MANCHESTER 1824
The University of Manchester

School of Electrical and Electronic Engineering

Embedded Systems Project

LINE FOLLOWING BUGGY

Title: Final Report

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: 01/05/2017

Table of contents

1. Executive summary

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.[1] 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.[2] The teams will face each other in a live race around a track with the fastest total time being awarded first prize (risk assessment provided in previous report). 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 into three design reports and four technical demonstrations. The design reports provided a theoretical framework for design, where all the buggy's main innovations were formalised into engineering drawings and methodological documentation. The technical demonstrations allowed for the buggy's previously designed theoretical characteristics and innovative features to be built and tested one after the other.

As the project evolved through the technical demonstrations with more and more features being added to the buggy one after the other, certain theoretical decisions had to be adjusted. A key objective modification was the abandonment of a speed encoder disc. Theoretical simulations revealed that the control algorithm would perform significantly worse without constant speed, suggesting a speed encoder disc was not only useful but essential to the buggy completing the race. However, when the third technical demonstration was reached, it was discovered that if the motors were run at a high enough speed, the hill would have a negligible impact on the control algorithm making a speed encoder redundant. Other modifications to our aims include tweaking the mechanical layout in order to shift the centre of mass to the front wheels, changing the sensor array standoffs from legs to skis and many more minor adaptations in order to better adhere to technical demonstration objectives.

A graphic of the buggy, with all these adaptations, in its final racing state is presented below:

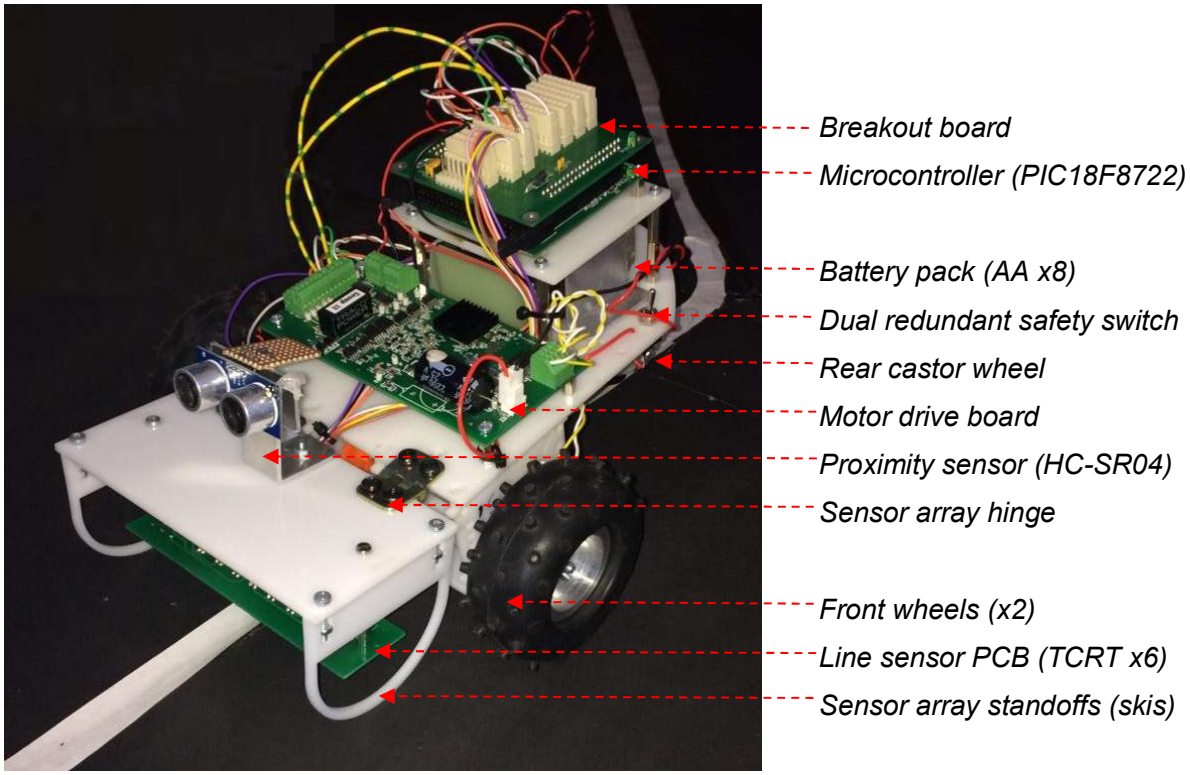


Figure 1.1 - Graphic of buggy in racing state

The final buggy was produced £0.94 under budget, on time, and adherent to all required criteria,

deadlines and technical documentation standards. It also cleared all obstacles in the heats.

The main required features of this buggy are: bipolar PWM-driven steering, proximity sensing, white line sensing, autonomous control and the ability to deal with an obstacle-ridden race track.

The main innovations that were added to give a competitive edge are: front wheel drive, a hinge-based sensor array, dual redundant software with both PID and PD control algorithms, dual redundant hardware with two sensor arrays (one array with 6x TCRT and one array with 4x TCRT and 2x CNY) and non-grip hinge standoffs for optimal tracking.

The rest of this report outlines in more detail the main system components, design choices, organisational planning, budgeting and an analysis of the buggy's final performance during race day.

2. System components

2.1 Mechanical components

The body of the buggy is divided into two major parts: the main chassis and the hinged sensor array. Since the buggy is front wheel driven, the main chassis baseplate is supported by a castor ball toward the rear end while the wheels and gear boxes are at the front of the chassis, immediately behind the hinged sensor array located at the very front of the buggy (see figure 1.1).

At the rear of the buggy, the ball bearing was modified for height adjustment to create an acute angle of 110 degrees allowing a shift of the centre of mass toward the front wheels. The angle was initially 90 degrees however, 110 degrees gives the wheels more downforce allowing for faster turns and better hill climbs. As we are using front wheel drive, unlike most other teams, this is a crucial aspect of our buggy as it is the main innovation that allows us to cope with hill climbs without a speed encoder, a feature that was removed from initial plans, in full performant control and stability[3]. Acetal was chosen as the material used for baseplates and standoffs as a full mechanical analysis revealed it was the optimal trade-off between strength and weight (see appendix 8.1.1).

The battery pack in its aluminium housing and the motor drive board are located on the first level of the chassis. For the most efficient spacing of the components, the microcontroller was mounted on a second level using metal pillars to secure it in place on top of the battery pack. This allows the battery pack to slide in and out on demand for fast battery swapping but also provides an excellent weight distribution to maximise downforce and minimise compressional and tensional stress (also shown in appendix 8.1.1).

Wiring and cable management was handled via the use of zip ties and a breakout board (see appendix 8.2.4), whose purpose is to enable connecting various components to the buggy's PIC microcontroller. All wires were multicore to increase durability and signal wires were either twisted with varying twist rate or isolated to reduce noise and interference respectively. This was a deliberate change only introduced after the third technical demonstration as interference was revealed to be a major cause for unpredictable behaviour.

Toward the front of the buggy's main chassis, gearboxes are held in place with an acetal assembly of baseplates. These gear boxes were designed such that the motors will provide enough torque for the buggy to climb the ramp while maximising theoretical maximum speed. This compromise was reached with a gear ratio of 15:1 which provides a high efficiency of 73% while the theoretical maximum speed remains at 1.6 m/s. This speed is more than sufficient as the software cannot cope with speeds beyond 1.5 m/s and this performed well on sloped as well as flat sections.

At the very front of the buggy, the sensor array is attached to the main chassis with hinges such that the line sensor array located at the bottom of the base plate can be as close to the line as possible in order to maximize its performance. The sensors are soldered onto a printed circuit board (PCB), and they are arranged in one line perpendicular to the motion of the buggy (see appendix 8.2.2). A feature that was added after the second technical demonstration was a spring-loaded mechanism and a shortening of the gap between the sensor PCB and the driving wheels. This allows a full mitigation of any hardware lag experienced in software readings of where the line truly

is. In fact, many different versions of this hinge were changed before settling on the one that performed best during tests. Some previous designs are presented below in figure 2.1.1.

