University of Sheffield

COM2009-3009 Robotics

Lab Assignment #2

'Team Robotics Challenge'

Lab Instructions

Dr. Tom Howard

t.howard@sheffield.ac.uk

Department of Computer Science

May 13, 2021

Version 3

Revision Notes

Version 1:

- Initial release.

Version 2:

- Adjusted the maximum exploration time available for Task 1 (see Section 4.1).

Version 3:

- Maximum time limits for Tasks 3 and 4 extended.
- Added Section 4.5 and Section 5 (previously a placeholder).
- Any changes to existing content are highlighted in magenta.

Contents

1 Overview (please read carefully)

In this lab assignment you will put into practice everything that you have learnt about ROS during Lab Assignment #1 and develop a ROS package to make a TurtleBot3 Waffle complete a *'Navigation, Search and Beaconing'* task in a simulated environment. You will do this in teams, and each team will compete to develop a robot control package that makes a robot complete this task in the fastest possible time!

Over the next 5 weeks (Spring Semester Weeks 8-12) you will work, in your teams, to develop a ROS package to control a robot to complete this *'Final Challenge'*! Your team's performance in this will form *part* of the assessment for Lab Assignment #2, but you will *also* be assessed on a series of intermediate programming tasks. The aim of these smaller tasks is to help you to break up the overall challenge into a number of discrete robotic behaviours, which you can develop and optimise in isolation, before combining everything to form your team's final ROS package.

Further details on the Lab Assignment #2 tasks and submissions are provided in Section 1.4.

This assignment is worth 30% of the overall mark for the COM2009-3009 course.

1.1 Lab Sessions

There won't be any taught content or weekly exercises for you to work through for this Lab Assignment. You will need to work more autonomously here and you should use the scheduled lab time to work in your teams on the Lab Assignment #2 tasks, as described in this document.

Lab sessions will still be held *most* weeks for Lab Assignment #2, but we will keep you updated on the plans week-by-week. As before, these will take place on-line via Blackboard Collaborate. The timetable is the same as it was for Lab Assignment #1, where sessions will take place on Thursdays from 14:00 to 16:00 for Lab Group A and Fridays from 09:00 to 11:00 for Lab Group B.

These sessions will be used to communicate to you any updates on the lab tasks, signpost you to any additional resources and provide you with direct access to the COM2009 Teaching Team (Tom Howard, Alex Lucas and the course GTAs) so that you can ask any questions that you may have as you are working in your teams.

1.2 Getting Further Support

Remember to use the Lab Assignment #1 Wiki as a reference throughout this work: taking the things that we worked on in the first 6 weeks of the COM2009/3009 course and applying them to your team's ROS control package.

Don't forget to use the COM2009 'PRACTICALS' Discussion Board on Blackboard to post any questions you may have regarding Lab Assignment #2 or ROS in general.

Over the next five weeks there will also be weekly appointment slots available for you to book in (as a team) for a chat with Tom Howard via video call to discuss anything that you may be struggling with.

1.3 The Teams

For this lab assignment you will work in teams of three (or - in some cases - four), which have already been assigned by the teaching team. The *'Team List'* is available in the Lab Assignment #2 area on Blackboard, so please refer to this to find out which team you have been assigned to and who your fellow team members are.

This assignment is not only designed to give you experience of programming robots, but also gives you the chance to develop and demonstrate essential *team working* and *project management skills*: planning and scheduling, assigning roles and responsibilities, delegating tasks, distributing workload appropriately and communicating effectively within your teams to deliver this assignment to the best of your abilities. It is up to you to decide on the best way to communicate effectively within your teams, but you should try to communicate regularly (both during and outside of the scheduled lab sessions) and you could use on-line meeting tools such as *Google Meet* to do this (for example). We have also created individual *'Team Workspaces'* on Blackboard, providing each team with a range of collaboration tools as well as a dedicated Blackboard Collaborate space for you to use. You should have already been assigned to the correct workspace, and should be able to access this from the Lab Assignment #2 area on the COM2009 Blackboard Course Page.

Note: *Please let a member of the teaching team know if you are unable to access your team's workspace, if you have been assigned to the wrong one, or if you haven't been assigned to a team!*

In addition to this, you are encouraged to use *GitHub* to share work and collaborate within your teams. As a University of Sheffield student, you can apply for the GitHub Student Developer Pack, which gives you access to a range of developer tools including *GitHub Pro*. GitHub Pro allows you to have unlimited collaborators on your repositories, which might help you to collaborate on your ROS package.

1.4 The Assignment

This assignment will require you to implement all the necessary behaviours to make a robot complete a *Navigation, Search and Beaconing Challenge* (more details on this in Section 2). As discussed above, we have split this assignment into a series of individual parts to help you break down the problem into a number of discrete behavioural stages and focus on things one at a time. Section 4 of this instruction document guides you through these individual development stages step-by-step and there is a **Task** associated with each. You will need to submit work to demonstrate your completion of each of these tasks *in addition to* the ROS package that you must submit for the final challenge (discussed in Section 5).

Each task is marked and goes towards your final grade for Lab Assignment #2. Further details and submission deadlines for all the Lab Assignment #2 Tasks are summarised in Table 1 below (click on the task number to take you to the section of this document where that particular task is defined).

Note: *You should work on each task in your teams, and you only need to make one submission per group to Blackboard for each task.*

As shown in Table 1, there are **100 marks** available in total for Lab Assignment #2.

Task	Details	Available Marks	Deadline
1	A working ROS package and basic obstacle avoidance controller	25/100	Wednesday 5th May
2	Developing a *Detection* behaviour	10/100	Monday 17th May
3	Developing a *Search & Beaconing* behaviour	15/100	
4	Developing a *Maze Navigation* behaviour	15/100	
5	Your *Final Challenge* controller	35/100	Friday 21st May

Table 1: Details of the Lab Assignment #2 tasks and submission deadlines.

At the end of this assignment each team will also be asked to submit a *'Peer Assessment Form,'* giving you the opportunity to indicate the level of contribution each team member has made to the Lab Assignment #2 work overall. This will then be used to distribute your team's final Lab Assignment #2 marks fairly amongst each of your team members.

Note: *It is up to you as a team to agree on this together and provide figures that accurately reflect the level of contribution that each team member has made.*

1.5 Submission Details

As shown in the table above, there are **five tasks** in total, delivered via **three submission deadlines** over the next five weeks. Each submission will require you to submit a ROS package (as a `.tar` file) to a submission portal on Blackboard. Further details and guidance on each submission is provided in the relevant section of this document as well as the associated submission portal on Blackboard.

As a team, you should create a single ROS package at the very start of this assignment (as detailed in Section 3.2.2), adding all the necessary functionality for each task as you go along. For each submission deadline, you will need to create a copy of your package in its current state by creating a `.tar` archive of it, submitting this to Blackboard by the specified date (the export process is explained in Section 1.5.2 below).

Once again! *You only need to make one submission per team to Blackboard for each task.*

Each of your submissions will be assessed by the COM2009 Teaching Team by downloading and running your package in the standard WSL-ROS environment that you have been using throughout the COM2009 labs so far, using one of the University Computers in Virtual Classroom 1.

In order to launch the necessary functionality within your package to satisfy a particular task you will need to include launch files, following naming conventions that are specified within this document for each task. This will allow *you* to ensure that all of the required functionality is executed when your submission is assessed, and also ensures that *we* know exactly how to launch this functionality in the WSL-ROS environment.

Note: *It is up to you to ensure that your functionality can be launched in the way that we specify and that all your functionality launches as intended to satisfy a given task. If it doesn't, then you will be awarded zero marks, so* ***make sure you test everything out in the WSL-ROS environment prior to submission!***

1.5.1 Submission Requirements

In order to be awarded *any* marks for *any of the tasks* outlined in Table 1 you must ensure that the following requirements are met with regards to the ROS package that you submit (as well as any additional requirements described in the later sections of this document):

1. Your package must be submitted to Blackboard as a `.tar` file with the following naming convention:

   `team{}.tar`

   Where the `{}` must be replaced with your own team number. See Section 1.5.2 below for more details on how to create a `.tar` archive of your package.

