University of Sheffield

COM2009-3009
Human-Machine Interaction and Robotics

Lab Assignment #2

'Multi-Layer Control'

Earlando Grant-Favourite

Sree Pittala

Alistair Cook

Robot: C2

Department of Computer Science

May 16, 2019

1 Overview (please read carefully)

In Lab Assignment #1 you investigated a single-layer negative-feedback control system. In particular, you developed a 'Proportional-Integral-Derivative' (PID) controller that was capable of steering a robot through a maze consisting of a long winding corridor. For Lab Assignment #2 your challenge is to navigate a more complex environment containing real-world hazards - and, in order to do this, you will need to add new layers of functionality to your existing robot through a *multi-layer* control system.

This programming assignment is worth 30% of the overall course mark.

1.1 Logistics

As before, the lab classes for COM2009-3009 take place in Computer Room 3 in the Diamond (DIA-CR3) from 13:00 to 14:50 on Mondays (for Group A) and 09:00 to 10:50 on Wednesdays (for Group B). **The teams have been re-randomised from Assignment 1. Your new team and robot are listed on MOLE**.

The assignment will again be judged in a league-based competition, and assessed on the basis of a group report (prepared using LaTeX).

The deadline for handing-in the group report (via MOLE) is ...

Midnight Friday 17th May 2019 (week 12)

Reminder: *You may not take a robot out of DIA-CR3 under any circumstances!*

Reminder: *Please be careful in and around the robot arena; the robots can move unexpectedly and the arena itself is high enough to be a trip hazard. There is a risk assessment for this in the folder on the teaching podium.*

Reminder: *At the end of each lab session, turn off your robot and return it to the correct charging cabinet, with the microcomputer (brick) being placed on charge by plugging in the power adapter in the centre section of the cabinet.*

Reminder: *If you have customised your robot, take a photo (as a memory aid) and return it to its original 'Driving Base' configuration and place all the Lego back in its box*[1].

1.2 Provided Materials

In addition to these instructions, you have been provided with a `.zip` file containing a number of items[2] that you will need for this assignment. In particular, you have been supplied with a LaTeX template - `COM2009-3009_Assignment-2_Robot-XY.tex` - which will you use to compile your response sheet.

- ☐ Before you start working with LaTeX, edit the filename of the provided `.tex` file by replacing `XY` with your robot id.

[1] This is necessary because, as before, the robots are shared between Group A and Group B.

[2] The provided `SystemDiagram.jpg` is simply a placeholder that you will replace with your own diagram.

As before, your edited `.tex` file should compile to produce a `.pdf` document with your names on the front cover, followed by your responses to a number of questions. You will be submitting the `.pdf` file when you have completed the assignment, *not* the `.tex` file.

1.3 Hand-In Procedure

Once you have completed the assignment, one member of each group should submit your response sheet (in `.pdf` format) via MOLE. You do NOT need to submit a paper copy, and do NOT submit your LaTeX source files.

The filename should be of the form `COM2009-3009_Assignment-2_Robot-XY.pdf`. Please make sure that your names and the id of your robot are shown correctly on the front page of your report.

Standard departmental penalties apply for late hand-in[3] and plagiarism[4].

Feedback (including provisional marks) will be provided via MOLE within three working weeks of the hand-in date.

The deadline for handing-in this assignment (via MOLE) is ...

Midnight Friday 17th May 2019 (week 12)

1.4 Increase in expected levels of autonomy

In Lab Assignment #1, care was taken to guide you through the various aspects of design that were required to build a maze-running robot. This included an introduction to the Lego EV3 hardware, sample code for programming your robot, and detailed guides on how to calibrate and program a PID controller.

In Lab Assignment #2, we expect teams to work in a more independent manner to find their own solutions to the problems posed. Part I below provides guides as to the type of behaviours you will need to implement and also links to lectures where common solutions to similar problems were covered. We strongly encourage students to research novel solutions, or even come up with their own. Novel solutions can be discussed in the submitted report.

