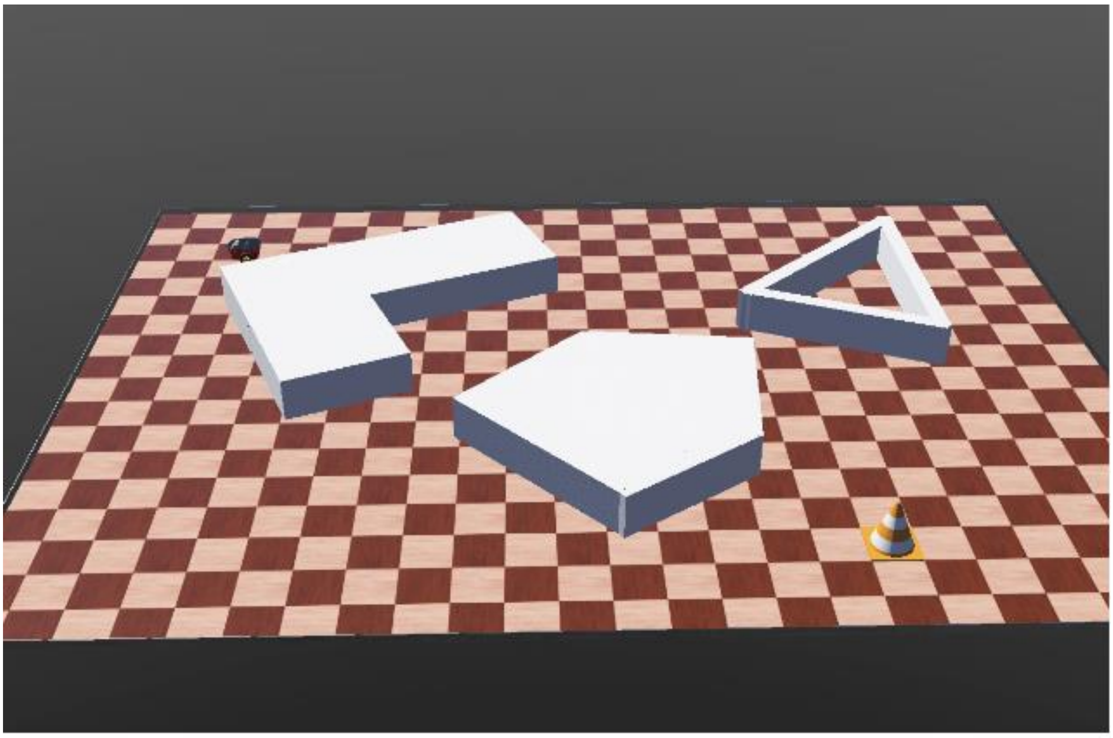
The deadline for the assignment is **1th January at 10.00 (note the time)**. If you believe you may experience problems with this deadline, then please discuss with me asap.

Note that this is an individual assignment. A typical student should aim to spend approximately 24 hours on this assignment in order to achieve the equivalent of a high B or low A grade (i.e. 65-75%). Ideally, you should aim to spend no more than 40 hours in total. Full details of the rubric used to mark the assignment are given below.

The aim of the programming assignment is **to implement an obstacle avoidance algorithm** to allow a robot to navigate from a start location, and to reach an end location (as represented by a traffic cone in the arena below). To do this, you will need to implement a robot controller that knows of a destination location, and can determine the appropriate direction to move to reach its destination, avoiding any obstacles in its way. You should not make any assumptions about the location or configuration of the obstacles in the code (this will result in a loss of marks). You should also track the distance your robot has travelled in reaching its location, and write up a short report on the approach taken. You can reuse any of the code released in the lab sessions, but extra marks will be gained for improving on this code to get better performance.



