**Faculty of Engineering, Environment and Computing**

M28COM – EVOLUTIONARY AND FUZZY SYSTEMS

**Assignment Brief 2018/19**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module Title  Evolutionary and Fuzzy Systems | Individual Project | | Cohort  Jan/May | Module Code  M28COM |
| Coursework Title  **Fuzzy Logic Controller (FLC) for Controlling a Mobile Robot** | | | | Hand out date:  30th Jan. 2019 |
| Lecturer  Dr. Vasile Palade | | | | Due date and time:  04th Mar. 2019, 6.00pm |
| Word Limit: 4-8 pages | | Coursework type: Assignment | | % of Module Mark 50% |
| * Submission arrangement: online via CU Moodle. * File types and method of recording: Submit your report as a PDF or Word document using the ‘Assignment 1’ link in the M28COM Moodle page. * Mark and Feedback date: 2 weeks after submission * Mark and Feedback method: provided in Moodle | | | | |

|  |
| --- |
| Module Learning Outcomes Assessed:   * Outline and represent fuzzy sets for linguistic variables using a variety of different fuzzy membership functions for fuzzification. * Apply and justify set-theoretic operators for calculating conjunction and disjunction of fuzzy sets as a basis for rule-based implication and composition operations used during the fuzzy inference process. * Design, implement and test simple fuzzy logic controllers for mapping system inputs to outputs through the use of If-Then rules, fuzzification, inference and appropriate use of defuzzification. |
| Task and Mark distribution:   |  |  | | --- | --- | | Task 1  Task 2  Task 3  Task 4 | 40  20  25  15 | |

# Section 1: Design and Implementation of the FLC and Design Justification

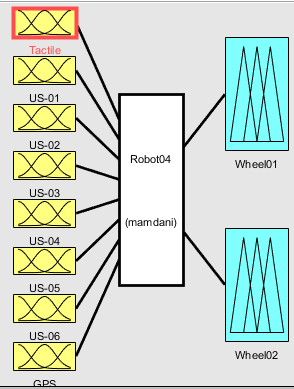
The fuzzy logic controller designed for the task was modelled using the Mamdani interface, accepting 8 inputs (tactile, 6 ultrasonic and the GPS) and outputting 2 outputs (2 wheel speeds).

Figure 1.1 - FIS

Whilst the robot does have 6 tactile sensors, due to their Boolean nature, and that they are only used to stop the robot in the case it crashes, the tactile sensors were combined into one single input.

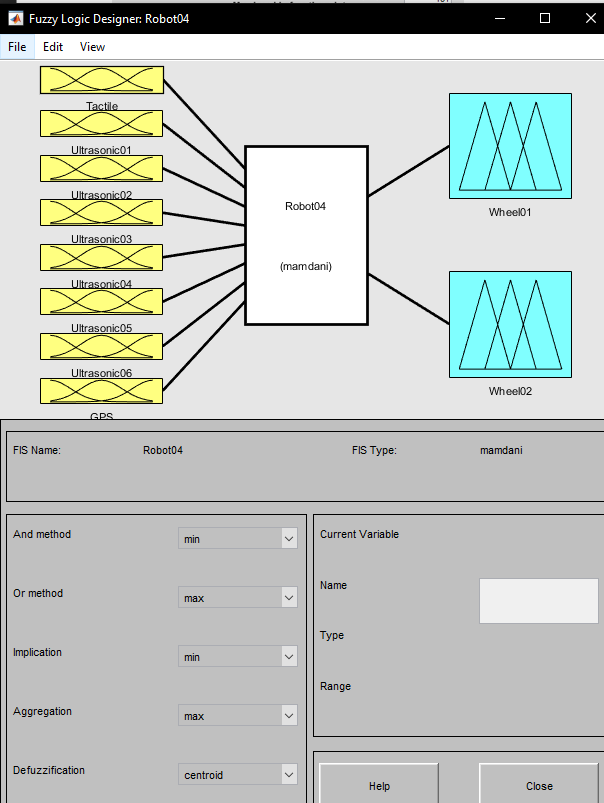
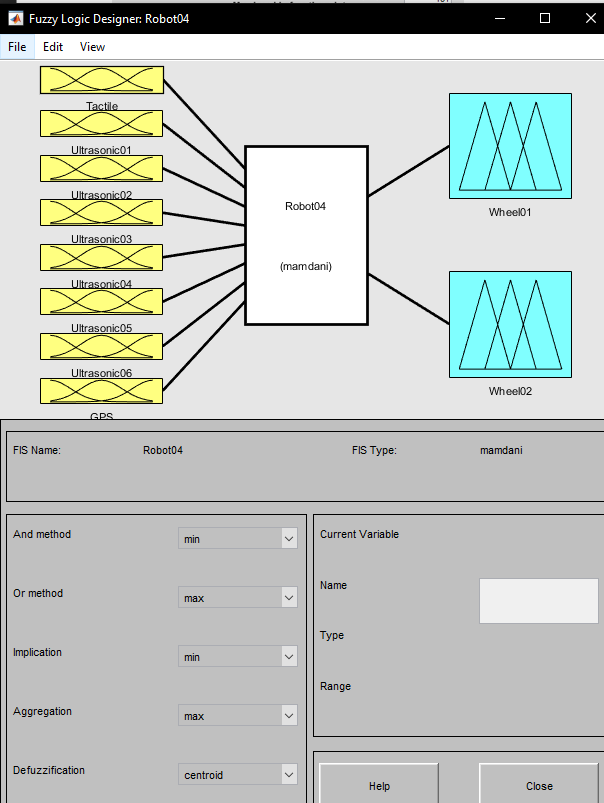
The wheel speed sensors were not used as inputs as for the simple simulation that information wasn’t needed, however, if the robot was to be built into a real-world device, then the wheel speed sensors would be needed as an input, so the robot can take acceleration into account when determining the new wheel speeds.

As for the GPS, as the robot has no one to determine which direction it is facing, using the GPS with its raw data (X and Y coordinates) would be impossible, so for the sake of the task, the input to the GPS sensor is a simple angle (further explained alongside the membership function).

The main reason for using Mamdani over Sugeno is because of the lack of a membership function for the outputs in a Sugeno inference system. The lack of a membership function means that the different states for the wheels cannot be implemented and thus the output of the Sugeno system would either be a constant or a linear equation however for the system to be as effective at avoiding walls and objects, a greater weight needs the be adding to the reverse states so that the robot prioritizes reversing in the case it gets close to a wall which allows the reversing rules to be more effective and more decisive in the final wheel speed.

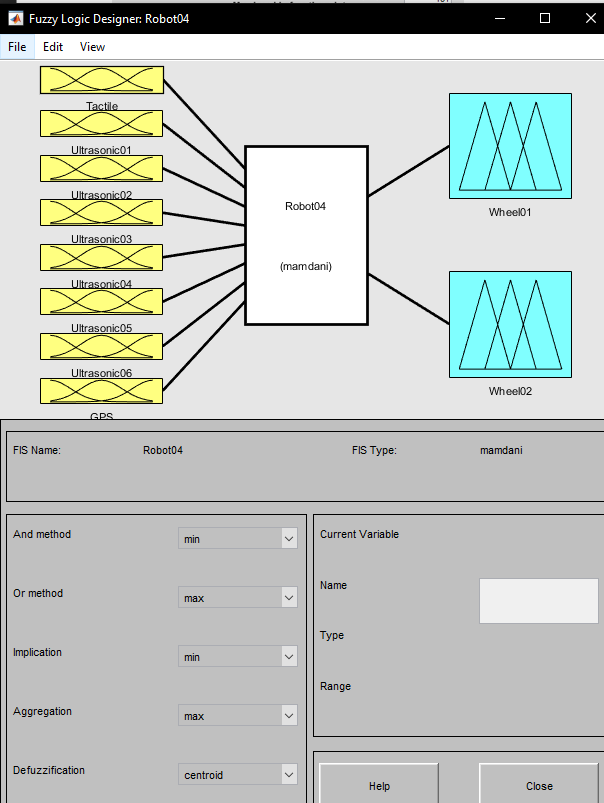
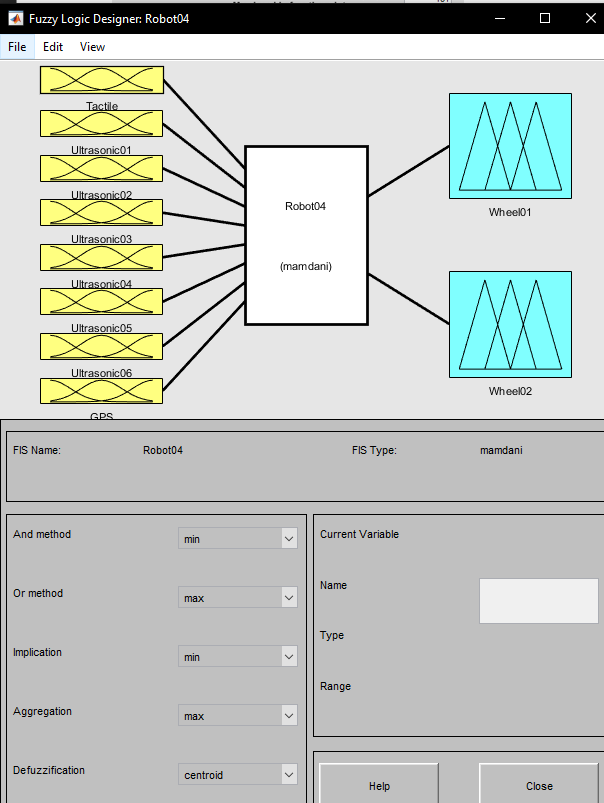
The method used for the AND rules is min (figure 1.2). The reason for using min is the min method can preserve the results of the AND operation whilst also extending to all real numbers between the scale

Figure 1.2 – And



The method used for the OR rules is max (figure 1.3). The reason for using the max method is because the alternative is to use the probor method (probabilistic), however as it’s a robot running with real crisp values, using a probabilistic method wouldn’t be as effective as the max method.

