# IN621 – Final Project

|  |  |
| --- | --- |
| Assignment Issued | **Week 9** |
| Assignment is due | **Week 11** |
| Total Marks | **100 marks (30% of your final mark)** |

## Learning Outcomes covered

3 - Analyse and select appropriate software development platforms for robotics/automated systems implementation.

4 - Design a simple robotics/automated solution to a specified problem following sound principles of interaction design.

5 - Use an appropriate software development platform to implement simple interactive robotics/automated systems.

## Goal Seek and Navigate Out of an Obstacle Course

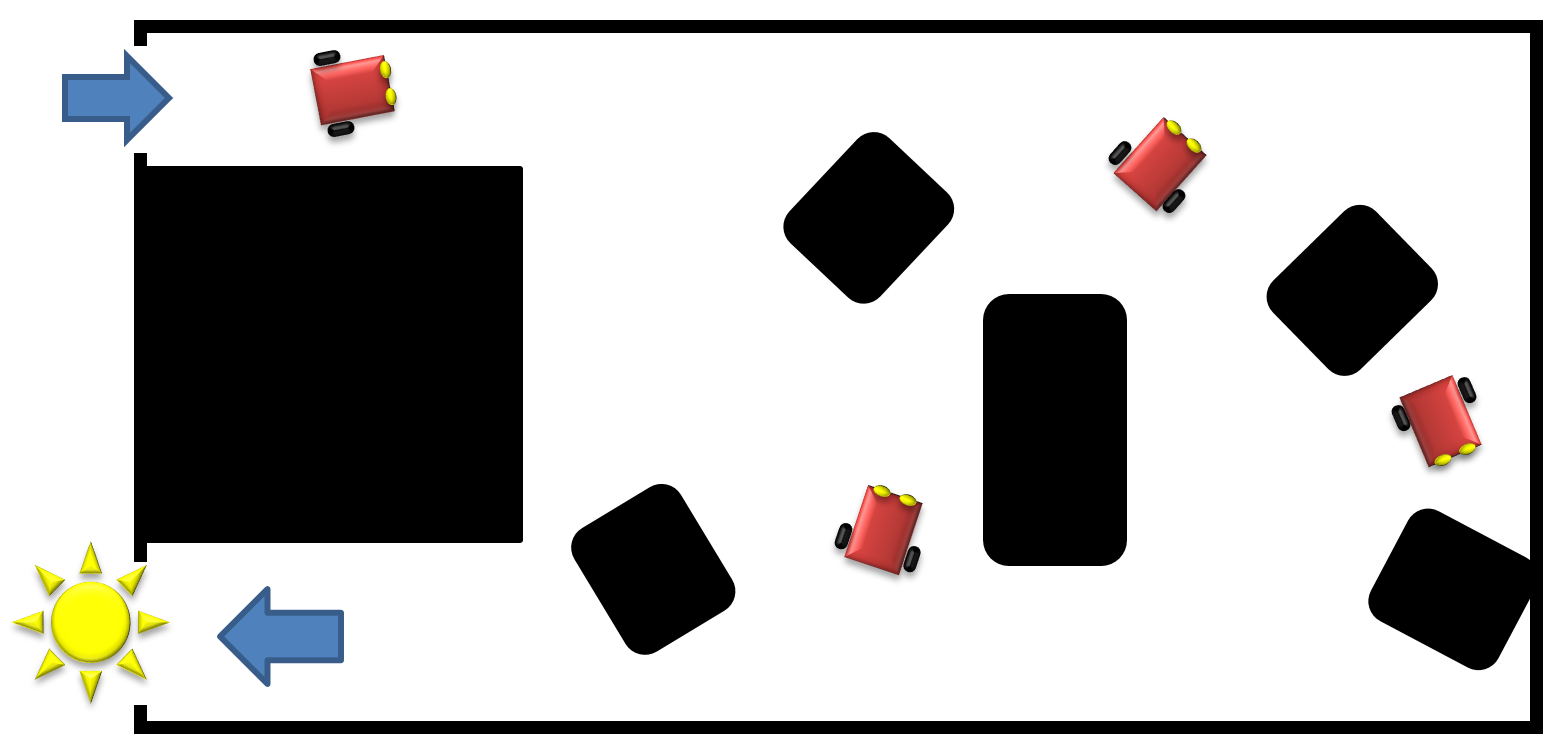


Figure 1 Example of layout

You have two main tasks to perform in the maze.

