the race involving electrical equipment or buggy motors running.	Fire	Evacuation, minor burns, destruction of laboratory equipment.	Anyone (up to 240 persons) present during race day.	CO2 fire extinguishers available at entrance to lab. Fire alarm point in adjacent corridor. Personnel & students appropriately trained.	2	2	4	Y
Any activity within the room involving students walking.	Tripping	Head injury, bruises, spills.	Anyone (up to 240 persons) present during race day.	Work areas kept tidy. Cables not left on ground (ceiling wiring) bags and coats put away.	1	2	2	Y
Running buggies around the track	Hot motor	Minor burns	Team members demonstrating their buggy.	Students are briefed to avoid hot motors and turn off buggies in case of overheating until it cools down.	3	2	6	Y
Electrical use, supplying power to the buggies and buggy touching during the race.	Electric shock from power supplies and faulty buggies.	Pain, discomfort & muscular contraction that could lead to spills. Interference with pacemakers.	Anyone (up to 240 persons) present during race day.	All equipment in compliance with IEEE EST regulation and frequently tested. Students warned and trained. HV buggies not allowed.	1	2	2	Y
Running of mechanical buggies and their rotational motion (base plates, chassis…)	Getting hands & hair stuck in buggies. Buggy collisions.	Bodily harm (minor or major cuts, bruises, hair damage)	Team members demonstrating their buggy.	Minimum margin of safety distance between track and team members. Hair must be tied up. Maximum torque of ~10mNm too small to cause injuries	2	2	4	Y
Any activity within the room involving fluids.	Spills & electrical damage	Material damaged by water. Risk of electrocution increased.	Anyone (up to 240 persons) present during race day.	Absolutely no food or drink permitted within the lab. All bottles closed when not in use. Absorbent paper.	1	2	2	Y
Buggies out of control	Broken parts flying off the track	Bodily harm (minor or major cuts, bruises, hair damage)	Anyone (up to 240 persons) present during race day.	Walls around the track built to prevent debris from reaching the audience	1	1	2	Y
Leaking batteries	Chemical contamination	Chemical burns, intoxication by inhalation	Anyone (up to 240 persons) present during race day.	New batteries used only at small voltages	1	1	3	Y

Assessment ID Number: EEN21000 **Activity:** Embedded Systems Project Race (2nd Year) **Location:** Renold_C9, The University of Manchester, UK.

				THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (MAX 12 MTHS)
MANAGER/SUPERVISOR	**NAME: Dr Laith Danoon**	**SIGNED:**	**DATE:**	
STUDENT	**NAME: Thomas Hollis**	**SIGNED:** 	**DATE: 26/11/16**	

IF THE ANSWERS TO ANY OF THE QUESTIONS BELOW IS YES THEN ADDITIONAL SPECIFIC RISK ASSESSMENTS MAY BE REQUIRED.

IS THERE A RISK OF FIRE?	Y	DOES THE ACTIVITY REQUIRE ANY HOME WORKING?	Y
ARE SUBSTANCES THAT ARE HAZARDOUS TO HEALTH USED?	N	ARE THE EMPLOYEES REQUIRED TO WORK ALONE	Y
IS THERE MANUAL HANDLING INVOLVED?	Y	DOES THE ACTIVITY INVOLVE DRIVING	N
IS PPE WORN OR REQUIRED TO BE WORN?	Y	DOES THE ACTIVITY REQUIRE WORK AT HEIGHT	N
ARE DISPLAY SCREENS USED?	Y	DOES THE ACTIVITY INVOLVE FOREIGN TRAVEL	N
IS THERE A SIGNIFICANT RISK TO YOUNG PERSONS?	N	IS THERE A SIGNIFICANT RISK TO NEW / PREGNANT MOTHERS?	N

***Severity value* = potential consequence of an incident/injury**

5	Very High	Death / permanent incapacity / widespread loss
4	High	Major Injury (Reportable Category) / Severe Incapacity / Serious Loss
3	Moderate	Injury / illness of 3 days or more absence (reportable category) / Moderate loss
2	Slight	Minor injury / illness – immediate First Aid only / slight loss
1	Negligible	No injury or trivial injury / illness / loss