Figure 1.3 - Or



The method used for implication is min (figure 1.4). The reason for using min is to limit the impact of the rules that are weighted less. The min method truncates the output, which as intended, limited the impact of said rules, whereas the prod method would scale the output accordingly which would increase their impact in comparison to the min method.

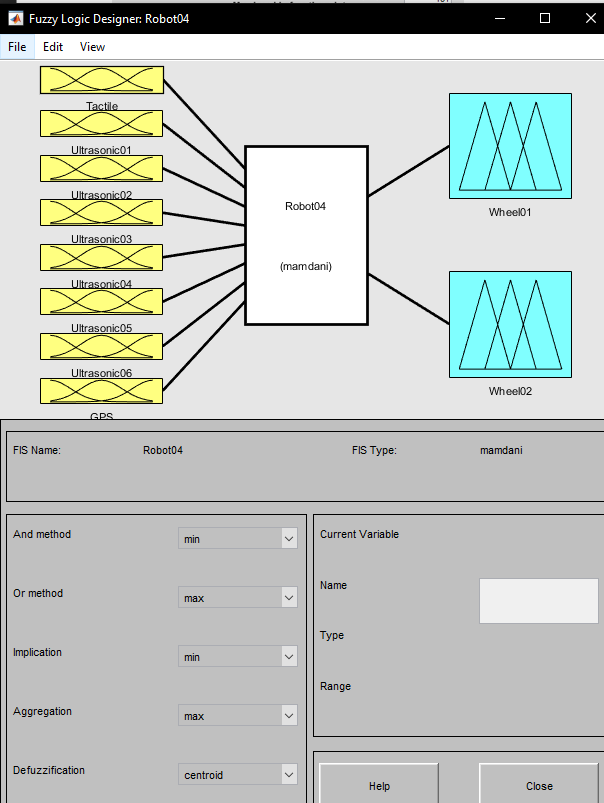
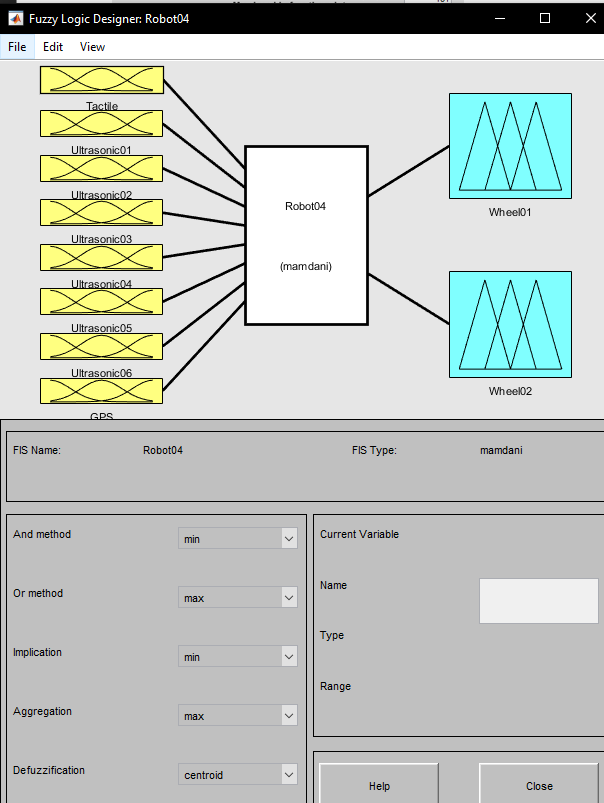


Figure 1.4 - Implication

The method used for aggregation is max (figure 1.5). The reason for using max is that the alternatives (prod and sum) reduce the impact on some of the important rules (such as the ones designed to prevent the robot from crashes), whereas max makes it so the lesser rules (or the rules with lower values at the time) have less of an impact.

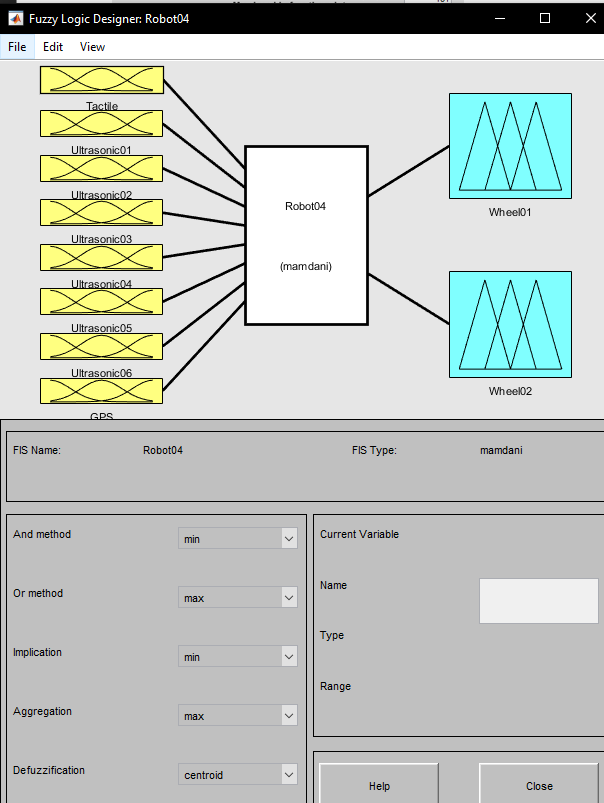
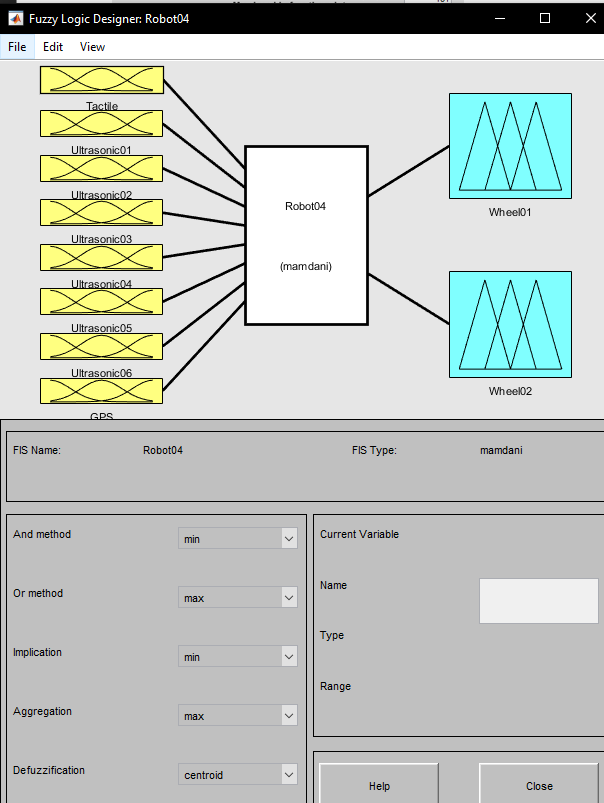
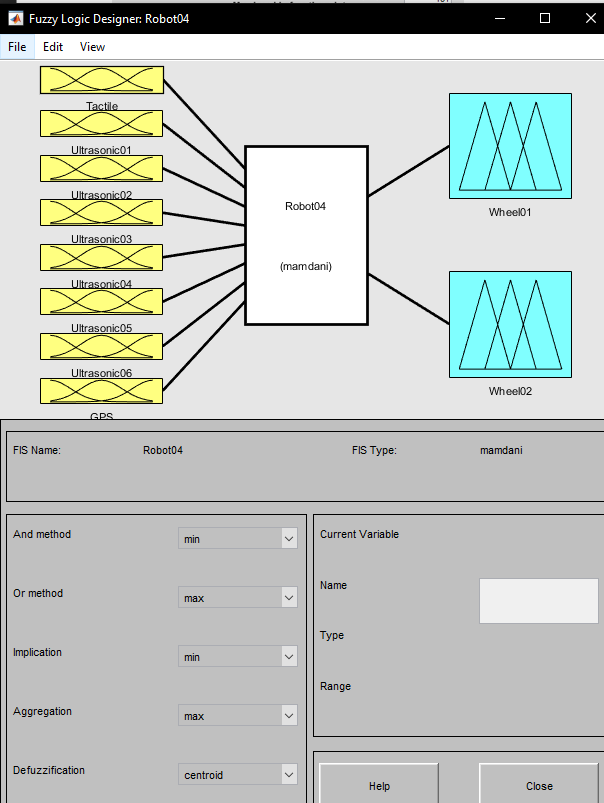
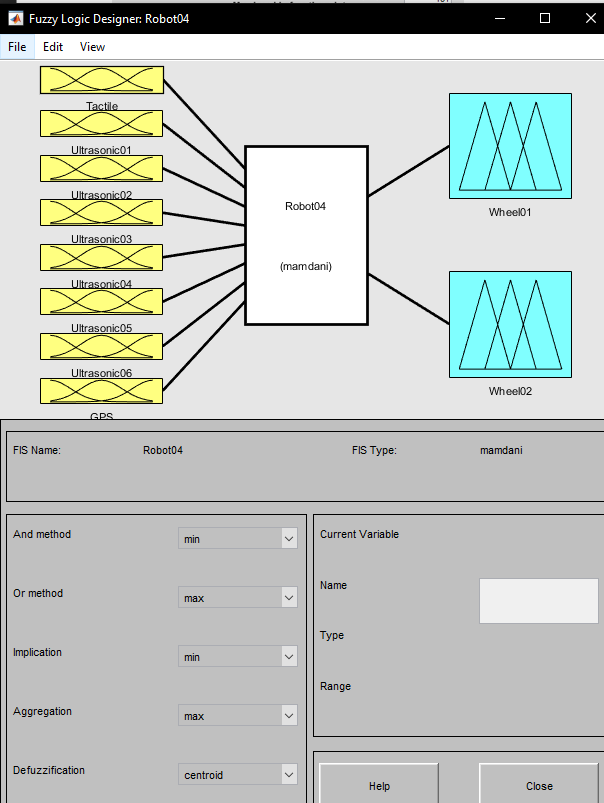


