Line Following Robot

COMPSYS 301 Design Project

Bechet, S.; Mohan, S.; Sukimin, I.S.; Yep, M.
GROUP 6

Abstract

This report encapsulates comprehensive details on the design and the implementation of our COMPSYS 301 Design Project – The Line Following Robot.

We have divided this report into three main components; Analogue design, PCB design and the Software design, as these were major sections in the design and implementation of our project. A mix of analogue, digital, and software solutions have been implemented for this project.

The report sections have been structured to follow the timeline and the progression of the three main project components. The design considerations were first discussed, followed by the design decisions chosen. In each of the design considerations we have discussed the main factors surrounded the required decisions to make, and possible options that could be chosen. In the design decisions, we touch back on the considerations required and explain the decisions that we made based on how the solutions fit the required functionality.

The processes of verification, validation and testing of the analogue solution were further discussed following the first two main sections of the report. These covered the simulation and extensive testing procedures completed on the light sensor circuit. A discussion on the components of the software solution along with testing procedures was then further elaborated.

A detailed appendix with relevant figures, schematics, graphs, and measurements has been attached along with the robot's datasheet.

Acknowledgements

With many thanks to the following:

Dr Nitish Patel, Dr Muhammad Nadeem

Andrew Lai, Joseph Tsoi, John Zhang

Fung Yang, Howard Lu

Contents

Introduction

A large part of computer systems engineering consists of sensing external variables, processing that information in an embedded system, and then modifying outputs to complete a task or function. This design project is intended to develop and showcase our skills at designing and implementing such an embedded system, which senses external inputs and utilises actuators to complete required functionality.

In the case of our autonomous line following robot, light sensors are used to detect the light intensity of specific areas above the robot – with the main goal of allowing the robot to traverse a maze projected downwards from a mounted computer projector. The project incorporates many fundamental concepts of computer systems engineering such as signal processing, embedded design, and actuator controllers.

The light sensor circuits on the robot included a phototransistor, a filtering state, and a gain stage with voltage offset. The microcontroller integrated circuit chip that we used was a PSoC 5, with both hardware and software programmability. We utilised pulse width modulation on the two independently controllable wheels to allow the robot to follow maze lines.

We also included many other popular embedded systems features in our project, to ensure that the robot would be able to carry out its functionality as best as possible. These included radio frequency UART receiving for positioning, quadrature output from the motors, and USB UART for debugging. We also designed and implemented a basic PI controller to maintain constant speed and straight-line movement.

The project also includes two maze traversal algorithms, DFS and A*, which are used to allow the robot to navigate through a maze and collect food pellets.

Analogue

Design Considerations

A total of three circuits design were proposed and considered for this project. Their features and functionalities were then evaluated when deciding on the final circuit design which is discussed further in the Analogue Design Decisions section.

The first circuit (see Appendix Figure 1) consists of three fundamental circuits which are the sensing circuitry, filtering circuitry and the gain circuitry. The DC offset from the signal is removed by the coupling capacitor in the high pass filter in the filtering circuitry. The high frequency signal is then amplified in the gain stage. The DC offset applied within the gain stage ensures that the output voltage is within the PSoC voltage range.

The second circuit (see Appendix Figure 2) proposed consists of two fundamental circuits which are the sensing, filtering and gain circuitries. The AC coupling removes the DC offset from the signal coming from the sensing stage which therefore increases the resolution of the signal measurement. The gain stage introduces a transistor which amplifies the signal. The usage of a transistor is less costly when compared to using an operational amplifier.

The third circuit (see Appendix Figure 3) consists of four fundamental circuits which are the sensing, filtering, buffering and the gain circuitries. Low frequencies are filtered out by the high pass filter, ensuring no amplified unwanted signals in the later gain stage. The buffering stage then isolates the output from the first circuitry from the input of the second circuitry. The DC offset applied then shifts the amplified signal to ensure that they are within the PSoC operational voltage.

Design Decisions

For the final circuit design, the third circuit proposed for the project were chosen as it encapsulates the features that enables the project to meet its specifications and goals.