***Likelihood value* = what is the potential of an incident or injury occurring**

5	Almost certain to occur
4	Likely to occur
3	Quite possible to occur
2	Possible in current situation
1	Not likely to occur

risk rating* = *severity value* × *likelihood value

***risk ratings* are classified as *low* (1 – 5), *medium* (6 – 9) and *high* (10 – 25)**

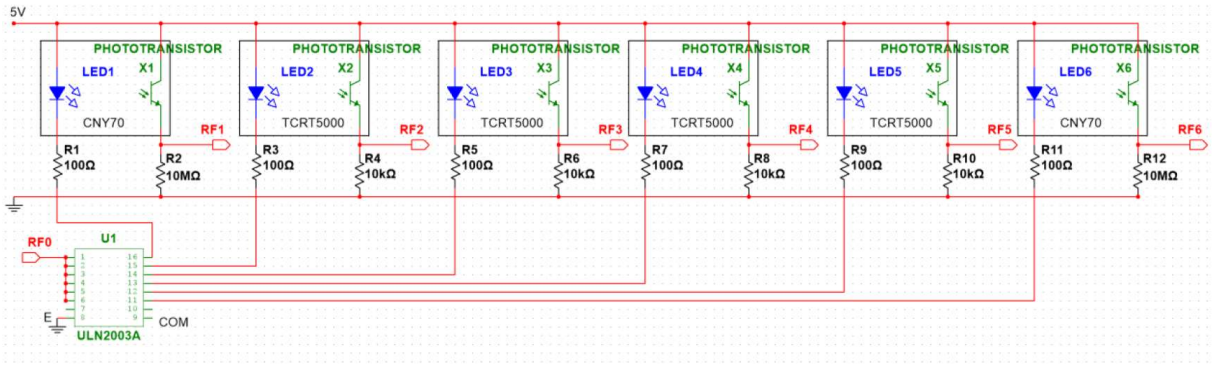
Risk Classification and Actions:

Rating	Classification	Action
1 – 5	Low	Tolerable risk - Monitor and Manage
6 – 9	Medium	Review and introduce additional controls to mitigate to "As Low As Reasonably Practicable" (ALARP)
10 – 25	High	Stop work immediately and introduce further control measures

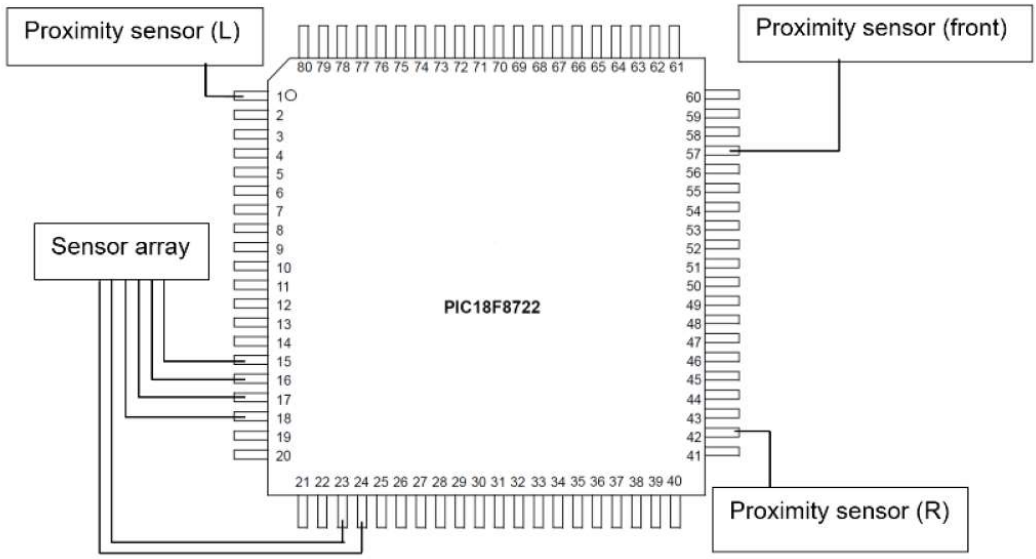
SEVERITY

LIKELIHOOD	1	2	3	4	5
1	Low	Low	Low	Low	Low
2	Low	Low	Medium	Medium	High
3	Low	Medium	Medium	High	High
4	Low	Medium	High	High	High
5	Low	High	High	High	High