Figure 1.5 - Aggregation

The method used for defuzzification is centroid (figure 1.6). The centroid method is the most commonly used method because it finds the centre of ‘mass’ within the aggregated results, thus allowing for a crisp value that is the average of the aggregated results.

Figure 1.6 - Defuzzification



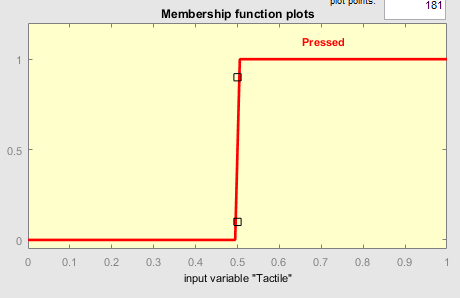
Whilst the robot has 6 tactile sensors, for the fuzzy logic controller they have been combined into one single input as the tactile is only responsible for stopping the robot if it bumps into anything, the ultra-sonic are responsible for controlling the direction and speed. Due to the Boolean nature of the tactile sensor (pressed or not), the membership function for it is a simple sigmoid function altered to cover only 0 and 1.

Figure 1.7 – Tactile Membership Function

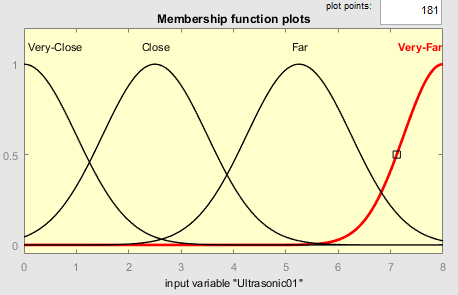
For the 6 ultra-sonic sensors, they all share the same membership function design (refer to figure 1.8). As the ultra-sonic sensor used has a range of 8 meters, the membership function ranges from 0 to 8 for its input values to fully utilize the sensors precision. Four states were devised along the scale to be used in conjunction with the rules (Very-close, close, far, and very-far) all of which are Gaussian functions. Very-close is used for the rules made to make sure the robot doesn’t crash, close is used to navigate around corners and follow walls, far & very-far are used to navigate the robot forward.

Figure 1.8 – Membership Function for all 6 Ultrasonics

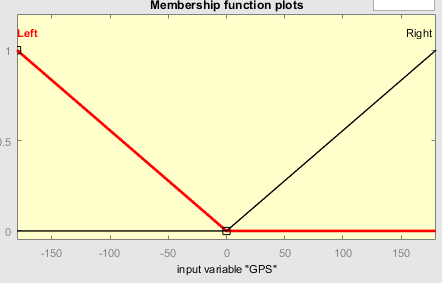
The GPS sensor receives two signals, an X coordinate and a Y coordinate. However, for a fuzzy controller, taking the GPS location of the robot and the GPS of the chosen location and using these to determine the chosen path is difficult as the robot has no one of telling which direction it is facing. That said, it is possible to figure of which direction the chosen location is through alternative algorithms, outside of the fuzzy logic controller and so for the sake of this controller the input to the controller is a single angle between -180 and 180 telling the robot which way to turn to face the chosen destination. The GPS input is designed to only have an impact in helping get the robot to the destination, if it was to run in a fork in the path for example, whilst not hindering the robot navigating the rest of the maze, and so the functions used are simple ... Each ranging from one end to the middle.

Figure 1.9 – GPS membership function

For both the wheels, they have the same membership function design (refer to figure 1.10). The scale for the wheel speed is -1 and 1, with -1 being reverse and 1 being forward. Like the ultrasonic sensors, the wheel speed outputs use gaussian functions for its states (reverse-hard, reverse, reverse slow, stop, forward-slow, forward and forward-fast). As the rules using the reverse-hard state are responsible for the robot not crashing, the state for membership function for reverse-hard was made to be quite large so it can scale smoothly when the robot gets closer and closer to the walls. The opposite is for the forward-slow state, the membership function is narrow so that it only adds a little bit of speed to the wheels.

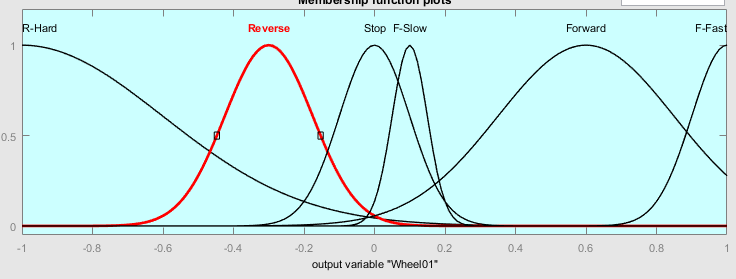


Figure 1.10 – Membership function for the two wheels

The logic controller is comprised of 13 rules to achieve its task, the rules are split into several sections, each responsible for different aspects of the controller. The first of which is a single rule, in the case of any of the tactile sensors are pressed then the vehicle reverses to make sure it’s no longer against a wall or object. The reason for it only reversing is to make sure that the tactile sensors are no longer pressed after it moves, the ultrasonic is then responsible for any form of turning and preventing it from crashing again.

Figure 1.11 – Rule set 1 (Tactile)

The second set of rules is a group of 3 responsible for making sure the robot doesn’t get too close to a wall and or object using the ultrasonic sensors. If the front sensors get too close to a wall it reverses, whilst if the side sensors get close, the wheel closest to the wall speeds up whilst the other wheel reverses, to turn the robot.



Figure 1.12 – Rule set 2 (Wall Avoidance)

The third set of rules are used to navigate the robot to a selected destination. It’s designed so in the case of a fork in the path, the rules will push it in the right direction, only slightly but enough for it to continue in that direction.

Figure 1.13 – Rule set 3 (GPS)

The fourth set of rules are for making the robot go forward, whenever the sensors don’t sense a wall or it’s far away, then the wheels move forward to make sure the robot is always moving. As this set of rules are responsible for the robot going forward, the weights of the rules were set to 0.5 so they don’t interfere with the other rules, the ones responsible with making sure the robot doesn’t crash.

Figure 1.14 – Rule set 4 (Navigation)

The fifth and final set of rules are designed to make the robot avoid any objects. As an object can be small, only one of the sensors will pick it up whilst the others say it’s fine to proceed thus a few extra rules were needed.