Since navigation of the robot is be dependent on the light from a projector, it was reasoned that noise would be one of the problems affecting the light sensing circuit. To help reduce it, a high pass filter was introduced with each individual phototransistor to help minimise noise without using any software-based solutions. This allows the cut-off of light with frequencies lower than that of the light projector (120Hz), especially the ambient light at 50Hz. Without these filters, the low frequency signals will be amplified in the amplification stage of the circuit. Resistor and capacitor values also had to be considered to make sure that the corner frequency would limit as much noise as possible without attenuating the frequency from the projected light. A lack of a filter stage would cause inaccurate analogue values when the output signal from the circuit is being fed into the ADC of the PSoC, resulting in the difficulty to differentiate between a wall and a path on the projected map.

The buffering circuitry in the proposed circuit provides isolation from the filter stage that has a high output impedance to the amplification stage which has low input impedance level. This ensure maximum energy transfer between two fundamental circuits while preventing the signal transferred to interfere with the desired operation of the whole circuit. This isolated the filtering and amplifying parts of the light sensor circuit from each other.

Verification

A majority of the analogue verification was done on LTSpice, to confirm the circuit was designed correctly and behaved as expected. When analysing the light sensor circuit, this allowed us to make sure each individual part of the circuit modified the signal as expected. As seen in Appendix Figure 4 the input signal is first filtered and has a DC offset of 2.5V added to it, resulting in a signal with a known offset with a lot of noise removed. To help the ADC read the signal more easily, the signal with this new offset is then passed through a non-inverting amplifier to be amplified. With the graphs obtained from LTSpice, it can be quickly noticed that the signal does not go below 0V or above 5V, which are the limits of the ADC in the PSoC. Similar verification was conducted for the rest of the circuit as well, which allowed for graphs of voltage, current and gain to be quickly generated, allowing for a simple yet efficient verification process of the analogue section.

Validation

After completing the analogue verification, one of the light sensor circuits was breadboarded to validate the design's functionality and check the behaviour more fully. The aim of analogue validation is to recreate the behaviour observed during analogue verification but with physical components and make sure a similar behaviour is observed in the desired environment. For the light sensor circuit, this also involved some testing that a difference between the projected path and wall could be detected. As seen in Appendix Figure 5 and Appendix Figure 6, the output signals from the light sensor circuit's when on a path and on a wall respectively, display a relatively similar behaviour to that of the expected one. Even though the measured responses of the analogue validation are not the identical to the ones predicted during the analogue verification, this is quite normal. Many factors such as the imperfection of the signal being projected, the imprecise values (relative to a simulation) for components used and other similar factors all add up and end up distorting the signal. After further tests with different conditions in the environment, the light sensor circuit was deemed as usable and could be implemented onto the PCB.

Testing

The analogue circuit testing was completed by designing and building a printed circuit board. The PCB was designed concurrently with a schematic diagram utilising multichannel design in Altium. It was designed as a double layer circuit board, so had tracks and planes on both sides of the board.

The PCB was assembled using surface mount components as these are smaller, allowing for additional space on the board for switches and headers. After each section was soldered on, we tested the PCB with continuity and resistance checks to ensure the solder connections were good. After all components were soldered on,

we tested the PCB under the same light conditions as the breadboard and scoped the output to get confirmation that the circuit was working perfectly. We could then read the analogue voltages on the PSoC ADC, and utilise them in our project.

PCB

Design Considerations

Many factors were considered when designing the PCB to ensure that the robot would behave predictably in its environment.

Since the voltage values obtained from the light sensor circuits would be quite small, a non-inverting amplifier was added to enlarge the response added, helping to more easily detect the difference between a path or wall on the projected map. It was also decided that two capacitors — one electrolytic and one ceramic — would be placed in parallel of the DC supply used across the whole circuit, to diminish the risk of providing voltage with noise.

