Jan Derlatka, 86, JN, Info (Team Leader)

Lucca Pereira Martins, 86, JN, Info

Maya Roy Prabhakaran, 99, M, Mech

Jenny Hardwick, 99, M, Mech

Szymon Piatek, 110, PEM, Elec

Noah Barker, 110, PEM, Elec

[Name, Lab Group, College, Sub-team]

*TEam 4 IDP L1*

Team Name: Porous a Drink

Robot Name: Wall-E 2.0

# Design and Sub-systems:

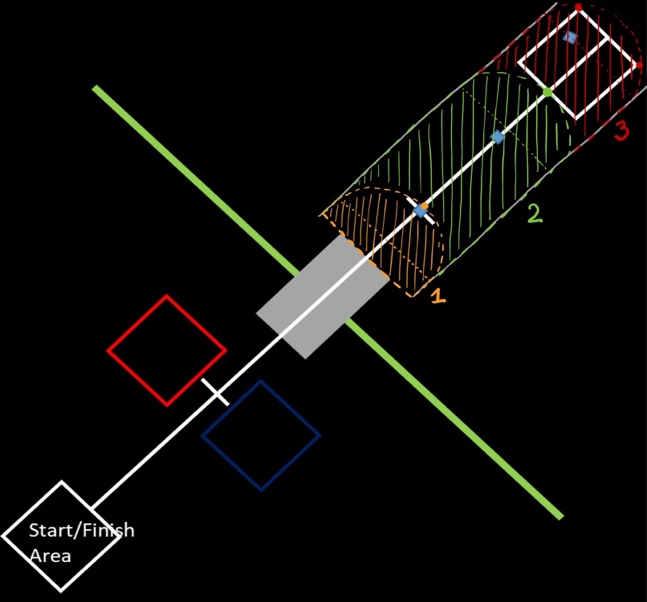


Figure 1 Sweeping Arcs for Each Cube

## Approach

We have opted for a general, works for all approach, where we account for all possibilities of where the cube could be by sweeping all possible positions, removing the need for cube detection (ref fig1.). To collect the cubes, we use two large arms which will open when sweeping, drive the car forwards to a maximum value, and close around the cube, but not squeeze it. At this point we will measure porosity by using infrared sensors at one corner of the cube. The robot will rotate and return over the ramp with open arms, pushing the cube forward. When depositing the cube back, the robot will turn into the box, putting the first cube deepest into the area, then reverse out before rotating and continuing.

## Moving

Keeping robot movement as simple and fundamental as possible and minimising the number of turns it must make (hence reducing the time spent off the central white line), will ensure an accurate and robust result. We decided on 2 rear wheels connected to a motor with 2 ball rollers at the front of the vehicle (ref fig2 & fig3). This would allow us to turn on the spot when delivering the cubes, but also micro adjust to ensure the robot stays on the central white line. The argument against 4 wheels, whether fixed or rotational, was that the requirement for an additional motor would increase the complexity and the weight of the robot, whereas the ball rollers would provide more flexibility in terms of motion and micro adjustments. We explored the possibility of driving backwards over the ramp and into the box, removing the need to turn on the spot, but to implement this we would have had to use a central axle with 4 ball rollers in each corner. Given the robot rotates about the rear axle, when trying to follow the line in reverse, the amplitudes of adjustments were higher, putting it off track. We decided not to pursue this idea given insufficient time to implement such a big design change.



