University of Sheffield

COM2009-3009
Human-Machine Interaction and Robotics

Lab Assignment #1

'Closed-Loop Control'

Megan Wright

Rafael Cavagnoli

Sohyun Park

Robot: B1

Department of Computer Science

March 6, 2019

Contents

1 Part I: Control Strategies

Before attempting to control the robot, it is first necessary to understand the behaviour of its actuators and sensors.

1.1 Actuators

The example program - `com2009-3009_ev3dev_test.py` - shows how the robot can be driven in a straight line by sending the same value to each motor. However, you need to be able to steer the robot by controlling the wheels 'differentially'. This can be achieved as follows ...

Let v be the base speed of the wheels and δ_v the speed differential between the two wheels, i.e. $v_L = v + \delta_v$ and $v_R = v - \delta_v$, where v_L and v_R refer to the left and right wheels respectively. If $\delta_v = 0$ and $v \neq 0$, then the robot will travel in a straight line. If $v = 0$ and $\delta_v \neq 0$, then the robot will rotate on the spot. If $v \neq 0$ and $\delta_v \neq 0$, then the robot will move along a curved trajectory.

- ☐ Implement the above scheme on the robot and confirm its correct operation by experimenting with different values for v and δ_v. Note that large values of v_L and v_R will be hard-limited by the maximum speed of the motors.

Question 1: *What happens when $v = \pm\delta_v$?* (Worth up to 2 marks)

When v and $\pm\delta_v$ have the same value and sign, the right motor speed becomes 0. If they are both positive the robot rotates anti-clockwise and if they are both negative the robot rotates clockwise. If the values are the same but the signs are opposite, the speed of the left motor if set to 0. If $\pm\delta_v$ is positive and v is negative then the robot will rotate clockwise, and if $\pm\delta_v$ is negative and v is positive, it will rotate anti-clockwise.

1.2 Sensors

The example program - `com2009-3009_ev3dev_test.py` - shows how to read the output of the ultrasonic sensor. However, for real-time control, sensor outputs need to be sampled at regular intervals.

- ☐ Implement an *inner* processing loop that samples the output of the ultrasonic sensor once every 10 msec.
- ☐ Implement an *outer* processing loop that calculates the running[1] mean, standard deviation, minimum and maximum values from the sensor over a period of 10 secs (i.e. 1000 samples). On completion, display the results on the EV3 screen and/or the `VSCode` debug window.
- ☐ Experiment with your robot facing various surfaces.

[1] Here's how to calculate a 'running' mean and standard deviation: `https://www.johndcook.com/blog/standard_deviation/`

Question 2: *How do the mean, standard deviation, minimum and maximum values from the ultrasonic sensor vary as a function of the actual distance to a surface?* (Worth up to 4 marks)

As the the number of samples increase the standard deviation starts decreasing towards zero. The final standard deviation value after a thousand samples is 0.00504 mm. However, the mean starts increasing towards a more consistent value, if analysed the real value which is 15 cm to the final mean after a thousand samples which is 14.8 cm it will all differ by 1.2 cm which for the experiment and type of application that the robot will be used for, is plausible. The minimum mean value differ by 9 to the real value and the maximum will differ by 1.

Question 3: *How do the mean, standard deviation, minimum and maximum values from the ultrasonic sensor vary as a function of surface angle?* (Worth up to 4 marks)

Taking in an account a comparison between the first experiment and this one, it can clearly be seen that as the surface angle changes the and accuracy of the distance will decrease because the sensor won't be able to calculate an exact distance. Therefore, the best way to calculate the distance between the robot and a wall is to keep the robot 90 degrees surface angle from the wall. That is for a better precision. Comparison between experiments with same straight distance between the robot and the wall: Standard deviation: 0.005, 0.08 (second) Mean: 6.78 cm, 14.8 cm (second)

1.3 Open-Loop Control

- ☐ Remove the ultrasonic sensor from your robot.
- ☐ Write a program that causes the robot to travel 0.5m in a straight line, rotate 180°, retrace its path, rotate 180° (and so on) continuously.
- ☐ Experiment with different surfaces.

