EE4308 Project 1

Turtlebot3 Burger Without Localization

AY 23/24 Semester 2
February 11, 2024

Lai Yan Kai lai.yankai@u.nus.edu
Xue Junyuan xue.junyuan@u.nus.edu
A/P Prahlad Vadakkepat prahlad@nus.edu.sg

This document is scaled for easier reading on phones.
Please consider a 2-page layout when printing.

Contents

1 Introduction

Move a Turtlebot3 Burger (turtle) around a **flat** area containing obstacles and narrow corridors. The turtle must reach three waypoints, in order, and within $0.1i$ m ($i \in 1, 2, 3$) of them, within 90s. This is a culmination of lab 1 and lab 2, and on real hardware.

The `.zip` containing the team's codes, the team's `.pdf` report, and the team's `.mp4` presentation and demonstration video have to be uploaded to Canvas by Week 7 Sunday, 10 March, 23:59h.

2 Loaning of Kits

Each kit contains multiple items including the Turtlebot3 Burger. A loan form needs to be completed. It contains information about the hardware that will be loaned. Teams are liable for any damages.

1. Complete the loan form digitally (all team members must sign), and submit it to `Canvas` → `Assignments` → `Project 1` → `Turtlebot3 Burger Loan Form`
2. After submitting the form, arrange a time with the teaching assistant(s) to collect the kit. Only one member needs to be present.

3 Signups

3.1 Sign up Teams

Every student has to go to `Canvas` → `People` → `Project Groups` to sign up for the teams.

3.2 Sign up for Lab Slots

For testing your robot on the actual field. Go to `Canvas` → `People` → `P1 Lab` to sign up for the slots (One team one sign up).

3.3 Sign up for Presentation Slots in Week 8

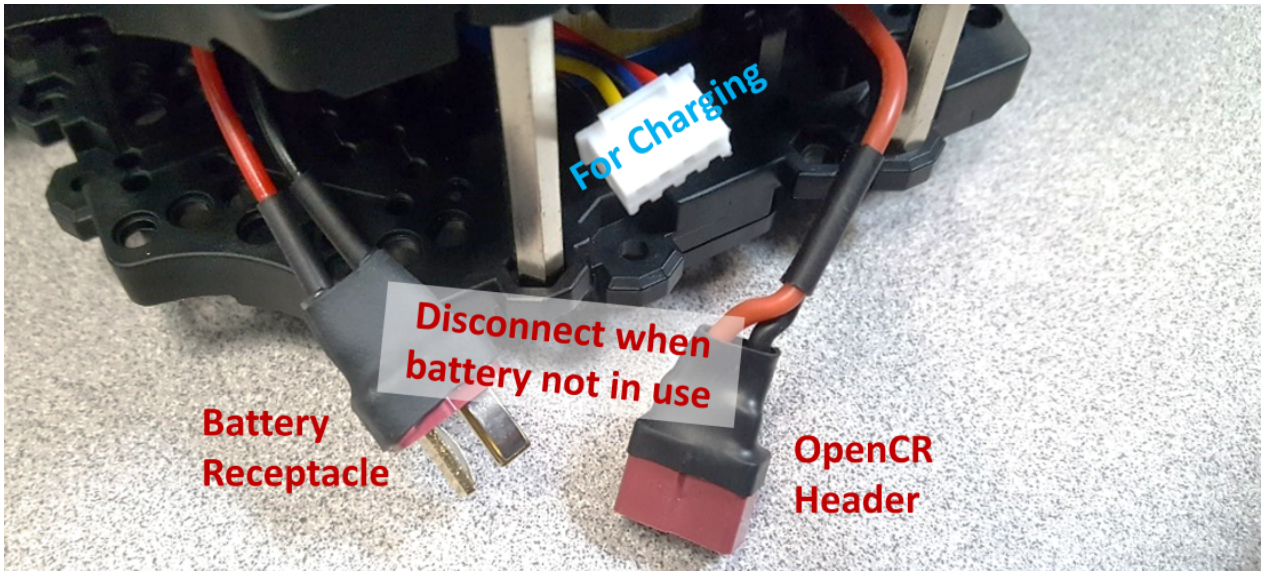
You are required to be present while the compiled video is played back, and to answer questions after that. Go to `Canvas` → `People` → `P1 Presentation` to sign up for the slots.