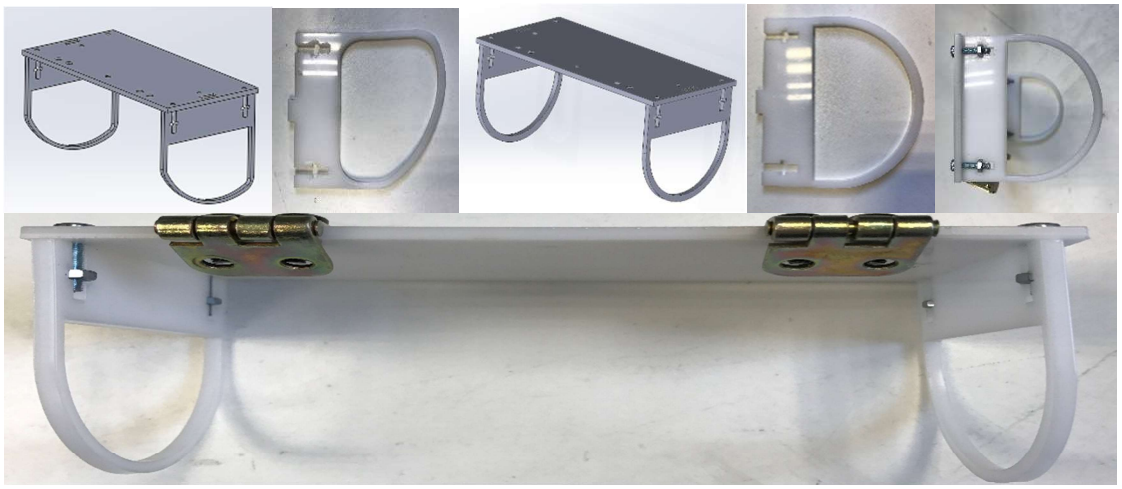


Figure 2.1.1 - Previous hinge designs developed and tested (semi-circle, circle & extended circle)

Following the final races, the most performant feature of our mechanical designs was revealed to be our innovative front wheel drive system and all the adjustments that went with it. Not only did the hinge prevent issues that other teams encountered during the hill section, but it also provided a resistive buffer that allows a smoother following of the line with sudden changes in direction being naturally damped. The front wheel drive provided us with more torque up the hill allowing for a stable control during oscillation at reasonable speeds without compromising uphill performance.

2.2 Electronic components

The main electronic components are divided into three major parts: power management, line sensing and proximity sensing.

Power management is handled by the motor drive board that takes 9V input from the battery pack and redistributes power to the motors and to the microcontroller. The input power to the drive board is controlled by a hardware on/off switch used to safely start and stop the buggy.

Line sensing is handled by two line-sensor arrays which consist of circuits of six sensors in a row. Unlike other teams, we chose to produce two separate line sensors that can be clipped on and interchanged in a fully modular fashion. The main benefit of this innovative modularity is the ability to adapt during race day to any unforeseen challenges allowing us to work towards a level of high performance without compromising reliability as our failsafe system can quickly be introduced to swap speed for reliability. These sensor arrays are presented below in figure 2.2.1.

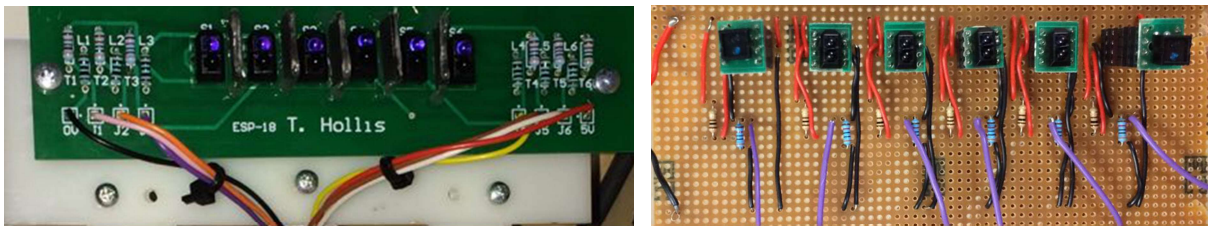


Figure 2.2.1 - Swappable sensor arrays (left: speed-focussed, right: reliability-focussed)

The first line sensor array is composed of six TCRT sensors and is designed with performance optimisation in mind. The sensors are placed very tight together for minimal oscillation about the white line allowing the buggy to reach its highest speeds (see appendix 8.2.2). However, this was an idea that was not present in initial designs. It was initially assumed that interference between sensors would not permit such a tight sensor array however a neat software trick was developed to not only mitigate this interference but use it to our advantage. This was a significant improvement on the overall performance of the buggy even if it was made during a later design review session.

The second line sensor array is composed of four TCRT sensors and two CNY sensors and is designed with reliability as its main priority. Having two different sensors types allows for a dual redundant array whereby if one sensor type is struggling to perform under race conditions for any reason, another sensor type can be used as a backup completely ignoring readings from the other. For example, if there is significant IR interference present inside the race room, the CNY sensors take over as they are fully shielded and unaffected by ambient conditions. This hardware innovation is only possible with a dual-branch software algorithm (see appendix 8.3.1). The sensors here are also placed wider apart to prevent line loss altogether. This array did not have to be swapped in during final performance as the buggy's speed-focussed sensor array performed as expected.

Both line sensor arrays were positioned with TCRTs 2mm from the track floor and CNYs 0.1mm from the track floor. These are the optimal operational heights as suggested by the datasheets and confirmed during laboratory testing. The CNYs are only hovering over the ground as they are inside a shielded package which protects from interference. This extremely close distance to the ground was only made possible due to our hinge which maintains constant sensor distance to the track.

Proximity sensing is handled by an ultrasound emitter/receiver whose main role is to sense the halfway wall in the track. The sensor triggers a hardware interrupt[4] in the processor which handles the 180-turn. The proximity sensor is attached to an 'L' plate such that it is operated at an optimal height to detect the halfway wall but not the side walls (see appendix 8.2.3). Initially there were designs to include two additional proximity sensors to detect side walls, preventing any collision. However, later it was discovered that this could be exploited to navigate the buggy using the walls, independently of the white line, and this was no longer permitted for use during the race.

Following the race, the most performant feature of our electronic designs was the optimised speed-focussed line sensor array. Without the fear of lacking reliability thanks to our backup reliability-focussed array, a lot of time was freed up focussing on performance instead of reliability. This meant that our buggy reached much higher speeds than most other buggies that used similar hardware.

2.3 Software components

The main software components are divided into three major parts: motor control, line following algorithms and proximity interrupts. This software was written onto the PIC's main EEPROM memory in the C programming language.[5]

Motor control was achieved by using two main functions: `motor_right()` and `motor_left()`. These control the power given to the right and left motors respectively. These use PWM to allocate a range of power to the wheels through the function `setup_PWM()`. A combination of these functions was used to create the `move_180()`, `move_stop()` and `move_algorithm()` functions that moved the buggy 180 degrees, stopped the buggy and moved according to the PID algorithm, respectively.

The main line following algorithms used were PID[6] for the performance focussed sensor array and PD[7] for the failsafe sensor array. The line sensors are first fed into a software array via the `read_line_sensors()` function, and the change in value of this array is used to calculate which sensor in the array is closest to the white line via the `calculate_proportional_error()` function. The `calculate_PID_error()` function then calculates the PID error associated with this proportional error before feeding this control loop back to the `move_algorithm()` function which adjusts accordingly.

The halfway wall is detected via an interrupt[4] rather than sequentially. This means that whenever the wall is detected the main line code is immediately interrupted to execute the 180-turn routine. This allows our code to have to poll a lot less sensors during every execution of the line following control loop allowing us to run the software a lot faster yielding smoother line fitting.

Originally the design included the use of an analogue feedback and error. However, during testing it was found that error was subject to differences in the separate sensors' sensitivity. This made it difficult to control as it yielded a non-linear gradient of error. Therefore, the design's main software change was to use quantization and a discrete error scale, which gave more flexibility and precision when tuning the buggy. This ultimately allowed the buggy's control system to deal with the error more consistently allowing for more predictable tuning parameters[8].

Tuning the buggy was a significant task which also contained many unforeseen design changes. The buggy's performance was found to be a factor of many parameters other than simply the three PID tuning constants K_p, K_d and K_i for example battery level and track surface. Overall, with all the necessary adjustments, the developed code made the buggy follow the white line tightly and smoothly in a reliable way after the 72nd version of the software. Frequent software updates and their corresponding tests allowed for an ideal development environment with little redundancy.

A few neat software tricks were also implemented in order to cope with some unforeseen challenges. The most significant of which was a method to deal with the slightly unpredictable behaviour of the proximity sensor. Many teams around us seemed to be cursed with the same issue as us where their proximity sensors would trigger unexpectedly during testing due to interference in the noisy test environment. A few concise lines were added to our code to only accept detections within a certain range. This completely alleviated our biggest source of instability.

Following the race, the most performant feature of our software components was the well-tuned PID parameters that allowed for a tight curve fitting at high speed in conjunction with the software tricks that allowed our buggy to survive obstacles that other buggies could not cope with. All this was helped by the efficiency of our code which was less than 300 lines long within a single C source file.