To be able to use the phototransistors efficiently and easily, the PCB had to be placed and aligned correctly on top of the rest of the robot. This required the use of long header pins to be able to superimpose it on top of the robot, which already had the PSoC on top of it. Surface mount technology was used to help reduce the size of each phototransistor circuit, which in turn allowed for them to be placed at specific places without complicating the circuit design.

Design Decisions

Throughout the designing of the PCB, many decisions were taken regarding which values would be more appropriate for components, which component arrangements were more appropriate, and what potential problems could arise from certain decisions. One of those decisions was the noise filtering, amplification, and offsetting of the signal received from each phototransistor. Although all three of these processes could have been completed in software, it was decided that it would be better to do so with hardware as more memory would be available to complete other tasks. Even if further signal processing on hardware was possible (such as the implementation of a Schmitt trigger), it was agreed that not doing so would allow further flexibility in the design.

Another decision taken was to not solder the phototransistors directly onto the PCB but rather make more modular and solder them onto small breakout boards connected to header pins so they could easily be replaced. The main motivation behind this idea was that, since the phototransistors play a crucial role as part of the robot when navigating throughout the map, they should be accessible and easy to replace if need be.

We also added extra switches on the board to allow for multiple modes to be programmed on the robot and defined by the arrangement of the switches. Some switches were also added to control the usage of LEDs used to track the robot when evaluated.

Regarding the values selected for resistors and capacitors, they were all chosen to make sure that more sensitive components would not be at risk. One such example would be the 100KOhm resistor R8_LS2 in Appendix Figure 3 which is the lowest it can be to maximise the size of the signal received without damaging the phototransistor due to excessive current values. At other times, these values were selected to make sure the signal would be preserved as best as possible: the resistor values for the non-inverting amplifier were selected to make sure the signal would be amplified as much as possible without reaching a state where clipping is observed.

Each operational amplifier on the board had a input filtering capacitor placed as close as possible next to it to ensure the voltage inputs were clean. The board also included a large ground plane to minimise eddy currents and thermal load on the ground plane. The 5V supply voltage track on the board ran in a trident shape emanating from the supply point. This ensured that the voltage track didn't have too much distance between any component and the supply voltage.

Software

Design Considerations

Our analogue light sensing design required an analogue to digital converter, rather than relying on a Schmitt trigger stage or some other comparator design to produce a digital output. This meant that our software design had to incorporate the ADC sampling, conversion, and calculations as well as the logic to control the robot's wheels. Therefore, in every iteration of the robot's main while loop we had to sample the analogue signal from each of the six sensor circuits. This took up a lot of processing time, so we had to keep the main while loop as clean as possible, only including the necessary functions and logic required for the specific task at hand.

The motor control software would need to run in every iteration of the main while loop, and although was relatively not memory or processor intensive. However, the motor control logic was time critical, as if the robot reacted too late to an external input then it may be past the time required to correct motor speed to complete a turn or stay on a line. This meant that we had to have the control logic directly after the ADC conversion code in the main while loop, to minimise delays between the input and outputs of the system.

The maze traversal algorithms were tested on MATLAB, but had to be implemented in C for them to work on the PSoC 5. For this to be successful, we had to take into consideration the memory used by the algorithms, to ensure that we were using as little as possible. These algorithms were also extremely processor intensive so we had to consider where and when we would run these algorithms.

Design Decisions

We decided to run the motor control logic sequentially after the ADC logic, rather than running the two processes concurrently. This ensured that we were deadlock and starvation free, and that no data was read or written incorrectly. This also meant that the motor control logic only operated once every light sensor value had updated, increasing the accuracy of our robot as it ensures all data is new, but decreasing the response time.

The radio frequency data that we obtained was in the binary format from our RF receiver module. We decided on this as it meant that no ASCII parsing was required, saving the execution time required. Additionally, the positioning data was read directly into a custom data structure so that the values could be read instantly and easily.

Based on which mode was selected, we ran the entire required maze traversal algorithm before the robot started traversing the maze. This meant that the robot did not need to compute its path while it was also reading from light sensors and controlling motors based on maze logic. The maze traversal algorithms used are discussed in the Maze Algorithms section below.

