Milestone 4

EEE3099S – Engineering Design

6 November 2020

Group 28

Matthew Terblanche
TRBMAT002

Michael Wetzel
WTZMIC001

Table of Contents

Introduction

Building a line following robot introduces an engineering student to real life problems and allows them to express their creativity and problem solving skills. Under normal circumstances a physical robot would've been built consisting of lego blocks and electronic components. Due to the corona-virus pandemic, access to the university and its equipment was not possible and the project had to be done virtually.

The line following robot was digitally modeled using simulink. Some parameters of the robot were given to us, namely the H-bridge, Sensor and Wheeled Robot model. Our instructions were to design a sensor system that consisted of no more than 5 sensors, with a penalty for more sensors. Our robot would have to follow a number of different paths, some with slight turns and other with very sharp turns, and come to a stop once it is at the end of the track. Half of our mark would be calculated by the implementation of the robot and the other half based on performance against the rest of the class.

We started the project testing various different layouts of sensors. We solved this problem in a trial and error way. As we were instructed to use as little sensors as possible we first tried the model with only 3. After testing this however provided complication when a track with sharper turns was presented. We then defaulted to using all 5 with 1 sensor being on the line, 2 right next to the line and the other 2 a bit further away from the line. The robot completed the tracks in a desirable time. Even though we did not actually build the robot, the virtual environment taught us a different set of skills that we will surely apply one day after our studies.

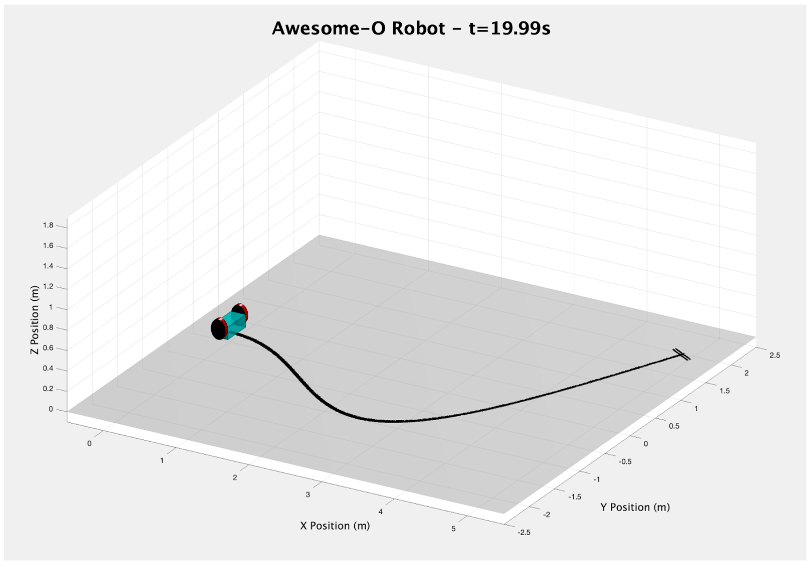


Figure 1: Animated view of robot simulation

Modelling & Experiment

In the real world when an engineer is presented with a problem, many of the pieces will already be present. In this project that parameter was the wheels. Instead of having access to the different wheel ratios we were given wheel parameters. This acted like nonlinearities and imperfect wheels. Most of the process consisted of trial and error. As we were instructed to use as little sensors as possible, up to a limit of 5, our first thought was to use the least amount of sensors.

Our first idea was to place one sensor on either side, very close to the line. If one sensor detects a line, for illustration purposes let's say the left side, the right side wheel should have its speed reduced. The right wheel will now turn at a slower rate than the left side with the result being that the wheel turns to the right. This is a recursive process that occurs as the robot moves from the left side to the right side. Many different wheel speeds were tried and code was implemented to account for the stop condition and this model worked for the first track. When the robot we tested on the second or third track this was not the case. Turns out that when the turns are too sharp or the robot doesn't perfectly start on the line, only 2 sensors does not suffice

Our second idea was to keep the remaining sensors but add one more on each side. After trial and error with wheel speed we managed to get the robot running on track 2 and 3 but still had complications with the stop command. We used the idea that if all the sensors detect that there is a line, then the robot should issue a 0 torque command to the wheels and it should stop. This model of the robot seemed to work well with the default value of sunlight. However, as soon as we changed the time of the day we kept getting errors. This prompted us to use our last available sensor to calculate a mean value for the floor reflectivity and the line.