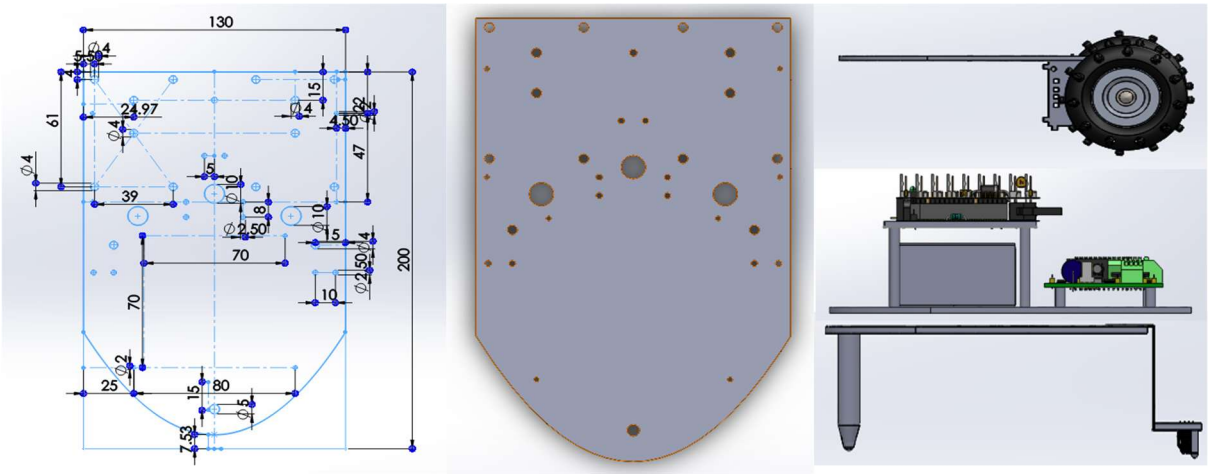
8.2 Engineering drawings



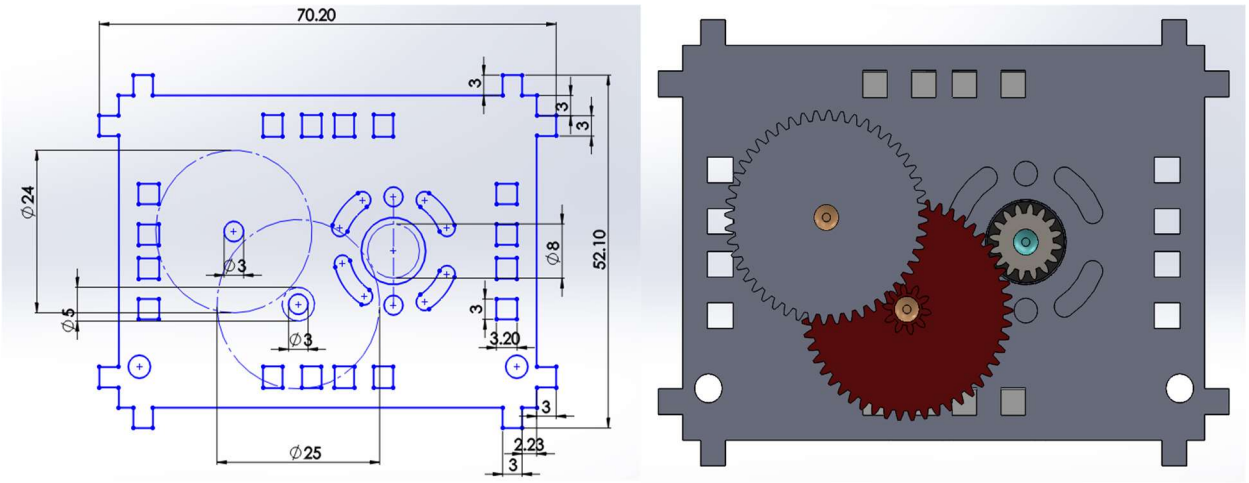
Appendix 8.2.1 - Sensor array schematic



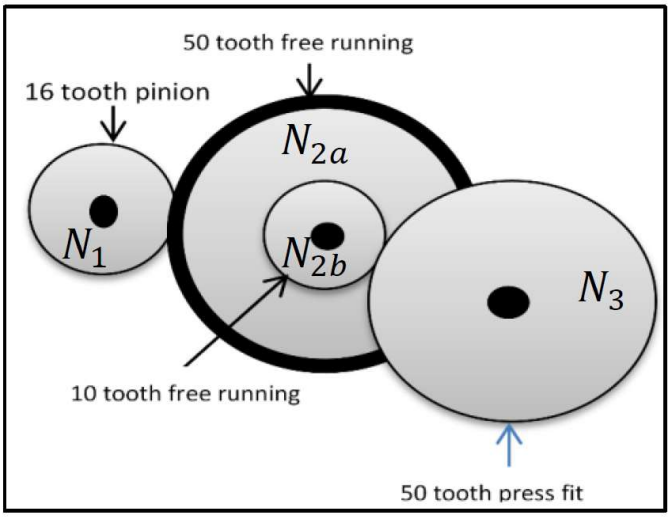
Appendix 8.2.2 - Sensor interface circuit with the PIC18F8722