Speed and distance measuring was done on a timed interrupt. This meant that we could easily divide distance travelled by the timer period to get the average speed for that period. This is more accurate than a polling approach as the time between readings is exactly the period required each time.

Motor Control

The two motors attached to the robot works independently. Each motor has its own PWM component connected. Due to the robot's hardware configurations, motor 1 naturally rotates at a faster rate as compared to motor 2. Therefore, the software plays a big role to ensure the uniformity and synchronisation of the two motors.

To regulate the speed of the motors, testing was carried out at different battery voltages. Through trial and error, it is found that within the robot's operational voltage range, the PWM value for motor 1 and motor 2 constantly differs by 2. This linear relationship is then applied in every single speed correction algorithm made in the program.

Motor speed measurements and PWM values calibrations were also carried out by always placing the moving robot on the ground as it

ensures robot is travelling at its natural speed which is when normal reaction due to its weight and frictional force is considered.

For the robot to traverse a maze autonomously, we had to handle all possible input and logic configurations so that the robot did not lose its position or encounter an unaccounted situation. When travelling along straight lines on the maze, both motors ran at a similar straight-ahead speed. If the light sensor array detected that the robot was going off course, then the corresponding motor would speed up to slightly turn the robot back on course. This is an example of a negative feedback controller. When arriving at an intersection, the robot checks what kind of turn it is required to make at the intersection. It then carries out the turn by stopping and turning on the spot – moving one wheel forwards and another backwards at the same rate. Turns are finished when the light sensor array shows the correct values.

Maze Algorithms

For the maze level 1 we decided to use a Depth First Search (DFS) algorithm, and for the maze level 2 we decided to use A*.

DFS is a graph traversal algorithm which starts at the specified start location and explores as far as possible along each branch before back tracking. It repeats this process until all nodes in a maze have been reached. One of the reasons we decided to use DFS for traversing the entire maze is because it is memory efficient, since memory requirement scales linearly with respect to the size of the maze, and the embeddedness of the system requires minimal memory usage. DFS has a similar time complexity of traversal to other algorithms such as Breadth First Search.

We chose to use A* for the maze level 2 because it was the best algorithm for finding the shortest path between two points. The A* algorithm solves problems by searching among all possible paths from the start to the target location, for the path that has the least cost, which is the least distance travelled in our case. A* calculates an F score for each node it visits using the formula $f(n) = g(n) + h(n)$, where $f(n)$ is the cost of getting from start to target location via that node, while $g(n)$ is the cost of getting from start to the current node, while $h(n)$ is a heuristic function that estimates the distance from the current node to the target location. The A* algorithm, begins from the specified start location and keeps finding the next adjacent node with the lowest F score until it reaches the target location.

For maze level 1, the DFS algorithm was used to generate a list of coordinates that needed to be travelled along to traverse the whole map. These coordinates were then converted into a list of turns that the robot then followed. Similarly, for maze level 2 the A* algorithm was used to generate a list of coordinates that would be traversed to consume all food pellets on the maze, taking the shortest path to each. These coordinates were then turned into an array of turns, identically to the maze level 1.

Testing

Testing the motor control software was undertaken alongside the development of the code. After a major section of the code was finished (such as handling a particular type of intersection), we would test the robot completing that intersection multiple times on the practice map. This reduced testing and debugging time as it was easier to pinpoint and fix errors.

To test the DFS and A* algorithms we used an online GCC compiler to simulate the robot going through the actual maze. To test the correctness of the DFS algorithm we ran it with different starting locations and manually checked the coordinates output list of the algorithm to determine that all parts of the maze were visited and that no invalid coordinates were present. We also determined that DFS was reasonably efficient at finding a path that visited the entire maze.

To test the correctness of the A* algorithm we ran it with different starting locations and found that each time it would find the shortest path that consumed all the food pellets. We also determined that no invalid path coordinates, such as walls or out of bounds were generated.