Detailed Task Description

The following are guidelines for your implementation. More information is given in the rubric below - please look at this carefully as part of your design so that you can decide how much you want to implement (based on the marks available). A total of 30 marks are available for the approach taken and work done on your solution. Up to 10 additional marks will be awarded on your style and quality of code, and up to 10 marks will be awarded for the report - **again see the associated rubric below**. You will need to create a Webots project, but can download the environment file for the environment above Download but can download the environment file for the environment above. It assumes that you are using **Webots W2022b**, but you can also use it with other versions. **If you have problems with another Webots version then let me know and I'll convert it for you**.

- **Your solution should be implemented in Java, using standard libraries (inc numpy).** You can make use of any of the Webots api, and reuse the code from the COMP329 labs, but please do not use other 3rd party code or libraries. If you are unsure about the acceptability of a library, please consult me first, and document these in the report. If this is not done, then a small number of marks may be lost.
- **Your solution should be tested on a 12x9 environment (pictured above)** downloaded from here Download downloaded from here, with a starting pose of (-4.5, 3, 0) and end at the location of the traffic cone (with an approximate location of (3.0,3.5) - there is no stipulation on the orientation). You can use any environment you like during development, but your solution will be assessed with the downloaded environment.
- **You should ensure your robot tracks its location by using the supervisor mode**, using the approach detailed in Lab 3.
- **You should report on your robots progress (and telemetry) as it navigates around each obstacle**. The specific data to be reported is not prescribed here, but ideally it should allow the user to follow what the robot is doing and why it makes the decisions it does. More marks will be awarded if this data is presented on a display using text or (better) graphically (see how to do this in Lab 4). By telemetry, I refer to at least information relating to the robot location, wheel speed, and current status.
- **Track the distance travelled by the robot and report this in your report**. You may also want to include the distance travelled in the data you report on, in addition to the telemetry.
- **You can use any obstacle avoidance algorithm you like**, but most of the marks will be awarded for implementing Bug2 or Bug 1 (with slightly more marks awarded for Bug 2). Extra marks can be obtained for comparing both of these approaches - see the rubric below.
- **You can make use of the PID and wall following algorithm** from Lab 4. Note that the code implements a basic solution, and there is scope for improvement, so additional marks will be awarded for improvements.
- **You should terminate when you reach the destination location** (i.e. where the traffic code is, which is approximately at (3.0,3.5)). Ideally, your robot should terminate because it realises that the goal location is not accessible (as it is occluded by the

traffic cone, acting as an obstacle), but you can chose to remove the traffic cone from the environment and end at this location.

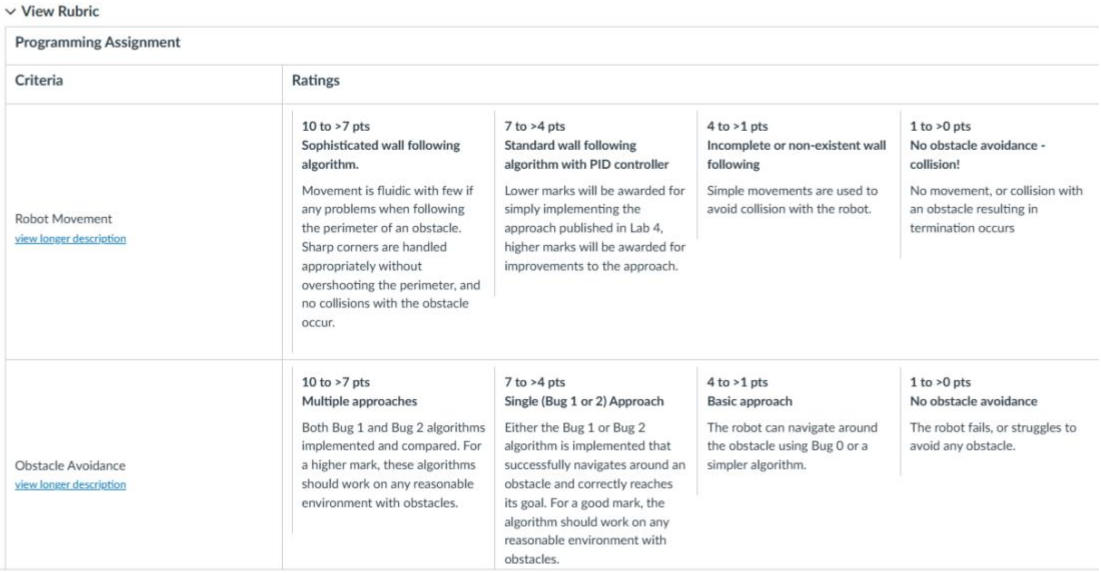
Assignment Report (worth 20% or up to 10 marks)