Note: *Don't spend too long on this. You'll find that it's quite difficult to achieve a satisfactory solution - and that's the point!*

Question 4: *In the above implementation, (i) how did you determine the control parameters for ensuring that the robot travelled the target distance and rotated the appropriate amount, (ii) how sensitive was it to the selected parameters, and (iii) how reliable was the best solution?* (Worth up to 4 marks)

(i) We tried a method in the ev3dev documentation called run_timed, which can rotate the wheel by calculating time and speed, however it didn't work in the way we needed for this question. So we implemented based on running the robot with motor speed and time.sleep. As we couldn't calculate the exact distance due to the delay in reaching the target speed we set, we tried multiple values to get 0.5m and 180 degrees. After a few times of experiments we could get speed=60, time=2.5 for moving 0.5 m and speed 10 and 2 seconds for rotating 180 degrees.

(ii) Not very sensitive in the sense that a slight change in value doesn't make a noticeable difference. However, a drastic change in values would completely change the way the robot moves.

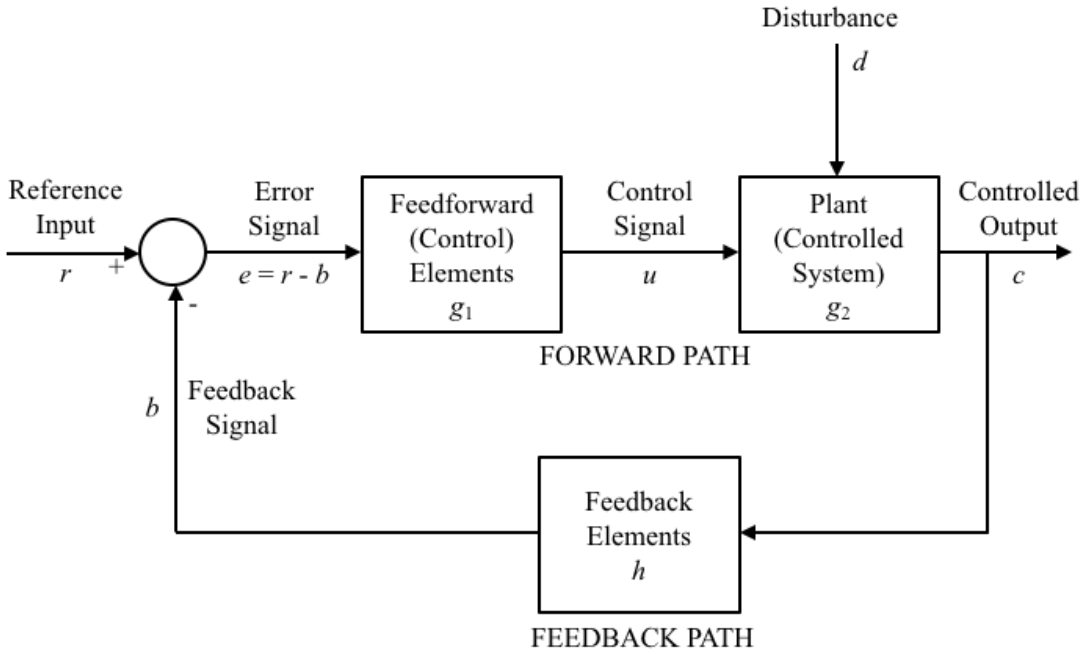
(iii) Not very reliable, we can't find the exact metric of speed thus can't get an approximate calculation before we actually run the code, and so most of our testing was down to trial and error.

1.4 Closed-Loop Control

In the control system you implemented above, you will have discovered that after a number of iterations backwards and forwards, the robot was no longer following its original track. Any disturbances (e.g. due to wheel slippage or variations in surface texture) and/or inaccuracies in the control parameters would cause the robot to deviate from its intended path. This is because the robot had no information about its position or orientation, i.e. it was using 'open-loop' control (otherwise known as 'dead reckoning').

Clearly, if a robot had 'feedback' from the environment, then it could use that information to maintain its intended path, i.e. 'closed loop' control. In particular, closed-loop control actions can be based on minimising the *difference* between a target value and a sensed value, and this is known as 'negative feedback' control.