Also, we cannot guarantee extra lab sessions as for Assignment 1 so please plan accordingly. It is very important to manage your development time on each task to ensure that you get to the end in time for the competition.

Reminder: *Lab Demonstrators are provided to guide your development and address technical questions that you might experience. Given the expectation of independent development they will not be able to debug code or provide full solutions as they will be specific to your robot.*

[3] `http://www.dcs.shef.ac.uk/intranet/teaching/public/assessment/latehandin.html`
[4] `http://www.dcs.shef.ac.uk/intranet/teaching/public/assessment/plagiarism.html`

2 Part I: Building A Multilayer Robot Control Architecture

2.1 The Search and Assist Challenge

In Assignment 2 you will again develop a robot to compete against your peers in a simulated life-like challenge within the same test arena as Assignment 1. Your task is to program a 'search and assist robot' that must find a stricken individual in an unstructured environment (test arena with randomly distributed obstacles) and remain there until further assistance arrives. The individual's location can be identified by an alarm beacon (light source). Your robot must navigate to the individual's location while avoiding contact with obstacles and the arena walls. Your robot's start position, and the positions of the target and obstacles will not be revealed until the competition session so you will have to design a controller that works in many conditions. You shall be ranked by the number of seconds that your robot spends at the target location within the 2 minutes allowed for each robot.

The following sections guide you through development of the various control behaviours that your robot will require to resolve this challenge but we expect and encourage teams to develop novel solutions that improve performance further. You will also notice that there is increased requirement to design and document unit tests to demonstrate the functioning of your subsystems. The design of the unit tests is for you to define - as you would in a real engineering or R&D position.

2.2 Building An Obstacle Avoidance Behaviour

In Assignment 1 you developed a PID controller to follow a maze without touching walls i.e. avoiding obstacles. In your new group, use the lessons learned from the last assignment to give your robot an obstacle avoidance capability.

- ☐ Program a PID-based obstacle avoidance behaviour. Take into account the hardware designs that produced the best results in the maze following competition. Refer to Assignment 1 documentation for reminders on calibrating your controller.
- ☐ Design and document unit tests to demonstrate that your robot succeeds in this task.

Question 1 Describe your PID-based obstacle avoidance architecture with accompanying unit tests that demonstrate that it functions correctly. ***(worth up to 5 marks)***

Hardware:
For our PID-based obstacle avoidance architecture, we used the 2 ultrasonic sensors which were mounted sideways in such a way that the angle between the sensors was 90deg. The left sensor would output a low value if the robot is placed close to the left wall and thus the same behaviour was observed with the right.

Software:
We firstly tested and adapted the PID controller from the previous assignment in

order to achieve our first step towards obstacle avoidance behaviour. If the robot encounters a scenario where it has moved close to obstacle either on the left or right side. Our 2 ultrasonic sensors will detect this and move in the opposite direction. For example, when an obstacle is encountered on the left side of the robot, the robot will turn right to move away from the obstacle and vice versa. An extension to the PID controller was making the robot move in a reverse arc if the distance between the obstacle falls below a threshold value. We set the threshold value to 100mm. For making the robot go backward in an arc, we increase the speed of one wheel and maintain a minimum speed of the second wheel. Thus, this is how we achieved our first task which was obstacle avoidance (check the Unit Tests).

Unit Tests:

Test Case	**Behaviour Observed with pre-defined values**
The robot is placed near the left wall at a distance = 300mm	Move right
The robot is placed near the right wall at a distance = 346mm	Move left
The robot is placed near the obstacle in various directions (N, S, E, W)	Move left or right depending on the value received by the sensor
Distance less than 100 received by the left sensor	Move in a reverse circular arc towards right for 1 second and speed = 70
Distance less than 100 received by the right sensor	Move in a reverse circular arc towards left for 1 second and speed = 70

2.3 Adding A Search Behaviour

Your robot will now be able to react to obstacles but needs a guidance strategy to explore the arena in search of the individual (light). Thus, your task is to add a search behaviour allowing your robot to explore the environment.