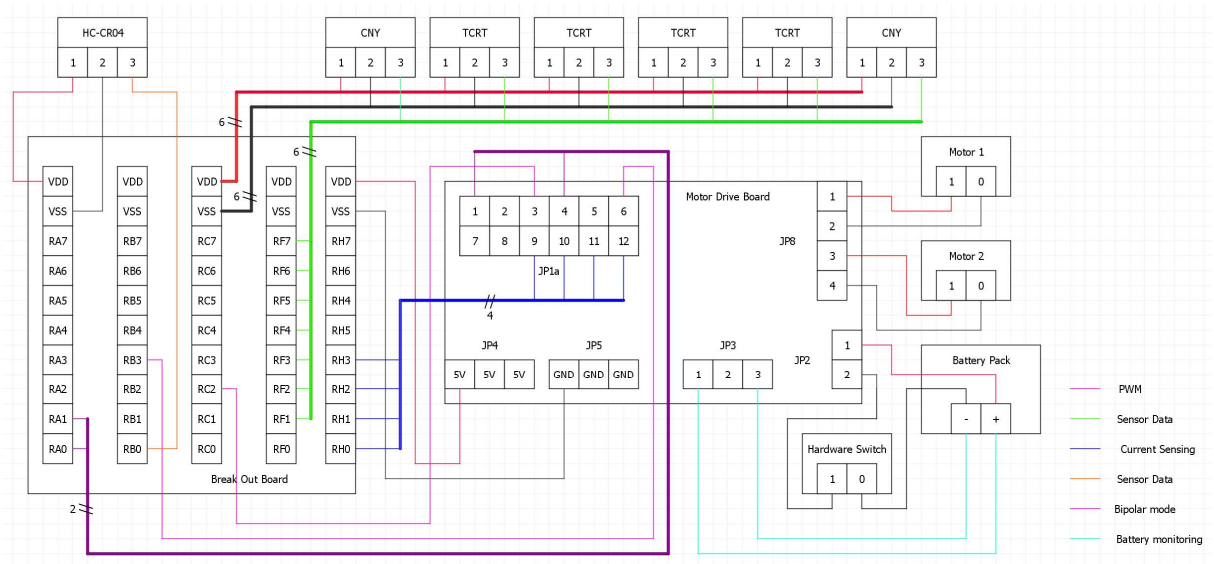
Appendix 8.2.3 - Fully dimensioned chassis