Conclusions

The aim of this design project was to develop our skills at the design and implementation of an embedded control systems project. From this line following robot project, we have learnt and developed important skills for future projects at both university and industry level. The experience of working as a team to design and implement an integrated hardware and software solution will be invaluable to our future careers. These skills not only include the technical aspects of designing and implementing both hardware and software solutions, but also the teamwork skills required to trust each other and work together efficiently.

The analogue light sensor circuit design and implementation provided us with an opportunity to design and implement a physical embedded system, which interacts with its surrounding environment to feed input data into a PSoC development board. The PCB design and implementation allowed us to practice our skills at developing our own unique method of detecting light patterns to provide useful information to the PSoC. Finally, the software design and implementation helped us to develop our embedded software skills, by implementing our own control system to allow the robot to navigate a maze autonomously.

Overall, we believe we were successful in completing the project to a high standard. The solution which we have designed and implemented is able to successfully navigate any standard maze in both maze levels.

Appendices

Figures

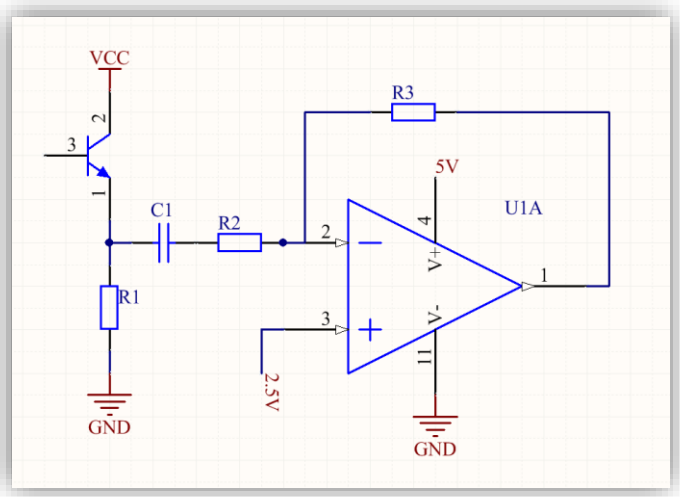


Figure 2: Proposed Circuit 1

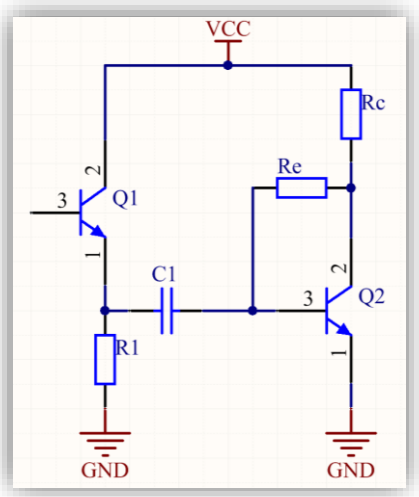


Figure 1: Proposed Circuit 2

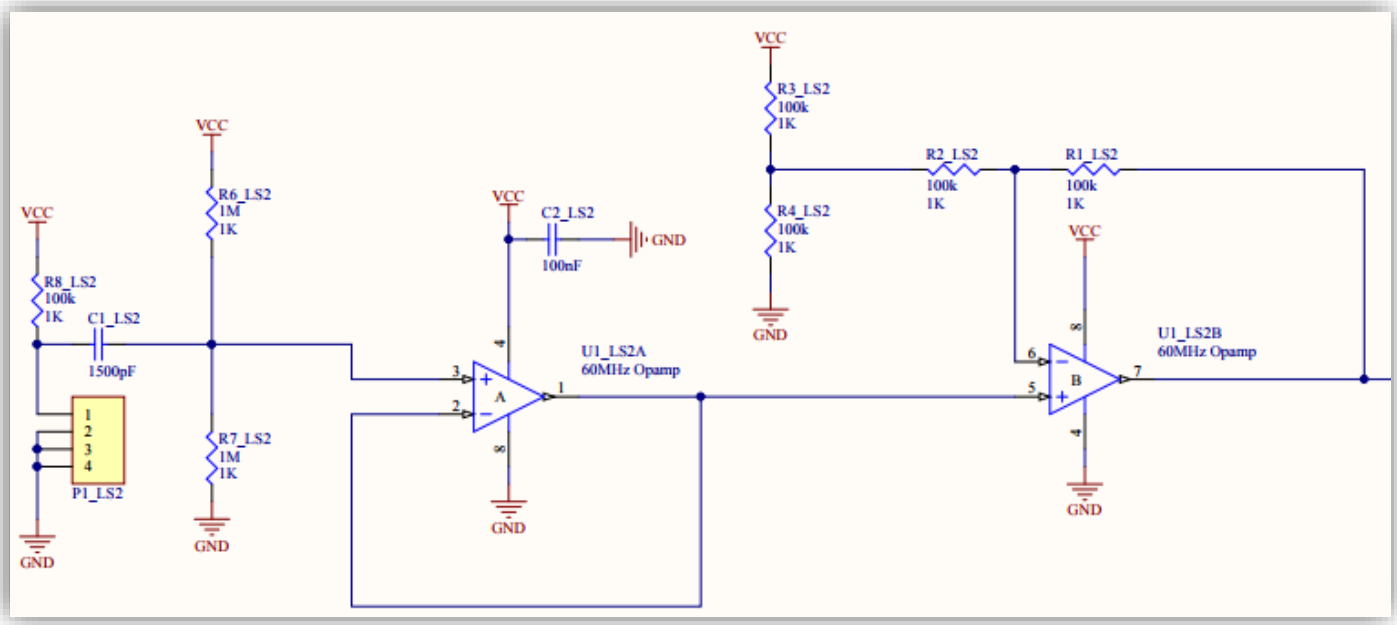
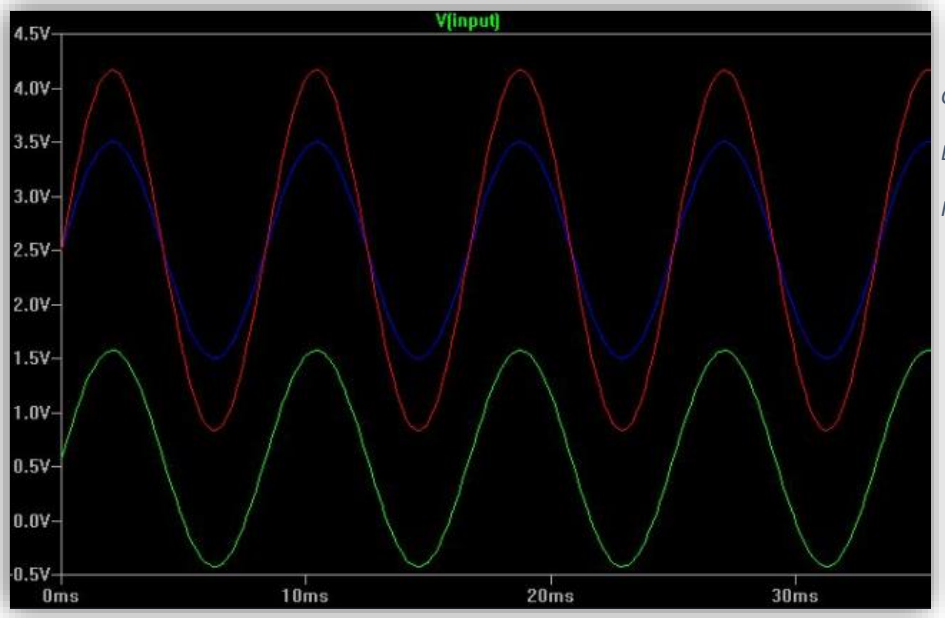


Figure 3: Proposed Circuit 3 (Selected Circuit for Project)

