

Multi-Layer Control

Contents

2	Part I: Building A Multilayer Robot Control Architecture	4
2.1	The Search and Assist Challenge	4
2.2	Building An Obstacle Avoidance Behaviour	4
2.3	Adding A Search Behaviour	6
2.4	Switching Behaviours Using A Subsumption Architecture	8
2.5	Adding A Stop Behaviour	9
2.6	Adding a Beaconing Behaviour	11
2.7	Final Multi-layer Control Architecture	13
2.8	Summary	14
3	Part II: Search and Assist Competition	15
3.1	The Competition	15
3.2	Your Team's Performance	16

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.




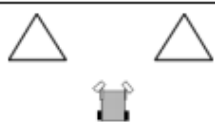
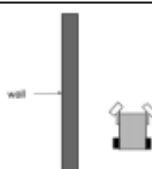
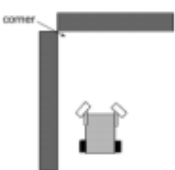
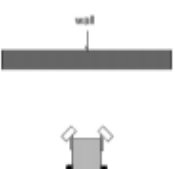
- ✓ Program a PID-based obstacle avoidance behaviour. Take into account the hardware designs that produced the best results in the maze following competition. Refer to Assignment 1 documentation for reminders on calibrating your controller.
- ✓ Design and document unit tests to demonstrate that your 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 - We positioned two ultrasonic sensors on the front of our robot, both facing 45 degree outwards. Initially, we tried positioning the sensors at 90 degree but it wasn't avoiding the obstacles efficiently. It did a good job but could be improved, especially when the obstacles are right in front of the robot. The robot would just bump into the obstacles. With the same code, we then tried positioning the sensors at 45 degree and it worked great, it avoids all the obstacles well even when the obstacles are right in front of it.

Software - The obstacle avoidance was heavily based on the work we did in Assignment 1. This used a closed-loop control system, the robot compares the target situation and the current situation and the resultant values closer to the desired situation. There is also a negative feedback loop to provide stability to the given resulting motor values. The goal for this part of the program is to stay a given distance away from any obstacles using an ultrasonic sensor on the left and right sides. The controller is PID and the feedback loop subtracts one ultrasonic sensor value from the other, this means that the robot is aiming for a feedback signal of zero, positive and negative feedback signals indicate it is closer to the left or right side respectively.

Unit Testing - We carried out a unit testing with a few scenarios. Then we tested the robot ability to avoid obstacles on the arena. The results are as follow:

Scenario	Predicted Result	Result
	Robot go straight to the obstacle	The robot turned left or right
	Robot turns right	The robot turned right
	Robot turns left	The robot turned left
	Robot moves and stays in the middle	The robot moved and stayed in the middle of the two obstacles.
	The robot follows the wall/ move straight	The robot moves straight following the wall
	Robot turns right at the corner	The robot turned right at the corner
	Robot turns left or right	The robot turned left or right according to where the initial position was

Most of the results are the same as the predicted results. However, some of the scenario gave different results from what we expected. For the scenario where the

robot is between two obstacles, when the robot is placed nearer to one of the obstacle, it tends to stick to the closer obstacle. This has something to do with the fact that we program the robot to stick to a closer obstacle when one of the readings of the sensor is too large. The test proved that our robot is capable of avoiding obstacles efficiently with the help of the two ultrasonic sensors.

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)

Hardware: Gyroscope attached onto the body of the robot to provide stability and maintain a reference direction in navigation, and the colour sensor attached directly in front of the robot with blocks of LEGO surrounding it to omit as much ambient light as possible and to get accurate readings.

Software: Our initial approach was to implement a Brownian walk and when the colour sensor was triggered by detecting a high enough value it would stop. This method was not the most effective as the robot would take a very long time to get into the position where it was both facing the light and close enough to it. We decided that it would be better to try and implement a modified Levy Flight algorithm where at a turn the robot would do a random turn but only if it hadn't been able to see the light around it at all. If it had seen the light then it would go towards it.