You should write a short (500-1000 word) discussion on the obstable avoidance strategy you have implemented, as well as a reflection on the approach taken and any challenges faced. In your report, ensure you include your name, the implementation language used, any non-standard libraries (check with me first), and a summary of the distance your robot travels from the start to the end location. If you test your solution on other environments, then include a screen shot of the successful solution.

Submission Instructions

Your solution should be submitted ELECTRONICALLY through canvas and should include all of the files for the Webots project (not just the code - i.e. java or python files), as a single zip file. Remember to include your student ID in your submission. If you experience any problems with submitting your assignment, then please email me (trp@liv.ac.uk) as soon as possible informing me of the problem, and include the latest version of your submission (as a zip file). If more than one submission is made, the latest submission will be marked.

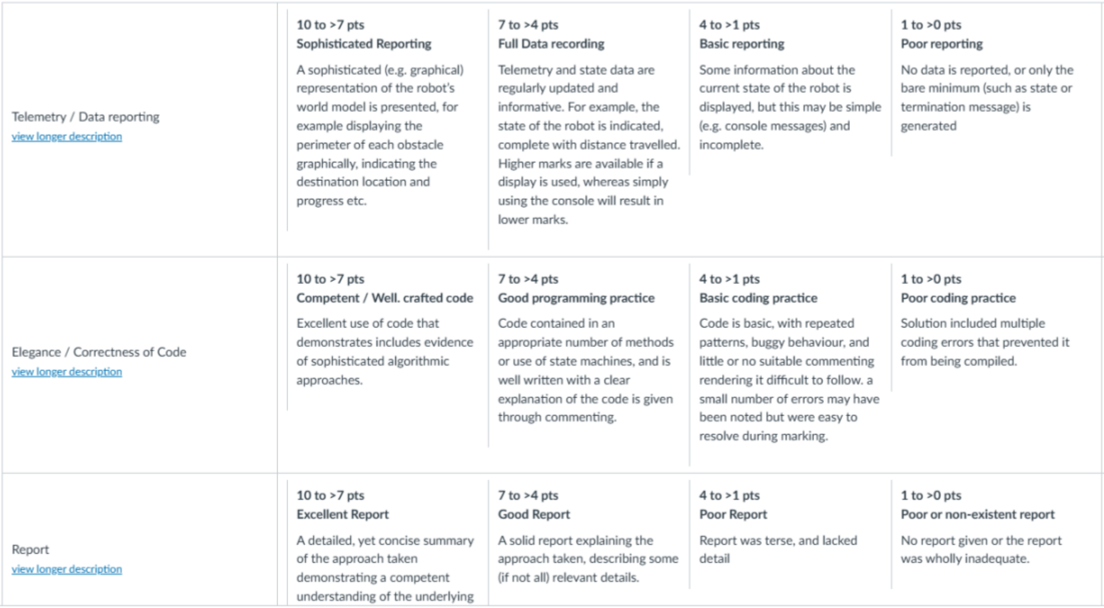
评分标准



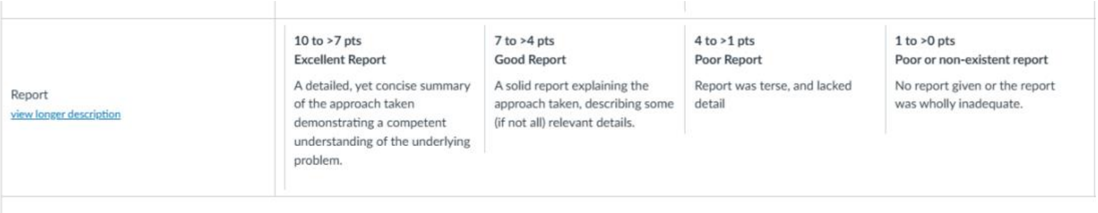
View Rubric

Programming Assignment

Criteria	Ratings			
Robot Movement view longer description	10 to >7 pts **Sophisticated wall following algorithm.** Movement is fluidic with few if any problems when following the perimeter of an obstacle. Sharp corners are handled appropriately without overshooting the perimeter, and no collisions with the obstacle occur.	7 to >4 pts **Standard wall following algorithm with PID controller** Lower marks will be awarded for simply implementing the approach published in Lab 4, higher marks will be awarded for improvements to the approach.	4 to >1 pts **Incomplete or non-existent wall following** Simple movements are used to avoid collision with the robot.	1 to >0 pts **No obstacle avoidance - collision!** No movement, or collision with an obstacle resulting in termination occurs
Obstacle Avoidance view longer description	10 to >7 pts **Multiple approaches** Both Bug 1 and Bug 2 algorithms implemented and compared. For a higher mark, these algorithms should work on any reasonable environment with obstacles.	7 to >4 pts **Single (Bug 1 or 2) Approach** Either the Bug 1 or Bug 2 algorithm is implemented that successfully navigates around an obstacle and correctly reaches its goal. For a good mark, the algorithm should work on any reasonable environment with obstacles.	4 to >1 pts **Basic approach** The robot can navigate around the obstacle using Bug 0 or a simpler algorithm.	1 to >0 pts **No obstacle avoidance** The robot fails, or struggles to avoid any obstacle.