Lastly we added the 5th sensor in the middle of the robot. This is to deal with the time of day and changing reflectivity. We scoped the middle sensors data and calculated a mean line value. From this we could determine how the floor model changes and thus how our line sensor model will change during different times of the day. After testing the different tracks with different day time parameters we found the performance to be satisfactory and finalized our design.

Mechanical Design

For the Mechanical design we used all 5 of the allowed sensors. Figure 2 shows the layout of the sensors and mechanical design of the robot.

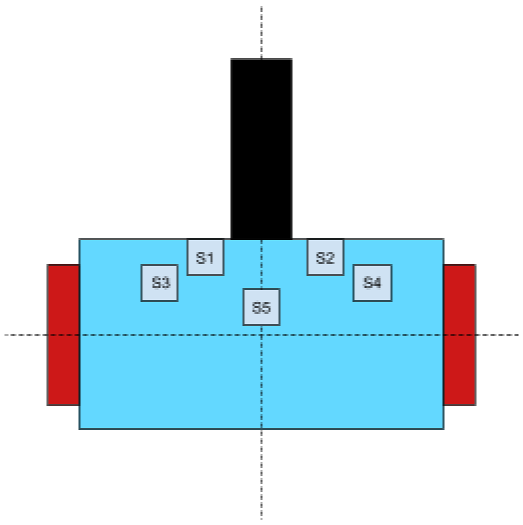


Figure 2: Robot Mechanical design

Sensor 1 and 2:

- These sensors were placed close to either side of the line. They were the general purpose sensors that would be used the most and are the most important to the robot (to be honest all 5 sensors are equally important). When the robot is going straight or is turning slightly, these are the sensors being used.
    - Positions are:
        - Sensor 1: (sX = 0.1, sY = 0.019)
        - Sensor 2: (sX = 0.1, sY = -0.024)

Sensors 3 and 4

- These sensors were placed a bit further from the line. Their use was on tracks that have sharp turns or when the robot starts at a turn when it isn't exactly on top of the line. In these cases sensor 1 and 2 aren't enough to accurately drive the robot. It is worth to note that these sensors are placed a bit behind sensors 1 and 2. The reason for this is if the robot arrives at the finish line and is a bit skew, the 2 front sensors will detect the double line first and straighten out the robot. At this time the 2 outer sensors will detect the double line and the robot will come to a stop.

- Positions are:
  - Sensor 3: (sX = 0.082, sY = 0.085)
  - Sensor 4: (sX = 0.082, sY = -0.082)

Sensor 5

- This sensor is placed in the middle of the robot and is used to calibrate the sensor values monitored. We are modeling this as a real life system and at different times of day the floor and tape will have different reflectivity values. The data from this sensor (which will always be on the line) is scoped and a mean value for the tape is determined.
  - Position is:
    - Sensor 5: (sX = 0.01, sY = 0)

Below is a diagram showing how the mechanical wheels connect to the robot and how the sensors are connected to the controller.