Appendix 8.2.4 - Fully dimensioned gearbox

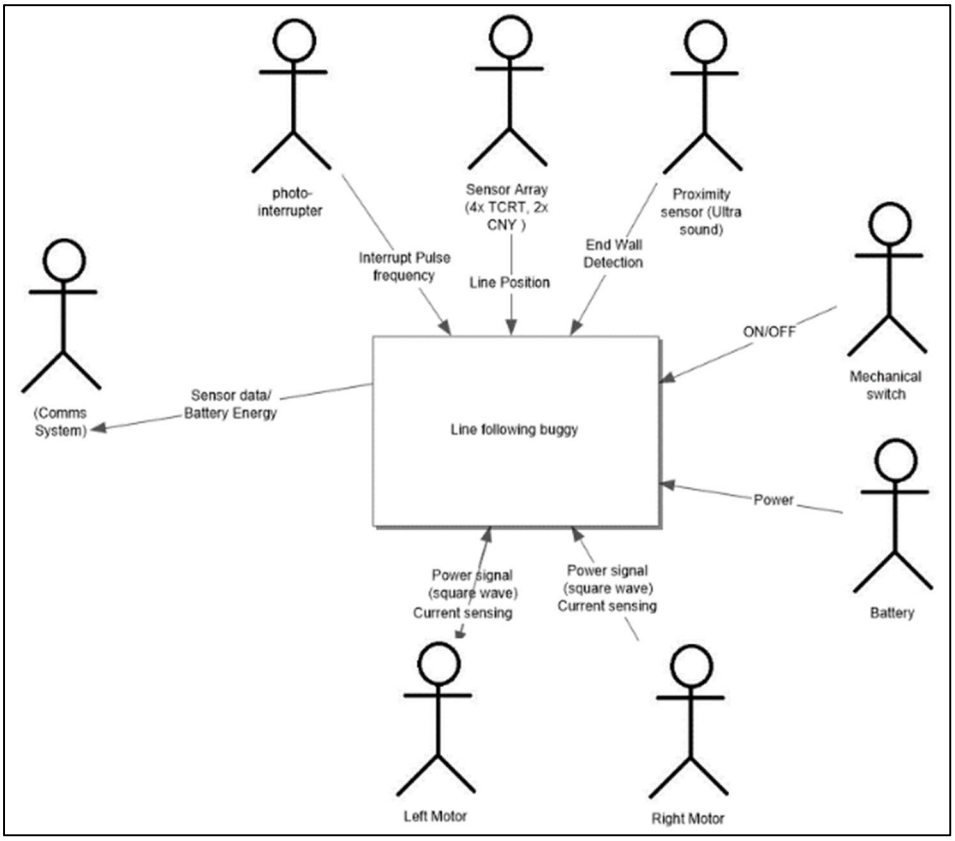


Appendix 8.2.5 - Gear combination chosen



Appendix 8.2.6 - Full wiring diagram

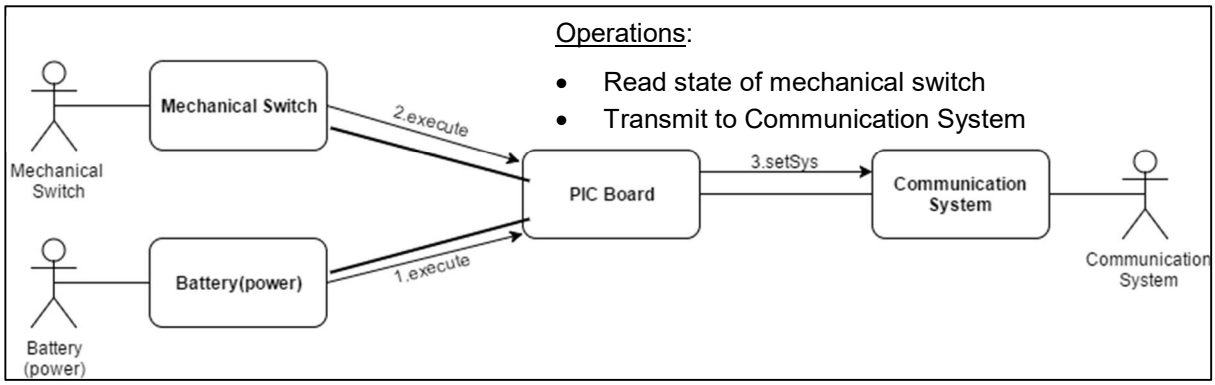
8.3 Use case details



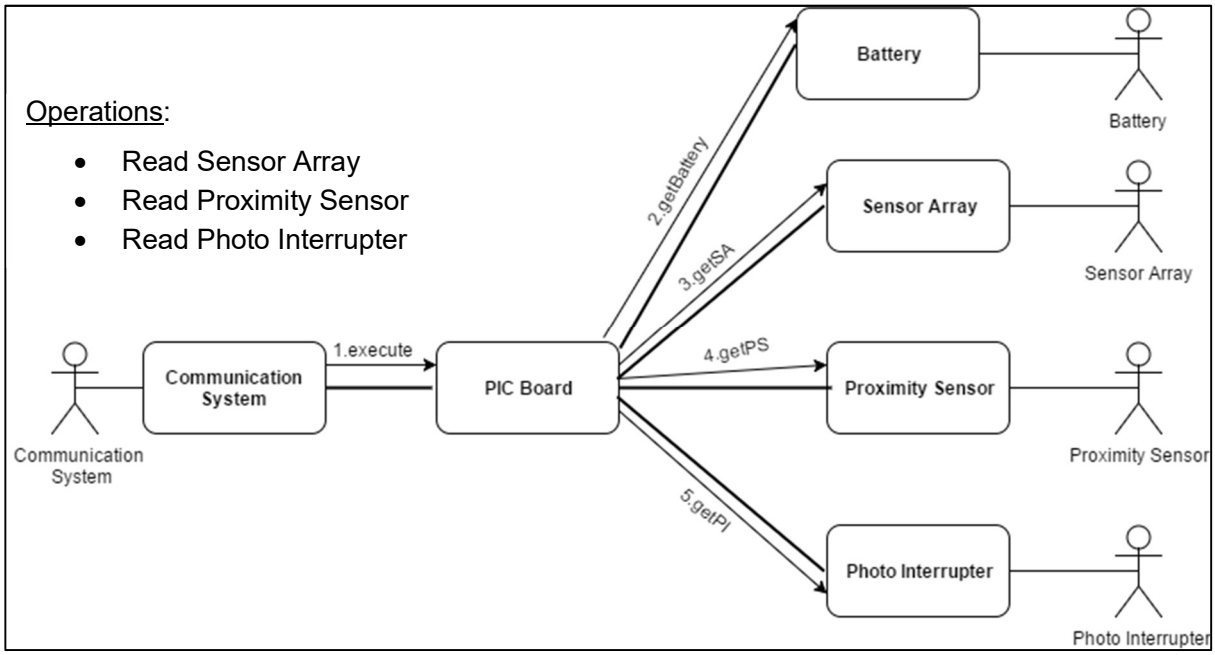
Appendix 8.3.1- Context diagram

Name	Description	Source	Destination	Interface	Size in bits	Arrival pattern	Comments
Line position	Position of line within the six sensors array	4 x TCRT500 sensors 2 x CNY sensor working as an array	Line following system-Control algorithm	Wired directly to PIC analogue input pins. ADC to software buffer.	10 bit at ADC 8 bit conversion by line following software	Periodic	
End Wall detection	Distance of buggy from the end wall of the track	MA40S4R/L Proximity sensor	Line following system-180deg spin function	Wired directly to the PIC analogue input pin	10 bit at ADC 8 bit conversion by line following software	Asynchronous	Signal will issue an interrupt to deal with the 180 deg spin of buggy.
Interrupt pulse frequency	Interrupt rate of encoder disk to photodiode	Phototransistor	Speed control system	Wire to PIC	1 bit	Continuous	Phototransistor is either on/off giving a square wave
Power	DC Power signal	Battery pack	System	Wired: PIC board, Motor drive board	n/a	Continuous	Power supplied to all required components
Battery monitoring	Measure of voltage and accumulated current in battery	Battery pack/Motor drive board	System	Dallas DS2781 integrated circuit on Motor drive board	8 bit	Periodic	Signals are supplied to the PIC from the Motor drive board
Current monitoring	Current value of motor	Isolated current sensing circuits on Motor drive board	System	Motor drive board	8 bit	Periodic	
PWM	Pulse width modulation used to regulate power to the motors	PIC	Motors	Wired to the motors through the Motor drive board	2 bit	Continuous	
Sensor date/ Battery energy	Data shared from the system to the outside world	PIC	I/O board	8 bit data bus	8 bit	Continuous	I/O board will only be connected when the data is required