Figure 1.15 – Rule Set 5 (Object Avoidance)

# Section 2: Analysis of Controller Performance

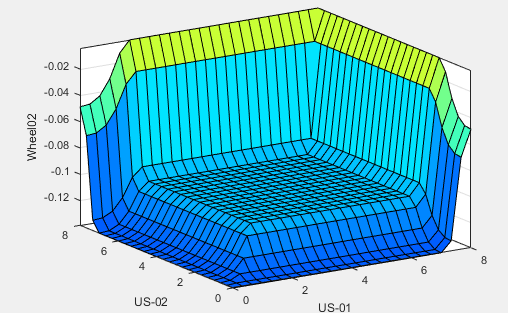
All surface plots shown are for wheel 2, with both wheels being reflections of each other, the surface plots which be the mirror images of one another.

Figure 2.1 – Surface Plot for Ultrasonics 1 and 2 for Wheel 2

Figure 2.1 Shows the surface plot for the first and second ultrasonic sensors, the side ones of the opposite side of wheel 2 (which would be US 5 and 6 for wheel 1). As seen, the closer the side ultrasonic sensors get to the wall, the more the wheels reverse.

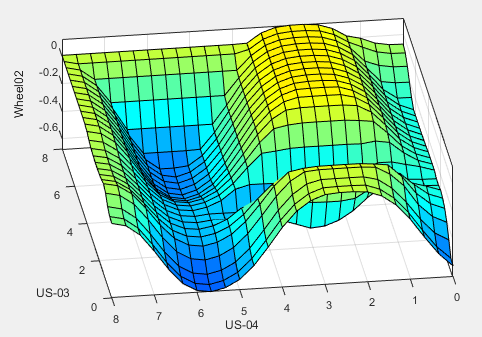
Figure 2.2 shows the surface plot for the third and fourth ultrasonic sensors, the middle ones. This surface plot would be mirrored for the first wheel. As seen, ultrasonic 3 has more of an impact on the wheel going forward whereas ultrasonic 4 has more or a varying impact on the wheel, this is vice-versa for wheel 1.

Figure 2.2 – Surface Plot for Ultrasonics 3 and 4 for Wheel 2

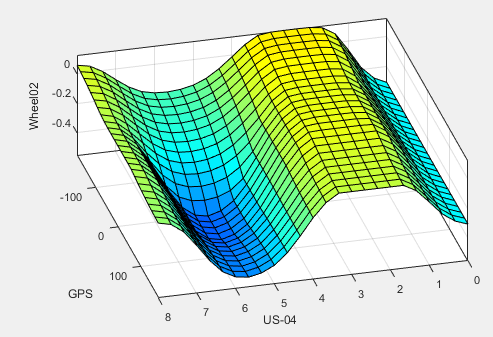
Figure 2.3 shows the surface plot for the fourth ultrasonic and the GPS, it’s to show the impact of the GPS on the system, the fourth ultrasonic was chosen as it’s an influential sensor for the second wheel. As seen, if the wheel is stationary or going forward then the GPS has little to no impact, however when the wheel is reversing then it has a huge impact.

Figure 2.3 – Surface Plot for Ultrasonic 4 and GPS for Wheel 2

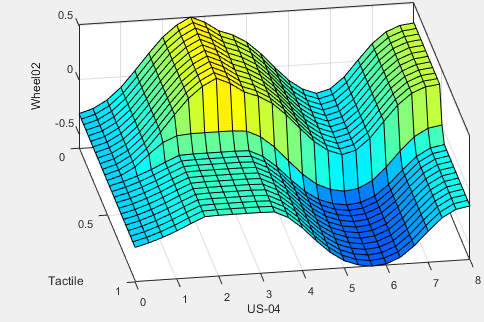
Figure 2.4 Shows the surface plot for the fourth ultrasonic and tactile. This is used to show the impact of the tactile in the case it is pressed and like before the fourth ultrasonic is used as the control. As shown if the tactile is pressed the wheel follows the same pattern just at a much lower speed (or even reversing).

