RS2 Technical Documentation Draft

**Project overview**

* + One to two sentences describing its purpose

We are part of a collaboration between 3 groups to create a robot guided art gallery experiences with service of drinks.

Our Project is to utilise turtlebot3 robots working in tandem to deliver drinks to art connoisseurs as they experience the art at Tom Ugly’s Art Gallery. The turtlebots will be required to navigate to known locations using the shortest path, and avoid known and discovered obstacles.

* + Key features / subsystem list

Our project has been broken into 4 subsystems which when integrated together allows the turtlebot to navigate the art gallery.

1. Path planning and object avoidance: Involves finding the shortest path to a known location, and updating the route when objects are discovered.
2. Movement Logic: Involves reducing the delay between movement, storing new goals and acting indefinitely without any loss in function, across multiple turtlebots
3. Simultaneous Operation and Goal distribution: Involves ensuring both turtlebots can operate from a single computer and dividing received goals between multiple robots to ensure efficient operation.
4. Initialisation and Object Detection: Involves initialising the turtlebots so they are localised in an environment, and detecting and identifying different known and unknown obstacles.

**Dependencies**

* + Hardware (parts list or Bill of Materials)
  + Computing Specs (if relevant)

Turtlebot3 x 2

Lidar detector x 2 (mounted to turtlebot)

Logitech Camera x 2 (mounted to turtlebot)

Computer running Ubuntu

* + Software (additional libraries that are not included in the installation steps, ROS version, MATLAB, etc.)

OpenCV

Multi-turtlebot library:

<https://github.com/dkhoanguyen/turtlebot3_multi_robot.git>

**Installation**

* + Hardware (custom end-effector, environment, work area, etc.)

Environment should be constructed out of foam padding for constraining the walls/space.

Artwork/tables can be represented with obstacles of known sizes from the lab

* + Software (step-by-step commands: git clone, rosdep, build, etc.)

OpenCV Installation: <https://medium.com/@redswitches/how-to-install-opencv-on-ubuntu-22-04-fe736d51c705>

Follow these steps to install OpenCV from the official Ubuntu repositories:

Open the terminal and update the system repository package:

# sudo apt update

Next, use the **apt**package manager to install OpenCV.

# sudo apt install libopencv-dev

This command installs the latest version of the OpenCV development library available in the Ubuntu repositories.

Verify the installation by checking the version of OpenCV.

# dpkg -l libopencv-dev

Multi-turtlebot library: <https://github.com/dkhoanguyen/turtlebot3_multi_robot>

In the workspace, clone the turtlebot3\_simulation package and build **WITH SIMLINK-INSTALL**

# mkdir -p multi\_tb\_ws/src

# cd multi\_tb\_ws/src

# git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3\_simulations.git

cd ../

# colcon build --symlink-install

In the same workspace, clone this package and build

# cd multi\_tb\_ws/src

# git clone https://github.com/dkhoanguyen/turtlebot3\_multi\_robot.git

cd ../

# colcon build --symlink-install

Remember to source the workspace whenever you want to run

# source multi\_tb\_ws/install/setup.bash

**Running the system**

* + Launch command / GUI

1. *Integrated launch code:*

TERMINAL 1: Launch file (takes about 4 mins total)

# ros2 launch issy multi\_launch.py

TERMINAL 2: example goals w service call

# ros2 service call /add\_goal issy/srv/AddGoal "{x: 1.0, y: 0.8}"

# ros2 service call /add\_goal issy/srv/AddGoal "{x: 2.0, y: 2.5}"

TERMINAL 2: service call for executing next goal in queue (only when turtlebot is in idle state/back at bar)

# ros2 service call /execute\_goals issy/srv/ExecuteGoals "{}"

1. *Path Planning*

TERMINAL 1- Spin Node

# ros2 run andy astar\_planner

TERMINAL 2- launch world

# export TURTLEBOT3\_MODEL=waffle\_pi

# ros2 launch turtlebot3\_gazebo Gallery\_Test2.launch.py

This will take a few minutes for the turtlebot3 to load into the environment but is quicker if you are not connected to the internet

