**Faculty of Engineering and IT  
School of Mechanical and Mechatronic Engineering  
41069 Robotics Studio 2**

WaitForMe

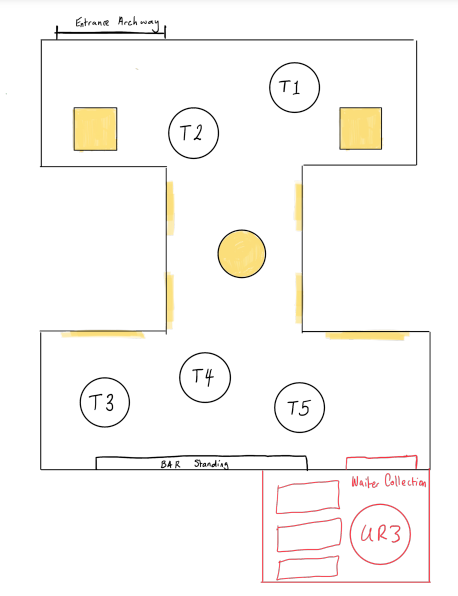
Client: Tom Ugly’s Art Gallery

Project Team:

|  |  |  |
| --- | --- | --- |
| **Full Name** | **Student ID** | **Expertise** |
| Andrew Goode | 13852898 | Manufacturing, WSL |
| Issy Pitt | 14040354 | Rover Team Lead, Assistant Robotics Engineer, Raytheon GradX, |
| Hallie Robins | 14253583 | Systems Integration Engineer |
| Thomas Dodgson | 13887791 | C++ , MATLAB, SolidWorks, Arduino IDE |

# Overview

Tom Ugly’s Art Gallery has tasked us with the automation of providing drinks to customer at the art gallery while they wait at tables. They have asked us to come up with a solution that utilises a TurtleBot design with multiple robots to delivery drinks in unison whilst avoiding objects and humans in the environment.



# Aims

1. Create a simulated environment of the Art Gallery, tables and drinks station.
2. Simulate at least 2 TurtleBot moving between the drinks station and the tables whilst avoiding objects.
3. Simulate 2 TurtleBot’s avoiding humans moving in the environment.
4. Implement functionality onto real TurtleBot’s.
5. Delivery drinks without dropping or spilling drinks during transportation

# Resources

Describe the resources required to complete the project. This includes: the robotics hardware & software available in the Lab, additional hardware & software you might need to source yourself, fabrication access (3D printing, laser cutting, etc.).

* Minimum of 2 TurtleBot’s for use in the lab
* ROS2 and gazebo

# Subsystems and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **Subsystem** | **Description** | **Lead by** | **Supported by** |
| Path Planning and Object Avoidance | Create logic to ensure object avoidance based on objects received by laser data. Creates optimal path planning in a known and changing environment | Andrew | Tom |
| TurtleBot Movement Logic | Logic includes travelling to selected tables based on information obtained from the host robot, and after delivering drinks, returning to the drinks station. | Issy | Hallie |
| Simultaneous TurtleBot Operation | Create logic to allow two TurtleBot’s to move without colliding with each other, and they will follow the same logic as above without duplicating movements (e.g. going to the same table.) | Hallie | Issy |
| SLAM | Localisation on known map environment. Utilises LIDAR data to identify objects in the real environment | Tom | Andrew |

# Evaluation

Additional information and clarification can be added here.

|  |  |  |
| --- | --- | --- |
| Subsystem 1: Path Planning, and Object Avoidance | | |
| P | TurtleBot’s can reach goals without colliding with any stationary known obstacles |
| C | P and TurtleBot can recognise and navigate around newly discovered stationary objects. |
| D | C and implements own rudimentary path planning. |
| HD | D and optimise to move along the shortest path. |
| Extension | HD and create a safety factor to move around the objects within a range that can be set. |
| Subsystem 2: TurtleBot Movement Logic | | |
| P | TurtleBot moves to a hardcoded table destination and returns to hardcoded drinks location |
| C | P but TurtleBot receives goal data from host for destinations, but goals may overwritten when new ones are published or node does not listen and save new goals during actions. Goals can be blocking and sent from terminal, instead of from a service. |
| D | C and TurtleBot can navigate to specific locations given by host and return to drinks stand, repeating action indefinitely/always listening for new goals from host. Uninterrupted movement even when goals are sent, services setup for sending 'go trigger' and goals are sent from a service and saved in a queue |
| HD | D and the logic is scalable for multiple TurtleBot’s to go to all of the given goals and return to drinks table. There will be a master node/functionality for scalability, not identical logic nodes running side by side. |
| Extension | HD and TurtleBot can navigate dynamic environments without losing functionality and still getting to the goals |
| Subsystem 3: Goal Management and Assignment | | |
| P | Receives communication from publishes and subscribers. Organises and distributes information to 2 different TurtleBot’s. both robots receive a single goal to inform an action. |
| C | P and TurtleBot’s can continue to receive new goals after completing a previous goal and returning home by referencing status updates from the bots and enqueuing and dequeuing goals. |
| D | C and optimising which data is sent to which TurtleBot before homing based on time to home, by including more detailed status updates from each TurtleBot and rearranging a vector of turtle bots to process what turtle bot takes priority. |
| HD | D and goals can be executed in non-chronological order to handle urgent or priority tasks, by executing 2 queues simultaneously to handle urgent task separately. |
| Extension | HD and Logic is scalable to a total of 5 turtle bots. |
| Subsystem 4: SLAM | | |
| P | Can localise itself on a map in a known environment |
| C | P and can distinguish between tables, people, and artwork from LIDAR data (use opencv feature detection) |
| D | C and can distinguish the turtlebot and track its movements with self-written function. |
| HD | D and distinguish objects without using a library (eg use ML such as SVM) |
| Extension | HD and create own mapping logic (.cpp) |