Figure 3 Plan View of Robot Design

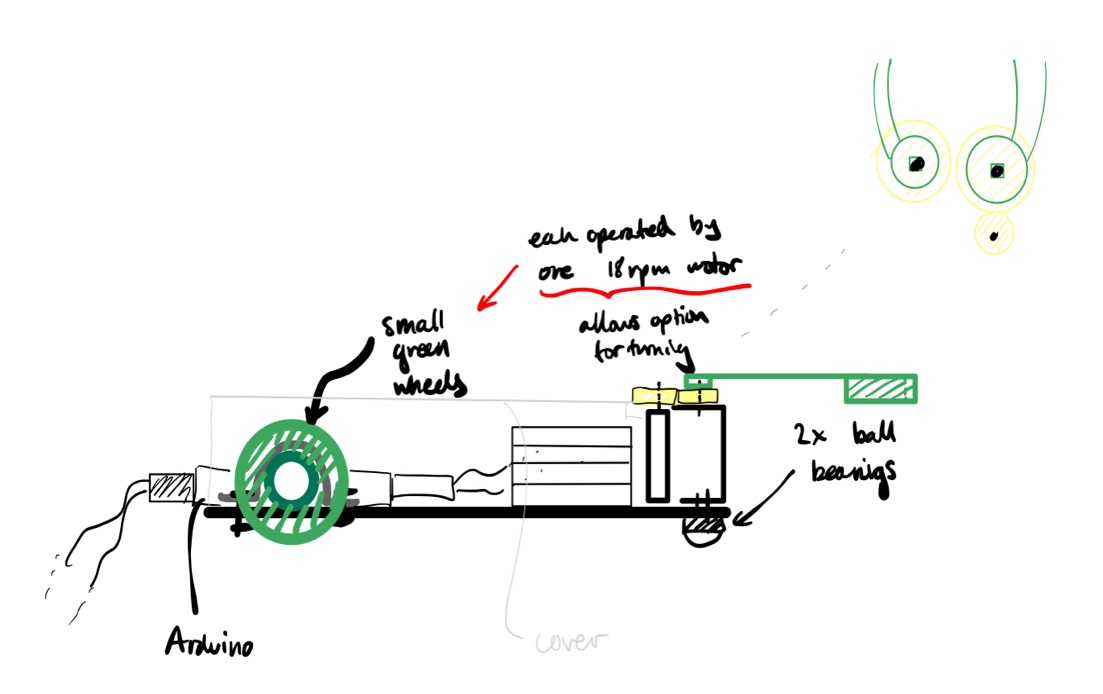


Figure 2 Side View of Robot Design

## Arms

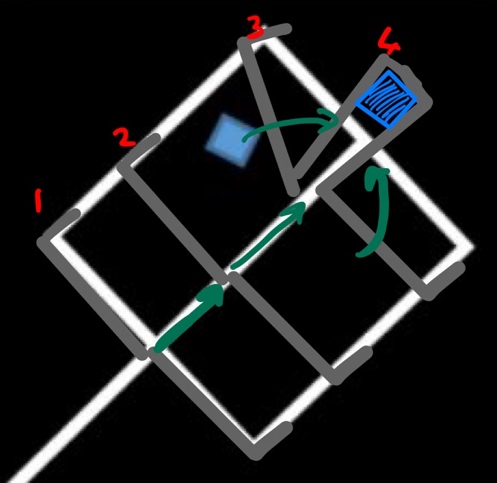


Figure 4 Pincer Motion for Arms

As mentioned above, the cubes will be collected by two arms in a pincer motion, sweeping a wide arc which covers all possible cube positions (ref fig4.). If the cube is towards the start of the possible locations, the robot will gently push the cube forward, and once the arms are closed and it has detected the cube type, the robot will turn around, open the arms and push the cube into the box. We decided on this approach to entirely remove the need for cube detection, which would have been one of the most complicated steps when finding the third cube. The main risk is knocking other cubes too far out of place, but careful and gentle arm movements will ensure this is not a problem. The arms were operated by a servo which we could adjust the angle of in the code and built from MDF, with a flange a similar height to the cube. The simplicity made the arms robust and easy to implement, giving the robot a certain flexibility regarding the task it approached, meaning it could easily be adapted to carry other objects.

## Line following

We used two line-sensors for the robots main steering (ref figs 5a, 5b & 6), placed either side of the white line and returning a binary output to the Arduino. Once the sensors detected the white colour, the motor would slow down one of the wheels accordingly to turn it back onto course, with a time delay in place to prevent repeated instructions being sent in a short time. When leaving the line to deposit the cubes, the sensors would be ignored, ‘deactivated’ so to speak, and the motion would be programmed. The robot turns into the box, placing the first cubes further in, before reversing out and rotating as to avoid knocking the cube with its arms. Similarly, when turning on the spot, the robot would turn an angle of 170 degrees before readjusting by detecting the line. This was the best option given the position of the sensors, which meant that if the robot was initially skewed, when using the sensors it would stop at the first detection of the line, which could be in forwards direction.