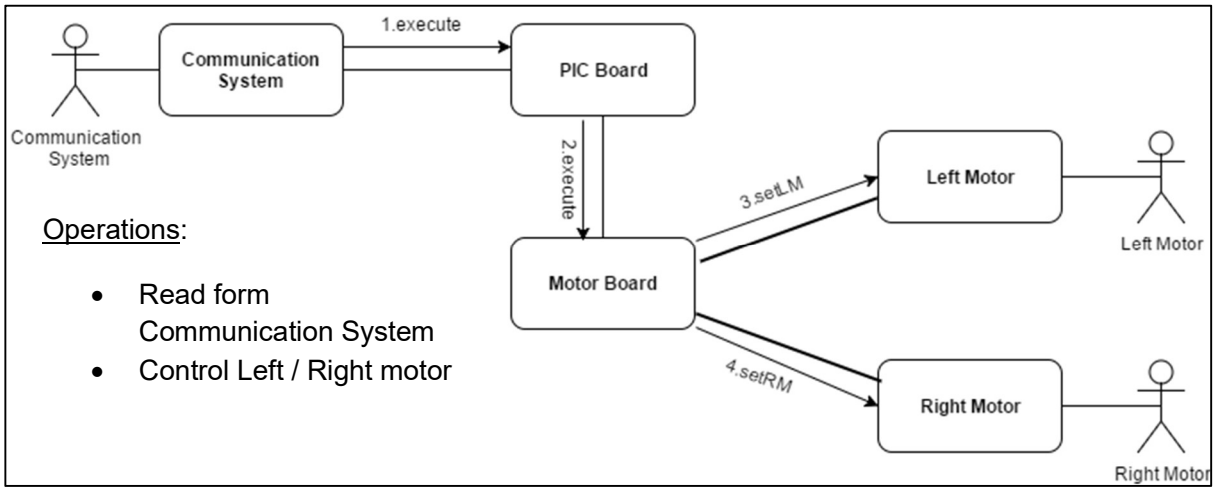
Appendix 8.3.2 - Message table



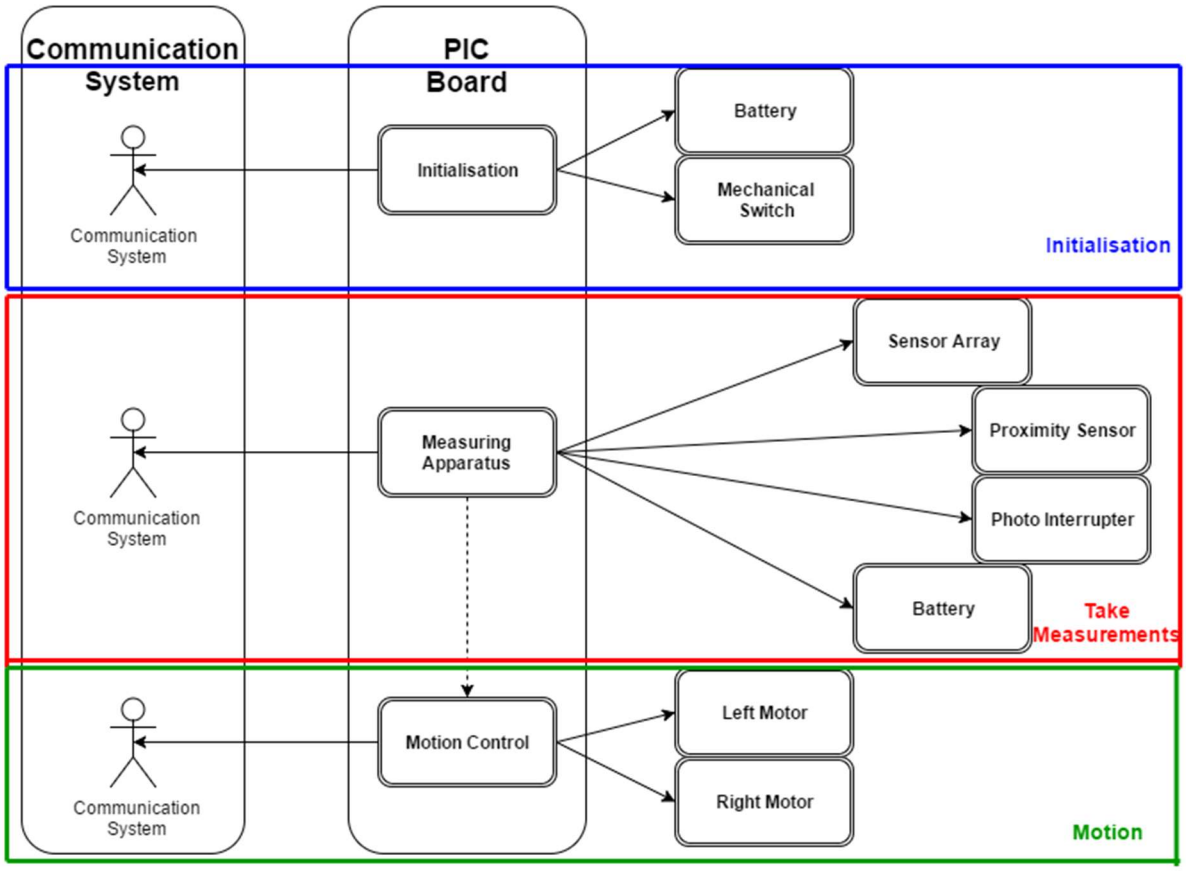
Appendix 8.3.3 - Initiation context diagram



Appendix 8.3.4 - Measurements context diagram



Appendix 8.3.5 - Motion context diagram



Appendix 8.3.6 - Composite diagram

8.4 Function prototypes & data structures

I. Motion control

A. PID/BB line following

```
void OpenADC(void);
void ConvertADC(void);
char BusyADC(void);
int  scan_line (void);
char calculate_error(char a);
char calculate_correction(char error, char previous_error);
void generate_PWM(int a);

char error;
char previous_error ;
char sensor_array[6];
```