- ☐ Program a search behaviour that will drive your robot around the arena. Consider the various search algorithms that you might use to most efficiently explore the test arena. *Hint: search strategies were covered in Lecture 6 but there are many others available.*
- ☐ In your robot kits you will find a Gyro Sensor which can be used to monitor turning angles and angular accelerations. This may or may not be useful to your specific search strategy. Refer to the online EV3 documentation (`https://ev3dev-lang.readthedocs.io/projects/python-ev3dev/en/stable/spec.html`), to understand how to access the data from the Gyro Sensor.
- ☐ Again, design and document unit tests to demonstrate that your search strategy performs as expected. If time allows, you may want to implement and compare two different search algorithms to assess which is best suited to the task, or even implement a novel hybrid search strategy.

Question 2 Describe the design of your search algorithm including discussion

of design choices and what testing you have implemented. ***(worth up to 15 marks)***

The first design we came up as a team:

Spiral Search Strategy: Initially, we thought of implementing the Spiral Search strategy by making the robot move in regular circles and gradually increasing the radius of it by increasing the difference in speed of the 2 motors. But while implementing this strategy we thought it wouldn't be easy for us to incorporate the obstacle avoidance behaviour because this could put the search strategy off track and lead to parts of the arena not being checked as the PID function would be called the moment in comes near the wall and would cause more problems.

The second design we came up as a team:

Spiral Search Strategy variation: Set up a time limit for the robot to move straight, then have the robot rotate a set amount of degrees using the gyro sensor to accurately detect how many degrees the robot has turned, similar to spiral search but instead of completing circles, it would complete other regular shapes. For example, rotate 90deg for 1.5 seconds with a speed of 80 every 4 seconds and create squares by increasing the length of the squares so that we can create bigger squares.

The final design we came up as a team:

The mixture of Spiral Variation and Brownian Motion:

Hardware: The hardware remains the same as the obstacle avoidance which means the same Ultrasonic sensors mounted sideways at 45 deg.

Software: Our Search algorithm works in such a way that the robot moves with a speed of 85 and we check the Left distance and the Right distance using the Ultrasonic sensors every 1 second. If the Left distance is less than the right distance then the robot turns right by 90deg and goes ahead for 0.8 seconds. Similarly, if the Right distance is less than the Left distance then the robot turns left and goes ahead for 0.8 seconds. This procedure is repeated recursively as the distance detected by the sensors always varies and thus makes the robot move in the whole arena so that it covers the maximum area. The pseudo code of our Search algorithm:

Pseudo code
speed = 85
left_distance = sensor_left.value()
right_distance = sensor_right.value()
time.sleep(0.5)
if left_distance < right_distance:
motor_left.run_direct(duty_cycle_sp= (-speed))
motor_right.run_direct(duty_cycle_sp= (10))
else
motor_left.run_direct(duty_cycle_sp= (10))
motor_right.run_direct(duty_cycle_sp= (-speed))
time.sleep(0.5)

Unit Tests:

Test Case	Behaviour Observed with predefined values
When the left distance reading was more than the right distance (1264mm > 678mm)	The robot turned left at 90deg and then started moving straight.
When the left distance reading was less than the right distance (785mm < 1324mm)	The robot turned right at 90deg and then moving straight
When the right distance reading was more than the left distance (693mm > 545mm)	The robot turned right at 90deg and then moving straight
When the right distance reading was less than the left distance (778mm < 969mm)	The robot turned right at 90deg and then moving straight
Enter all 4 grid quadrants	The robot eventually visits all 4 quadrants

2.4 Switching Behaviours Using A Subsumption Architecture

Your robot should now have two functioning, single layer behaviours: obstacle avoidance and search. You now need to consider how your robot will select which strategy to apply and when.