Telemetry / Data reporting view longer description	**10 to >7 pts** **Sophisticated Reporting** A sophisticated (e.g. graphical) representation of the robot's world model is presented, for example displaying the perimeter of each obstacle graphically, indicating the destination location and progress etc.	**7 to >4 pts** **Full Data recording** Telemetry and state data are regularly updated and informative. For example, the state of the robot is indicated, complete with distance travelled. Higher marks are available if a display is used, whereas simply using the console will result in lower marks.	**4 to >1 pts** **Basic reporting** Some information about the current state of the robot is displayed, but this may be simple (e.g. console messages) and incomplete.	**1 to >0 pts** **Poor reporting** No data is reported, or only the bare minimum (such as state or termination message) is generated
Elegance / Correctness of Code view longer description	**10 to >7 pts** **Competent / Well. crafted code** Excellent use of code that demonstrates includes evidence of sophisticated algorithmic approaches.	**7 to >4 pts** **Good programming practice** Code contained in an appropriate number of methods or use of state machines, and is well written with a clear explanation of the code is given through commenting.	**4 to >1 pts** **Basic coding practice** Code is basic, with repeated patterns, buggy behaviour, and little or no suitable commenting rendering it difficult to follow. a small number of errors may have been noted but were easy to resolve during marking.	**1 to >0 pts** **Poor coding practice** Solution included multiple coding errors that prevented it from being compiled.
Report view longer description	**10 to >7 pts** **Excellent Report** A detailed, yet concise summary of the approach taken demonstrating a competent understanding of the underlying	**7 to >4 pts** **Good Report** A solid report explaining the approach taken, describing some (if not all) relevant details.	**4 to >1 pts** **Poor Report** Report was terse, and lacked detail	**1 to >0 pts** **Poor or non-existent report** No report given or the report was wholly inadequate.



Report view longer description	**10 to >7 pts** **Excellent Report** A detailed, yet concise summary of the approach taken demonstrating a competent understanding of the underlying problem.	**7 to >4 pts** **Good Report** A solid report explaining the approach taken, describing some (if not all) relevant details.	**4 to >1 pts** **Poor Report** Report was terse, and lacked detail	**1 to >0 pts** **Poor or non-existent report** No report given or the report was wholly inadequate.