3. Team organisation and planning

3.1 Requirements analysis

The main aim of the project was to build an electronic line following buggy that would navigate a track in the fastest time possible. A major prerequisite of the project was to be able to work as a team to produce a winning buggy and it was decided early on that a strong team organisation and planning is crucial to do so. The deadline to produce the final product for race day was 29 weeks. The buggy had to be designed and built from scratch with a budget of £40. To optimise our design for the buggy, requirements analysis was crucial. This was most often done using a checklist. A summary of these requirements in the same checklist style is presented below in table 3.1.1.

Requirement description	Requirement feasible?	Requirement designed for?	Requirement met in race?
Performance requirements			
Follow the line using an automated algorithm	Y	Y	Y
Do not touch side walls	Y	Y	Y
Deal with line bends toward right and left (45°)	Y	Y	Y
Navigate up and down a hill	Y	Y	Y
Detect midpoint wall & turn 180° within 30cm	Y	Y	Y
Stop within 10cm of line ending	Y	Y	Y
Deal with any obstacles (break, chicane, bump…)	Y	Y	Y
Complete the track as fast as possible	Y	Y	Y
Perform the above in one uninterrupted run	Y	Y	N
Architectural requirements			
Total build time of less than 29 weeks	Y	Y	Y
Mechanical build must be in safe conditions	Y	Y	Y
Only 6 line sensors allowed	Y	Y	Y
Total budget of £40 maximum	Y	Y	Y
High gearbox efficiency	Y	Y	Y
Maximum two motors (provided)	Y	Y	Y
Maximum two end wall sensors	Y	Y	Y
Innovative features	Y	Y	Y
Programming requirements			
Follow the line using an automated algorithm	Y	Y	Y
Maximum three wireless interventions allowed	Y	Y	Y
Clean code documentation	Y	Y	Y

Table 3.1.1 - Requirements analysis summary checklist

3.2 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 3.2.1). This is where additional ad-hoc project meetings are called.

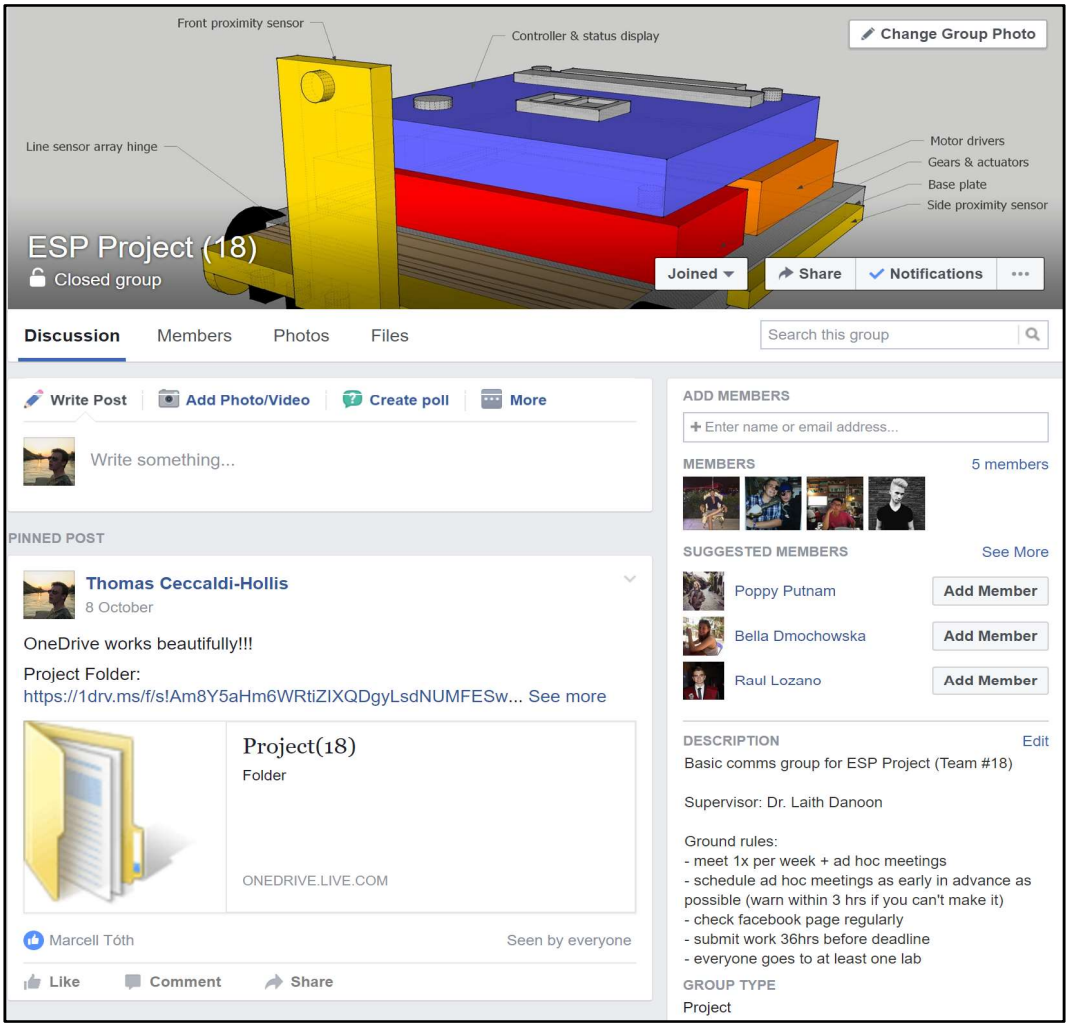


Figure 3.2.1 - ESP Group (18) Facebook Group

The ground rules are clearly outlined in the group description and team members can easily communicate urgent information via group messages on a social 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 3.2.2). This worked excellently throughout the entire project with no team members missing any meetings and no communication mishaps.

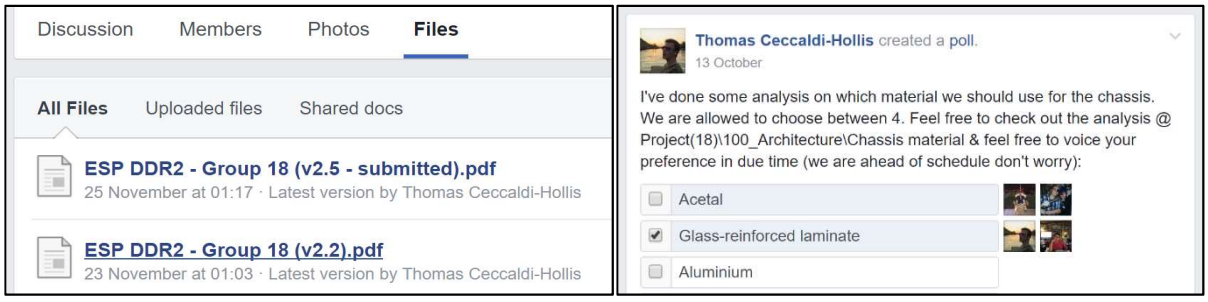


Figure 3.2.2 - File upload and poll creation on group page

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' personal computers (see figure 3.2.3 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.

This was a considerable programming task to setup, and it took almost a full day to get all team members familiar with the material however it yielded excellent results with no lost files, no version collisions and seamless file collaboration in real time with scheduled weekly offline backups just in case.

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 allowed the team leader to verify, easily and fast, that tasks were done are of appropriate quality and on time. In addition, all members kept 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.

Weekly reports are also submitted to the supervisor, as a double-sided page clearly detailing the individual responsibilities and tasks carried out each week.