2. Your ROS package must work 'out-of-the-box': the Teaching Team will not make any modifications or fix any errors for you.

1.5.2 Exporting your ROS package for Submission

It is important that you follow these steps carefully in order to create an archive of your ROS package that is appropriate for submission, when required:

1. First, navigate to the `catkin_ws/src` directory in a WSL-ROS terminal instance:

   ```
   $ cd ~/catkin_ws/src/
   ```

2. Then, use the `tar` command to create an archive of your package:

   ```
   $ tar -cvf /mnt/u/wsl-ros/team{}.tar team{}
   ```

   (Again, replacing `{}` with your own team number). This will create the `.tar` archive in the `U:` Drive of the *Windows* filesystem.

3. In *Windows*, open up Windows Explorer, click "This PC" in the left-hand toolbar and locate your own personal `U:` Drive in the "Network locations" area.

4. In here there should be a `wsl-ros` folder, which should contain the `team{}.tar` file that you have just created.

5. Submit this `.tar` file, as it is, to Blackboard via the appropriate submission portal.

2 The Navigation, Search and Beaconing Challenge

The *Final Challenge* for Lab Assignment #2 is to develop a ROS package which allows a TurtleBot3 Waffle to complete a Navigation, Search and Beaconing Task. There are a number of stages to this, summarised as follows:

1. The robot will need to *navigate a maze* as quickly as possible without touching the maze walls.
2. After this, the robot will reach a *'Search Area'*: a more open space containing a number of different coloured objects (i.e. *'beacons'*). It will need to *search* this area for the single beacon within the environment that is of a particular *'Target Colour'* (each beacon within search area will have a different colour).
3. The robot will begin the challenge (at the start of the maze) in a *'Start Zone'*, three examples of which are illustrated in Figure 1 below. The colour of the Start Zone indicates the Target Colour for the challenge, i.e.: the colour of the beacon that the robot must find in the Search Area. So, it will need to detect this colour, remember it and then search for the beacon of the same colour once it gets to the Search Area.
4. Once found, the robot will need to move towards the target beacon (i.e. *'beaconing'*) and stop within a *'Stop Zone'* printed on the floor surrounding it. The robot must stop in the Stop Zone without touching the beacon!

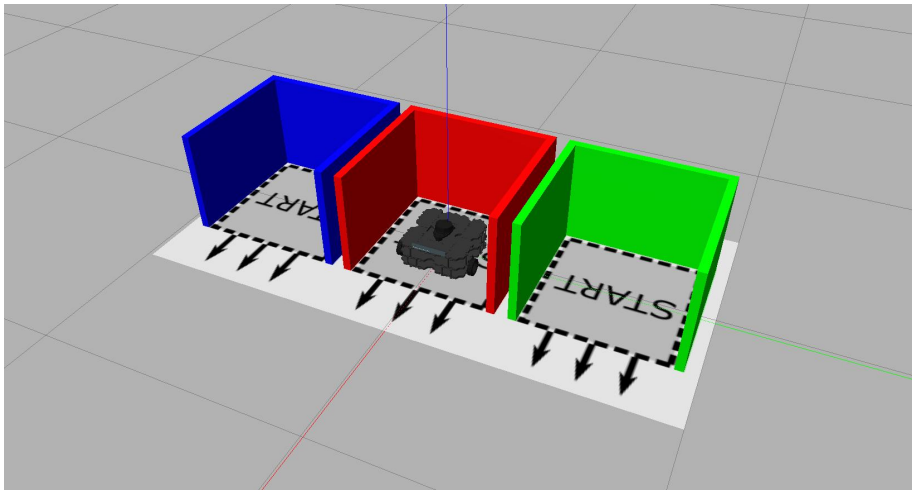

Figure 1: Examples of *'Start Zones'* rendered in `Blue`, `Red` and `Green`.

Section 4 guides you through the development of the various behaviours that the robot will require in order to solve this challenge, and also outlines all the intermediate tasks that you must complete and submit in addition to your final ROS control package. Section 5 discusses the Final Challenge and associated marking in more detail.

You should refer to the work that you have done during Lab Assignment #1 to help you with this team-based ROS programming assignment, but you may also need to do your own independent research, in your teams, to explore ROS further and leverage more advanced features of the framework. We encourage you to think creatively when approaching this assignment: *try to think beyond what is provided in this document to come up with novel solutions that may enhance your robot's performance!*

2.1 The Final Challenge Arena

The exact configuration of the *Final Challenge Arena* (i.e. the format of the maze; the location, shape and colour of the beacons in the search area; the colour of the target beacon) will not be revealed until after the final submission deadline. You will therefore need to develop your robot control package robustly to accommodate an unknown environment. Figure 2 illustrates what this arena *might* look like.

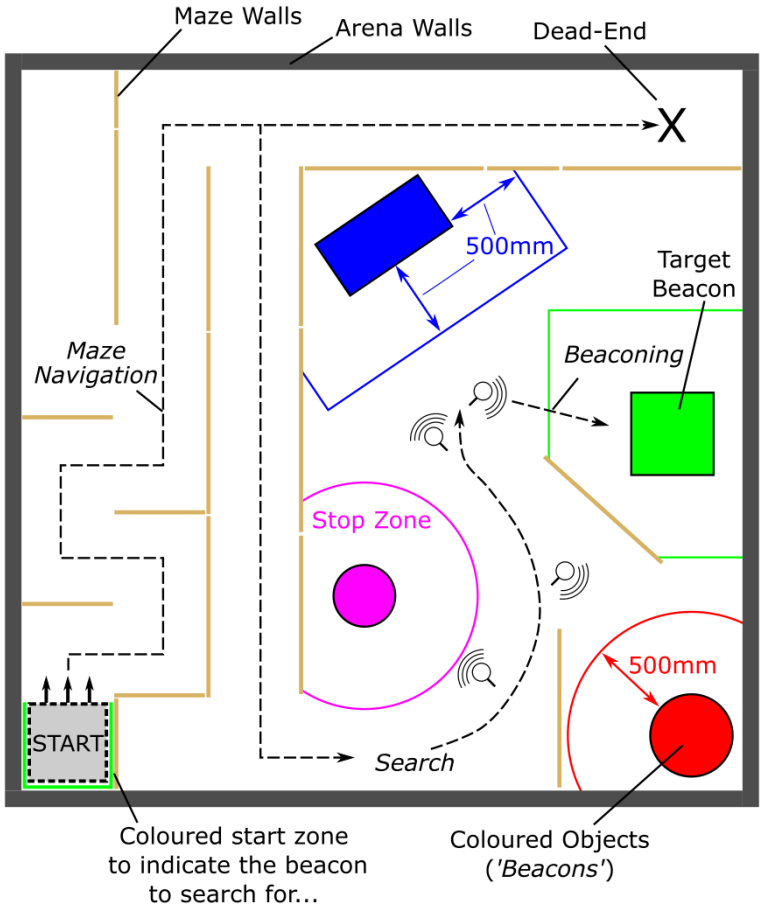

Figure 2: An example configuration for the *Final Challenge Arena* to illustrate the nature of the task.

To support you with this assignment we have also put together a ROS package containing a range of simulation resources. A *final challenge example arena* will be included as part of this (discussed in Section 5, once available). You should use the resources within this package to develop and test out your robot control package as you are working through Lab Assignment #2. See Section 3.1 below for further details on how to download and install this in your WSL-ROS environment.

Be aware that the Final Challenge Arena *will be different* to any of the example simulations that we provide you with! It *will* however have the following design features (this also applies to any alternative arenas that your robot may need to perform any of the intermediate tasks in):

1. Arenas will be square, 5x5m in size (unless stated otherwise for a particular task) and will be bound by grey arena walls of 300mm height.