Figure 2.4 – Surface Plot for Ultrasonic 4 and Tactile for Wheel 2

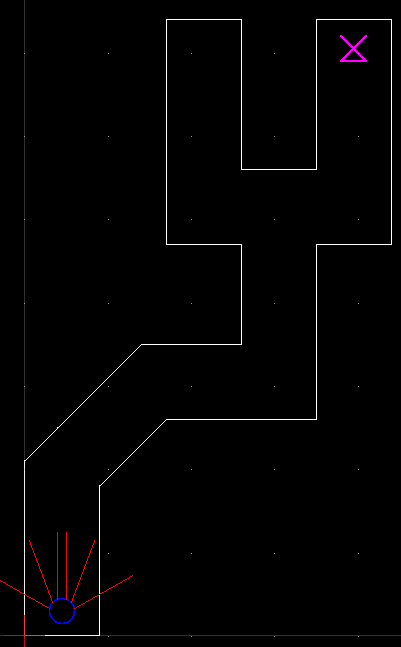
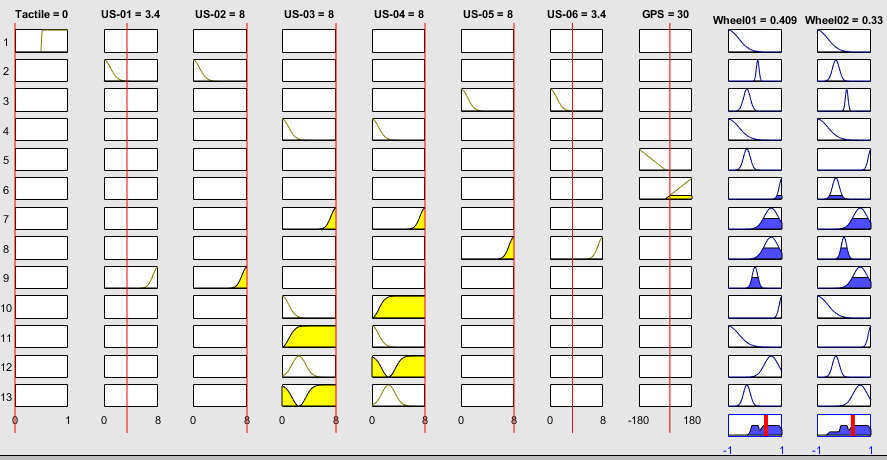
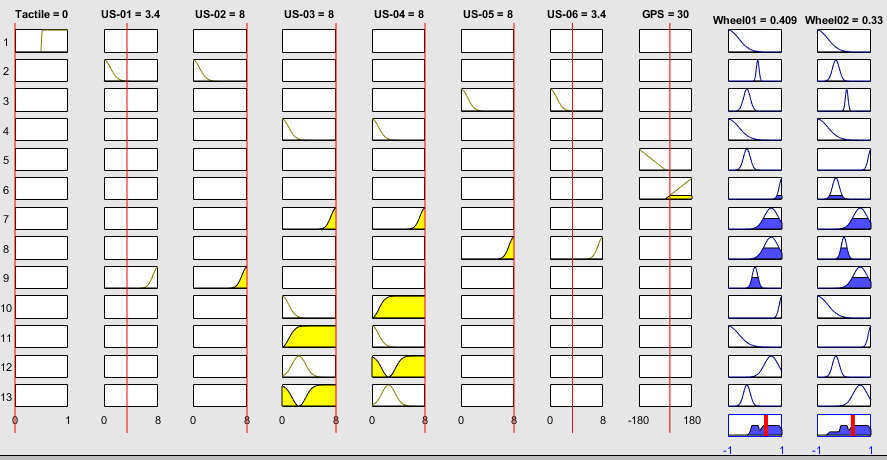
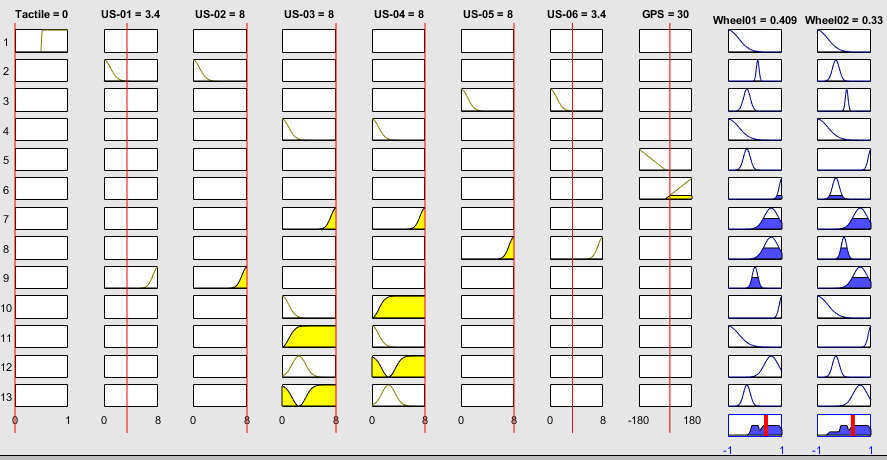
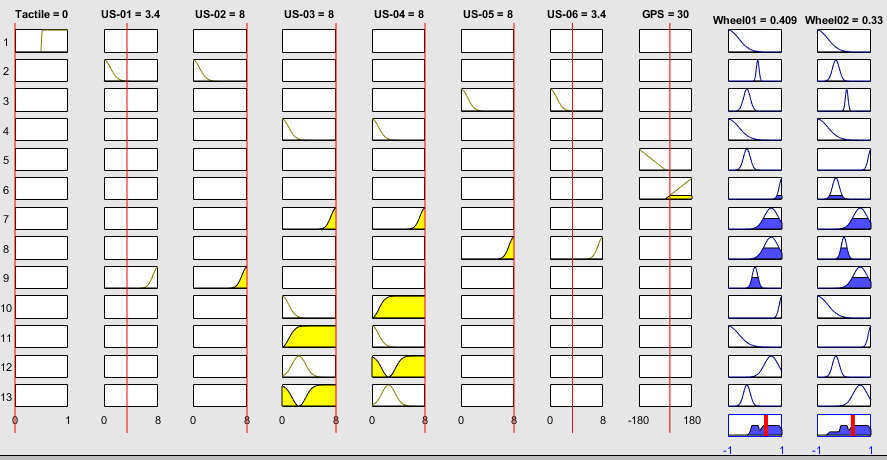
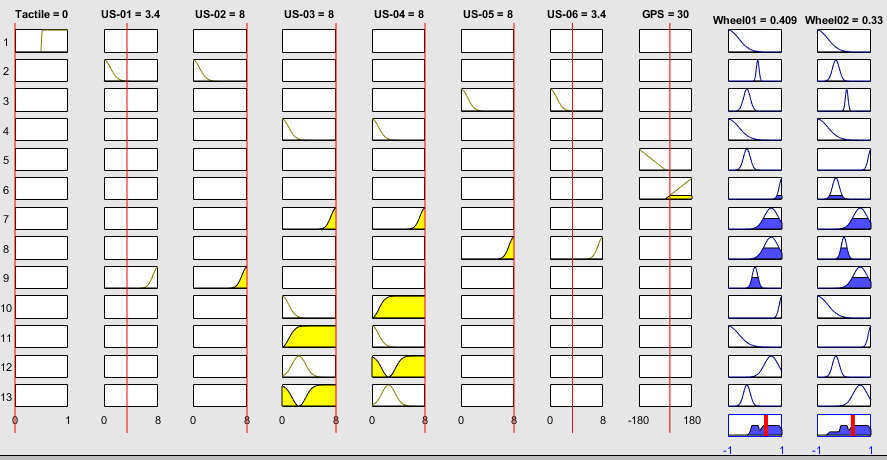
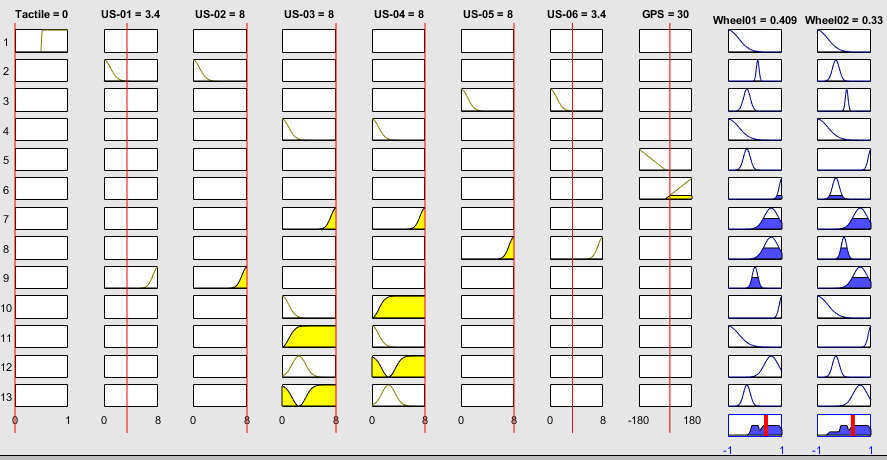
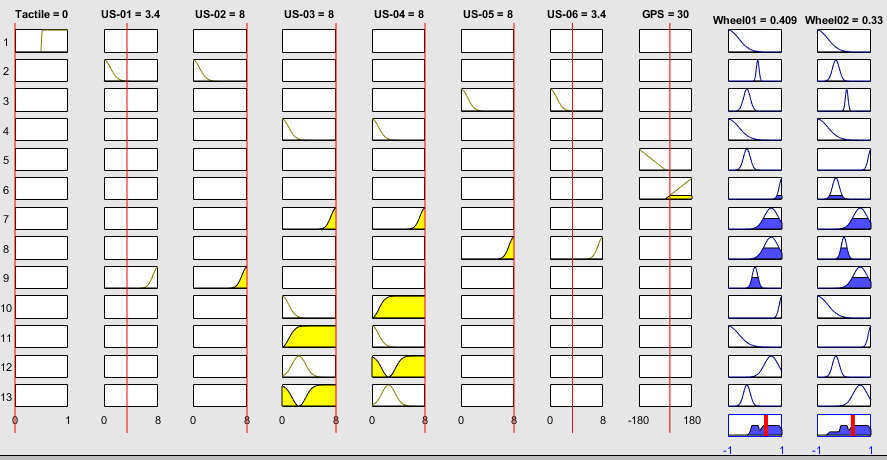
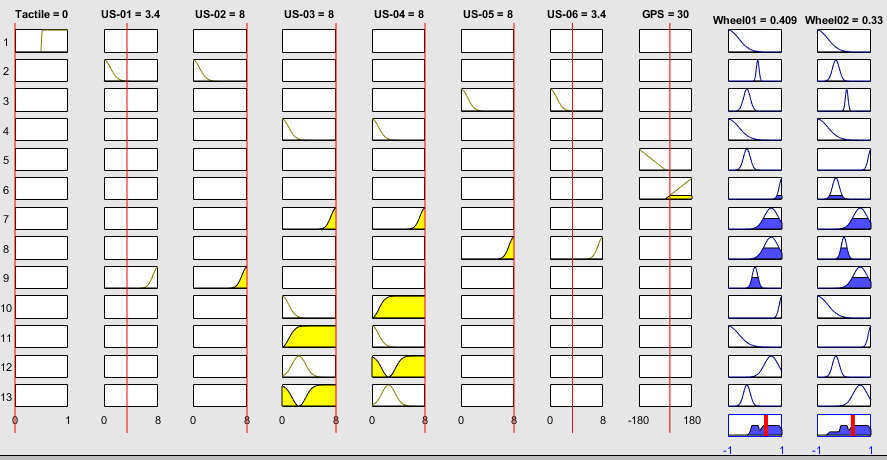
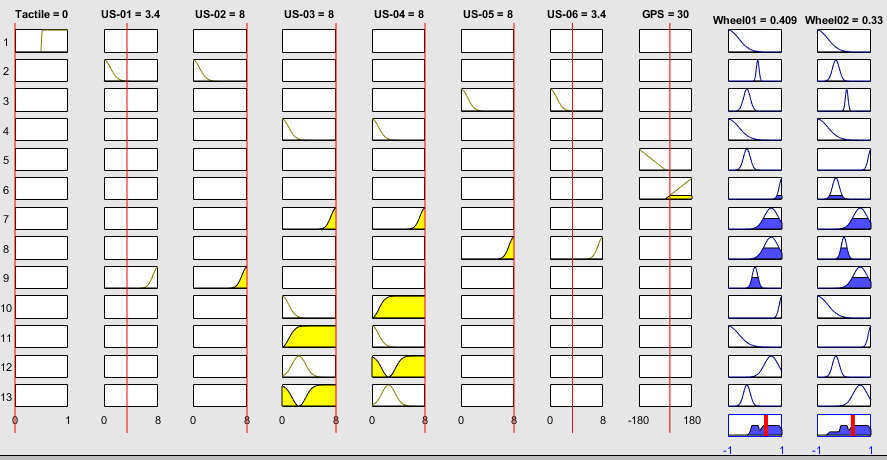
The figure to the right (figure 2.5) is a simple path for the robot to traverse, with the pink symbol being the destination, and below is a number of situations that prove the ability of the logic controller. All the situations were drawn up in CAD to get accurate angles and distances for the ultrasonic sensors, in combination with equations for calculating how far a robot travels for the different outputs, all of this to get accurate readings whilst testing the robot. For all situations explained below, there is a table of iterations with the input and outputs, the inputs are the tactile sensors, the 6 ultrasonic and the GPS in that order. Also, alongside each situation is a CAD picture showing the robot in the first iteration of the situation with all its input values labelled. For all situations only a few iterations are included, enough to demonstrate that it achieves the given task at that time.