A classic closed-loop negative-feedback control system is structured as follows:



... where the input reference signal r specifies the 'setpoint' for the system, i.e. the target

value of the feedback signal b. Based on the value of the error signal e, the controller g_1 sends signals to the actuators g_2 causing some behaviour c. Sensors h detect the consequences of the behaviour and provide feedback b, which is compared to the reference r. The difference between the feedback signal b and the reference r generate the error signal e, and the loop continues (iteratively minimising $r - b$).

The controller g_1 is commonly configured as follows:

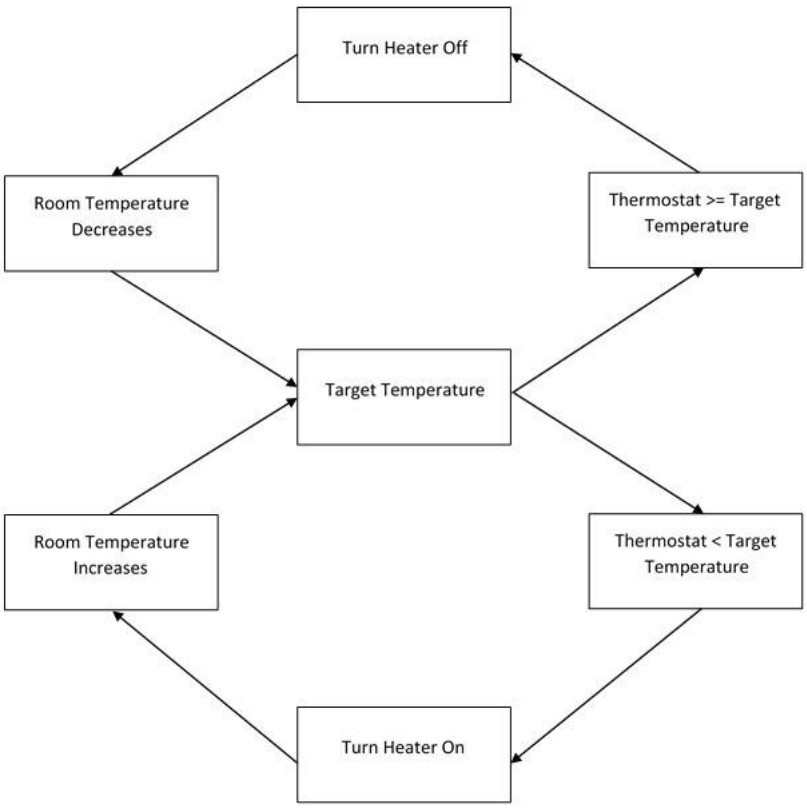
$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de}{dt}(t),$$

... where K_P is the 'proportional' gain, K_I is the 'integral' gain and K_D is the 'derivative' gain. This is known as a 'Proportional-Integral-Derivative' (PID) controller. Note that a simple controller might only use K_P (i.e. $K_I = 0$ and $K_D = 0$).

Question 5: *In a negative-feedback control system, what is the significance of $e = 0$?* (Worth up to 2 marks)

When e = 0, the difference between the reference input and the feedback signal is 0, and so there is no error. This means the sensed value and the actual value that is expected are the same.

Question 6: *Give an example of a real-world negative-feedback control system and describe its operation.* (Worth up to 8 marks)



The heaters we use indoors use a negative feedback system. The heater heats up the room and

uses the thermostat (sensor) to check the current temperature. If the room temperature is lower than the set temperature, then it keeps the heater on, making the room warmer. If it's the room temperature is the same or higher then it turns the heater off. This is so that it can maintain or cool the room.

Question 7: *Assuming that $K_I = 0$ and $K_D = 0$, what would happen to your example system if (i) $K_P = 0$ or (ii) $K_P < 0$?* (Worth up to 4 marks)

If $K_I = 0$ and $K_D = 0$ then the given formula will be the same as $u(t) = K_P e(t)$ it means that the control variable $u(t)$ will depend on K_P and error value $e(t)$.