- ☐ With reference to lecture 4 implement a subsumption strategy allowing your robot to search across the environment while avoiding obstacles.
- ☐ To verify that your robot is performing as you intend, design some unit tests that demonstrate appropriate switching between different behaviours.
- ☐ Consider what parameters to optimise to improve performance.

Question 3 Describe your subsumption architecture, tests that you have implemented, and performance improvements that led to them. ***(worth up to 15 marks)***

Hardware: Our hardware remained the same as above.

Software:
For our subsumption architecture design, we defined distance limits to determine the robot's behaviour. We set the value for our PID as > 100mm and < 400mm which means that if the sensors detect the value to be in between 100mm and 400mm then PID function is called whereas if the sensors detect that the value is less than (<) 100mm then switch to the reverse function. For making the robot go backward in an arc, we increase the speed of one wheel and maintain a minimum speed of the second wheel. The robot would execute the reverse arc for 1 second. If right and left distance from the obstacles are greater than 450mm then execute our searching algorithm. That was our Subsumption architecture.

Unit Tests:

Test Case	Behaviour Observed with predefined values
The searching algorithm still works correctly when combined with obstacle avoidance	All searching algorithm unit test pass
Obstacle avoidance still works correctly when combined with the searching algorithm	All obstacle avoidance unit test pass
Right sensor detects the distance to be 625 mm	Executes the searching algorithm, by checking the left and right distance continuously.
Left sensor detects the distance to be 890 mm	Executes the searching algorithm, by checking the left and right distance continuously.
Right sensor detects the distance to be 250 mm	Executes the Obstacle avoidance behaviour by turning left for 1 sec
Left sensor detects the distance to be 294 mm	Executes the Obstacle avoidance behaviour by turning right for 1 sec
Right sensor detects the distance to be 90 mm	Executes the reverse function where it turns in a reverse arc towards left for 1 sec with speed 70
Left sensor detects the distance to be 70 mm	Executes the reverse function where it turns in a reverse arc towards right for 1 sec with speed 70

2.5 Adding A Stop Behaviour

In the final competition, the individual's location can be identified by an alarm beacon (omnidirectional light source) that can be used to know when your robot has found the target.

- ☐ In your robot kits you will find a Colour Sensor which can be used to classify colours, measure reflected light intensity or measure the ambient light intensity. You should mount this sensor on your robot and connect to the EV3 using the same port array where your ultrasonic sensors are already connected.
- ☐ Refer again to the EV3 documentation to find the command that allows you to access the Colour Sensor. You should use Ambient Light Intensity mode.
- ☐ Document light readings in various parts of the arena - far away from the light, close to the light (remember that conditions might adapt due to the windows - can you document this?).
- ☐ Using the above information consider where to position the Colour Sensor on your robot and any adaptations that you might make to your robot morphology to improve the signal.
- ☐ You should now have a sufficiently good understanding of the information available to program a stopping behaviour for your robot. Program this and add it as another

layer of your submsumption architecture.

Question 4 Describe the stopping behaviour including describing your hardware and software solutions, any calibration attempted and any unit testing performed. Also describe if you added via a subsumption architecture or some other control strategy *(worth up to 15 marks)*

Hardware:
Our hardware was the same except for which we mounted a colour sensor at the front of the robot facing forwards, and covered by a shroud to minimise ambient light detection such as sunlight.

Software:
We made a separate function called "checkLight" and had defined, if the light intensity was below 27 then run the Subsumption architecture whereas if the light intensity was above 27 then stop the robot in it's current position by locking both the motors at -0 and -0 and eventually terminate the program. We found out that our threshold was 27 by placing the robot at a distance near the light source and eventually recorded the readings at different distance. We observed that the farther the distance from the light source the lesser the readings. Thus, for our robot we chose the value 27 which was 200-250mm away from the light source. When it hits this reading it will stop running any further functions.