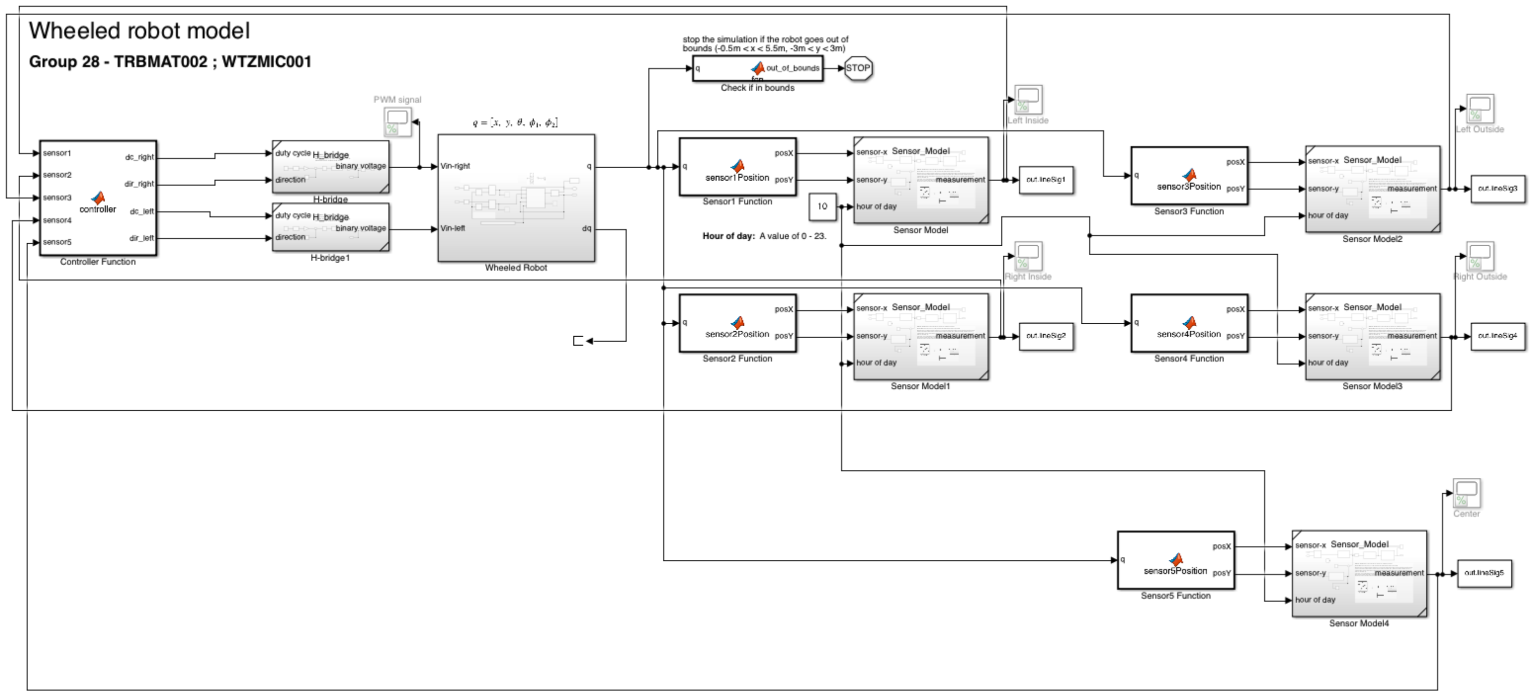


Figure 3: Digital Model of mechanical system

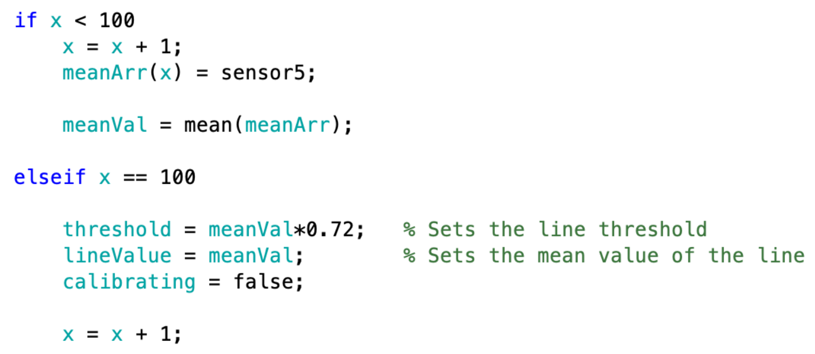
Algorithm Design

We first tried a simple strategy of if-else statements. If the line was detected by the left sensor then the robot must turn left, otherwise if the line was detected by the right sensor then the robot must turn right. If no sensors detect the line then the robot must go straight. And lastly, if the line was detected on both sensors the robot must stop.

A fixed line threshold was used for this method, originally set at around 0.75.

This method did not work on all tracks and at all times.

We then decided to put a sensor in the centre of the robot so that it is always on the line at the start. The controller would then take the output from this sensor and record the first 100 samples and calculate the mean line value. This value is then the new line threshold to be used as it accounts for the changes in lighting conditions for various times of day. The calibration code is shown in Figure 4.



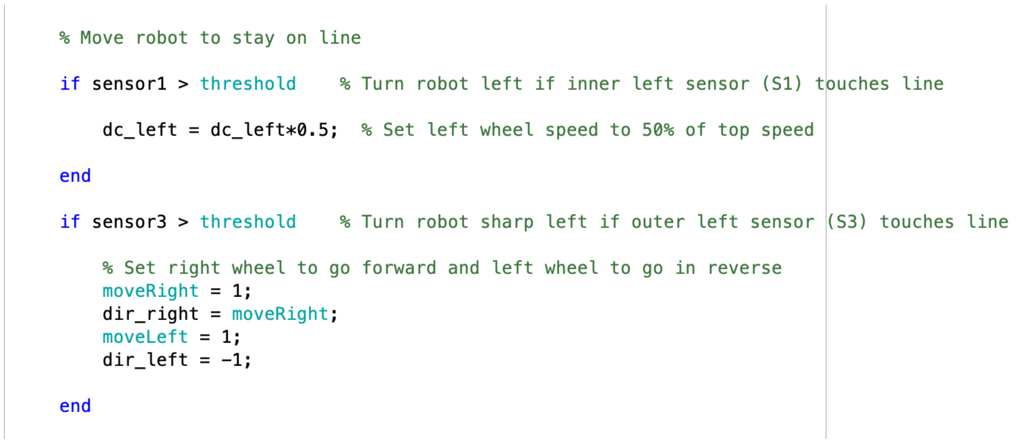
```
if x < 100
    x = x + 1;
    meanArr(x) = sensor5;

    meanVal = mean(meanArr);

elseif x == 100

    threshold = meanVal*0.72;    % Sets the line threshold
    lineValue = meanVal;         % Sets the mean value of the line
    calibrating = false;

    x = x + 1;
```