2. Internal arena walls will be 250mm high, 30mm thick, of various lengths and have a 'wooden' appearance and colour.

3. Internal walls will be constructed so as to ensure there is always enough space for a TurtleBot3 Waffle robot to pass through any apertures or corridors.

4. Mazes may be comprised of more complex twists and turns than those illustrated in the example in Figure 2, so not all corners will necessarily be right-angled.

5. Beaconing objects will be modelled as *boxes* or *cylinders*. In the X-Y plane (the plane of the arena floor) box geometries may be square or rectangular, having a maximum dimension of 800mm in the X or Y axis. Cylindrical beacons will always be circular in the X-Y plane, having a maximum radius of 300mm. All beacons will be between 200mm and 400mm in height.

6. The 'stop zone' surrounding a target beacon will be 500mm greater than the beacon's dimension in the X or Y axis.

7. There are a total of **six** possible beacon colours that may be used in the simulations for the beaconing and search tasks, so you only need to develop search algorithms to accommodate these. The colours are listed in Table 2 below, and there is also a simulated environment in the `com2009_simulations` package called `beacon_colours` to illustrate these too (see Section 3.1 for details on how to download and install this package in the WSL-ROS environment).

Colour Name	**Example**
`Red`	
`Yellow`	
`Green`	
`Turquoise`	
`Blue`	
`Purple`	

Table 2: The range of possible colours that could be applied to beacon objects for any beaconing and search tasks.

3 Getting Started

3.1 Installing the Development Resources

A ROS package called `com2009_simulations` has been put together to help you with this assignment. This package contains a range of simulation resources to help you work through the tasks and prepare your robot for the final challenge.

This package must be downloaded and installed in your WSL-ROS environment. Follow the steps below to do this:

1. First, in a WSL-ROS terminal instance, make sure that you are located in the `src` directory of your *Catkin Workspace*:

   ```
   $ cd ~/catkin_ws/src/
   ```

2. Then, clone the `com2009_simulations` package repository from GitHub:

   ```
   $ git clone https://github.com/tom-howard/com2009_simulations.git
   ```

3. Check that there are no errors in this, or any other packages in your Catkin Workspace, by running `catkin_make` from the root of the `catkin_ws` directory:

   ```
   $ cd ~/catkin_ws/ && catkin_make
   ```

4. If this completes without any errors then you should now be able to launch the `beacon_colours` arena that was referred to above:

   ```
   $ roslaunch com2009_simulations beacon_colours.launch
   ```

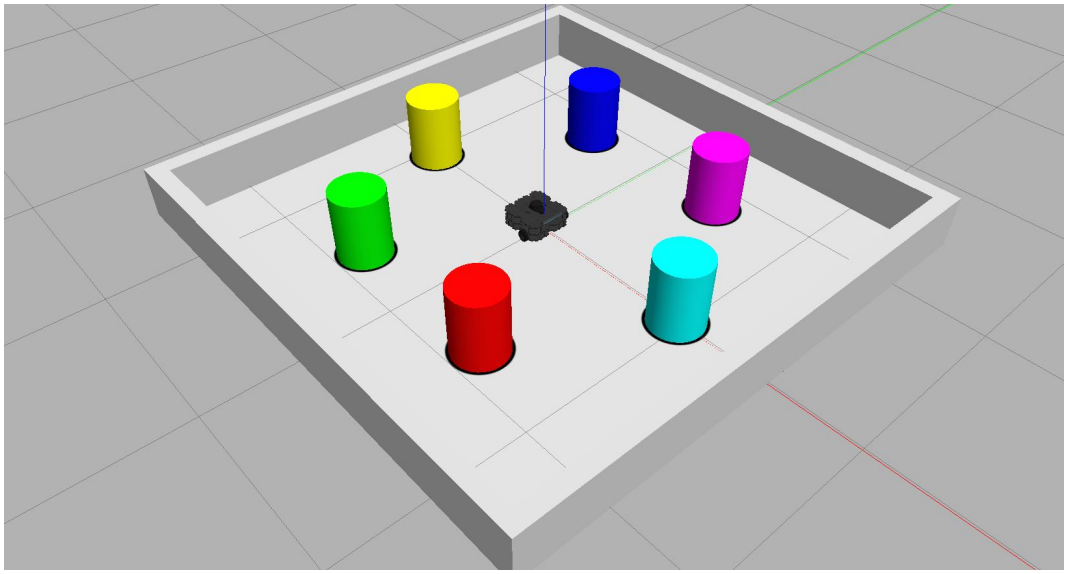

Figure 3: The `beacon_colours` arena, illustrating all possible colours that could be used for beaconing and search tasks.

3.2 Your Team's ROS Package

As discussed earlier, everything that your team submit for this lab assignment must be contained within a single ROS package. Within this package you will develop all the necessary nodes that will allow your robot to complete each task (including the final

challenge). Each task will be assessed by the COM2009 Teaching Team by launching the required functionality from within your package that you specify in a *launch file*. As such, you will need to create one launch file per task.

Note: *It is important that you follow the naming conventions specified within this document when defining your package and creating your launch files.* ***If you don't then you will receive no marks!***

Create your team's Lab Assignment #2 ROS package now by following the steps below. Only one team member needs to do this and, once created, we recommend that you upload it to GitHub and sharing it with the rest of your team.

Note: *If you do plan to use GitHub, then you will each need to set up your individual* `git` *configurations in the WSL-ROS environment before you go any further...*

3.2.1 Configuring `git` in the WSL-ROS Environment

`git` will automatically set your name and email address using the WSL-ROS username and the hostname of the remote machine that you are working on. It is important that this is changed to match your own identity!

1. From a WSL-ROS terminal instance located in your home directory run the following command to edit the `git` configuration file:

   ```
   $ git config --global --edit
   ```

   This will open up the `.gitconfig` file in the `nano` text editor. By default, this file should have the following content:

   ```
   # This is Git's per-user configuration file.
   [user]
   # Please adapt and uncomment the following lines:
   #       name = COM2009 Student
   #       email = student@TENxxxxxxxxxxx.shefuniad.shef.ac.uk
   ```

2. Uncomment the `name = ...` and `email = ...` lines by removing the `#` at the start of each of these lines in the file. Then, change the `name` and `email` fields to match your own personal identity!

3. Once done, press `Ctrl+X` to exit `nano`. Before it closes, you will be asked if you want to save the changes that you have made, so enter `Y` to confirm this and then press `Enter` to confirm that you want to keep the same `/home/student/.gitconfig` file name and overwrite the original content.

4. Finally, don't forget to run `rosbackup.sh` to save these changes to your external rosbackup file, so that they will always be restored whenever you run `rosrestore.sh` in a fresh WSL-ROS instance.

3.2.2 Creating the Lab Assignment #2 ROS package

Remember: *Only one team member actually needs to do this, but you will then need to share it with the rest of your team members via GitHub.*

1. In a WSL-ROS terminal instance, navigate to the `catkin_ws/src` directory of the Linux filesystem:

   ```
   $ cd ~/catkin_ws/src/
   ```

2. Use the `catkin_create_pkg` tool to create a new ROS package in exactly the same way as you did multiple times during Lab Assignment #1:

   ```
   $ catkin_create_pkg team{} rospy
   ```

   Replace the `{}` in `team{}` with *your* team number!