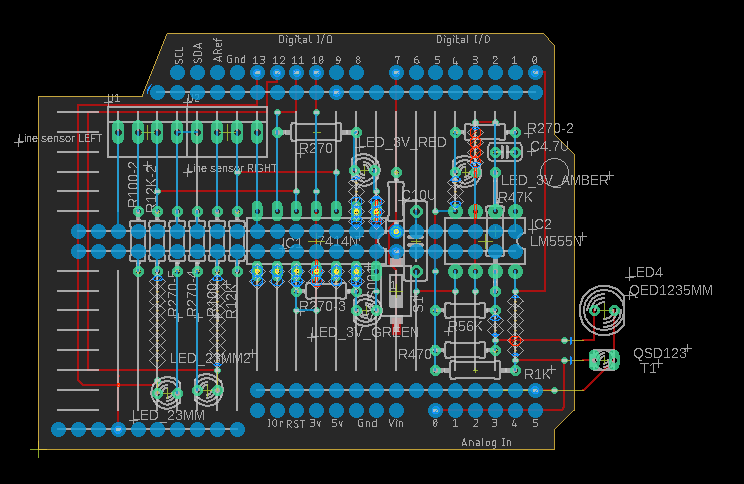


Figure 5b Bottom Layer of PCB

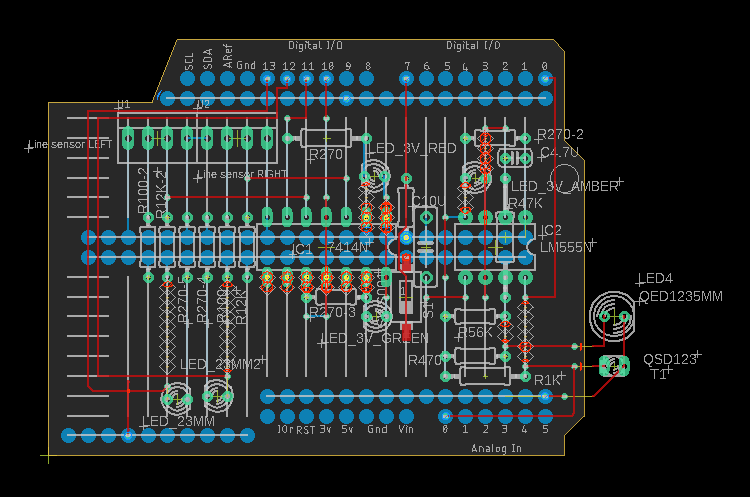


Figure 5a Top Layer of PCB

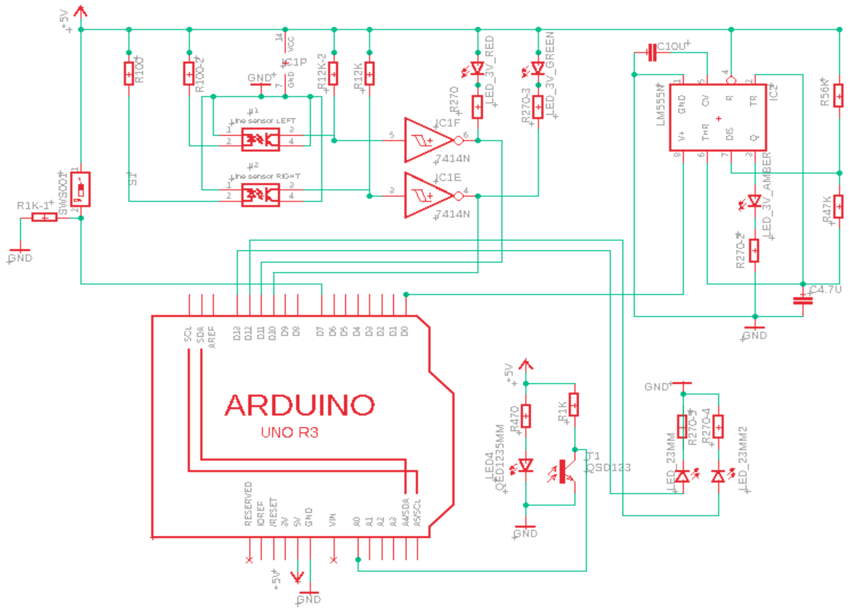


Figure 6 PCB Layout

## Determining Cube Porosity

When determining cube porosity, we opted for an infrared sensor placed in the corner of the arm, and hence shining and receiving light through a corner of the cube (ref fig 7). The more porous cubes would return a higher reading, which we could convert into a digital output. Initially, we tested the sensors at either side of the cube; an emitter in one arm and the receiver in the other but noticed that the difference in measurements between the two cubes was insufficient to be robust, as well as other interfering factors such as cube location.