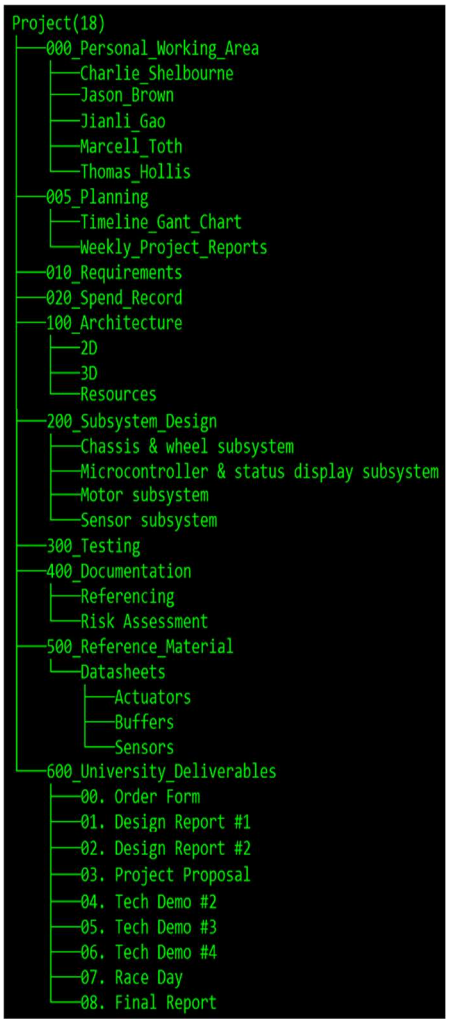


Figure 3.2.3 - Folder hierarchy

Finally, code is mostly developed on GitHub[9] before the final version is uploaded to the main folder structure. Code on GitHub is easily collaborated on by creating multiple branches containing different versions of software written by different people. This is shown in figure 3.2.4 below:

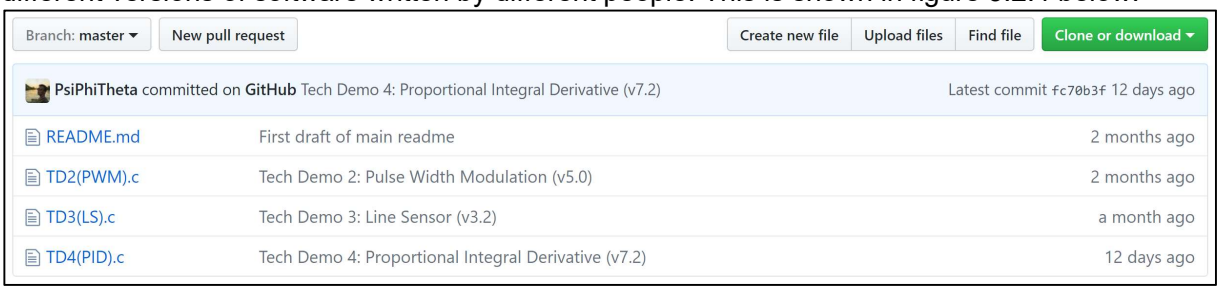


Figure 3.2.4 - GitHub repository of ESP-18 source code

Project organisation was a definite success yielding no significant organisational issues throughout.

3.3 Project planning

The project was planned based on the its main deliverables and their interdependencies (see figure 3.3.1). 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).

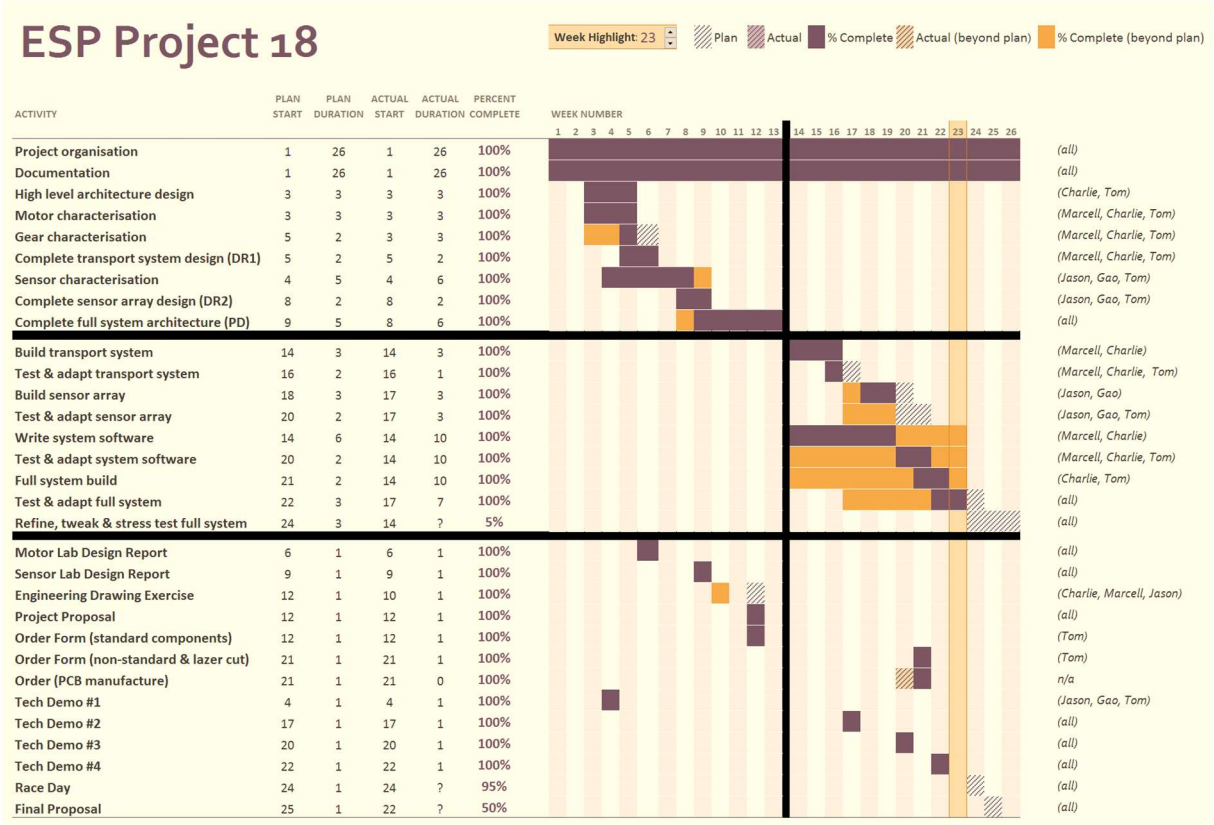


Figure 3.3.1 - ESP Team Project Plan (Gantt chart)

This plan was used to set strict weekly deadlines for each team member to complete SMART[1] tasks. The amount of manpower allocated to each task was specified and amended as required. Each individual's responsibilities were outlined in their Personalised Project Plan and reviewed by the team leader. On occasion, some team members felt that they were not receiving enough work so these tasks were amended at the start of the week, during the weekly meeting.

The main obstacle in the project's planning was a significant amount of delay due to language barriers between team members. This was overcome by scheduling additional ad-hoc meetings to give certain team members time to speak and ensure they fully understand what is expected of them.

In order to further help address this issue, group meeting rooms were booked before every major deliverable where the team leader explained to all group members all aspects of the expected requirements are criteria. During these sessions, code was displayed on the overhead screen, each team member was quizzed on their knowledge and any grey areas were cleared up. The team also organised presentation rehearsals before each technical demonstration to ensure that a professional pitch was delivered every time. This resulted in receiving full marks for every single tech demo presentation with shining feedback such as:

"This group was easily the most impressive demonstration in the lab. They showed what I would consider to be a master class in group presentations." - Tech Demo 2

"As expected, perfect. This group clearly spent a great deal of time rehearsing their presentation and it showed." - Tech Demo 3

"Excellent presentation in all aspects. Group members are able to provide very detailed explanation of technical details." - Tech Demo 4

This customer-centric approach to presenting our work, inspired by existing business models[10], was perhaps the best way we found of turning our largest issue into a strength.

[1] 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.

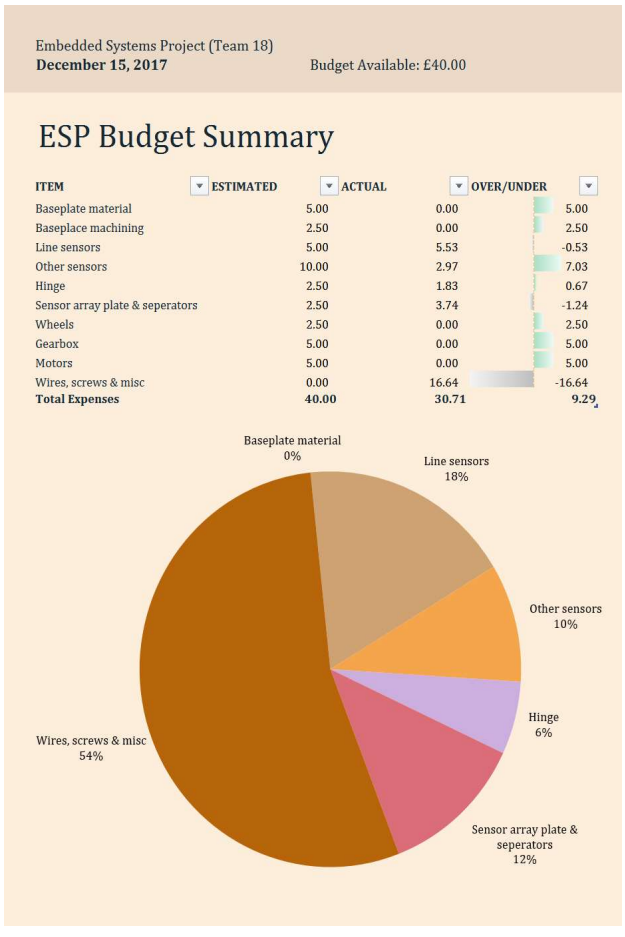
Fortunately, as the group organisation was a major priority in this project, all deliverables were achieved on time and at the level required however some difficulties within the group arose nonetheless. As you can see from Figure 3.3.1, a few tasks had to be started a lot earlier than initially planned (see in orange) such as writing system software and testing since certain technical demo requirements dictated a specific demonstration of functionality that was not foreseen.