Figure 4: Line calibration code

The threshold was set to 72% of the mean line value calculated. The 72% was found by running many experiments and trial and error.

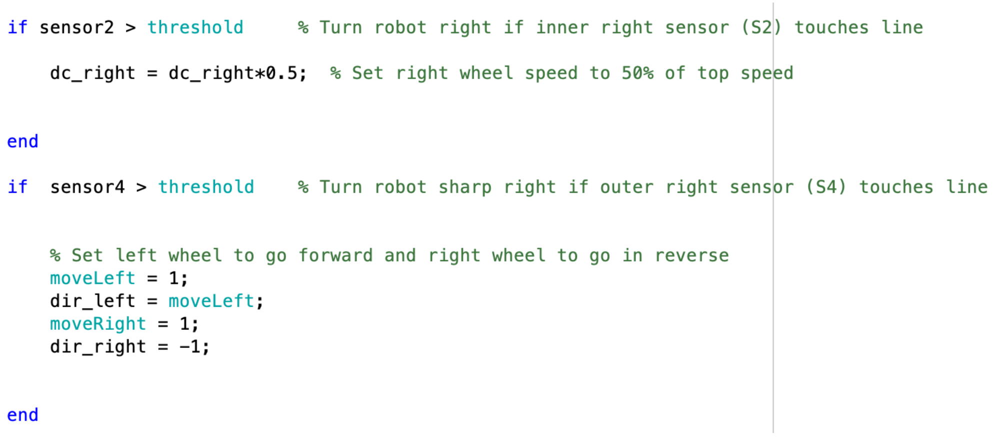
We then used this new threshold to control the robot. When sensor 1 is above the threshold the left motor is set to 50% speed to turn the robot left. When sensor 2 is above the threshold the right motor is set to 50% speed to turn the robot right.

If sensor 3 is above the threshold then a sharp left is needed so the left motor is set to reverse and the right motor to forward. If sensor 4 is above the threshold then a sharp right is needed and the right motor is set to reverse and the left motor to forward.



```
% Move robot to stay on line

if sensor1 > threshold    % Turn robot left if inner left sensor (S1) touches line

    dc_left = dc_left*0.5;  % Set left wheel speed to 50% of top speed

end

if sensor3 > threshold    % Turn robot sharp left if outer left sensor (S3) touches line

    % Set right wheel to go forward and left wheel to go in reverse
    moveRight = 1;
    dir_right = moveRight;
    moveLeft = 1;
    dir_left = -1;

end
```

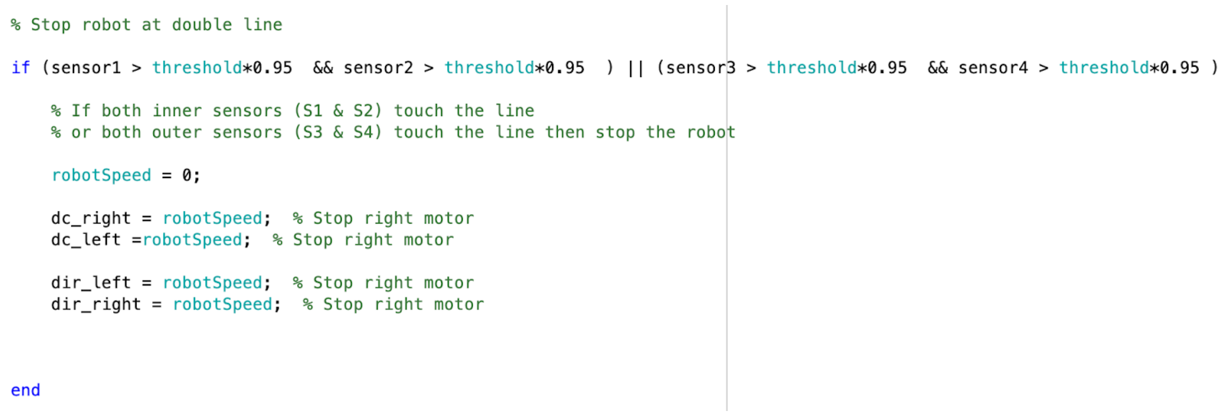
Figure 5: Move robot left code



```
if sensor2 > threshold     % Turn robot right if inner right sensor (S2) touches line

    dc_right = dc_right*0.5;  % Set right wheel speed to 50% of top speed


end

if  sensor4 > threshold    % Turn robot sharp right if outer right sensor (S4) touches line


    % Set left wheel to go forward and right wheel to go in reverse
    moveLeft = 1;
    dir_left = moveLeft;
    moveRight = 1;
    dir_right = -1;


end
```