The way our robot behaves is it will initially start with a search for light. It does this by rotating on the spot in a random direction, left or right, until it is either triggered the colour sensor detecting a strong enough light signal or by it completing one full rotation. If it thinks it has seen the light then it will rotate slightly more since the

sensor detects light on the edge but we want the robot to face directly towards the light. If it has not seen the light it will be facing the direction it started in so it will generate a random angle to rotate to and rotate to that angle. It will then do a walk forwards for a random amount of time before searching again. Assuming the light isn't immediately found the actions of the robot will be search, walk, search, walk, ..., light seen, walk, light found.

The values for the random variables for how long to walk and the angle to turn to are important to calibrate. If the robot is walking for too long it may be within line of sight of the light but never stop to search for it, so after some experimentation we decided that random values between 1 and 7 seconds were best. For the angle to turn to we just wanted a random angle between no rotation and one full rotation. This values changes depending on the surface the robot is on and also the battery level, so for now it is set to 270 but this might need to be calibrated again in different conditions.

If the robot is triggered by the colour sensor value thinking it is facing the light it will proceed to walk towards the light, avoiding obstacles on the way, until either the light signal is gone below the trigger value in which case it has missed the light or until the light signal triggers the value which shows the robot has found the light and is within the circle when the program will stop. It is important to get the trigger values for both seeing a light from a distance and knowing it has found the light correct. If the trigger for seeing the light from a distance is too low then it will see light sources that are not there, if it is too big then it will take a long time or maybe will never get close enough to trigger it and we have the same problem as before. If the trigger for knowing it has find the light is too small then it could stop before it is within the circle which would be very annoying, and if it is too big then it might crash into the light or never stop meaning it misses the circle. The testing we implemented to come up with these values was to put the robot in the test arena, pointed it in the direction of the light and moved it closer while recording the colour sensor values. We decided to set the value for seeing the light to roughly double the radius of the circle and the value for finding the light for being a few centimeters inside the circle.

Once the light is found the robot will constantly be checking that the threshold value is still being reached, otherwise we can assume that it finding the light was an anomaly or that the robot missed the light in which case we will continue searching and walking.

Another method we tried to implement but couldn't get to work was to only trigger the robot to start walking towards the light when we observed the peak of the colour sensor value while we were rotating. We would do this by observing the colour sensor value increase as it rotated more and more towards the light until it would peak and then when it started to drop we would know we were facing directly towards the light. We then adjusted for the extra rotation we had completed and then walked towards it. Any negligible drops in the colour sensor value would be ignored. This proved difficult in practice though since there is a lot of interfering light in the arena and the values needed to be fine tuned.

Unit Testing: We performed two unit testing. The first one is basically just getting the colour sensor value when the robot is at different distance from the light. The second test is to get the colour sensor value and robot's distance from light for when

it founds the light.

Seeing Light Unit Testing:

Distance from Light (cm)	Colour Sensor Value
100	6
90	7
80	7
70	6
60	9
50	10
40	12

Found Light Unit Testing:

Distance from Light (cm)	Colour Sensor Value
40	12
35	13
25	15
15	17
10	19

The unit tests demonstrate (unsurprisingly) that that the further away the robot is to the light source, the lower the colour sensor reading, and the closer it is to the light source, the higher the colour sensor reading. However, there was not a big variance in readings as we expected the light reading to be much higher when the sensor was directly in front of the light compared to when it was further away - this means that we will have to choose our thresholds carefully so that the robot can detect when it has actually found the light source.

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: 2x ultrasound sensors for obstacle avoidance, gyroscope to reference direction and a colour sensor for searching and beaconing. This hardware will provide the necessary foundation of data for our subsumption architecture to work in practice.

Software: Subsumption architectures implement a strategy that accomplishes tasks sequentially and uses open loops which don't use feedback. When we were designing

the switching behaviours we kept these principals in mind. There are two main loops that switch between each other when the robot hasn't found the light, walking and searching. The walking will switch to search after a random walk time or if the light is seen. The searching will switch to walking after it has done a full rotation and rotated to a random angle or if the light is seen. When the light is seen the robot will change its obstacle avoidance behaviour to go towards objects namely the light while walking forwards.

Unit testing: We carried out one unit testing to make sure that the obstacle avoidance and search behaviors that we implemented are functioning well.