We had also defined this function (checkLight) in the end of Subsumption so that it works recursively and checks the light always while searching.

Unit Tests:

Test Case	**Behaviour Observed with predefined values**
Light intensity = 27 when distance = 220mm	Robot stops
Light intensity = 30 when distance = 150mm	Robot stops
Light intensity = 35 when distance = 100mm	Robot stops
Light intensity = 40 when distance = 50mm	Robot stops
Light intensity = 22 when distance = 300mm	Robot continues to run subsumption architecture loop
Light intensity = 18 when distance = 375mm	Robot continues to run subsumption architecture loop
Light intensity = 14 when distance = 637mm	Robot continues to run subsumption architecture loop
Light intensity = 05 when distance = 900mm	Robot continues to run subsumption architecture loop

Light intensity readings at different points in the arena:

Location	Light Intensity
Facing light source at a distance $<$ 200mm	R24+
Facing light source at a distance $>$ 200mm	17-24
Facing away (ambient light)	12-17
Very far from the light $>$ 1000mm	1-6

2.6 Adding a Beaconing Behaviour

Your robot now has all the components required to solve the task: search; obstacle avoidance; and stop criteria. However, performance can be improved further by appreciated that the light also provides information that your robot can use to approach the light source. You are now going to add this beaconing behaviour to your robot.

- ☐ Consider how the light provides information that could be used to guide your robot to the target. Also consider how you would design a control algorithm that will drive your robot towards the light source (e.g. up a gradient) using a single sensor. You will need to work as a group to develop both a robot morphology and control algorithm to perform this task. *You might want to refer to material covered in Lecture 6.*

Question 5 Describe the design of your light following behaviour including describing your hardware and software solutions, any calibration attempted and any unit testing performed. ***(worth up to 15 marks)***

Hardware:
Our hardware has remained the same except we mounted a Gyro sensor on top of the robot so that we keep a track of the angles when it spins.

Software:
For our light following behaviour, we came up with a solution which works as follows: After finalising our search strategy as shown above and adding a stopping behaviour after it detects the light intensity above 27, our design involves spinning the robot every 5 seconds. We set our conditions as such the robot spins only when the light intensity it detects is above 17. Otherwise run the subsumption function which has the search strategy and the PID obstacle avoidance behaviour. So, when the robot spins for the 1st time, we keep a record of the light intensity and pass it into an array. After it completes 1 full spin (360 deg), we find the angle where the maximum light intensity was detected by the Gyro sensor which we placed above our robot and thus our robot turns back to that angle where it was maximum and then continues to search the light.

However we encountered issues with our connection of gyro sensor as it used to terminate the program due to port error. Hence, we came up with an alternate solution. What we did was to have the robot rotate for 3 seconds. While rotating the colour sensor would continuously record the light intensity and save it in a count variable. The count would then reset and allow the robot to rotate again until the next count equals the the previous best count where light intensity was the greatest. This was how we implemented the Beaconing behaviour without the Gyro sensor and

made it search the light only when the intensity was above 17. If light intensity is below 17, it would simply follow the Subsumption function and thus trigger this piece of code only when it reaches 17.

Unit Tests:

Test Case	**Behaviour Observed with predefined values**
Ambient light below 17	Continue the subsumption function (Searching, PID and Reverse)
Ambient light above 17 briefly	It would pause Subsumption to execute a spinning light scan. As it doesn't find any light over 17 intensity, it goes back to Subsumption
Static light source with an intensity above 17	Subsumption would pause, a spinning light scan would be executed and upon detecting light above 17 and below 27, the robot would turn towards the maximum light intensity found after 1 complete spin where that was detected and move forward
Ambient light intensity higher than 27	Robot would halt

2.7 Final Multi-layer Control Architecture

You will need to incorporate this new beaconing behaviour into your robot control strategy.