3. Then navigate into the package directory that should have just been created:

   ```
   $ cd team{}/
   ```

4. This should already contain a `src` folder for you to populate with all your Python ROS nodes. Create a `launch` folder in here too, which you will use to store all your launch files:

   ```
   $ mkdir launch
   ```

5. If you *do* intend to collaborate on this in your team using GitHub, then follow the steps here to initialise this as a local repository and push this to GitHub for collaboration. Follow the steps in the link for a **Linux** operating system, but be aware that the WSL-ROS environment uses a slightly older version of `git`, so a couple of the steps need to be adapted as follows:

   (a) In Step 4, use:
   ```
   $ git init
   ```
   not:
   ```
   $ git init -b main
   ```

   (b) In step 9, use:
   ```
   $ git push origin master
   ```
   not:
   ```
   $ git push origin main
   ```

4 Developing your Robot Behaviours

4.1 Part I: Obstacle Avoidance

One of the most important behaviours that your robot will need to exhibit is the ability to move around in an environment autonomously without crashing into things! As your first programming task in this assignment you will need to implement this type of behaviour.

In Week 3 of Lab Assignment #1 you were introduced to the LiDAR sensor on the robot and what the data from this sensor tells us about the distance to any objects that surround the robot in its environment. We talked in Week 5 about how this, in combination with the *ROS Action framework*, could be used as a *feedback signal* to develop a simple obstacle avoidance behaviour that would allow a robot to explore an environment autonomously whilst avoiding any objects that might be present.

This is one approach that you could use to achieve this first task, but there are other (and potentially simpler) ways that this could be achieved too.

Consider Lecture 3, for instance, where you were introduced to *Cybernetic Control Principles* and some of *Braitenberg's "Vehicles"* were discussed and implemented on a Lego robot. In particular, Braitenberg's *Vehicle 3b* might well be relevant to consider as a simple method to achieve the desired behaviour here.

TASK 1 ***(25/100 marks)*** **Due Wednesday 5th May**
Submit a working ROS package containing the ROS node(s) to make a TurtleBot3 Waffle robot explore an environment for 90 seconds without contacting any walls or objects.

4.1.1 Task 1 Simulation

For this task, your robot must autonomously navigate around the `obstacle_avoidance` environment from the `com2009_simulations` package. This simulation can be launched using the following `roslaunch` command:

```
$ roslaunch com2009_simulations obstacle_avoidance.launch
```

You should use this environment to develop your ROS node(s) for this task. Please note though that ***the location of the objects in the environment will be changed when we assess your submission***, so make sure you program your node(s) to accommodate variation here!

4.1.2 Task 1 Details

1. The robot must explore the `obstacle_avoidance` environment for 90 seconds without touching any of the arena walls or the objects within it.
2. If any contact with the environment is made before the 90 seconds has elapsed then the attempt will be stopped.
3. The robot must enter as many of the nine zones within the arena as possible during the attempt.
4. Your robot must maintain a linear velocity of at least $0.1m/s$ throughout the entire duration of the task except for brief periods (of no more than a few seconds) where

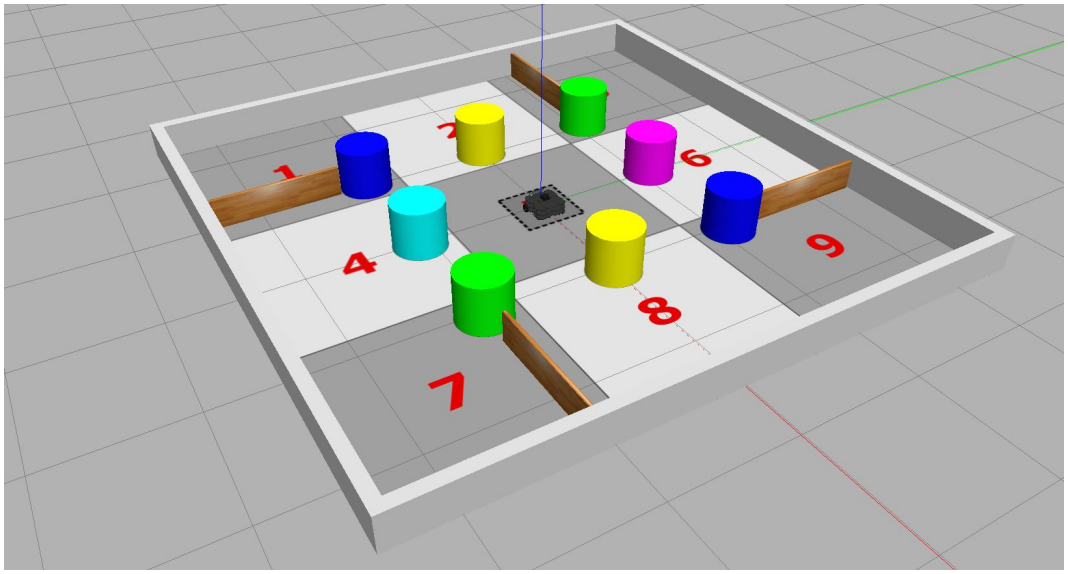

Figure 4: The `obstacle_avoidance` arena for Task 1.

it may stop to turn on the spot, if required.

5. The ROS package that you submit must contain a launch file called `task1.launch`, which will be used by the teaching team to launch the node(s) from within your team's package that you have programmed to make your robot complete this task. This functionality must be launch-able via the command:

   ```
   $ roslaunch team{} task1.launch
   ```

   (The `obstacle_avoidance` environment will already be running before the teaching team attempt to execute your launch file).

4.1.3 Task 1 Marking

There are 25 marks available for this task in total, awarded based on the criteria outlined in Table 3.

The aim of this task (in part) is for you to get to grips with submitting your ROS package correctly for the further task submissions in Lab Assignment #2. This task will be assessed by the COM2009 Teaching Team during the Week 10 lab sessions where you will be asked (as a team) to attend a video meeting with one of the Teaching Team whilst they run and assess your submission. If things don't work, then you can use this as an opportunity to observe what went wrong and find out how this could be resolved for the further Lab Assignment #2 task submissions.

If you *do* receive zero marks for this task on the first attempt then your team will be offered an opportunity to resubmit this before the end of Week 12, giving you a second chance to get things right and obtain some marks here. Any resubmissions will not be eligible for any marks for Criterion A on the second attempt though, so only 15/25 marks will be available in total for Task 1 in such cases.

Note: *We will be offering a resubmission opportunity* ***for this task only****. Resubmission of any further tasks will not be possible.*

<table>
<tr><th colspan="2">Criteria</th><th>Marks</th><th>Details</th></tr>
<tr><td>A</td><td>A working ROS package</td><td>10/25</td><td>You will be awarded full marks here if you submit a ROS package containing a launch file as specified in Section 4.1.2 and that you submit your package according to the submission requirements outlined in Section 1.5.1. No partial credit will be awarded here, and if any of these requirements are not met then you will be awarded zero marks for this entire task!</td></tr>
<tr><td>B</td><td>Run time</td><td>6/25</td><td>You will be awarded marks for the amount of time that your robot spends exploring the environment before 90 seconds has elapsed or the robot makes contact with anything in its environment:
<table>
<tr><th>Time (Seconds)</th><th>Marks</th></tr>
<tr><td>0-9</td><td>0</td></tr>
<tr><td>10-19</td><td>1</td></tr>
<tr><td>20-29</td><td>2</td></tr>
<tr><td>30-39</td><td>3</td></tr>
<tr><td>40-49</td><td>4</td></tr>
<tr><td>50-59</td><td>5</td></tr>
<tr><td>60+</td><td>6</td></tr>
</table></td></tr>
<tr><td>C</td><td>Exploration</td><td>9/25</td><td>You will be awarded 1 mark for every zone of the arena that your robot manages to enter. The robot only needs to enter each zone once, but its full body must be within the zone marking to be awarded the mark.</td></tr>
</table>

Table 3: Marking Criteria for Task 1.