Figure 2.5 – Reference Map

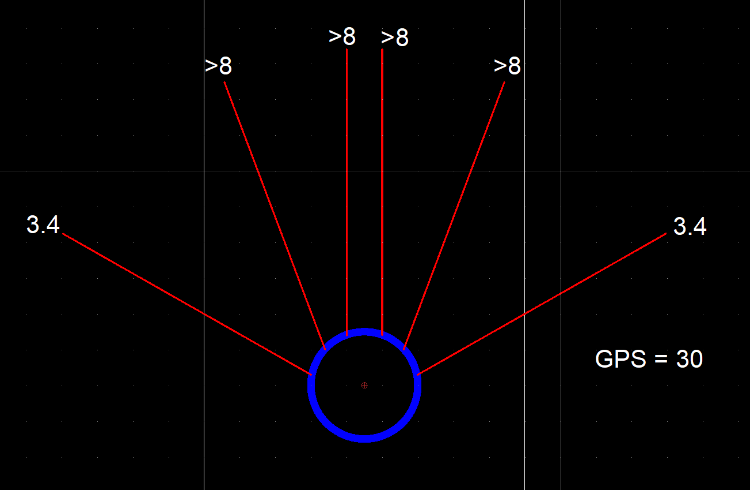
Figure 2.6 shows the rules activation with the inputs being the first iteration of situation 1 (refer to page 7).

Figure 2.6 – Rule Activation for 1st iteration of Simulation 1



|  |  |  |
| --- | --- | --- |
| Situation 1 | | |
| Iteration | Input | Output |
| 1 | 0, 3.4, 8, 8, 8, 8, 3.4, 30 | 0.409, 0.333 |
| 2 | 0, 3.5, 8, 8, 8, 8, 3.3, 27.72 | 0.408, 0.336 |
| 3 | 0, 3.6, 8, 8, 8, 7.8, 3.3, 25.56 | 0.408, 0.343 |

Figure 2.7 – Situation 1 Map

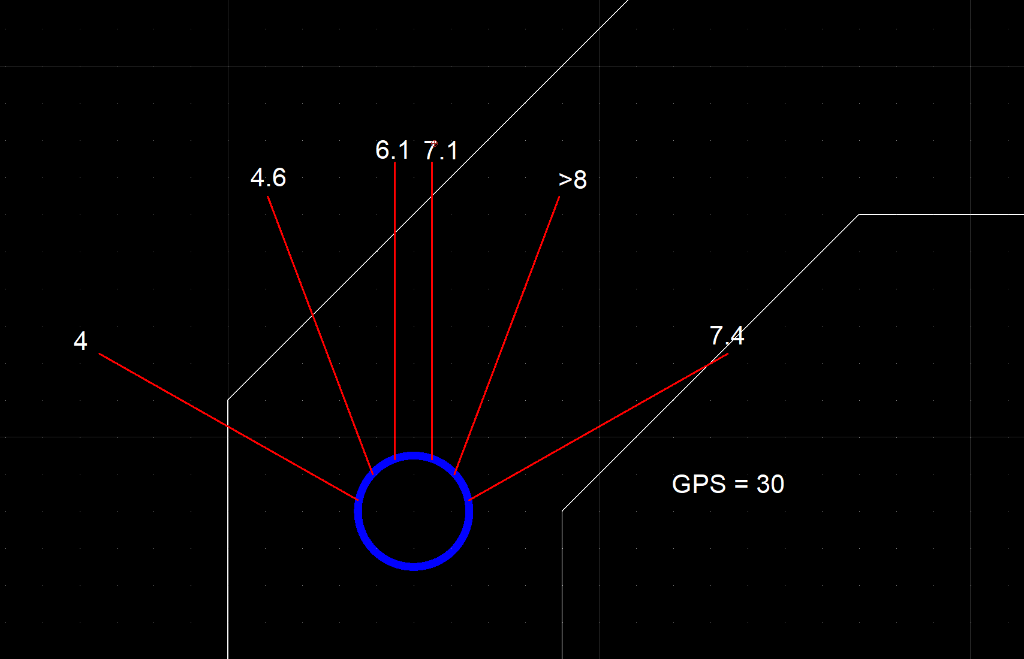


3.4

Table 2.1 – Situation 1 Iterations

The first situation is the robot going down a straight corridor, figure 2.7 shows the starting position and values for the input. As seen by table 2.1, the outputs from each iteration make the robot lean towards the right, this is because of the GPS, but with each iteration the difference between both wheels lessens and the car follows the wall, this is because as it gets closer to the wall, the ultrasonic sensors make the right wheel go faster to make sure it doesn’t crash.

Figure 2.8 – Situation 2 Map



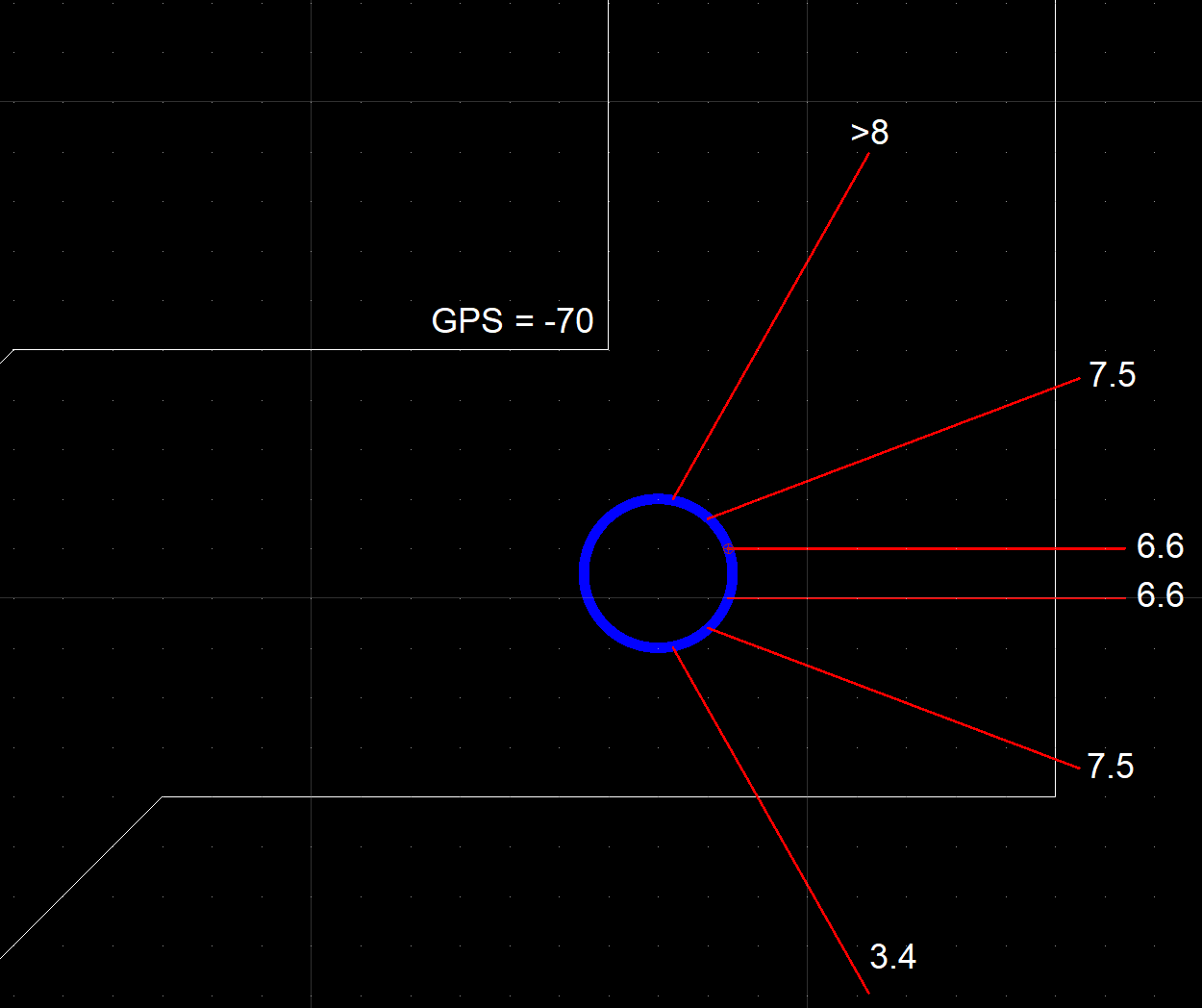
4

|  |  |  |
| --- | --- | --- |
| Situation 2 | | |
| Iteration | Input | Output |
| 1 | 0, 4, 4.6, 6.1, 7.1, 8, 7.4, 30 | 0.561, 0.217 |
| 2 | 0, 4, 4.7, 7.3, 8, 8, 4.7, 19.68 | 0.561, 0.3548 |
| 3 | 0, 3.7, 4.9, 8, 8, 8, 4, 13.47 | 0.56, 0.37 |

Table 2.2 – Situation 2 Iterations

The second situation is the robot navigating a corridor which takes a 45 degree turn to the right, a half corner, figure 2.8 shows the starting position and values for the input. As seen in table 2.2, with the left wheel going faster than the right, the robot starts to turn following the turn in the wall.