Another difficulty within the group arose during a technical demonstration where the buggy's code, which had been performing flawlessly in previous conditions, was performing very poorly due to a change in environmental conditions. With only half an hour to fix the issue, all team members had to stay calm, put their ideas forward to try to find and fix the bug in the code, reprogram the buggy and pass the technical demonstration as expected. All team members contributed different ideas of where the bug may have come from and until it was found to be due to a software threshold defining what a white line looks like to the line sensors. This was fixed using a `#define` statement that had been strategically placed in the code as a preventative measure in case errors like these occurred. It was due to careful prevention techniques such as `#define` that the difficulty was surmounted and the technical demonstration was passed with a flawless performance.

4. Budget vs. outturn

4.1 Itemised costs

The budget was managed using spreadsheets and a summary pie chart. The full itemised cost breakdown is first presented 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 4.1.1 - Budget & cost breakdown

This list excludes the provided standard parts of the buggy such as motors, gears, wheels, battery pack, PIC microcontroller, motor drive board and breakout board. 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 were ordered in advance with a total cost of £30.71, within our £40 budget with £9.29 to spare for redesigns.

4.2 Final cost and outturn

The £9.29 left over from the initial order of components was nearly completely used for prototypes, replacements and minor modifications outlined in section 2 of this document.

Prototype costs are identified in the table below:

Prototype	Cost
Prototype stripboard line sensor array	£4.88
Prototype stripboard proximity sensor	£0.00
Prototype stripboard speed sensor	£1.10

Table 4.2.1 - Prototype costs

Replacement costs are identified in the table below:

Replacement	Cost
TCRT replacements	£1.96
CNY replacements	£0.00
Proximity sensor replacements	£0.00

Table 4.2.2 - Replacement costs

Minor modification costs are identified in the table below:

Modification	Cost
Spring loaded hinge	£0.00
Ball bearing modification	£0.00
Wiring interference modifications	£0.00
Tighter sensor array PCB redesign	£0.00
Modified proximity sensing of side walls	£0.00

Table 4.2.3 - Modification costs

The above costs add up to £7.94 and were kept within budget by reusing soldered components in the case of the speed sensor and using the prototype as a final working sensor in the case of the proximity sensor as well as other smart cost-cutting strategies through using free university services. Final cost and cost for 100 buggies are identified in the table below:

	Cost
Itemised costs	£30.71
+ Prototypes, replacements and modifications	+ £7.94
+ Standard components (PIC + motor drive board + breakout board)	+ £115.00
= Final buggy cost	**= £153.65**

Itemised cost for 100 buggies (via economies of scale and bulk discounts)	£ 2 699
+ Standard components for 100 buggies (already in bulk pricing)	+ £11 500
Cost per buggy (mass produced buggies)	**= £141.99**

Table 4.2.4 - Final cost and cost for 100 buggies

While the proposed budget of £30.71 does differ from the final cost of £38.65 by approximately 28%, this was an eventuality that was expected and planned for. By allowing a margin of safety for future buggy developments, the buggy outturn was produced within budget. Moreover, with further bulk orders and economies of scale itemised costs can be driven down significantly by 30% to £26.99. This resulted in the cost per buggy dropping from £153.65 to £141.99, a 7.6% decrease, highlighting the scalability of our buggy's design and its realistic mass production potential.

5. Race Analysis

5.1 Race preparation

While technical demonstrations served as major stepping stones to prepare for the race, there was still a lot of programming and additional work to be undertaken after the final demonstration.

The main form of preparation was refining existing code to reach faster and faster speeds without losing the line. This code was written, programmed to the buggy and then the buggy was tested on a custom practice racetrack. This was done repeatedly to stress test performance.

The custom practice racetrack that was built was integrated into the floor of the accommodation of the team leader. The track was built using black paper and rolls of white tape to create the line.

All the obstacles that could be present during the race were simulated on different sections of the custom practice racetrack as follows:

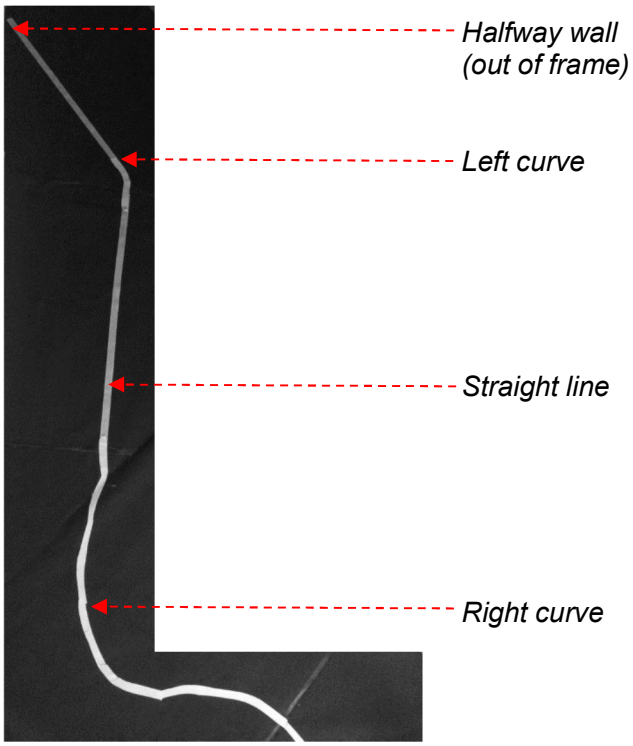


Figure 5.1.1 - Practice racetrack (section 1)

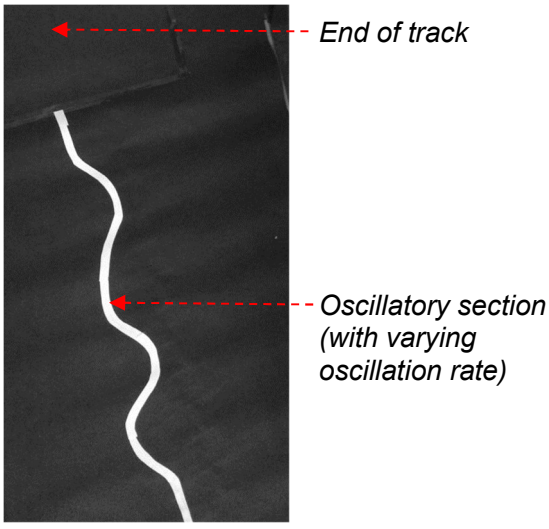


Figure 5.1.2 - Practice racetrack (section 2)

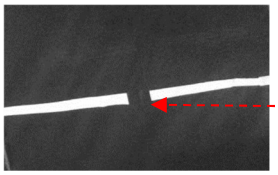


Figure 5.1.3 - Practice racetrack (section 3)

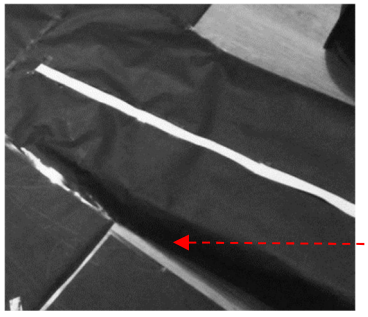


Figure 5.1.4 - Practice racetrack (section 4)

This strategy worked well and it allowed our team to avoid the stressful last minute programming that other teams inevitably went through due to limited access to a practice racetrack.

5.2 Testing procedures

Testing documents were also produced as part of the race preparation and software development, in order to achieve a formal testing methodology. An example of this is shown below:

	Testing document	
		Result
Control parameters		
	Software version	v6.1
	Test number (out of 10)	#3
	Battery level (before tests)	High
	Ending battery level (after tests)	Mid
	Previous record time	00:59
Required parameters		
	Straight line	Pass
	Right curve	Pass
	Left curve	Pass
	Halfway turn	Pass
	Oscillatory section	Fail
	Line break	Pass
	Ramp	Pass
	End of track	Pass
	Entire track completion without reset	Fail
Additional parameters		
	Max possible speed	60%
	Smoothness of motion	Poor
	Total time taken (excluding resets)	01:15

Figure 5.2.1 - Testing document example (result of program version 6.1)

The testing documents shown above were a checklist of how the buggy performed during all of its 10 stress tests on the custom practice racetrack. This checklist was completed every single time new code was downloaded onto the buggy to keep a written log of the physical performance of each version of the code. This allowed the accurate fine tuning different aspects of the code while maintaining a written record of previous performances.