4.2 Part II: Object Detection

Your robot will need to be able to recognise the colour of the start zone that it is launched within when it begins the Final Challenge. The start zone colour denotes the *Target Colour* for the *Search & Beaconing* part of the challenge, and the robot will be able to observe this as soon as it is launched into its simulated world by analysing the walls that it is surrounded by to the left, right and behind. Once determined, the robot then needs to search for a beacon of the same colour within its environment. We did something similar to this in Week 6 of Lab Assignment #1: using `OpenCV` to analyse the images published (as ROS messages) to the `/camera/rgb/image_raw/` topic on the ROS network by our robot. Use the work we did in Exercises 2 and 3 here as a starting point for achieving the desired behaviour for Task 2.

TASK 2 ***(10/100 marks)*** **Due Monday 17th May**
Develop the ROS node(s) that allow a TurtleBot3 Waffle robot to detect an object in its environment of a particular *target colour*.

4.2.1 Task 2 Simulation

For this task your robot must be able to search within an environment called `detection`, which contains four uniquely coloured pillars. This simulation can be launched in the WSL-ROS environment using the following `roslaunch` command:

```
$ roslaunch com2009_simulations detection.launch
```

The arena here is slightly smaller than the others within the `com2009_simulations` package, measuring 3x3m only. The arena contains a *Start Zone*, which your robot will be located within when the simulation is launched (using the above `roslaunch` command). The colour of the start zone indicates the *Target Colour* for the detection task: the robot must take note of the colour of this starting zone and identify the pillar of the same colour within the arena.

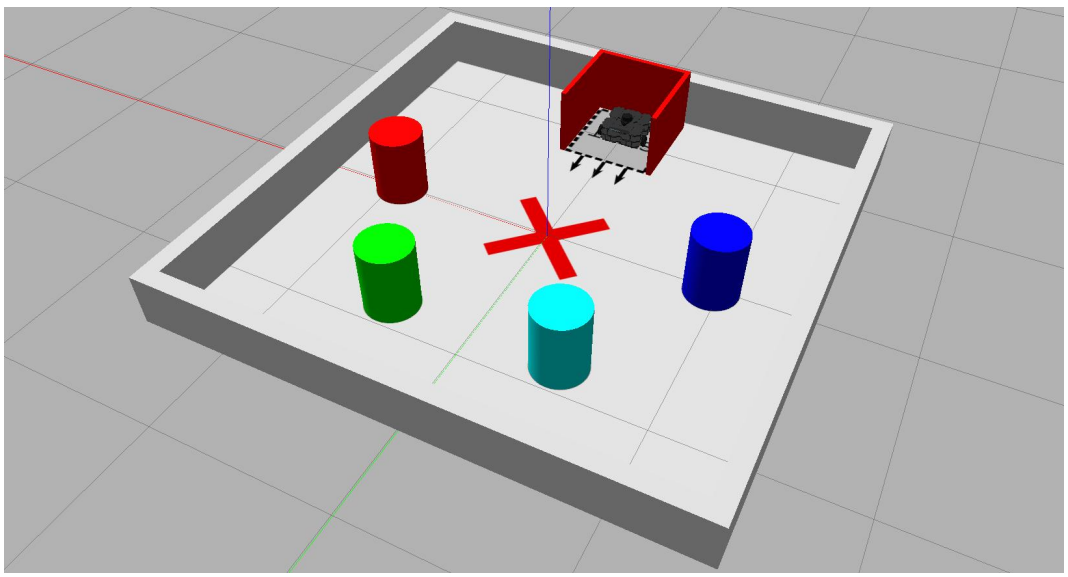

Figure 5: The `detection` arena for Task 2.

You should use this environment to develop your ROS node(s) for this task. This arena

will be used by the Teaching Team to assess your submission for this task, but note that ***the colour of the start zone and the pillars will be changed in the environment when we assess your submissions!***

4.2.2 Task 2 Details

1. Your robot will first need to determine the *Target Colour* by analysing the *Start Zone* that the robot will be located in when the simulation is launched.
2. Once the target colour has been determined **a message must be printed to the terminal** to indicate which colour has been identified. This terminal message should be formatted as follows:

   `SEARCH INITIATED: The target colour is {}.`

   Where `{}` is replaced by the *name* of the target colour as defined in Table 2.
3. The robot should then move forwards to locate itself in the centre of the arena. From the start zone, it will need to move directly forwards by approximately $1m$ to position itself on the red cross in the centre of the arena floor (this is just for guidance though, it doesn't have to be located exactly at this point).
4. There are four coloured pillars in the arena distributed in a 180° arc about the arena centre point (marked by the red cross). The robot will need to turn by 90° in either direction in order observe all four pillars.
5. The robot must stop turning when it is directly facing the correctly coloured pillar. **A message must be printed to the terminal** to indicate that the target beacon has been identified. This terminal message should be formatted as follows:

   `SEARCH COMPLETE: The robot is now facing the target pillar.`
6. The robot will have a maximum of 60 seconds to complete this task.
7. Your team's ROS package must contain a launch file called `task2.launch`. This will be used by the Teaching Team to launch the node(s) that you have developed for this task when we assess your submission. The intended functionality within your package must be launch-able via the command:

   ```
   $ roslaunch team{} task2.launch
   ```

   (The `detection` environment will already be running before the teaching team attempt to execute your `task2.launch` file).

4.2.3 Task 2 Marking

There are *10 marks* available for this task in total, awarded based on the criteria outlined in Table 4. No partial credit will be awarded unless specifically stated against any of the criteria.

Criteria		Marks	Details
A	Identifying the target colour	5/10	Whilst located within the start zone, a ROS node within your package must print a message to the terminal to indicate the target colour that has been detected and that will subsequently be used as the search criterion when analysing the colour of the four pillars in the arena. You will receive the full marks that are available here provided that the terminal message is presented *and* formatted as specified in Step 2 above.
B	Detecting the correct pillar	5/10	You will receive the full marks available here for stopping the robot once it is facing the correctly coloured pillar in the arena *and* ensuring a message is printed to the terminal to indicate that this has been achieved. The terminal message must be formatted as specified in Step 5 above.

Table 4: Marking Criteria for Task 2.

4.3 Part III: Search & Beaconing

Your robot should now (in isolation) be able to successfully navigate its way around an environment without crashing into things *and* detect an object of interest within an environment based on the colour of a start zone. Combining what you have done for Tasks 1 and 2 effectively provides your robot with the ability to *search* its surroundings whilst avoiding obstacles.

Remember that in the Final Challenge the environment will be populated with objects of different shapes, sizes and colours as detailed in Section 2.1. The target colour will be provided at the starting point of the maze and your robot will need to detect this, remember it and then search for the beacon that matches this colour.

Think now not only about how to merge the two behaviours that you have developed so far, but about how the ability to search could be optimised. You might wish to consider some of the examples of robotic search strategies that were discussed in Lecture 8, but there are many other ways to achieve this too, so feel free to be creative!

Having located the target object within the environment, your robot will then need to move towards it and stop within the allocated stop zone. This technique is known as *Beaconing* and some strategies for this were also presented in Lecture 8. You might also consider *Braitenberg's Vehicles* again, as introduced in Lecture 3, and consider how some of these simple models could be applied to this problem in order to control your robot's trajectory and approach to the target object.

The concept of *Visual Servoing*, discussed in Lecture 7, might also be worth considering as a method to control the position and trajectory of a robot based on images from its camera. Remember that the data from the robot's LiDAR sensor might also be very helpful to achieve this too.

TASK 3 ***(15/100 marks)*** **Due Monday 17th May**
Program your robot with a beaconing behaviour, combining what you have done in the previous tasks to allow your robot to move around an environment safely, search for an object and stop in close proximity to it.

4.3.1 Task 3 Simulation

When working on this task you should use an environment called `beaconing`, which can be launched in the WSL-ROS environment using any of the following three `roslaunch` commands:

1. `$ roslaunch com2009_simulations beaconingA.launch`
2. `$ roslaunch com2009_simulations beaconingB.launch`
3. `$ roslaunch com2009_simulations beaconingC.launch`

This arena contains *three* start zones: A, B & C; each of a different colour, as well as a number of uniquely coloured 3D objects (or *'beacons'*). There is a single beacon in the arena to match each of the three start zones, plus a couple more beacons to act as red herrings! The three `roslaunch` commands above can be used to launch your robot in each of the start zones, which determines the beacon that must be targetted for this task. You can therefore develop and test out your beaconing algorithms in three unique scenarios.