Figure 2.9 – Situation 3 Map



3.4

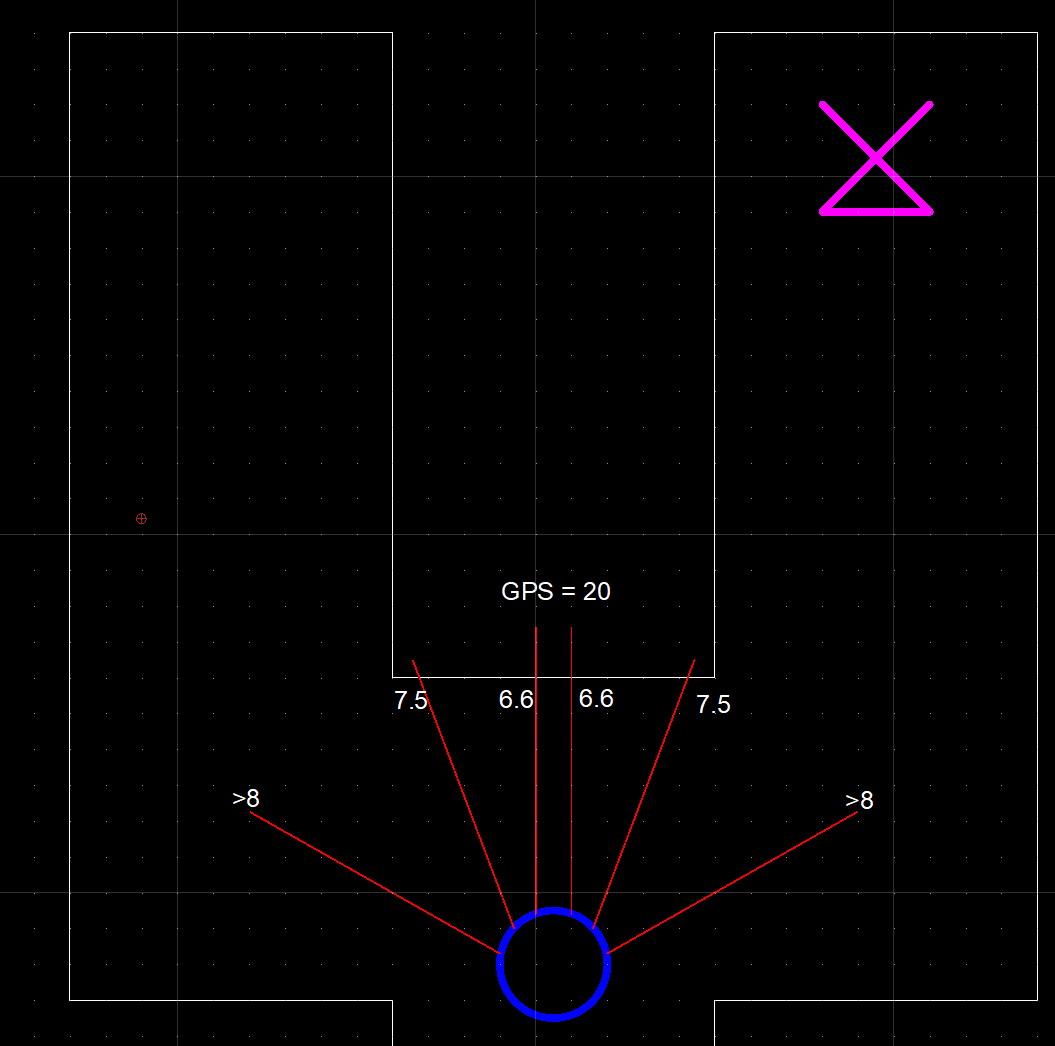
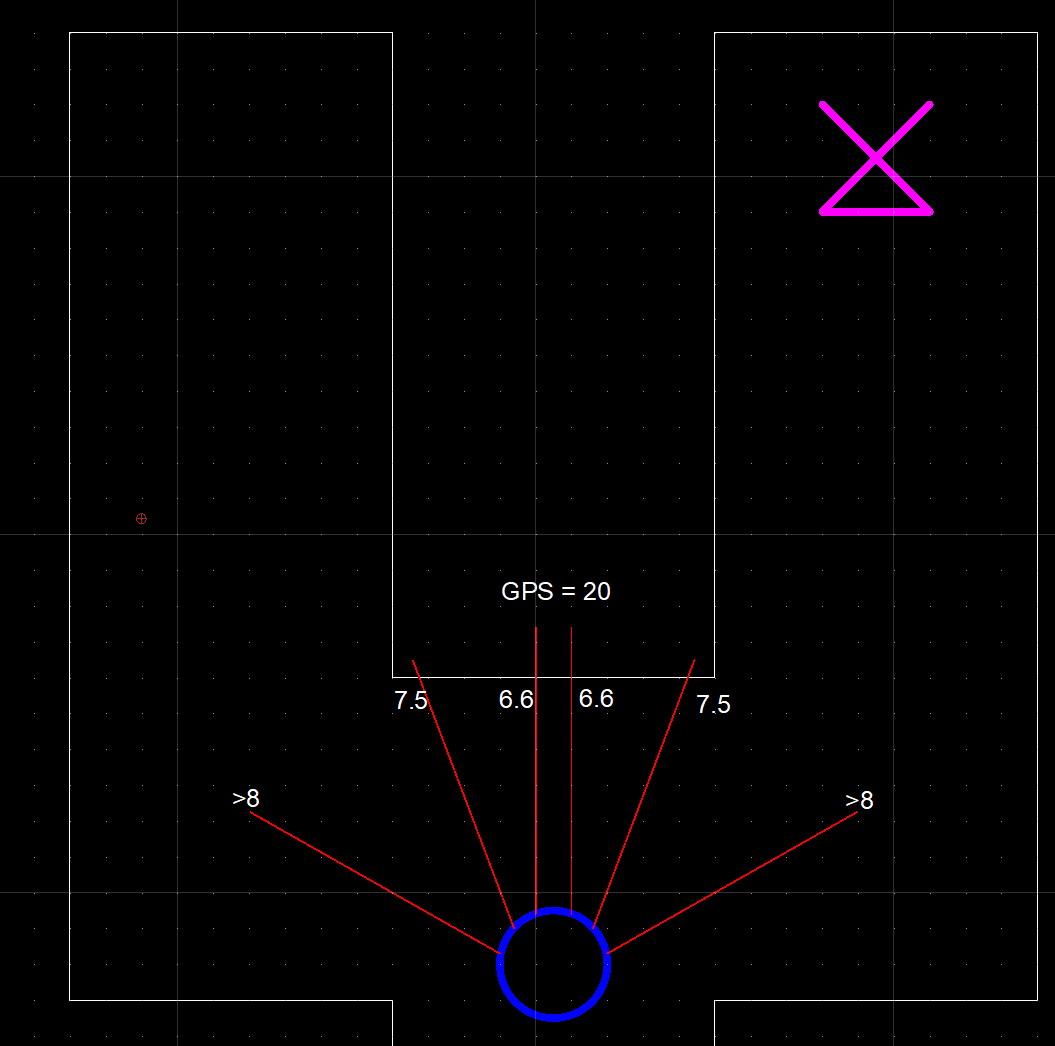
GPS = -70

|  |  |  |  |
| --- | --- | --- | --- |
| Situation 3 | | | |
| Iteration | Input | Output |
| 1 | 0, 8, 7.5, 6.6, 6.6, 7.5, 3.4, -70 | 0.223, 0.435 |
| 2 | 0, 8, 7.6, 6.4, 6.2, 6.8, 3.8, -63.64 | 0.0504, 0.513 |
| 3 | 0, 8, 8, 6.7, 6.3, 6, 5, -49.762 | 0.043, 0.557 |

Table 2.3 – Situation 3 Iterations

The third situation is the turning of a corner, figure 2.9 shows the starting position and the input values. At the first iteration, the robot turns to turn the corner, as it gets closer to the wall the left wheel starts to slow down making the robot take a sharper turn.