Figure 6: Move robot right code

If both inner sensors, S1 and S2, detect the line or if both outer sensors, S3 and S4, detect the line then the motor directions are set to 0 which turns them off and the robot stops.



```
% Stop robot at double line

if (sensor1 > threshold*0.95  && sensor2 > threshold*0.95  ) || (sensor3 > threshold*0.95  && sensor4 > threshold*0.95 )

    % If both inner sensors (S1 & S2) touch the line
    % or both outer sensors (S3 & S4) touch the line then stop the robot

    robotSpeed = 0;

    dc_right = robotSpeed;  % Stop right motor
    dc_left =robotSpeed;  % Stop right motor

    dir_left = robotSpeed;  % Stop right motor
    dir_right = robotSpeed;  % Stop right motor


end
```

Figure 7: Stop robot code

This method worked for most times of day but not all of them.

The final step was to use the mean line value to find the time of day and set a unique robot speed for each time of day group. See Table 1 for time of day groupings.

By testing each time of day group and setting them for unique speeds the robot was made to work on all tracks at all times of day.

```
    % Sets unique robot speed for various times of day

    if abs(lineValue-0.5243) < 0.001    % Hour 0,1,2,3,4,5,6,21,22,23
        robotSpeed = 0.59;

    elseif abs(lineValue-0.5592) < 0.001    % Hour 7
        robotSpeed = 0.59;

    elseif abs(lineValue-0.5943) < 0.001    % Hour 20
        robotSpeed = 0.59;

    elseif abs(lineValue-0.6290) < 0.001    % Hour 8
        robotSpeed = 0.6;

    elseif abs(lineValue-0.6641) < 0.001    % Hour 19
        robotSpeed = 0.59;

    elseif abs(lineValue-0.6998) < 0.001    % Hour 9
        robotSpeed = 0.57;

    elseif abs(lineValue-0.7700) < 0.001    % Hour 13,14,15,18
        robotSpeed = 0.55;

    elseif abs(lineValue-0.8043) < 0.001    % Hour 10,12,17
        robotSpeed = 0.61;

    elseif abs(lineValue-0.8396) < 0.001    % Hour 11,16
        robotSpeed = 0.57;

    else
        robotSpeed = 0.5;   % Default
    end
end
```

Figure 8: Set unique robot speed code

Table 1: Hour groupings with shared mean line values

Mean Line Value	Hour of day
0.5243	0,1,2,3,4,5,6,21,22,23
0.5592	7
0.5943	20
0.6290	8
0.6641	19
0.6998	9
0.7700	13,14,15,18
0.8043	10,12,17
0.8396	11,16

Testing (Mechanical)

For the mechanical testing we first tried 4 sensors. Two inner sensors for turning left and right, and two outer sensors for sharp turns.

After running the simulation numerous times and not succeeding on all tracks and times we added a 5th sensor in the center to get the mean line value. The mean line value was then used to calculate the new line threshold unique to the time of day.

Each track was run many times with slight adjustments made to sensor positions until the robot worked for all tracks at most times of day.

We then had an issue with the robot not stopping at the end of the track due to the robot crossing the finish line slightly skew during some simulations.

The decision was then made to move the outer sensors backwards so that the inner sensors would detect the line first and straighten the robot and after which the outer sensors would detect the line at the same time and stop the robot from moving. Figure 9 and Figure 10 show the initial and final sensor positions before this adjustment was made.