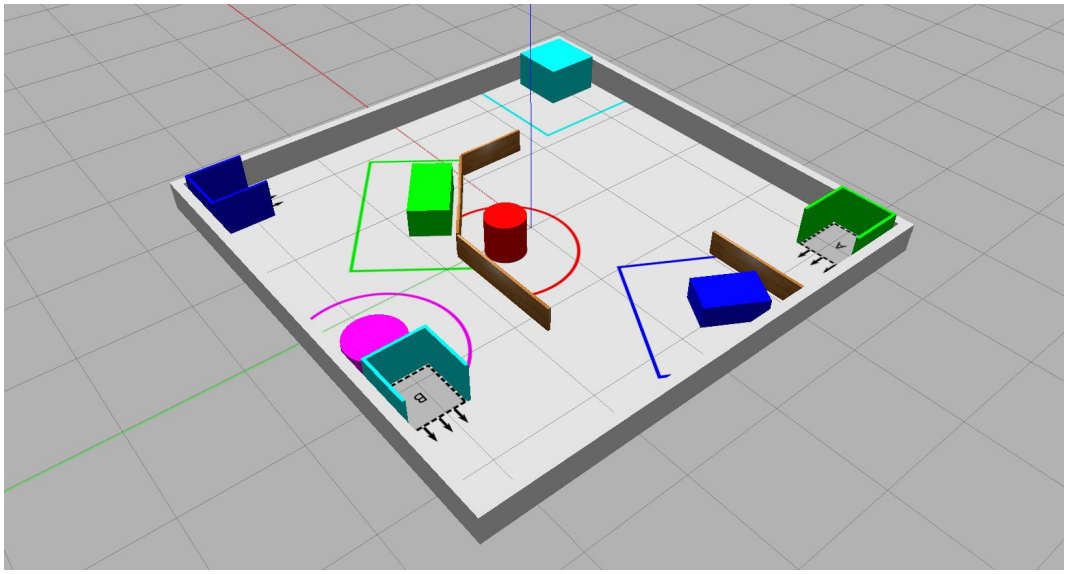

Figure 6: The `beaconing` arena for Task 3.

This arena will be used by the Teaching Team to assess your submission for the beaconing task, but note that ***your robot may be launched in any of the three start zones (selected at random) and the colour of the start zones and beacons may also be modified for the assessment!***

4.3.2 Task 3 Details

1. Your robot will first need to determine the *Target Colour* by analysing the *Start Zone* that it has been placed in on launching the simulation (selected at random by the assessor).

2. Once the target colour has been determined **a message must be printed to the terminal** to indicate which beacon colour will be targetted. This terminal message should be formatted as follows:

   `SEARCH INITIATED: The target beacon colour is {}.`

   Where `{}` is replaced by the *name* of the target colour as defined in Table 2.

3. The robot then needs to navigate the arena, avoiding contact with any of the objects that are located within it whilst searching for the beacon whose colour matches the target colour.

4. Once the target beacon has been detected, **a message must be printed to the terminal** to indicate that this has been achieved (the robot doesn't need to stop or pause, but the terminal message needs to be clearly visible and readable to the assessor). This terminal message should be formatted as follows:

   `BEACON DETECTED: Beaconing initiated.`

5. The robot then needs to start moving towards the beacon, stopping when it is close enough to be within the stop zone surrounding it, but not close enough to actually make contact. As shown in Figure 2, the stop zone surrounding each object will be $500mm$ greater than the beacon dimensions in the X-Y plane.

6. A further message must be printed to the terminal to indicate that the robot has successfully and intentionally stopped within the designated area. This terminal message should be formatted as follows:

   `BEACONING COMPLETE: The robot has now stopped.`

7. The robot will have a maximum of 90 seconds to complete this task.

8. Your team's ROS package must contain a launch file called `task3.launch`, which will be used by the Teaching Team to launch the node(s) that you have developed for this task when we assess your submission. The intended functionality within your package must be launch-able via the command:

   `$ roslaunch team{} task3.launch`

   (The `beaconing` environment will already be running before the teaching team attempt to execute your `task3.launch` file).

4.3.3 Task 3 Marking

There are *15 marks* available for this task in total, awarded based on the criteria outlined in Table 5. No partial credit will be awarded unless specifically stated against any of the criteria.

Criteria		Marks	Details
A	Identifying the target colour	2/15	Whilst located within the start zone, a ROS node within your package must print a message to the terminal to indicate the target colour that has been detected and that will subsequently be used to identify the target beacon. You will receive the full marks available here provided that the terminal message is presented *and* formatted as specified in Step 2 above.
B	Detecting the correct beacon	3/15	You will receive the full marks available here for ensuring a message is printed to the terminal to indicate that the target beacon has been identified. The terminal message must be formatted as specified in Step 4 above.
C	Stopping in the correct stop zone	5/15	Your robot must **stop** inside the correct stop zone within the 90 second time limit *and* a message must be printed to the terminal to indicate that this has been done intentionally. You will receive the full marks available here provided that this is achieved successfully and the terminal message that is presented is formatted as specified in Step 6 above. If your robot manages to stop, but part of its body lies outside of the stop zone then you will be awarded half-marks here.
D	An 'incident-free-run'	5/15	If your robot completes the task (or the 90 seconds elapses) without it making contact with anything in the arena then you will be awarded the maximum marks here. Marks will be deducted for any contact made, to a minimum of 0/5 (i.e. there will be no negative marking here: the minimum mark that you can receive for this is zero). Your robot must be moving within the arena continually to be eligible for these marks though, simply turning on the spot for 90 seconds is not enough!

Table 5: Marking Criteria for Task 3.

4.4 Part IV: Maze Navigation

As described in Section 2, your robot will need to navigate a maze composed of complex corridors without touching the walls. You therefore now need to think now about how you will build a behaviour for your robot that uses feedback from its on-board sensors to detect these corridor walls and maintain a safe distance from them, whilst also informing the robot of which direction to move in order to progress through the maze system. The LiDAR sensor may (again) prove rather useful here, but the data from the robot's camera might also provide some valuable information too.

Remember from Figure 2 that the maze may also contain *dead-ends*. This is a timed challenge, so you may want to ensure that time is not wasted exploring the same regions of the maze more than once: perhaps the robots *odometry* that is published to the `/odom` topic (and that we learnt about in Week 2 of Lab Assignment #1) could help with this?

One common method for solving mazes is to use a wall following algorithm. The Final Challenge maze won't contain any islands, that is: it will be *simply connected*, so this is one method you might choose to adopt here. Also consider what was discussed in Lecture 10: 'Maps and Path Planning' too, which may be useful when considering your approach to this part of the challenge.

TASK 4 ***(15/100 marks)*** **Due Monday 17th May**
Develop the ROS node(s) to make a TurtleBot3 Waffle robot navigate a maze in 150 seconds or less without crashing into anything.