Figure 2.10 – Situation 4 Map

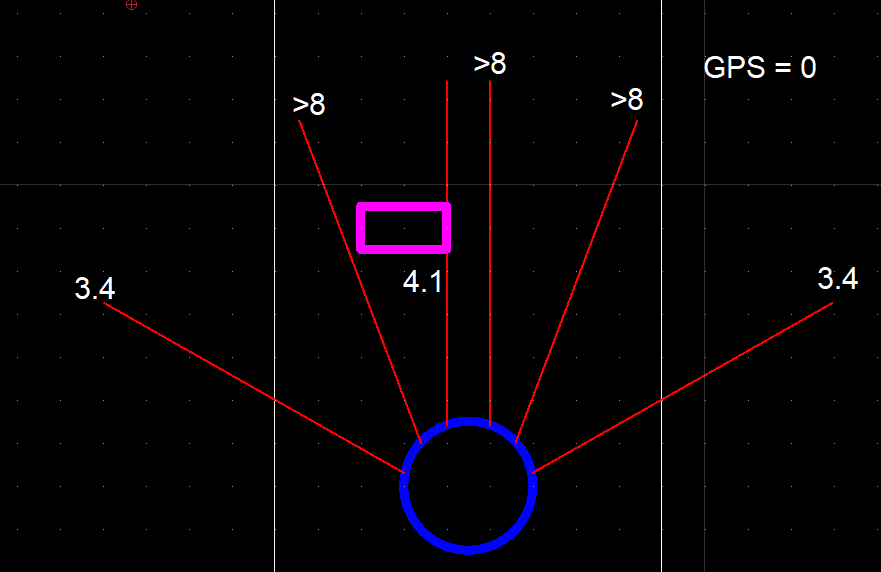


|  |  |  |
| --- | --- | --- |
| Situation 4 | | |
| Iteration | Input | Output |
| 1 | 0, 8, 7.5, 6.6, 6.6, 7.5, 8, 20 | 0.411, 0.354 |
| 2 | 0, 8, 7, 6.2, 6.2, 7.2, 8, 18.29 | 0.41, 0.358 |

Table 2.4 – Situation 4 Iterations

The fourth situation is a fork in the path like mentioned earlier, it is used to demonstrate the effectiveness of the GPS sensor as the intended destination is down the right path, figure 2.4 shows the starting position and input values. As seen in table 2.10, the robot starts to turn slowly to the right, whilst it is only a little bit it’s enough for the robot to continue to turn the corner in that direction.

Figure 2.11 – Situation 5 Map



3.4

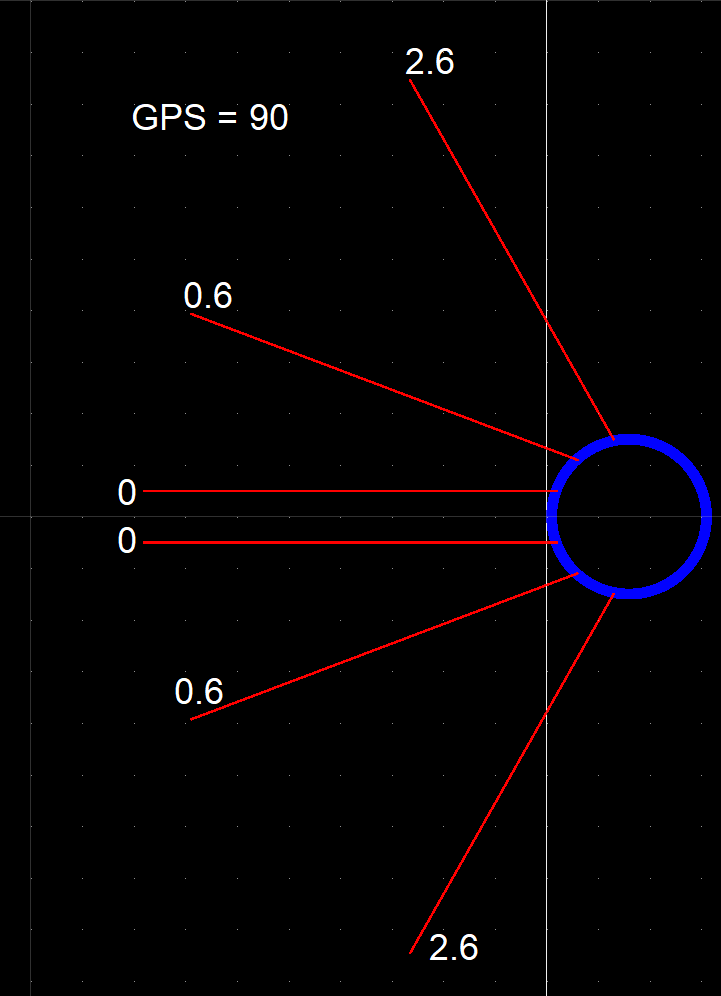
3.4

|  |  |  |
| --- | --- | --- |
| Situation 5 | | |
| Iteration | Input | Output |
| 1 | 0, 3.4, 8, 5.9, 8, 8, 3.4, 0 | 0.409, 0.291 |
| 2 | 0, 3.7, 4.3, 8, 8, 7.9, 3.2, -3.54 | 0.537, 0.411 |

Table 2.5 – Situation 5 Iterations

The fifth situation is the introduction of an object in the path to demonstrating the ability of the logic controller, figure 2.11 shows the starting location and input values. At first, the robot starts to take a sharp turn to avoid the object, after which the turning rate is slowed, but it continues until it clears the object.

Figure 2.12 – Situation 6 Map



2.6

|  |  |  |
| --- | --- | --- |
| Situation 6 | | |
| Iteration | Input | Output |
| 1 | 1, 2.6, 0.6, 0, 0, 0.6, 2.6, 90 | -0.319, -0.439 |
| 2 | 0, 2.9, 0.9, 0.5, 0.6, 1.1, 4, 86.46 | -0.27, -0.332 |
| 3 | 0, 3.2, 1.2, 0.8, 0.9, 1.5, 5.2, 84.6 | -0.187, -0.217 |

Table 2.6 – Situation 6 Iterations

The sixth and final situation is to demonstrate the ability of the tactile sensor. The ultrasonic sensors and the rules connected to them are designed so that the tactile sensors are never pressed and so this situation was needed to show if they work or not, figure 2.12 shows the starting location and input values. As the robot is perfected in line with the wall, the robot would back off but not turn as it wouldn’t know which direction to turn so for the sake of this situation the GPS sensor is set to tell the robot to turn to the right, and as seen in table 2.6 the robot’s right wheel reverses faster, thus turning the robot to the right and the robot continues this turn whilst reversing.

# Section 4: Combining FLC with Other Computer Intelligence approaches

There are two big hybrid fuzzy systems that have promising results, Neuro-Fuzzy systems and Genetic Fuzzy systems (evolutionary-fuzzy), both of which fuse fuzzy systems with another powerful system in data science. The first of which is the combination of fuzzy systems with neural networks, this grants the fuzzy system the ability to learn from its environment and adjust its membership functions and fuzzy rules. The Neuro-Fuzzy system is similar to a normal multi-layer neural network, with a crisp input layer, a crisp output layer (defuzzification layer), and three hidden layers (input membership functions, fuzzy rules and output membership functions). Having the fuzzy system designed as a neural network, it allows for the system to determine and adjust the membership functions and rules in accordance with the inputted data which in turn allows for a more precise and effective fuzzy system especially if the system has a lot of raw data inputted. One of the main advantages of a neuro-fuzzy system is the lack of need for expert knowledge, to develop a functional fuzzy system the developer needs to have expert knowledge in the area to be able to determine the rules and membership functions whereas with a neuro-fuzzy system as the membership functions and rules are devised by the network itself, less expert knowledge is then needed to implement it. For the robot, this would be very useful in creating a more flexible system, one that would operate more effectively than one with human input.

This design can be extended into an Adaptive Neuro-Fuzzy system which is designed to function in the same way the Sugeno inference system does. It follows a similar neural layout as the normal neuro-fuzzy system with the inclusion of a 6th layer which summarizes the outputs of the defuzzification neurons. This type of hybrid system has the ability to achieve the same results but much quicker similar to how the Sugeno is quicker than the Mamdani in a normal fuzzy system. One of the main advantages of the ANFIS is the similarities to the Sugeno, which is best suited to mathematical analysis whereas Mamdani is better suited to human input, this is especially important for the hybrid systems as the neural network will likely receive a lot of raw data rather than human inputted data. For the robot, the neuro-fuzzy would be more effective then this, however, if the robot was to be more advanced (more sensors) and thus more mathematical data, then the ANFIS would be more effective and powerful.

Another powerful hybrid system is a Genetic-Fuzzy system, the combination of an evolutionary system and a fuzzy system. Like the neural-fuzzy system, the introduction of evolutionary computing is used to generate fuzzy rules and adjust the membership functions through the use of genetic algorithms. The most important aspect of the genetic-fuzzy system is that its capable of reducing the number of if-then rules needed for the system to work, this allows for a small lightweight system that is still as powerful, and also allows the system to be easily analysed by a human expert. As the introduction of genetic algorithms to a fuzzy system reduces the number of rules, whilst still maintaining the effectiveness of the system, it would be very important for the robot as it would allow the FIS to be smaller thus allowing it to be implemented only a robot with limited storage capabilities.

# References

Negnevitsky, M., 2002. *Artificial Intelligence: A Guide to Intelligent Systems.* 1st ed. Harlow: Pearson Education Limited.

# Appendix

Link to the Fuzzy Logic Controller and a text file with the iterations for the situations in the analysis section

<https://drive.google.com/open?id=1A-SK3UOLVc1J3V9YmsVDzh5fnz7Ln62u>