(i) K_P is proportional tuning, so it involves correcting a target proportional to the difference. If the difference of the target and sensed values is 0, then it wouldn't perform tuning.

(ii) It will make the $u(t)$ value negative. A PID controller uses a negative feedback system. If the $u(t)$ is negative, then it will make the system unstable. (This could lead to increasing oscillations and possible destruction of the mechanism.)

1.5 Basic PID Controller

The next step is to program the robot to position itself *autonomously* at a set distance from a vertical surface (such as a wall). In order to do this you will need to implement a closed-loop negative feedback system incorporating a PID controller.

- ☐ Re-attach the ultrasonic sensor.
- ☐ Implement a PID controller of the form $u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de}{dt}(t)$, where the 'reference input' r corresponds to the desired distance between the robot and the surface, the 'feedback signal' b corresponds to the output of the ultrasonic distance sensor, and the 'control signal' u corresponds to the value that is sent to the motors. You will have to measure dt (the loop time) using the system clock.

Once you have implemented the code, you should follow these steps:

- ☐ Set $r = 50$ cm.
- ☐ Set $K_P = 1$, $K_I = 0$ and $K_D = 0$.
- ☐ Place the robot 100 cm from a vertical surface with the ultrasonic sensor facing towards it.
- ☐ Run the code.

If all is well, then your robot should move slowly towards the surface slowing down and eventually stopping as it approaches the 50 cm point.

Note: *If your robot appears to do the opposite of what you expect, try setting $K_P = -1$. If that solves the problem, then you have implemented a positive-feedback rather than a negative-feedback loop. This means that you need to swap the polarity of the signal being sent to the motors (and reset $K_P = 1$).*

Question 8: *What happens if, after the robot has arrived at the 50 cm point, you manually push it towards the surface?* (Worth up to 2 marks)

If we manually push the robot towards the surface, it goes backward. Similarly, if we place it further then it moves toward to get to the 50cm point again. To check the code and the sensor is working correctly, we put an obstacle right in front of the robot's sensor when it was moving, and it went backward. When removed, it went forward. As the robot keeps getting the value from the sensor, it repeated this behaviour every time we put and remove the obstacle.

Now implement an outer loop that swaps r between 30 cm and 50 cm every 10 seconds. As a result, your robot should now change its position at regular intervals. These step changes in r will make it easier to observe the robot's behaviour for different values of K_P, K_I and K_D.

Question 9: *With $K_I = 0$ and $K_D = 0$, what happens when you experiment with different values of K_P? In particular, (i) what is a 'good' value for K_P (and why), (ii) what value of K_P causes the robot to start to oscillate continuously backwards and forwards around the target r, and (iii) when it starts to oscillate continuously, what is the oscillation period? Hint: start with $K_P = 1$, then increase it gradually.* (Worth up to 5 marks)

We have tried when K_P is 0, 0,5, 1, 5, 7, 10, 20, 50.

(i) Good value for K_P is the value that could stabilise the robot in a short time and shouldn't be too high or too low. We placed the robot about 20cm away from the wall. When we had 20 and 50 for K_P value, it went backward in very high velocity, missing the exact point. When we had 0 or 0.5 for the K_P value, it took longer to get the right point than we had the good value for K_P. In our case, when we had 5 for K_P value, it found the exact 500 and 300 point within 5 seconds.

The value of K_P that causes the robot to oscillate backwards and forwards continuously is known as the 'ultimate gain' K_U, and the oscillation period is designated as T_U.

Note: *Be aware that T_U is a measure of time, not frequency. I.e. if the robot oscillates backwards and forwards at a rate of five times a second, then $T_U = 1/5 = 0.2$ secs.*

Question 10: *What is the relationship between the 'good' value of K_P that you discovered by experimentation and the value of K_U that you measured?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

1.6 PID Tuning

There are several methods available for tuning the parameters of a PID controller. The *manual* method involves setting $K_P = 0.5K_U$, $K_I = 0$ and $K_D = 0$. Next, K_I is gradually increased to improve the convergence. Then K_D is gradually increased to improve the responsiveness.