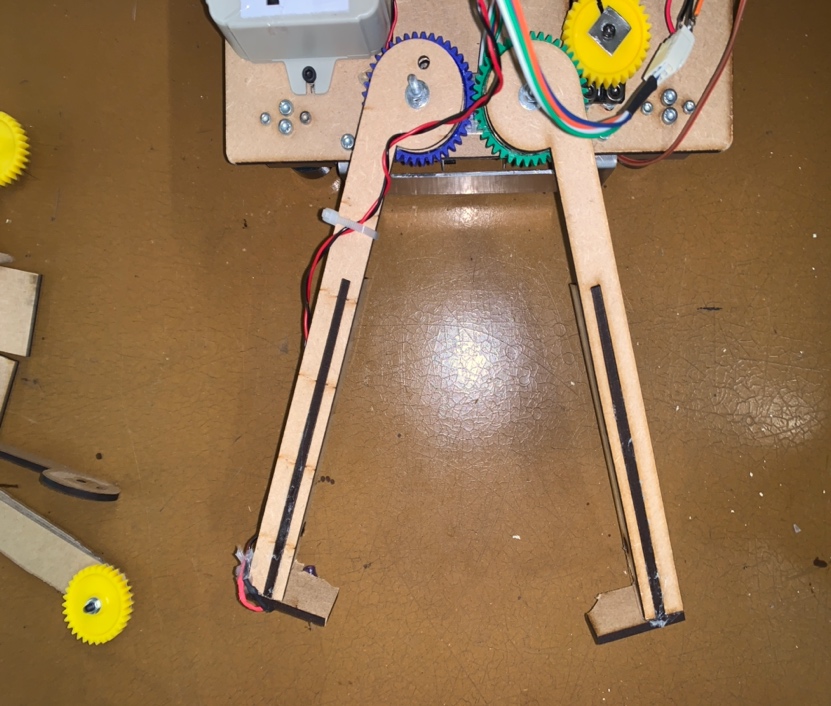


Figure 7 Infrared sensor in corner of arm

## Time-Based Distance Measurement

To keep track of the robot’s position and know when to stop sweeping areas, we used a time-based distance measurement. Manually experimenting with the robot’s speed on flat ground, rotating on the spot and when crossing the ramp and calibrating it appropriately, we found that our values for the time taken to complete each section were very accurate, translating to errors of only a few millimetres on the table. To further reduce error, given our ability to detect intersections with the line-sensors, we could ‘reset’ the time at each intersection, breaking it up into smaller tasks rather than one long execution.

## Implementation:

Figure 8 shows how we aim to integrate the three teams, with orange representing mechanical, white: information, and blue: electrical.