Scenario	Predicted Result	Final Result
Robot gets too close with an obstacle	Robot gets too close with an obstacle	Robot turned left and right and successfully avoided the obstacles
Robot in the middle of nowhere	the robot rotates randomly then facing the direction it started in, it will generate a random angle to rotate to and rotate to that angle. Robot then do a walk forwards for a random amount of time before searching again.	the robot rotated randomly then rotated a bit and walked for a random amount of time then repeat until it found a significant light intensity then it started walking to the direction of the light
Robot detects light while rotating	rotates to the angle it detected the light and moves straight to the light	after a full rotation, the robot rotates back to the angle where it detected the light and then walked straight at that angle

This test shows that the single layer behaviours: obstacle avoidance and search that we implemented so far are functioning well together. The robot was able to successfully avoid obstacles while doing search.

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 subsumption 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: We positioned the light sensor on the lower front of our robot. Considering that the light source is positioned a bit low, we decided that it is the best position that the robot can detect the light from the alarm beacon. The color sensor on our robot focus on sensing light straight in front of it.

Software: We firstly measured light readings in various parts of the arena. With that readings we decided to mark the arena by circles. The light source acts as the centre of the circle. The first and smallest circle is about 30cm radius. Around this circle, we measured the light intensity as 15. So when our robot sense light intensity of a value 15, it will walk toward the light source for a random amount of time between 1 to 7 seconds. Then, it will stop and prints out "LIGHT FOUND" on the terminal. We programmed the robot to stop when it gets into the smallest circle. The distance of the robot from the light source differs according to the time that it takes to walk after it detected a value of 15. Once the robot is triggered to stop by finding the light source it does not stop permanently, it is constantly checking if the value is still above the threshold. If the trigger was erroneous then the robot will be able to deal with the anomaly. The light values are hard coded, if we had more time a feature we would have implemented would be to allow the robot to vary the threshold depending on the situation because something we observed was different light levels in the room on different days would effect how it behaved.

Unit testing: We performed one unit testing which is mainly to test whether the robot stops when it is put at a certain distance from the alarm beacon. The results is shown in the table below. For this unit testing, we disabled all the other sensors and only uses the color sensor to detect lights. The robot just walk straight from where it was placed or stops when the alarm beacon is found. In the table below, the color sensor reading is recorded to prove that the robot stops when the value of the light intensity detected by the color sensor is equal or more than 15.

Distance from the alarm beacon, cm	Predicted Result	Color Sensor Reading
60	Keep moving	9
40	Keep moving	12
35	Keep moving	13
25	Stop	15
15	Stop	17
10	Stop	19

Additional Insights: If we were given time, we would get the current reading of intensity of the light source right before the competition. The value that we used before this were from a different day and on the competition day itself, the light intensity were different. We estimated the light intensity for stopping on the day but like many other teams, we got it wrong. Therefore, if given the time, we would like to get the current reading in front of the alarm beacon to get the accurate light intensity reading to be used as the threshold.

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 beaoning 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 - We positioned the light sensor at the front of the robot, surrounded by long pieces of LEGO that blocked out any ambient light that could interfere with the light source it is trying to detect. This minimises erroneous light sensor readings from the robot's surroundings, therefore providing precise information required to guide the robot towards the light.

Software - Initially we set the light sensor thresholds too high, which caused the robot to endlessly wander around the test arena without reacting to any light gradient. After some extensive trial and error, we tuned our light sensor readings enough for the robot to accurately react to them. During it's rotating search for light, if a reading above the threshold is detected, it will walk (whilst avoiding obstacles) in a straight line from the angle it was found at for a period of time, before searching again. We essentially implemented the beaoning behaviour by reactive control, with an uncrossed negative connection that would lead to approach and stop behaviour.

We also tried implementing a similar behaviour, where the robot would start to walk towards the light once it had observed the peak of the colour sensor value whilst rotating. It would observe the maximum colour sensor value in a full rotation and log the angle it was found at, then proceed to walk in that direction. This proved to be quite a challenge in practice, so we adopted a more simple, yet more effective strategy.