This procedure was particularly useful in identifying unexpected behaviour such as the impact of the current level in batteries on the performance of the buggy. By testing the buggy 10 times before rewriting the code and keeping a written record of it, we noticed early on in the development process that a buggy with batteries at full charge over-corrected itself if it was tuned at medium charge.

Other testing documents were also produced throughout the development stage in order to ascertain individual component-level behaviour and make sure all sub-sections of the buggy work as expected, minimising debugging time. This is shown in figure 5.2.2 on the following page.

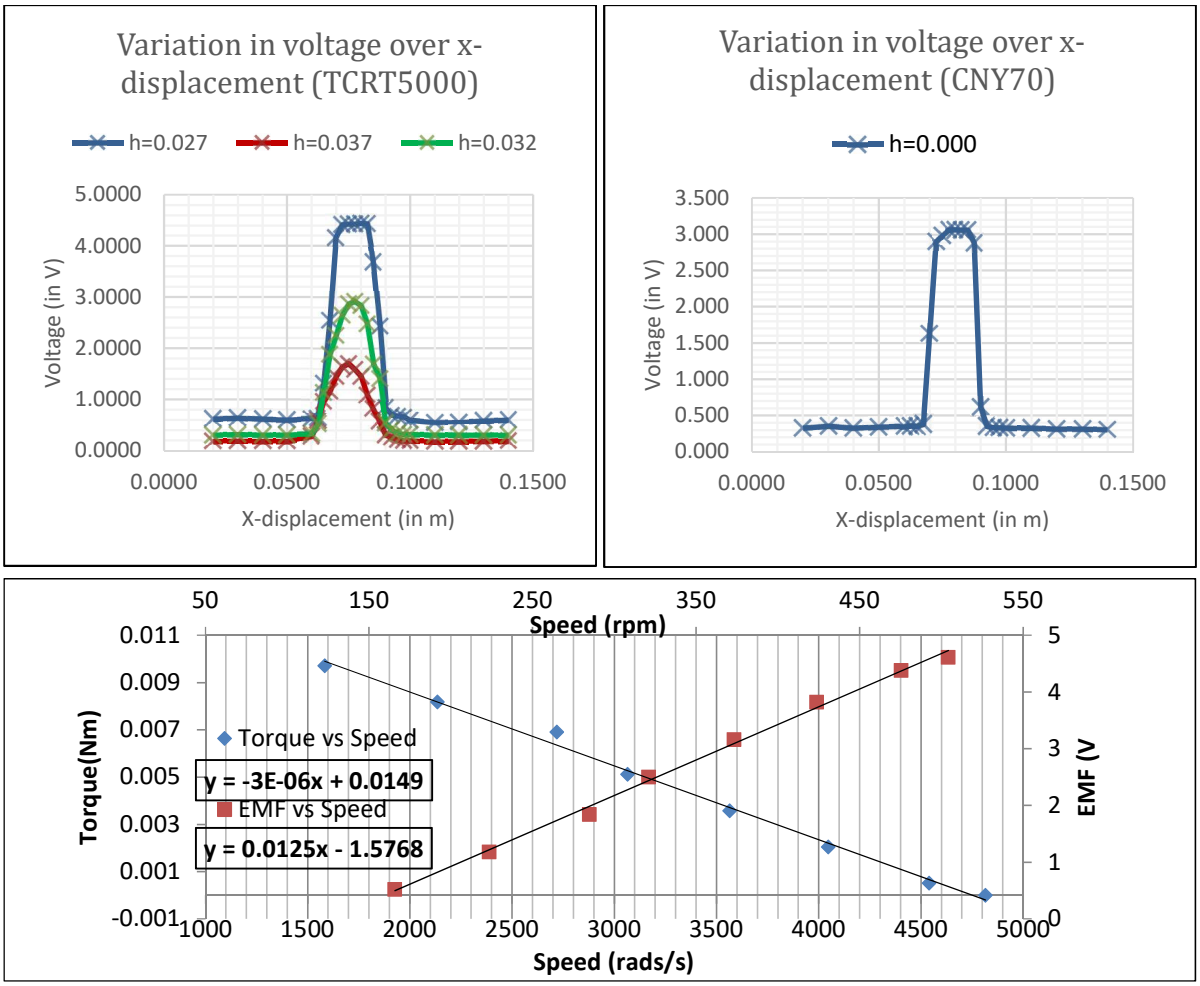


Figure 5.2.2 - Component-level testing results

Figure 5.2.2 shows the results of individual component-level tests (here: line sensors and motor performance) which were used to verify that the hardware components of our buggy were behaving as expected and close to our theoretical predictions. This was a very useful asset to minimise the time spent debugging as we did not have to waste time wondering if the issue was software or hardware based as the hardware had already gone through rigorous quality control.

5.3 Summary of race performance & critical analysis

Just before the heats, the buggy was tested 10 times on a duplicate version of the track to ensure reliability and it passed all of those tests without any issue. The buggy was deemed stable enough and was then released to attempt its assessed runs.

All aspects of the race were successfully completed during the assessed runs with the exception of the race being completed in a single run. The buggy ran into issues on two occasions.

The first issue arose during the first assessed run. The buggy missed the line while going over the brow of the ramp on its way down. This error was not foreseen as the duplicate test track was not quite identical to the assessed track. The assessed track had a slight gap between the flat section and the brow of the ramp. It also had a piece of white tape on it that was not quite straight. This feature was not present on the test track thus this error was not anticipated.

This error occurred since, during the brow of the hill, the sensors are at a uniquely elevated position with respect to the line. While the buggy should indeed have been able to cope with such inconsistencies in between different pieces of race-track, the buggy's failure was due to lack of

attention rather than lack of performance capability or poor programming. Indeed, this error was very easily rectified by adjusting the hinge and the net position of the line sensor with respect to the track. The rear castor wheel was elevated slightly more than usual to minimise the risk of line loss. This minor adjustment confirmed our previous hypothesis as ten more runs on the problematic piece of track confirmed the issue had successfully been eradicated.

The second issue arose during the second assessed run. The buggy performed all aspects of the track flawlessly and in excellent time (under 24.00s), until a few excruciating centimetres before the end. Just before the end, while in the chicane section, the buggy lost the line. While the buggy performed the chicane without any issue on the first half of its journey and in all the previous practice runs, it failed to do so again on its way back.

This error was hypothesized to be mainly due to excessive speed. Indeed, the buggy had two versions of code that could be used. One version ran at a speed of 80% of maximum speed while another version ran at 90% of maximum speed. The PID tuning parameters are significantly different in each code as to cope with the speed of each version of code. Since our first assessed attempt was ruined by a mindless mistake we had a strategic decision to make: either run the slow code and get a guaranteed mark of 80% with all the criterion met or take a risk, run the faster code and get a mark between 70%-100% depending on the quality of our competition. The team democratically agreed to take the 10% gamble due to the encouraging test results of the 10 previous test runs. Unfortunately, the gamble was punished due to slightly inconsistent buggy behaviour.

The behaviour of the buggy when it comes to its precise tuning is very volatile and this was a major challenge throughout the project. The buggy struggled to maintain control after the downhill stretch due to a speed build up and, even if the PID control did correct this, such volatile behaviour could have been improved by using the current sensing capability of the motor drive board and adjusting power to the motors accordingly in downhill sections. This volatility was also due to a large variety of sources of systematic and random error.

The main sources of systematic error which impacted the buggy's accuracy were uneven tightness of wheel bearings, asymmetry of the gear boxes and different ages of the motors. All these made the buggy turn slightly to the right when it was instructed to go in an uncontrolled straight line. This was artificially compensated for within the software but in hindsight perhaps it would have been better to resolve these issues using a hardware solution. The reason behind this is that following the heats, the buggy was visually inspected, open circuit tested and short circuit tested. Upon this extensive inspection, we had noticed one of the wheels had a slightly less strong grip on its axle. This could have caused a slip which may have resulted in the loss of the line. This made us realise that compensating for hardware faults in software is a risky strategy as hardware faults deteriorate over time and the software cannot predict this. For future projects, we will make sure not to solve hardware solutions with software tricks but instead to address the hardware issue with its corresponding hardware solution.

The main sources of random error which impacted the buggy's precision were fluctuating battery levels, differences in racing conditions, differences in track conditions and crosstalk interference between PWM wires. These sources of random error may also contribute to the volatility of the buggy's behaviour. The main way in which these sources of error would be improved in future projects would be via the use of shielding wires that prevent crosstalk interference, and a more rugged hardware assembly with leaves less to chance.