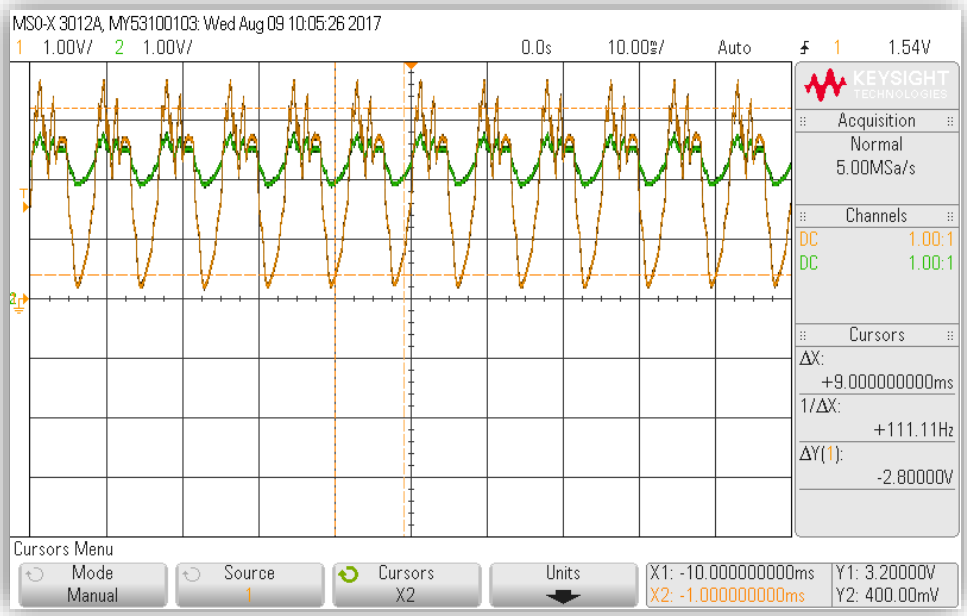


Green: Input Voltage

Blue: Before gain stage

Red: Output Voltage

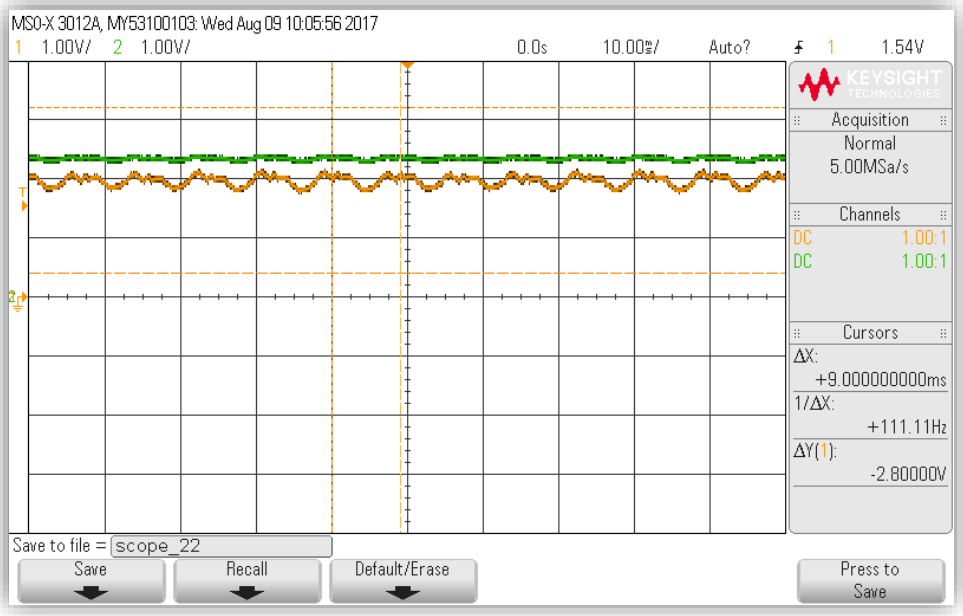
Figure 6: LTSpice Simulation of Light Sensor Circuit