## Major Decisions

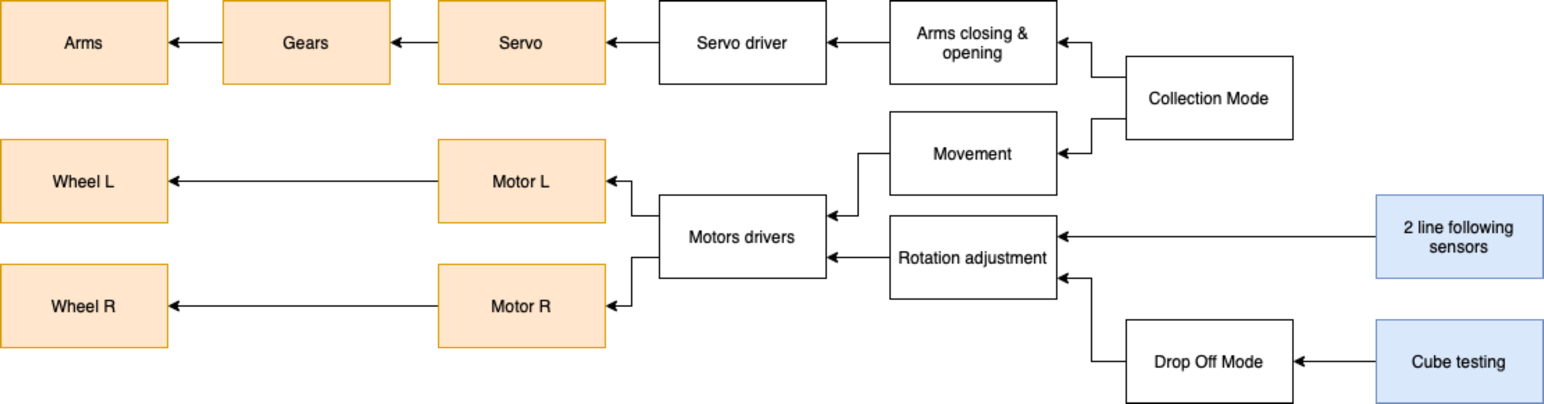


Figure 8 Sub-team Integration

Although it would be unreasonable to say we made no major decisions, the reality was there were very few. The final product was close to what we had originally designed, and the method was also very similar, but some of the key choices we had to make include:

1. Rotating the robot rather than reversing it

Initially, we intended on grabbing the cube and reversing over the ramp with the arms closed. This would allow us to reverse up to a point near the starting box and turn a wide arc into the drop-off areas. However, when attempting to follow the line in reverse, we found that given our use of a rear axle, the adjustments the robot would make had much higher amplitudes than when driving forwards. The only solution to this would have been to move the axle to a central location and use 4 ball rollers, one in each corner instead. This meant a complete overhaul of the design and current robot build, which given the time constraints and a lack of necessity, we chose to rotate the robot on the spot instead. The problem of knocking any cubes ahead out of place arose, but this was easily avoided by reversing a little before rotating.

1. Opening the arms when crossing the ramp

The idea of hinged arms was discussed, where the arms could elevate as they contacted the ramp while crossing over, a concept very much related to reversing with the arms closed. However, this not only proved impossible, but also inefficient. We discovered that by adding a small metal plate to the front body of the robot (ref fig9), we could safely push the cube along without risking losing it or it lodging under the robot. Given the design of the arms, which covered the span of the ramp, this was very unlikely. This meant that once we had determined cube porosity and turned around, we could leave the arms open for the remainder of the delivery.

Figure 9 Metal Plate at Front Body of Robot



# Risks:

**[RISK, LIKELIHOOD, MINIMISING]**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Likelihood**  **(1:least -5:most)** | **Risk Reduction** |
| Failing to collect cube or slipping out in transit | **1** | Ensure arms sweep the entire area gently and arms are similar height to cube |
| Knocking into other cubes | **3** | Best avoided through trial and error, with careful consideration of all possibilities |
| Failing to cross ramp | **5** | Adjustments have been considered should this happen, ie. raising arms |
| Failing to determine porosity | **4** | Ensure sensors are accurate and well-placed, and code is robust at evaluating |
| Losing track of white line | **2** | Sufficient testing and robust code to ensure adjustments are smooth & correct |

# Project Management:

## Gantt Chart

The Gantt Chart helped keep all team members on track and aware of upcoming deadlines. We updated it frequently and kept track of deadlines and expected milestones, allowing some leeway for each task to ensure punctuality. The initial estimates for time were accurate, although integrating new parts together took longer than expected.

Chart

Description automatically generated

Chart 1 Gantt Chart

## GitHub Repository

Using a single repository to share and store all files such as code, CAD models, sketches and graphs, helped simplify distribution and version control across teams, maximising coordination and communication.

# Testing

Some of the problems which arose during testing and prior to the competitions were:

1. Scrapping of the arms against the ramp

Some of the ramps were slightly different across the different tables, meaning on some of them the open arms would scrape against the ramp, and almost prevent the robot from crossing it entirely. We solved this by filing down and curving the corners of the arms, which although didn’t completely remove the scrapping, it decreased it significantly.

1. Increased sensitivity of line-sensors when using circuit board

For the first competition, the electronics were on a breadboard before we transferred them to a soldered circuit board ahead of the final competition. We noticed the line-sensors became much more sensitive after doing so and would detect intersections at places which there weren’t any, such as going up or coming off the ramp. This was detrimental as our time-based system relied on correctly detecting the intersections to move the robot onto the next phase. There wasn’t an easy solution to this and would require rewriting the intersection detection code to include some form of verification.

1. Calibration of infrared sensors

Setting up the emitter and receiver at such an angle and position as to get consistent results proved much harder than expected. A slot was cut into the arm, allowing the sensor to be adjusted and moved around before ultimately being glued in place. However, we struggled to find said position and because of this, the cube detection was largely inconsistent and incorrect.