The most successful features were the isolation of the proximity sensor wires and the corresponding software trick which prevented a very common issue that most teams faced where their proximity sensor would get triggered by accident at strange places in the track. By setting a custom range threshold in the interrupt routine we were able to drastically improve our buggy's reliability which allowed it to successfully pass all the obstacles of the course, albeit not all in one continuous run. The most helpful innovative feature of the buggy was its front wheel drive mechanics and hinge concept that allowed our buggy to power up the hill at very high speeds using a gear ratio focussed on maximising speed not torque. This gave us a significant speed advantage over other teams.

6. Specification summary

The Embedded Systems Project's buggy was successful as a line following robot, built within budget, within deadline and capable of dealing with all the requirements specified. However, it is important to highlight the project's overall performance and be critical of its shortcomings to determine what could have been improved.

The buggy's main innovations were a front wheel drive system, a mechanical hinge, two modular sensor arrays, one of which containing two different sensor types for dual-redundancy and a unique two branch software written in under 300 lines of code. The innovations focussed on maximising speed without compromising reliability.

The buggy's basic features included an efficient gearbox with a practical torque to speed trade-off gear ratio (15:1), a light chassis designed to withstand significant tensile strength, a sensitive line following six-sensor array printed on a custom PCB, a proximity sensor circuit properly calibrated and a reliable PID algorithm written in C.

While the team was disappointed that the buggy was not selected for race day due to it not completing the track in a single run, this does reveal a lot of different sections that could have been improved throughout the buggy production process.

At the design stage, more out-of-the-box ideas should have been considered instead of assuming these were not going to be allowed. The buggy's software would have been much easier to write in Python over a more powerful microcomputer such as the Raspberry Pi[11]. Moreover, a different type of line sensor may have allowed for a more performant top speed or even a more elaborate line sensing array layout.

At the hardware development stage, it is clear that the buggy would have been better off with sturdier 3D mechanical CAD designs and a more reliable mechanical assembly. A different motor and gearbox assembly could drastically help reduce the sources of systematic error highlighted in section 5.3 such as asymmetry in the axle and wheel subsystem. In addition, using new parts instead of used ones would help decrease sources of random error such as a difference in performance between both motors.

At the software development stage, the main testing apparatus was a custom-built racetrack which proved to be very useful in avoiding issues that other teams faced with respect to testing-time limitations. However, relying on test results from a slightly different environment suggests that a more accurate tuning could have been achieved through tuning the buggy directly on the racing environment or through a calibration routine for the buggy to execute automatically before the race. A calibration routine was initially written but this was deemed too slow to be beneficial in racing conditions and was abandoned. Another software improvement that may have improved the buggy's performance could be the use of a more intelligent self-learning algorithm that would be able to record the path taken in the first half of the track and use it to return safely back at a much higher speed.

From an organisational point of view, while there was a lot of time that was spent overcoming language barriers, this time was evidently not wasted as the team's effort to overcome it resulted in very positive feedback during technical demonstrations.

If this design is to be taken forward any further the main areas of improvements identified are a more accurate and precise software tuning to prevent unexpected line loss, a more rugged mechanical structure and a stricter testing methodology to avoid any outcomes to be left to chance.

Finally, while the project did have its issues, no system is infallible and once we accept our limitations, we must always seek to go beyond them.

7. References

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

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

[3] Hindiyeh, R., Gerdes, C. (2010) *Analysis and control of high sideslip manoeuvres*. International Journal of Vehicle Mechanics and Mobility

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

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

[6] Thirupathi, A., Matla, R., Sundeep, K. (2016) *Design of PID controller for DC Motor Speed Control Using Arduino Microcontroller.* International Research Journal of Engineering and Technology.

[7] Normey-Rico, J., Alcalá, I., Gómez-Ortega, J. and Camacho, E. (2001). *Mobile robot path tracking using a robust PID controller*. Control Engineering Practice. pp.1209-1214.

[8] Lanzon, A. (2017) *Control System 1 Lecture notes*. School of Electrical and Electronic Engineering, The University of Manchester.

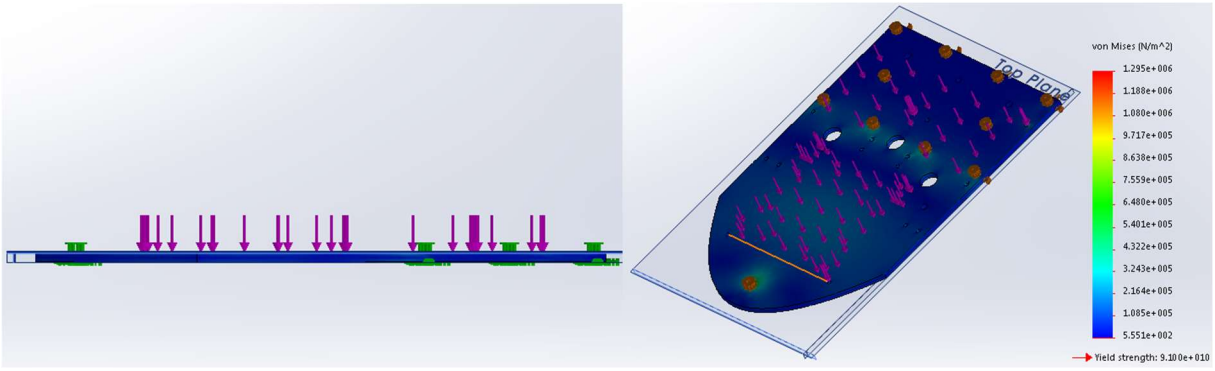
[9] Hollis, T. *ESP-18,* GitHub. Available at: github.com/PsiPhiTheta/ESP-18 (Accessed: 02/05/17)

[10] Jirinova, K., Kolis, K. (2013) *Research of Customer-Centric Approach and Involvment of Customers Into Innovation Process*. International Statistics and Economics, Prague.

[11] Raspberry Pi, *Hardware Datasheet*. Available at: www.raspberrypi.org/documentation/ (Accessed: 03/05/17)