TERMINAL 3- launch map (aka: rviz nav2 map thingy)

# export TURTLEBOT3\_MODEL=waffle\_pi

# ros2 launch turtlebot3\_navigation2 navigation2.launch.py use\_sim\_time:=True map:=$HOME/ros2\_ws/src/waitforme/GalleryMapHD.yaml

In Rviz localise by using /initial\_pose button

Add MarkerArray and subscribe to topic /Astar\_path

TERMINAL 4- run service call to send goal

# ros2 topic pub /astar\_goal geometry\_msgs/msg/PoseStamped "{header: {frame\_id: 'map'}, pose: {position: {x: 2.0, y: 0.5, z: 0.0}, orientation: {x: 0.0, y: 0.0, z: 0.0, w: 1.0}}}" --once

1. *Object Detection & Localisation*

TERMINAL 1- launch world

# export TURTLEBOT3\_MODEL=waffle\_pi

# ros2 launch turtlebot3\_gazebo Gallery\_Test2.launch.py

This will take a few minutes for the turtlebot3 to load into the environment but is quicker if you are not connected to the internet

TERMINAL 2- launch map (aka: rviz nav2 map thingy)

# export TURTLEBOT3\_MODEL=waffle\_pi

# ros2 launch turtlebot3\_navigation2 navigation2.launch.py use\_sim\_time:=True map:=$HOME/ros2\_ws/src/waitforme/GalleryMapHD.yaml

In rviz add the marker to sidebar – will need to add topic after detection is running

TERMINAL 3 – Run localisation node

# ros2 run tom localisation\_node

Type into terminal

# setpose 3.12 3.31

TERMINAL 4 – Run Detection node

# ros2 run tom detection\_node

In Rviz change the topic for maker to be /visualization\_marker

TERMINAL 5 – Run people detection

# ros2 run tom people\_detector

1. *Dual Turtlebot launching* 
   * Expected outcome (with screenshots, pictures)
2. *Object Avoidance*

As the turtlebot moves around the map, it should discover and publish the positions of tables and artworks, publishing them onto the rviz map as black circles, and red squares respectively.

***Picture***

1. *Path Planning*

As a goal is sent to the Astar planner the Rviz will update showing the shortest path.

***Picture***

1. *Dual Turtlebot*

Both turtlebots will appear in rviz and the world.?

***Picture***

**Subsystem specifics**

* + Purpose

NOTE: Copied from above

1. *Path planning and object avoidance*: Involves finding the shortest path to a known location, and updating the route when objects are discovered.
2. *Movement Logic:* Involves reducing the delay between movement, storing new goals and acting indefinitely without any loss in function, across multiple turtlebots
3. *Simultaneous Operation and Goal distribution:* Involves ensuring both turtlebots can operate from a single computer and dividing received goals between multiple robots to ensure efficient operation.
4. *Initialisation and Object Detection:* Involves initialising the turtlebots so they are localised in an environment, and detecting and identifying different known and unknown obstacles.
   * Key topics/services/files
5. *Path planning and object avoidance:*
   1. Topics: /map, /odom, /astar\_goal, /astar\_path, /planned\_path
   2. Services: nil?
   3. Files: .yaml & Gazebo
6. *Movement Logic:*
   1. Topics: /tb1/astar\_goal, /tb2/astar\_goal, /tb1/cmd\_vel, /tb2/cmd\_vel, /tb1/planned\_path, /tb2/planned\_path, /tb1/odom, /tb2/odom
   2. Services: /add\_goal, /execute\_goal
   3. Files: .yaml & Gazebo
7. *Simultaneous Operation and Goal distribution:*
   1. Topics
   2. Services
   3. Files: .yaml & Gazebo
8. *Initialisation and Object Detection:*
   1. Topics: /scan, /odom, /visualisation\_marker, /initialpose, /camera/image\_raw, /people\_detector/image, /people\_marker
   2. Services: non used
   3. Files: .yaml & Gazebo
   * Configurable settings (if relevant)

**Troubleshooting & FAQs**

Just so much has gone wrong.