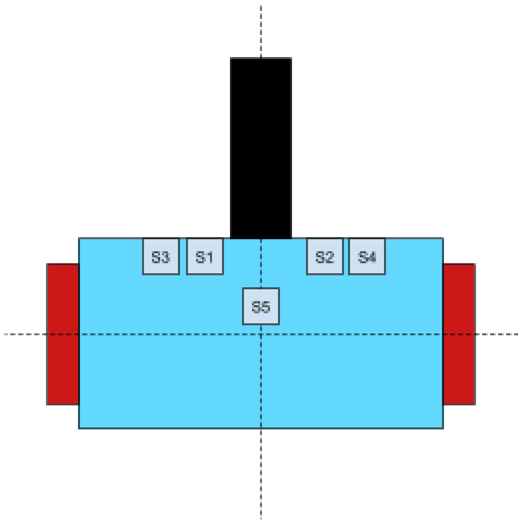


Figure 9: Initial 5 sensor positions

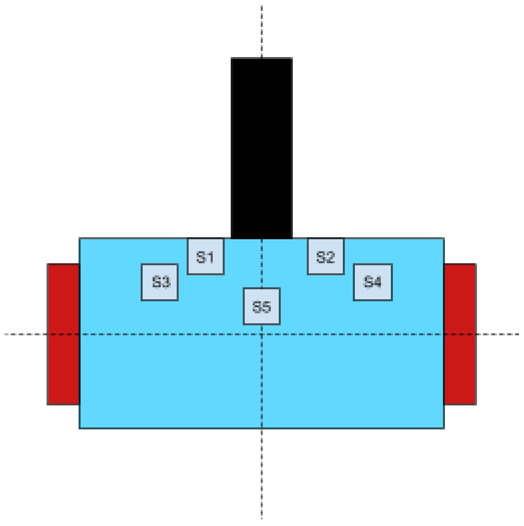


Figure 10: Final 5 sensor positions

Testing (Software)

For the software testing the simulation was run many times with adjustments being made to the line threshold and robot speed in an attempt to get the robot to complete all tracks at all times

We had issues with the robot stopping at the end at some times of day due to the varying lighting conditions.

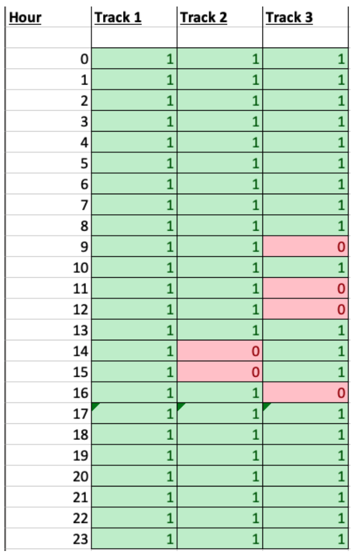
The decision was made to decrease the line threshold so that a line detection happened sooner to account for noise.

We then used the mean line value to work out the various times of day and put the hours with the same line value in a group.

Each group was then tested for every track and any hour from that group to ensure the robot worked for all of them with the results recorded in tables shown below. If the robot did not complete the track and stop at the end, the robot speed was increased and decreased to find a speed that worked on all tracks for that time group.

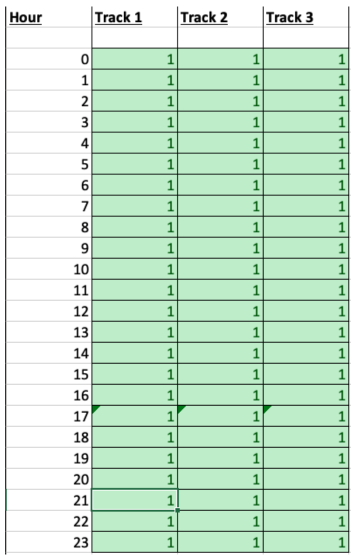
After finding a unique speed that worked for each group the robot was tested again on all tracks for all times of day to ensure they all worked. The tests were run again and the speed was increased to find the fastest speed that the robot could run at and complete all tracks for the chosen time group. The tables below were used to keep track of the results. Table 2 shows the final results of the tests when a constant speed was used for all times. Table 3 shows the results of setting a unique speed for each time group. Green blocks mean the robot was successful and red blocks mean the robot either did not complete the track or it did not stop at the end.

Table 2: Test results for constant speed



Hour	Track 1	Track 2	Track 3
0	1	1	1
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	0
10	1	1	1
11	1	1	0
12	1	1	0
13	1	1	1
14	1	0	1
15	1	0	1
16	1	1	0
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1

Table 3: Test results for unique time group speeds



Hour	Track 1	Track 2	Track 3
0	1	1	1
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1