Green: Before gain stage

Orange: Output Voltage

Figure 6: Oscilloscope output from breadboarded circuit, under light

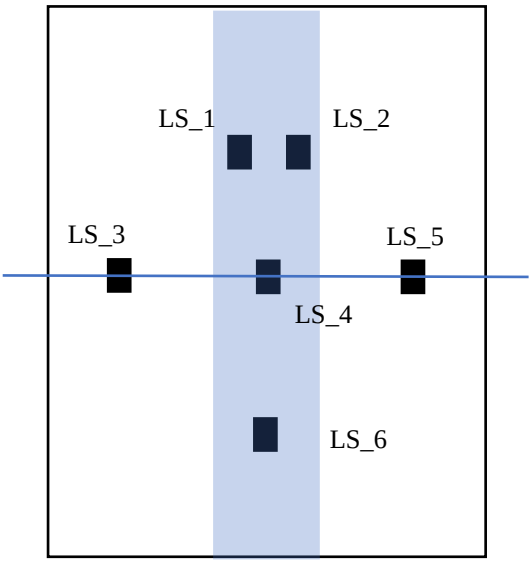


Green: Before gain stage

Orange: Output Voltage

Figure 6: Oscilloscope output from breadboarded circuit, under line

Light Sensors



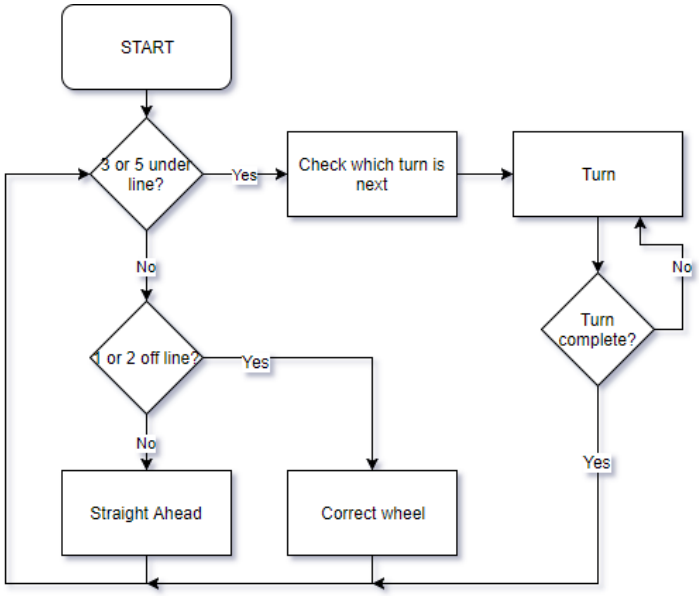
The above figure shows the light sensor array that we used. The blue line represents the wheel axis, and the blue shaded rectangle represents the straight-line shadow. Light sensors 1 and 2 are used to keep the robot moving straight ahead on a straight line, if one of the sensors goes off the line then the corresponding motor speeds up to get it back on the line. They are also used to detect when turns are finished, and the robot is now facing the new line to travel straight on.

Light sensors 3 and 5 are used to detect intersections, as they are placed wide enough to not be affected by the straight line the robot is travelling on. When an intersection is detected, the robot checks which direction to turn and completes the turn.

Light sensors 4 and 6 are mainly used for correction and dead-end detection. These ensure the safety and reliability of the robot to ensure it does not go off the line and get lost.

Software Flowchart

Basic software flowchart for the maze traversing logic.



Contributions

Project

- *Sylvain:* Analogue Design, Verification, Validation, Testing, PCB Design, robot benchmark testing.
- *Savi:* A* MATLAB code, A* C code, DFS C code, Robot Distance and Speed Algorithms, Software Testing.
- *Ira:* Analogue Design, Maze and Benchmark Turning Algorithms, Maze Turn Following Algorithms.
- *Mark:* Software Motor Control, PCB Construction, Maze Turn Following Algorithms.

Report

- *Sylvain:* PCB Design Considerations, PCB Design Decisions, Analogue Verification (LTSpice), Analogue Validation (Breadboard)
- *Savi:* Software Maze Algorithms, Software Testing, Datasheet
- *Ira:* Analogue Design Considerations, Analogue Design Decisions, Software Motor Control, Abstract
- *Mark:* Software Design Decisions, Software Design Considerations, Analogue Testing (PCB), Introduction