Unit testing: - The following unit tests describe the robot's behaviour when positioned at different angles from the light source. For this unit testing, all of the sensors are used in order to perform a reliable beaconing behaviour. The robot will perform a rotation and proceed to walk from the angle at which it found a colour value above the threshold (15) at an initial distance of 90cm from the light source (positioned to the left of the starting position).

Angle = 0°	Color Sensor Reading	Distance from alarm beacon (cm)
	9	81
	11	77
	16	62
	15	49
	17	30
	18	23

Angle = 90°	Color Sensor Reading	Distance from alarm beacon (cm)
	7	87
	10	77
	11	72
	11	63
	13	55
	13	41

Angle = 180°	Color Sensor Reading	Distance from alarm beacon (cm)
	9	80
	13	75
	15	63
	16	48
	17	32
	19	24

Angle = 270°	Color Sensor Reading	Distance from alarm beacon (cm)
	11	81
	14	64
	16	50
	18	39
	20	27
	22	18

The unit tests show how when initially pointed towards the light source, the robot is able to walk towards it proficiently - indicating that the higher the light value, the closer it is to the light source. This hypothesis is complemented with the results from starting the robot at 90° (away from the beacon), showing that it took longer

to reach the light source when less light was detected. The other starting positions displayed the middle ground in terms of (distance to light / light intensity) and had similar results due to the same starting angle displacements.

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)

Our subsumption architecture mainly revolves around the value the robot receives from the light sensor. At the lowest level of the hierarchy we have the obstacle avoidance module, which is triggered when the value from one of the ultrasonic sensors gets too small and the robot tries to correct itself by steering away from the obstacle. It uses the same closed loop control and PID system from Assignment 1 that was proven to be quite proficient at avoiding obstacles.

Above this, we have our walk module which adopts a modified Levi Flight algorithm. The robot would perform a random turn but only if it hadn't seen light during its rotation. If it does detect light then it would go towards it; which leads onto the next module in the hierarchy - beaconing (or search). Initially it will start with a search for light by rotating either left or right, and if during this rotation it detects a light sensor value higher than the threshold, it will proceed to walk in a straight line from the angle it found the value at. If it does not detect a threshold during the full rotation, it will pick a random angle to walk in. With these three layers, the robots behaviour essentially boils down to [search / walk / search / walk / light seen / walk / light seen / light seen ...] - all whilst avoiding obstacles.

Finally, we need to be able to stop the robot when it reaches the light source, which brings us to the stop module; the final layer to our hierarchy. This is only ever used to kill the motors if the light sensor value is above a certain threshold. This threshold was calculated by taking averages around a radius of 30cm from the light source, which turned out to be 22. Once the stop behaviour is triggered by this value it does not stop permanently, as it will be checking if the value is above the threshold continuously - this avoids any erroneous readings that could occur during the run (e.g. reflections or light through the window) and will stop completely if a constant light value above the threshold is detected (i.e. in front of the light source).

We believe our approach and implementation towards the subsumption architecture was well thought out and proved to be quite effective. A few changes we made along the way were mainly involving our search strategy, as some methods proved to be difficult in practice and did not interact well enough with the other hierarchies.

Unit testing: We carried out one unit testing to make sure that all of the behaviours work well together:

Scenario	Predicted Result	Final Result
Robot gets too close with an obstacle	Robot gets too close with an obstacle	Robot turned left and right and successfully avoided the obstacles
Robot in the middle of nowhere	Walks a bit and turns randomly then continue walking again	the robot rotated randomly and walked for a bit then rotated randomly again until it detects a light
Robot detects light while rotating	Moves straight at the angle where it detected the light from	after a full rotation, the robot rotates back to the angle where it detected the light and then walked straight at that angle
Robot in front of the light	Robot stops	The robot stopped and "LIGHT FOUND!!!" was printed on the terminal
When the robot stopped in front of the light, move it to an angle not facing the light	Continue moving (searching)	The robot start moving again, it turned and walked randomly to search for the light

The unit tests show that the robot proved to be quite effective during practice, with our predicted results matching the final results. However, in preparation for the competition we must ensure that the light sensor thresholds are tuned accordingly for the day so that it performs as well as it did during practice.

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 mins 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
0	10 or more?	2