4.4.1 Task 4 Simulation

Within the `com2009_simulations` package there is an arena called `maze_nav`, which you can use to develop and test out your robot's maze navigation behaviour. The simulation can be launched using the following `roslaunch` command:

```
$ roslaunch com2009_simulations maze_nav.launch
```

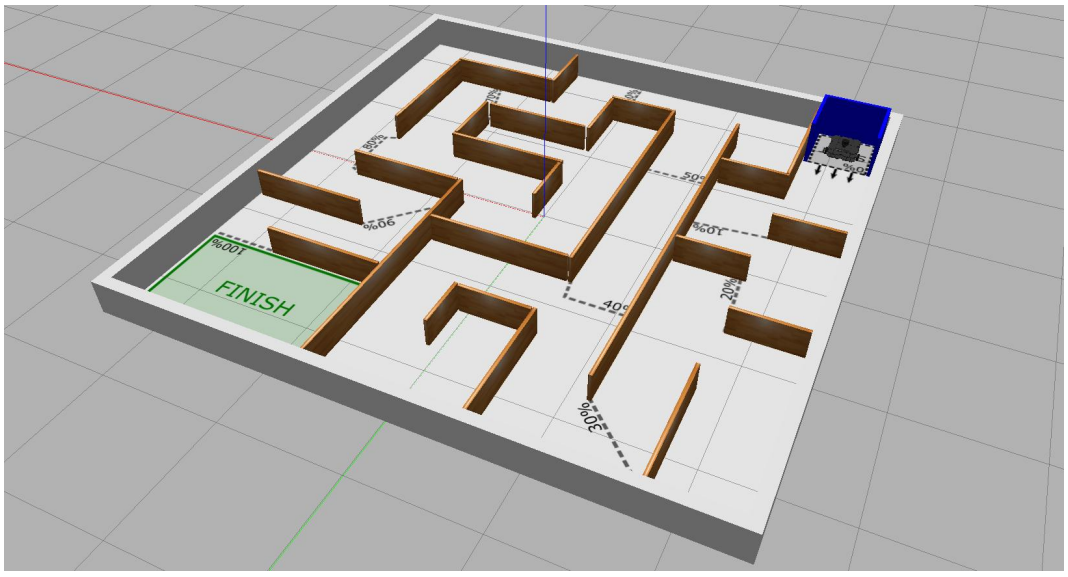

Figure 7: The `maze_nav` arena for Task 4.

This arena will be used by the Teaching Team to assess your submission for the Maze Navigation task. ***Nothing about the arena will be modified when we assess your submissions for Task 4.***

This arena contains a maze *similar* to that which your robot will need to navigate for the Final Challenge. The Final Challenge maze will be slightly smaller than this: the overall arena will still be 5x5m in size, but there will *also* be a Search Area which will consume some of the available space. The maze in the Final Challenge Arena will be constructed of the same types of wall: varying in length but maintaining the same height, width and 'wooden' appearance. Maze corridors will all be of a similar size: sufficient for a TurtleBot3 Waffle to pass through comfortably.

Remember: *Corners in the final challenge arena won't necessarily be at right angles: they may be more complex than those used in this example arena!*

4.4.2 Task 4 Details

1. On launching the simulation the robot will be located in the `Blue` Start Zone and you must program your ROS node(s) so that your robot can autonomously navigate the complex series of corridors in order to find its way to the `Green` Finish Zone, or as close as it can get to it.
2. The robot must be able to do this without touching anything in the arena and penalties will be applied for those that do (See Table 6 below).
3. The robot's progress will be measured using the progress markers printed on the arena floor at increments of 10%.
4. The robot will have a maximum of 150 seconds to complete this maze navigation task.
5. Your team's ROS package must contain a launch file called `task4.launch`, which will be used by the Teaching Team to launch the node(s) that you have developed for this task when we assess your submission. The maze navigation functionality within your package must be launch-able via the command:

   ```
   $ roslaunch team{} task4.launch
   ```

   (The `maze_nav` environment will already be running before the teaching team attempt to execute your `task4.launch` file).

4.4.3 Task 4 Marking

There are *15 marks* available for this task in total, awarded based on the criteria outlined in Table 6.

Criteria		Marks	Details
A	Progress through the maze	10/15	Marks will be awarded based on the largest progress marker that your robot passes within the 150 second time limit (it doesn't matter if the robot happens to turn around and move back behind that progress marker again at any point during the run). Marks will be awarded at 10% increments only (i.e. no fractional marks) and the **whole** of your robot must have crossed the progress marker in order to be awarded the associated marks.
B	An 'incident-free-run'	5/15	If your robot completes the task (or the 150 seconds elapses) without it making contact with anything in the arena then you will be awarded the maximum marks here. Marks will be deducted for any contact made, to a minimum of 0/5 (i.e. no negative marking). Your robot must *at least* pass the 10% progress marker to be eligible for these marks.

Table 6: Marking Criteria for Task 4.

4.5 Submission and Assessment for Tasks 2, 3 and 4

You should only make one submission for Tasks 2, 3 and 4: i.e. you should submit a single ROS package (as a `.tar` file) which contains the functionality for all three tasks. You should submit this to the Submission 2 Portal on Blackboard by the deadline shown in Table 1 (and on Blackboard).

Unlike Task 1, these tasks will be assessed *offline* by the Teaching Team. We will however still be using the WSL-ROS environment on a University computer in Virtual Classroom 1 to carry out the assessments. Results will be made available via Blackboard as soon as they have all been assessed.

See Section 1.5 for the standard submission requirements that must be fulfilled and for a reminder on how to prepare your package for submission (as a `.tar` file).

Remember: *It is up to you to ensure that your functionality can be launched in the way that we have specified for Tasks 2, 3 and 4 and that all your functionality launches as intended to satisfy each task. Failure to do this will result in your team being awarded zero marks for the Task in question, so* ***make sure you test everything out in the WSL-ROS environment (on a Virtual Classroom 1 machine) prior to submission!***

5 The Final Challenge

The last part of this assignment is the *Final Challenge*, where you will need to combine all of the behaviours that you have developed for your robot throughout Section 4.

Remember: *The Final Challenge Arena* ***will not be revealed*** *until after the Task 5 submission deadline, but we have put together an example arena to illustrate the* type *of environment that your robot will be faced with (see Section 5.2 below).*

5.1 Submission and Assessment for Task 5

Similarly to Tasks 2, 3 and 4: Task 5 will be assessed *offline* by the Teaching Team in the weeks following the submission deadline. The results will be released on Blackboard as soon as they are available and we will also publish a league table on Blackboard - for all to see - showing which robots managed to complete the challenge, where each team ranked in the competition and who the winners were!

You'll need to submit your ROS package for Task 5 to the Submission 3 Portal on Blackboard by the deadline shown in Table 1 (and on Blackboard). Remember to refer to Section 1.5 for the standard submission requirements that must be fulfilled and for guidance on how to prepare the `.tar` archive of your package for submission.

Once again: *It is up to you to ensure that your functionality can be launched in the way that we have specified for this Task (see below) and that all your functionality launches as you intend it to. So* ***make sure you test everything out in the WSL-ROS environment before you submit*** *to avoid being awarded zero marks!*

TASK 5 ***(35/100 marks)*** **Due Friday 21st May**
Combine everything that you have developed throughout Section 4 to produce a ROS package capable of making a TurtleBot3 Waffle robot complete a *Navigation, Search and Beaconing Challenge* and compete to finish the challenge in the fastest possible time!