A more formal tuning approach is known as the *Ziegler-Nichols* method. Once K_U and T_U have been determined, the PID gains may be set according to the following heuristic:

	K_P	K_I	K_D
P	$0.5K_U$		
PI	$0.45K_U$	$0.54K_U/T_U$	
PID	$0.6K_U$	$1.2K_U/T_U$	$3K_UT_U/40$

Question 11: *What values of K_P, K_I and K_D are given by the Ziegler-Nichols method for a P, PI and PID controller (based on the values of K_U and T_U you measured previously)?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

☐ Confirm the effectiveness of the Ziegler-Nichols method by setting the values for K_P, K_I and K_D in your controller.

Question 12: *Using the values for K_P, K_I and K_D given by the Ziegler-Nichols method, what happens when you decrease the sampling rate (e.g. by changing the value of the 'wait' function in the loop)? What is the significance of your observations?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

1.7 PID-based Steering

Finally (for Part I), reposition your ultrasonic sensor so that it faces sideways from the robot.

Now re-program your robot with a PID controller that maintains a fixed distance from a wall as the robot travels along parallel to it. This will require the speed to be set at some fixed value, and the PID-based control loop will handle the steering. Use the Ziegler-Nichols method to determine appropriate values for K_P, K_I and K_D (this can be done while the robot is stationary and changing the target distance a few centimetres once every 10 seconds, as before).

Note: *It is particularly important that the sensor is mounted well forward of the driving wheels, otherwise it will not be sensitive to changes in direction when the robot is stationary.*

Question 13: *What values for K_U and T_U did you measure, and what are the resulting values for K_P, K_I and K_D given by the Ziegler-Nichols method?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

☐ Confirm the effectiveness of these values for K_P, K_I and K_D by having the robot travel along a wall maintaining a constant distance from it.

1.8 Summary

You have successfully completed Part I of the assignment, and you should now have all of the skills necessary to move on to Part II.

2 Part II: Maze-Running Competition

2.1 Objective

The aim of Part II of the assignment is to use the theoretical knowledge and practical experience you have acquired in Part I to design and implement a robot that is capable of competing in a maze-running competition.

The challenge is to navigate a corridor (that will be set up in the robot arena) in the fastest time possible and without touching the walls. The precise layout will not be revealed until the final lab session. However, you will be able to practice in the arena beforehand.

You need to decide which control principle(s) to use and how to set up your robot's sensors and actuators. Be sure to take into account information you have learnt in Part I. Also, since it's a competition, you will need to to think about how your robot manages 'risk', i.e. is it better to be slow-and-careful or reckless-but-fast? You will probably need to experiment with various alternatives before deciding on a final approach.

Note: *You will need to adopt good working practices for (i) organising your team and (ii) software version control. The latter is important as robots are notorious for failing after a so-called 'improvement', so being able to revert easily to an earlier version will save a lot of pain and grief.*

Note: *Each Lego kit contains two ultrasonic sensors.*

Note: *The maze will be a long winding corridor. There will be no dead-ends, no loops and no junctions. However, there may be chicanes and small breaks in the wall.*

2.2 The Competition

In order to give every team an equal chance, 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. Any number of practice runs may be made (subject to the availability of the arena).

2. Each team must register their arrival at the arena (with their robot) 10 mins prior to their league's designated time slot, after which no further technical development will be permitted.
3. Each team will be called forward in turn to place their robot on the starting line.
4. An 'official' run for each team's robot will be timed by the lab demonstrators.
5. If a robot fails to complete a run within a **2 minute time limit**, the demonstrators will record the furthest position it reached.
6. Each 'wall touch' will incur a **10 sec penalty**.
7. If a robot needs to be 'rescued', a *designated* team member may place it back on the course at the location where things went wrong. However, the clock will keep running.
8. The designated rescuer must stand *outside* the arena next to the starting position, and return to that position after each rescue.
9. Up to two rescues are permitted.
10. Each rescue will incur a **20 sec penalty**.
11. A third rescue is NOT permitted. Instead the run will be terminated and the position reached will be recorded.
12. Teams will be ranked in their league according to their finishing time (including any penalties) or their distance travelled.
13. Teams who travelled the same distance will be ranked according to the time taken to get to that point (including any penalties).
14. A team which fails to arrive at the arena at the designated time will incur a **60 sec penalty** and may NOT get a run.