- ☐ Can this behaviour be easily added to the subsumption architecture used previously? Or is an alternative method now required? (*you might want to refer to material covered in Lecture 4.*) In your team you should develop a multi-layer control strategy allowing your robot to complete the final task for the Search and Assist competition. Use unit testing to assess if your design choices were correct and justify any changes made.

Question 6 Describe your final multi-layer control architecture. You should discuss if you chose to stick with a subsumption architecture or use an alternative and justify this design choice. Support your conclusions with data from unit tests. ***(worth up to 15 marks)***

Hardware:
The hardware remains the same except we removed the gyro sensor and just retained the 2 ultrasonic sensors and the colour sensor.

Software:
We chose to stick with our subsumption architecture, however, in our final multi-layer architecture we did not incorporate all behaviours.

We changed our Search strategy as shown in Q2. Our Search strategy was a bit different this time, it works in such a way that we just make the robot go straight and rotate right or left unless and until it finds an obstacle or a wall. If the Left distance detected by the sensor is critically less than the set value which in this case was set to (150) inside the Subsumption architecture, then the robot would just opt for reverse direction for 1 second depending on which sensor catches the low value. For example: If the left sensor detects value as 100mm, then it would call the reverse function which makes the robot perform a reverse arc towards right by running the left wheel backwards at speed (80) and the right wheel at (5) vice-versa. If the distance is not critically low which means that if any of the sensors detect the a value which is in between 240 and 400, then run the PID obstacle avoidance behaviour. The behaviour is very much similar to the above but it doesn't check the left and right distance anymore and we increased the values for e.g. reverse distance, speed and the left and right distance.

Our design for the light following behaviour initially was that the robot will run a light intensity scan, if the intensity is above the threshold the robot will attempt to follow the brightest light intensity. If not, it will run the search function for five seconds before running another scan. This is interrupted by the PID and the reverse function if it gets close to the wall or obstacle.

In our final program, we decided that the light scan was not reliable enough to locate the brightest light source(beaconing), so instead, we relied on the subsumption consisting the search strategy, PID obstacle avoidance and reverse. In order for it to find the light we just implemented a function named checkLight as shown above and put the stopping behaviour as such if the light intensity that it receives is above 27 then just stop the robot when it searches. Thus, all of these are called recursively so that the robot checks, avoids obstacles and also stops when it detects the light.

Unit Test:

Test Case	**Behaviour Observed with predefined values**
The robot is placed near the left wall at a distance = 300mm	Move right
The robot is placed near the right wall at a distance = 346mm	Move left
The robot is placed near the obstacle in various directions (N, S, E, W)	Move left or right depending on the value received by the sensor
Distance less than 100 received by the left sensor	Move in a reverse circular arc towards right for 1 second and speed = 80
Distance less than 100 received by the right sensor	Move in a reverse circular arc towards left for 1 second and speed = 80
When the left distance reading was more than the right distance (400 > 390)	The robot turned left at 90deg and then started moving straight.
When the left distance reading was less than the right distance (289 < 412)	The robot turned right at 90deg and then moving straight
Enter all 4 grid quadrants	The robot eventually visits all 4 quadrants
The searching algorithm still works correctly when combined with obstacle avoidance	All searching algorithm unit test pass
Obstacle avoidance still works correctly when combined with the searching algorithm	All obstacle avoidance unit test pass
Right sensor detects the distance to be 625 mm	Executes the searching algorithm, by going straight
Left sensor detects the distance to be 890 mm	Executes the searching algorithm, by going straight
Right sensor detects the distance to be 250 mm	Executes the Obstacle avoidance behaviour by turning left for 1 sec
Left sensor detects the distance to be 294 mm	Executes the Obstacle avoidance behaviour by turning right for 1 sec
Right sensor detects the distance to be 120 mm	Executes the reverse function where it turns in a reverse arc towards left for 1 sec with speed 70
Left sensor detects the distance to be 140 mm	Executes the reverse function where it turns in a reverse arc towards right for 1 sec with speed 70