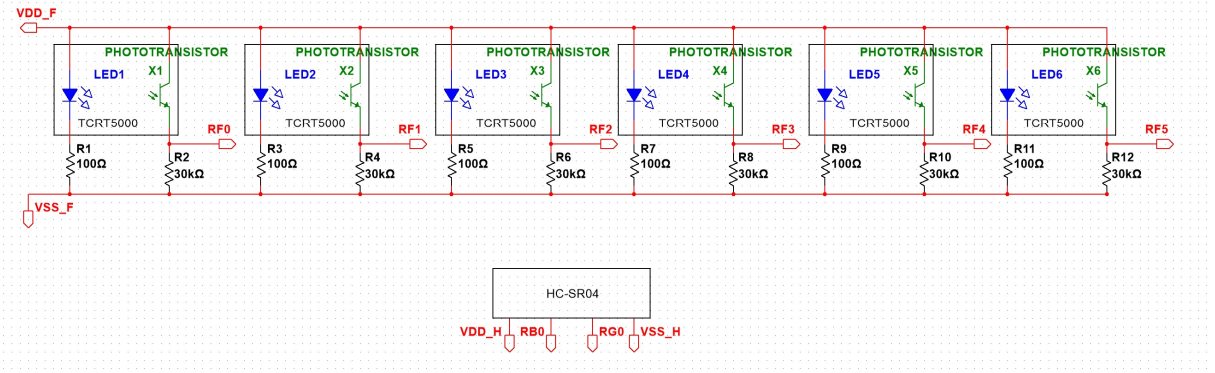
8. Appendices

8.1 Updated engineering drawings

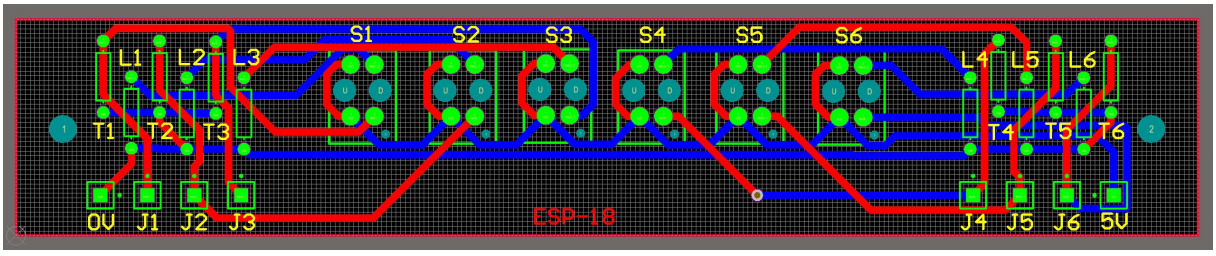


Appendix 8.1.1 - Updated stress analysis of baseplate

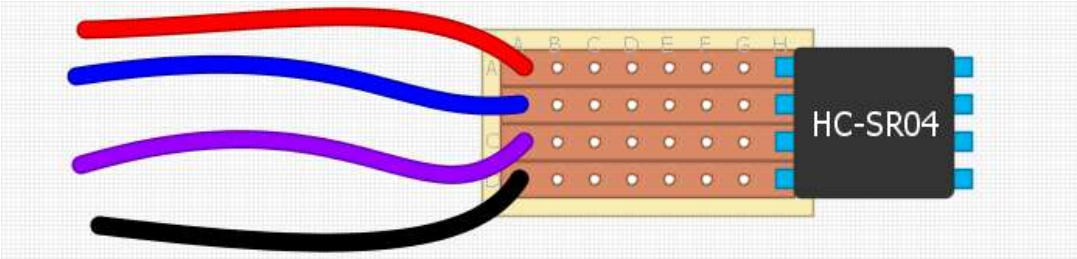
8.2 Updated circuit/wiring diagrams



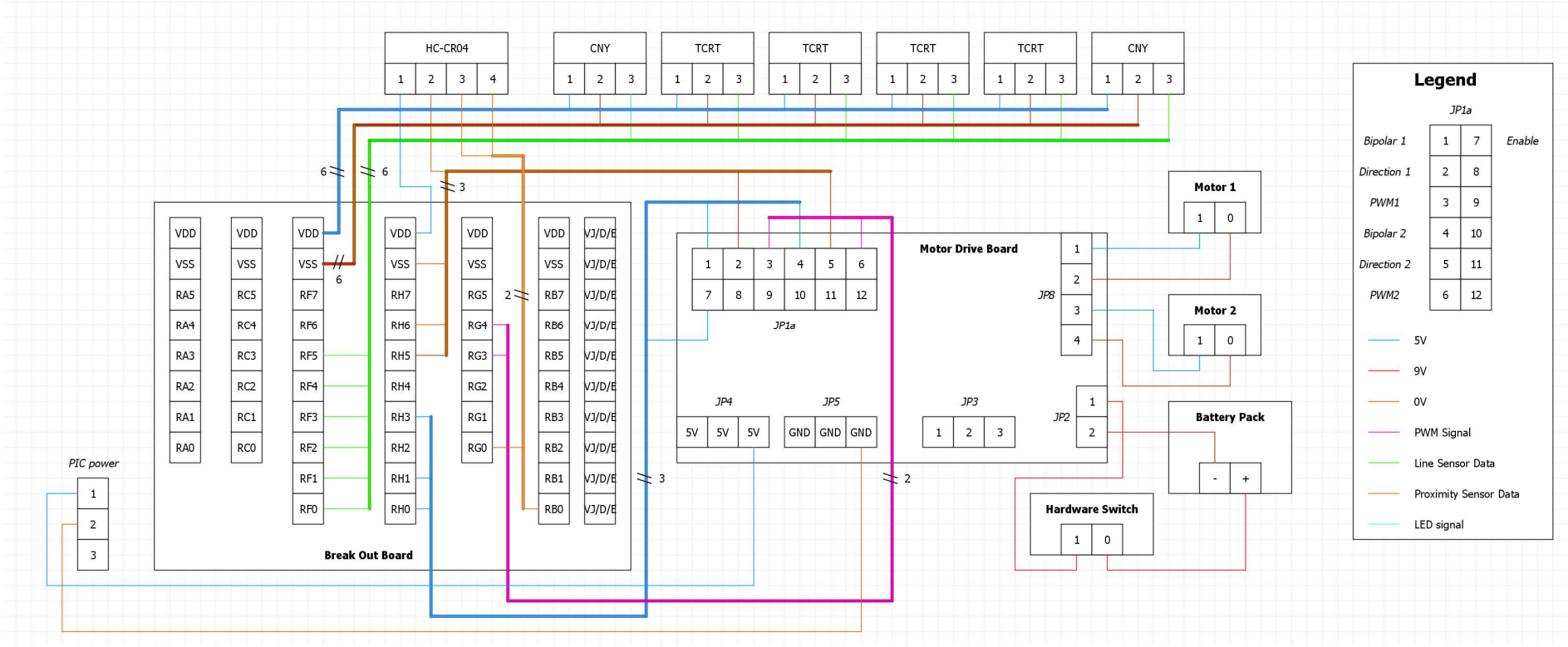
Appendix 8.2.1 - Updated circuit diagram



Appendix 8.2.2 - Updated line sensor layout diagram

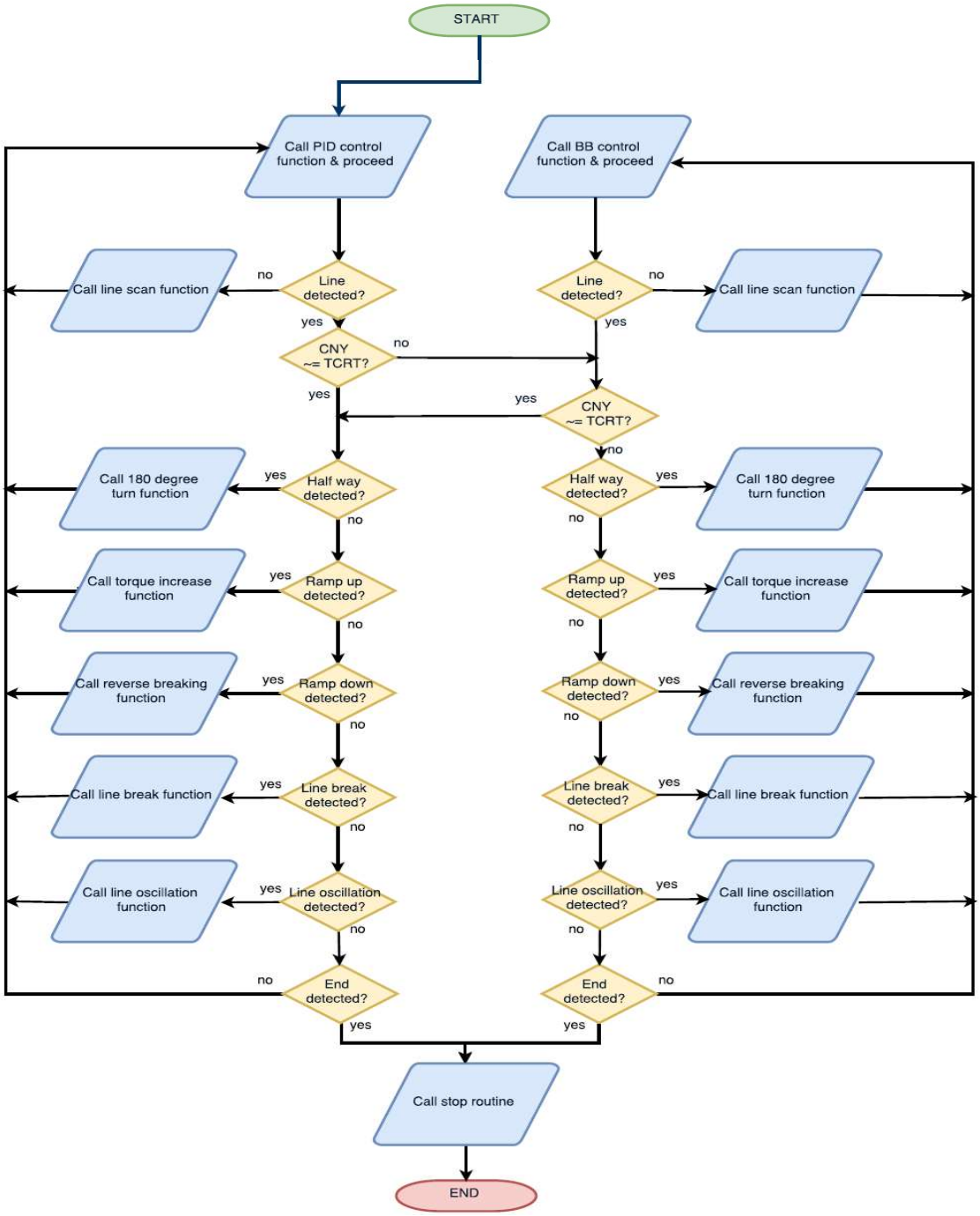


Appendix 8.2.3 - Updated proximity sensor layout diagram



Appendix 8.2.4 - Updated wiring diagram

8.3 Updated pseudocode



Appendix 8.3.1 - Updated pseudocode flowchart clearly showing dual branch software