Marks will be awarded as follows:

- 15 marks will be awarded to the team with the best run within their league, 9 marks to the second best team, and so on.
- 5 *bonus* marks will be awarded for a run completed within the 2 minute time limit with *no* wall touches and *no* rescues.
- A team who fails to appear at the starting position will receive 0 marks.

2.3 Your Team's Solution

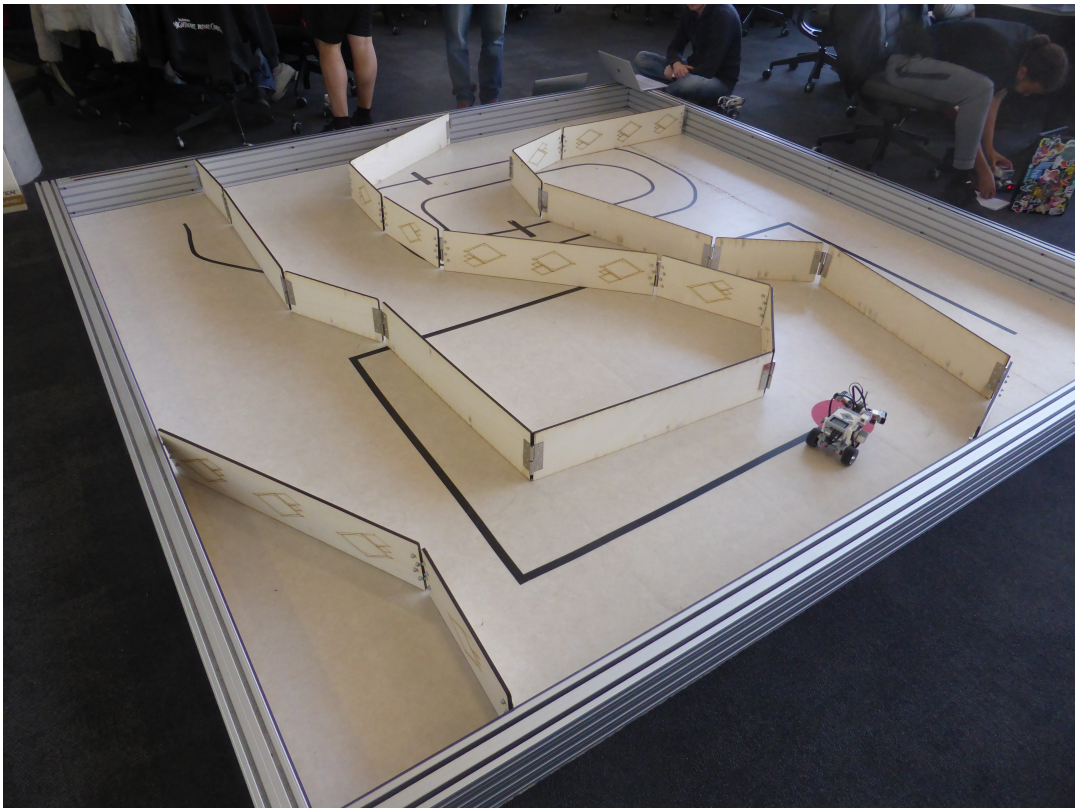
Question 14: *In terms of your robot's software architecture, what principles did you explore, and what was the final approach taken?* (Worth up to 15 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this

text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

Question 15: *What was your robot's final physical configuration (include a photo)?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.



(This is a photo of a possible maze. Replace it with a photo of your robot.)

Question 16: *Are there any points you wish to make about your robot's performance in the competition? For example, if you had had more time, what might you have done differently?* (Worth up to 5 marks)

Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this

text with your answer. Replace this text with your answer. Replace this text with your answer. Replace this text with your answer.

Question 17: *What was the result of your official attempt?* (Worth up to 20 marks)

Distance	Time	Wall Touches	Rescues
end reached/? metres	00:00	0/1/2/...	0/1/2/...

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	%
?	?	?
?	?	?
?	?	?