Light intensity = 27 when distance = 300mm	Robot stops
Light intensity = 30 when distance = 150mm	Robot stops
Light intensity = 35 when distance = 100mm	Robot stops
Light intensity = 40 when distance = 50mm	Robot stops
Light intensity = 22 when distance = 250mm	Robot continues to run subsumption architecture loop
Light intensity = 18 when distance = 375mm	Robot continues to run subsumption architecture loop
Light intensity = 14 when distance = 637mm	Robot continues to run subsumption architecture loop
Light intensity = 05 when distance = 900mm	Robot continues to run subsumption architecture loop
Facing light source at a distance < 200mm	R24+
Facing light source at a distance > 200mm	17-24
Facing away (ambient light)	12-17
Distance > 800mm	1-6

**NOTE: Some of the Unit Tests have been repeated because we had retained and original design and made a few changes only to our search algorithm.

2.8 Summary

Your robot is now ready to compete in the final competition!

3 Part II: Search and Assist Competition

3.1 The Competition

As for Assignment 1, the competition will be organised into a number of leagues, each consisting of several teams (and their robots). Marks will be awarded based on each robot's performance **within its league**. This will mean that each team will be competing against teams that have had an equal amount of time devoted to the development of their respective robots. The competition will take place during the final lab session.

Note: *If there is time at the end of the league battles, the winning robot in each league will compete again to find the overall champion. This 'championship' contest will not count towards the marks for the assignment, but a (small) prize may be awarded.*

As you can imagine, running such an event with a large number of teams/robots requires very precise time management, so we will be issuing a strict timetable for the final lab session. This should appear on MOLE the week before. It is essential that you prepare carefully for your designated time slot (otherwise you may lose marks - see below).

The rules for the competition are as follows:

1. Each team must register their arrival at the arena (with their robot) 10 min prior to their league's designated time slot, after which no further technical development will be permitted.
2. Each team will be called forward in turn to place their robot on the starting position.
3. An 'official' run for each team's robot will be timed by the lab demonstrators / lecturers. They will record the accumulated time that the robot spends in the designated target area surrounding the light source during the 2 minutes allowed.
4. Each 'touch' with a wall or obstacle will incur a **-10 second penalty**.
5. If a robot needs to be 'rescued', a *designated* team member may place it back in the arena the course at the location where things went wrong. However, the clock will keep running.
6. The designated rescuer must stand *outside* the arena next to the starting position, and return to that position after each rescue.
7. Up to two rescues are permitted.
8. Each rescue will incur a **-20 sec penalty**.
9. A third rescue is NOT permitted. Instead the run will be terminated and the total time accumulated at the target recorded.
10. Teams will be ranked in their league according to their accumulated time at the target (minus any penalties). [what to do with people clustered on 0min?]

Marks will be awarded as follows:

- 15 marks will be awarded to the team with the best run within their league, 14 marks to the second best team, and so on.
- 5 *bonus* marks will be awarded for a robot that stops within the target location within the 2 minute time limit with *no* wall or obstacles touches and *no* rescues.

- A team who fails to appear at the starting position will receive 0 marks.

3.2 Your Team's Performance

Question 7: *What was the result of your official attempt?* (Worth up to 20 marks)

Total Time at Target	Wall/Obstacle Touches	Rescues
00:02	1	0

Note: *The demonstrators will have already recorded the above information. However, please include it here as a cross-check.*

Finally, how did you organise your team? I.e. what was each member's role and responsibility, and what was each person's contribution as a % (adding up to 100%)? Please fill in the Table below:

Team Member	Role	%
Earlando Grant Favourite	Helped with coding all functions	33.3
Alistair Cook	Helped with coding all functions	33.3
Sree Pittala	Helped with coding all functions	33.3