5.2 The Final Challenge Example Simulation

The `com2009_simulations` package has been updated since the initial release of this document and now also contains an arena called `final_challenge_example`. If you haven't downloaded the `com2009_simulations` package to your WSL-ROS environment yet then follow the steps in Section 3.1 to do so. If you *have* already downloaded it then all you need to do now is to pull down the most recent updates to the package repository from GitHub:

1. First, in a WSL-ROS terminal instance, make sure that you are located in the root of the `com2009_simulations` package directory:

   ```
   $ cd ~/catkin_ws/src/com2009_simulations/
   ```

2. Then, download the new files from GitHub:

   ```
   $ git pull
   ```

You should then be able to launch the `final_challenge_example` simulation using the following `roslaunch` command:

```
$ roslaunch com2009_simulations final_challenge_example.launch
```

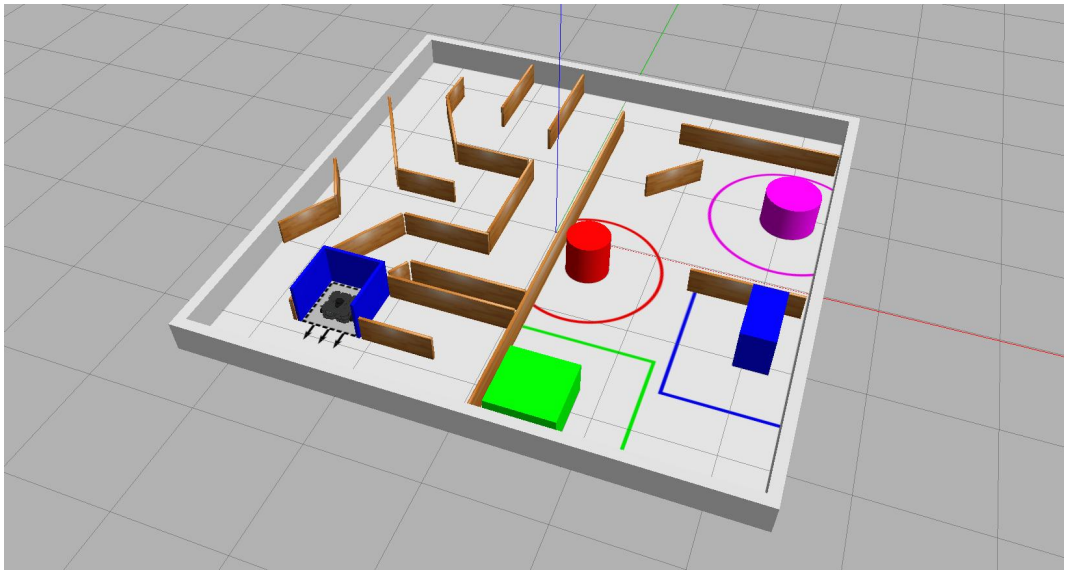

Figure 8: The `final_challenge_example` arena.

Remember: *The actual arena for the final challenge will be different to this and the example provided here is only to illustrate the type of challenge your robot will be faced with.*

Notice here that at the end of two of the maze corridors the start zone is visible: *how will you make sure your robot doesn't interpret this as being the target beacon?* Odometry data could provide a useful point of reference here...

5.3 The Challenge

Full details of the Final Challenge are provided below. Some of this is re-iterated from Section 2, but there are some additional details too, so please read carefully.

1. Your robot will be launched into a simulated world inside a *'Start Zone'*, which will be located at the start of a maze.
2. As illustrated in Figure 1, the Start Zone will be comprised of three walls, all of the same colour. The colour of the start zone will not be revealed beforehand, and could be rendered in any of the colours listed in Table 2.
3. To begin with, your robot will be facing the open side of this Start Zone (as shown in Figure 8).
4. First, the robot will need to detect the colour of the start zone walls as this denotes the *'Target Colour'* for the *search and beaconing* part of the challenge. Once this has been determined **a message must be printed to the terminal**, which should be formatted as follows:

   `TARGET COLOUR DETECTED: The target beacon is {}.`

   Where `{}` should be replaced by the colour name as defined in Table 2.
5. Your robot will then need to *navigate through the maze* as quickly as possible.

6. At the end of the maze, your robot will find itself in a *'Search Area'*, containing several beacons of various colours. No two beacons will have the same colour, and any of the colours defined in Table 2 may be used. The beacons will be 3-dimensional objects of cylindrical or box geometry.

7. Your robot will need to identify the *'Target Beacon'* within this space, that is: the beacon whose colour matches that of the start zone at the beginning of the maze. Once the robot has identified this, **a message must be printed to the terminal**, formatted as follows:

   `TARGET BEACON IDENTIFIED: Beaconing initiated.`

8. Your robot must approach this target beacon and stop inside a *'Stop Zone'* surrounding it. The perimeter of the stop zone will be printed on the arena floor and will be $500mm$ greater than the target beacon in its X and Y axes.

9. Your robot must stop within this zone without touching the beacon itself and once it has done this **the following message must be printed to the terminal**:

   `FINAL CHALLENGE COMPLETE: The robot has now stopped.`

10. Your robot must complete all this in the quickest possible time. Up to a maximum of 210 seconds (3 minutes and 30 seconds).

11. Throughout all this the robot will need to avoid touching anything in the environment!

12. Finally, the ROS package that your team submit for Task 5 must contain a launch file called `task5.launch`, which we (the Teaching Team) will use to launch all of the functionality that you have developed for this final challenge. The node(s) that you have developed for this task must be launch-able via the command:

    ```
    $ roslaunch team{} task5.launch
    ```

    ... once again, replacing `{}` with your own team number (we will already have the final challenge arena simulation up and running before we do this).

5.4 Marking

A: Progress through the maze *(10/35 marks)*:

Marks will be awarded for the furthest progress marker that your robot manages to pass whilst completing the maze navigation part of the challenge. Progress markers will be printed on the arena floor at 10% increments. As with Task 4, it doesn't matter if the robot happens to turn around and move backwards again: we will award marks based on the furthest marker that was passed at any stage during the run. Marks will be awarded at 10% increments only (i.e. no fractional marks) and the **whole** of your robot must have crossed the progress marker in order to be awarded the associated marks.

B: Search & Beaconing *(5/35 marks)*:

The marks here will be awarded as follows:

1. **Identifying the target colour** *(1/5 marks)*: At the start of the challenge and whilst your robot is still located in the start zone a terminal message must be printed to the terminal to indicate the target colour that has been identified for the search

& beaconing task. The terminal message must be formatted as specified in Step 4 above.

2. **Finding the target beacon** *(2/5 marks)*: Once your robot is inside the search area, a terminal message must be printed to the terminal once the robot finds the target beacon. The message must be formatted as specified in Step 7 above. It must be clear that the robot is actually looking at the target beacon when the terminal message is printed and the message must only be printed once.

3. **Stopping in the stop zone** *(2/5 marks)*: Your robot must stop inside the stop zone surrounding the target beacon (as printed on the arena floor). The whole of its body must be inside the stop zone markings and the robot must not make contact with the beacon. The message shown in Step 9 above must be printed to the terminal and the robot must have actually stopped in order to be awarded these marks!

C: Finish Time *(10/35 marks)*:

If your robot completes the challenge in under **210 seconds** then you will be awarded marks for the speed with which it does this:

Time (seconds)	**Marks**
210+	0
200-209	1
190-199	2
180-189	3
170-179	4
160-169	5
150-159	6
140-149	7
130-139	8
120-129	9
<120	10

Once 210 seconds has elapsed then the assessment will be stopped.

D: An 'incident-free-run' *(5/35 marks)*:

If your robot completes the challenge (within the maximum time limit) without it making contact with anything in the arena then you will be awarded the maximum marks here. Marks will however be deducted for any contact made, to a minimum of 0/5 (no negative marking). Your robot must at least pass the 10% progress marker in the maze to be eligible for these marks.

E: Exceptional performance *(5/35 marks)*:

A final 5 marks are available to teams whose robot performs exceptionally well in this final challenge, and to solutions that demonstrate innovation and creativity in their approach. This will be based on what we observe in the Gazebo simulation as your robot completes the challenge: we will not be scrutinising your code.

5.5 Final Step: Peer Assessment

The final step in this lab assignment is for you, **as a team**, to submit a *'Lab Assignment #2 Peer Assessment Form.'* This will be available for you to download as a `.pdf` file from the Peer Assessment Submission Portal on Blackboard. ***It is important that all teams complete and submit this form***.

Download the form from the submission portal (when available) and use the form fields in the document to provide the names of all of your team members along with their contribution to Lab Assignment #2 as a % (adding up to 100% in total for your team). Save the completed form using the format `team{}.pdf` (replacing `{}` with your team number), then head back to the submission portal to upload it.

This should be completed by no later than **Friday 28th May**.

Lab Assignment #2 Complete.