B. Constant Speed

```
int read_photo_interrupt(void);
int calculate_average_speed(int a);
void isr(void);
void generate_PWM(int a);
void ClosePWMx(void);
void OpenPWMx(char period);
void SetDCPWMx(unsigned int dutycycle);

const char N_ticks = 12;
int T_period;
unsigned int dutycycle;
```

II. Initialization & configurations

```
void test_routine (void);
void calibrate_sensor (void);
void config_IO (void);
void config_timer0 (void);
void config_global_interrupts (void);
```

III. Measurement controller

A. Sensor Array Functions

```
char sensor_array[6];
```

B. Proximity Functions

```
void read_proximity(void);
void isr(void);

int reference_within_30cm;
```

C. Photo Interrupter Functions

```
int read_photo_interrupt(void);

int calculate_average_speed(int a)

const char N_ticks =12;
int T_period;
```

D. Battery Functions

```
int read_battery(void);
int calculate_energy(int v, int i);

int battery_energy_data[];
```

8.5 Record-keeping evidence

	Lab 2 data post processing started & pictures added to folder Design report 2 started, marking criteria examined.
5	Organised sensor & motor lab report internal deadlines Organised everyone to have a task in the report writing for both reports Extensive non-standard sensor research (~1.5hrs) & discussion with Jason to settle optimal sensor Extensive preliminary research in programming control algorithms (~3hrs) Pseudocode for control algorithms written (LC, PC, PID & MPC), PID control suggested as optimal Preliminary research in other line following robots: realistic speed, programs, sensor arrays... Lab 1 – first draft full review & revision of draft v1.0 Lab 2 – first draft started, verbally aided further data collection to increase report quality High Level drawings reviewed to accommodate for track environment (ramp)
6	Lab 1 – extensive revision of draft up till v2.3 (7+ hrs) More extensive pseudocode written for control, line loss, start-up, end & test subroutines Meeting recap posted & coordinated Lab 1 – final report written and submitted (~5hrs) High Level architecture 3D model created & updated to v2.1 (~3hrs)

Figure 8.5.1 - Work log extract (weeks 5 & 6)

Weekly Report (Group 18) week 8

Date:21/10/2016

Main tasks completed this week have been completion of high level architecture for the buggy and the completion and amalgamation of each team member's individual part of DR#2.

Tasks to be completed: Review of DR#2, possible extras added to the report and allocation and start of project proposal

Individual Student Contributions

Charles Shelbourne:
-Wrote Navigation Strategy of report amalgamating Marcell's input. Report focus on options for control systems and how our choice would cope with the track.
-Received guidance email from Frank pod, which led to a restructure of the navigation strategy part of report for which I rewrote plan, delegating Marcell the initial review of navigation strategies and myself the decision and high level details of our chosen strategy.
-Reading into potential obstacles buggy will face
-Made final decisions on how the buggy will deal with obstacles, researching into how a constant speed will implemented for the buggy
-Gave input into final design of buggy, including positioning of sensors and proximity sensors to best deal with track
-Amalgamated mine and Marcell's part of report and sent to Tom for amalgamation with first draft of DR#2

Marcell Toth:
Further research on navigation algorithms

Completing my part of the report (Navigation Strategy)

Proofread Charlie's part of the report as we did Navigation Strategy together.

After receiving the clarification e-mail on the report we had to do it from scratch.

Further reading in the ESP Procedures Manual Handbook and the Technical and Management Manual

Writing my part of the report again.

Thomas Hollis:
-Coordination with Jason & Gao for a wide range of sensor data collection. All components characterised. External components also characterised.
-Report writing in collaboration with Jason (up to v1.1)
-Line following algorithm discussions with Marcell.
-Potential budget spending discussion with Frank Podd, mentor & team members.
-Investigations & research into external ultrasound proximity sensor.
-Timeline updated & approach strategy refined to minimise stress near deadlines.
-Updated high level architecture to v2.2.
-Full layout, connectivity & schematics for sensor array drawn.
-Full simulations for sensor array undertaken.
-Heavy report writing (DR2) up to first deliverable draft. Merging of software section with hardware section.

Jason Brown:
-Sensor discussion with Tom.
-Met up with Gao to collect data on CNY70 sensor (the sensor having only arrived the weekend of week 7).
-Modified graphs and formatted them to make them easy to compare within lab report.
-Finished draft of line characterisation within the lab report.

Jianli Gao:
-soldered and tested another line sensor

-collected and analysed the data, (proved that this sensor could be used only when it was closed to the black-white surface.)

-decided and tried to test the proximity sensor

Figure 8.5.2 - Weekly report extract (week 8)