# Specification

|  |  |  |
| --- | --- | --- |
| **Robot Specification Criteria** | **Criteria Met (Y/N)** | **Comments** |
| No sharp edges and safe around humans | Y | Any edges considered sharp were filed down before competition |
| Displays a flashing amber light when moving | Y | Use of a 555 timer circuit to generate flashing behaviour |
| Modular construction & standard components | Y | Each component was easily removable and interchangeable |
| Robot started in a controlled manner | Y | Started using a physical button |
| Neatly installed colour-conforming cabling | Y | Refer to figure 10 |
| Robot must fit entirely in start area | Y | Robot was relatively small so this was never an issue |
| **Task Specification Criteria** | **Criteria Met (Y/N)** | **Comments** |
| Robot reaches alternate side of table | Y | Easily completed |
| Block transported to opposite side | Y | The arms functioned correctly and were able to transport the blocks, although occasionally the cube could get jammed under the arm |
| Correct LED displayed when transporting the block | Y & N | As discussed above, the cube detection lacked the robustness we desired, meaning it could only correctly detect the cube 50% of the time, which is what happened in the final competition, once correctly and once incorrectly. |
| Delivering block to correct area | Y & N | Similar to above, this meant some of the cubes were in the right boxes and some were not, however all were within the area limits and the robot did not push any cubes out when manoeuvring out of the box. |
| Robot returns to start area | Y | Although complicated by the intersection detection, we could manually experiment and calibrate the robot to return to the start area |

# Competition Performance

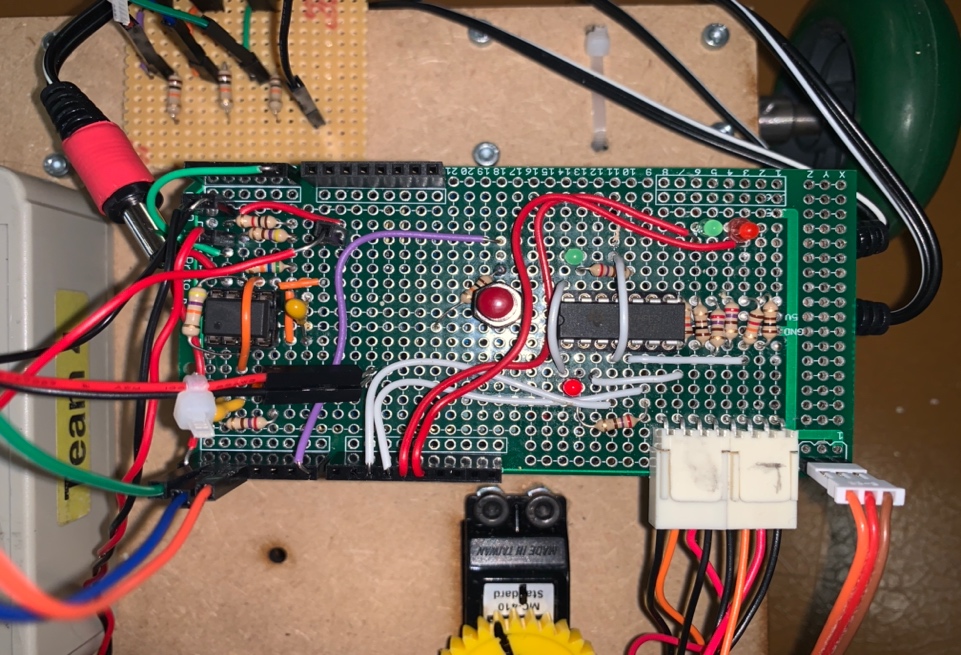


Figure 10 Physical PCB Board with Correct Cabling

The issues that prevailed in the competition were the incorrect detection of intersections and the inconsistent use of the infrared sensors to determine cube porosity. We could consider two solutions for the increased sensitivity in the line-sensors, causing the robot to move onto movement stages ahead of time and even miss out cubes entirely. A physical adjustment would be to alter the resistors and circuit in a way that the signal threshold would be higher, meaning there would be no output on the fringes of the line, only when the sensor was entirely on or off, providing a sort of buffer. The other alternative, which is software based and the more likely one, would be to include this buffer in the code, which could be done by requiring repeated inputs indicating an intersection before an output was provided, or by an overhaul of the distancing process.