## Task 1 – Autonomous Navigation

Using **any sensor payload of your choice**, you will devise a robot that will **autonomously navigate a maze**. The overarching goal for your robot is to **seek the exit of the maze** which will be indicated by a bright light source.

## Task 2 – Broken Robot

You will repeat the goal of the previous task – your robot will autonomously navigate its way out of the same obstacle course. This time, **I will randomly select one sensor from your robot and neutralise it**. Using its remaining sensors your robot must successfully navigate its way out of the maze. This will demonstrate sensor validation and redundancy in your control software. You must only have one version of your control software, not multiple versions to be loaded depending on which sensor is masked.

## General Conditions

* You are allowed a total of **five pit stops** so you can fine tune the performance of your robot, fix broken parts, tweak code, etc.
* There will be a **random arrangement of static and moving obstacles** in the test area, including the potential of **other robots and people. I may move obstacles at any time**.

Your robot needs to be able to deal with a number of challenging conditions inside the maze. For example:

* Corners (Canyoning)
* Getting stuck against objects that don’t trigger a sensor (a version of Stasis)
* Losing the light source
* Anything else!

It would be a sensible idea to plan out the code for your robot before implementing it, to try and catch all of the contingencies.

## General Requirements

1. The robot must exhibit behaviours (sensor inputs 🡪 actuation)
2. The robot must use a closed-loop control system for *Servo Behaviours*.
   1. bang-bang control or other.
   2. The robot may also have an open-loop system for *Ballistic Behaviours*. Eg: panic, randomising to prevent canyoning or stasis states.
3. The robot must have a schema for mediating behaviour demands for control of the hardware (Fixed Priority Arbitration, use of a Finite State Machine, or Other)
4. The robot must reliably find its way out the obstacle course. It must be able to do this **twice** (no flukes). Your robot should be able to respond to changing conditions in the test area.
5. Software: your software should be substantially your own work, be well written, use methods, be indented correctly, and exhibit other good coding practice. No spaghetti in loop(). Un-credited code from other sources is plagiarism and gets a zero mark.

**Marking Schedule – 100 marks in total**

Your robot performance is marked in class. Please make sure you code is uploaded to Moodle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task ↓** | **Worthy of top marks**  **10 - 9** | **Honest attempt to be better than average**  **8 - 7** | **Struggling**  **6 – 5** | **Insufficient**  **4 – 0** |
| **Software:**  **Weight:55%** | * Behaviours are evident (Servo and Ballistic) and well thought out. * Control system is in pragmatic use. * Behaviour arbitration or similar is evident, and allows for graceful degradation. * Code is well written and follows good practice. | * Behaviours are evident (Servo and Ballistic). Some logical issues (i.e. missing behaviours). * Control system is in pragmatic use, but roughly implemented * Behaviour arbitration or similar in use. Could be better. Graceful degradation clunky, but works. * Code is ok. Some tendency to untidiness, but tries to follow guidelines. | * Behaviours poorly implemented. Missing, illogical or scope too small/poorly planned. * Very poor use of feedback and control. * Behaviour arbitration or similar is brittle and transitions between behaviours are poorly done. Graceful degradation is problematic * Code is a hot mess. (But it does work, after a fashion) | * Robot is implemented as ballistic behaviours (Not cleverly) * Mostly open loop control * No behaviour arbitration or graceful degradation * Code not fit for purpose. |
| **Robot: Navigates obstacle course twice**  **Weight: 45%** | Responds to changing conditions gracefully.  Actively seeks goal.  Robot completes task | Robot mostly responds to changing conditions, with minor problems.  Robot may appear to lose goal, but reacquires.  Robot completes task | Robot has moderate issues with conditions.  Robot appears to lose goal without regaining, or mostly completes task by avoiding obstacles.  Robot completes task | Robot is unable to complete task |

Pit stops: ❑❑❑❑❑