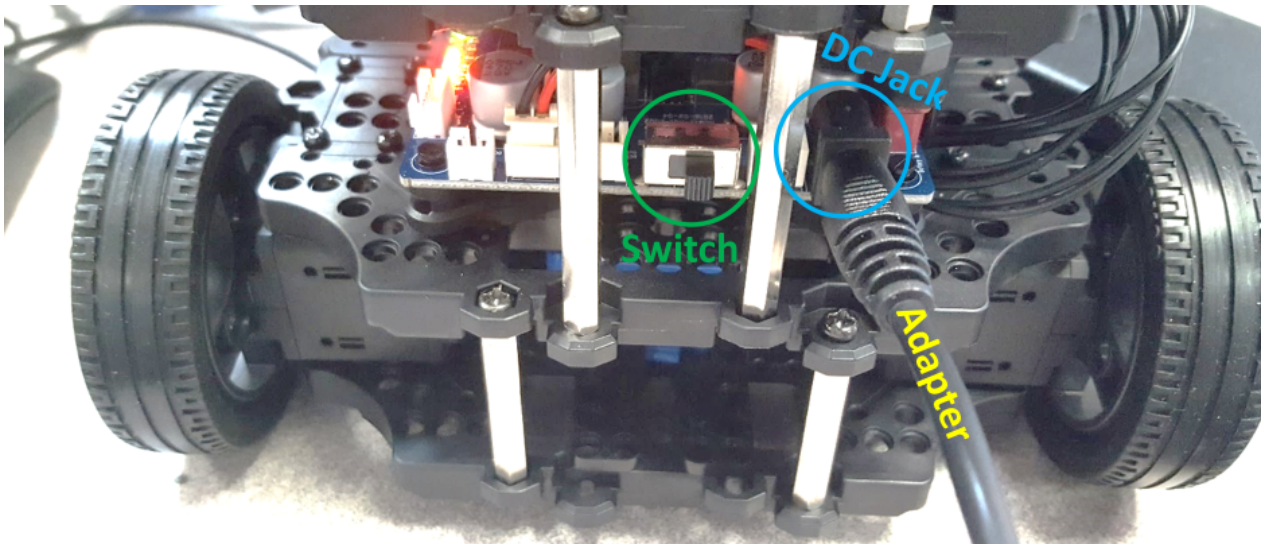
4 Important Hardware Information

Please read this section to avoid spoiling the robot.

4.1 Tethered and Portable Modes

1. For the **portable** node, connect the battery to the OpenCR board using the red plugs, and unplug the DC jack. Use this mode when the robot has to run in the playing field.





2. For the **tethered** mode, plug the DC jack in to the OpenCR board. The red plugs from the OpenCR board and the battery need not be disconnected. This is the **preferred** mode when the robot is turned on and not running any programs.
3. When the power sources are connected, turn on / off the robot using the switch on the OpenCR board. **Keep it turned off when not in use**.

4.2 Important Power Information

1. When the robot is turned on and not running any programs, always use the tethered mode to prolong the battery life. Plugging in prevents power from being drawn from the battery.
2. Never deplete the battery completely, as doing so will shorten its shelf-life.
3. Always disconnect the battery when not in use.
4. Always try to use `sudo shutdown now` (Sec. 7.7) from the robot (via SSH or keyboard) before turning off with the switch on the OpenCR board.

4.3 Battery Charging

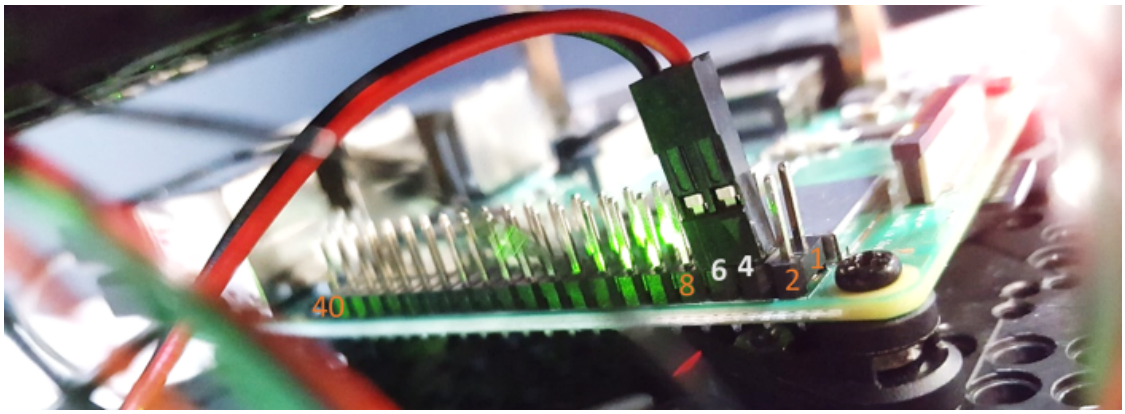
1. Before charging, **disconnect the battery from the OpenCR board**.
2. Connect the white plug of the battery to the charger.

3. The charger will show a red light if charging, and green light when the battery fully charges.

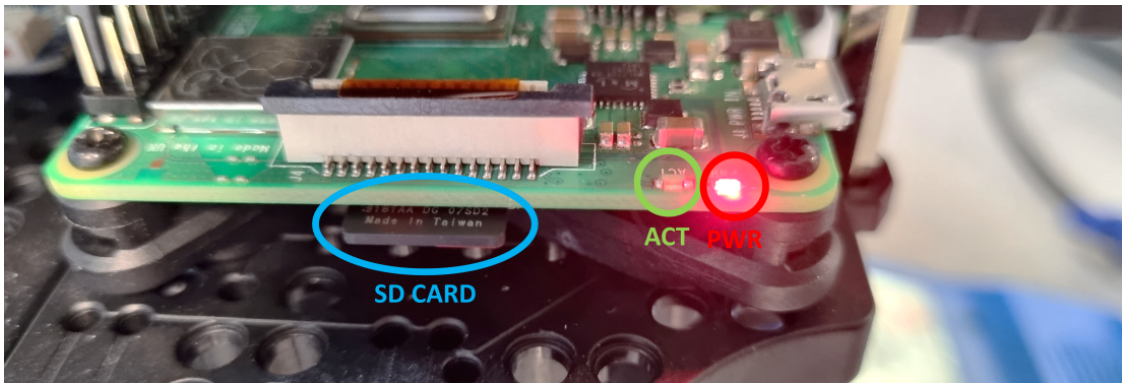
4. Once charging is complete, **do not pull out the charger by pulling against the battery wires. Instead, use a flat screwdriver to push out the white connector of the battery.**

4.4 Others

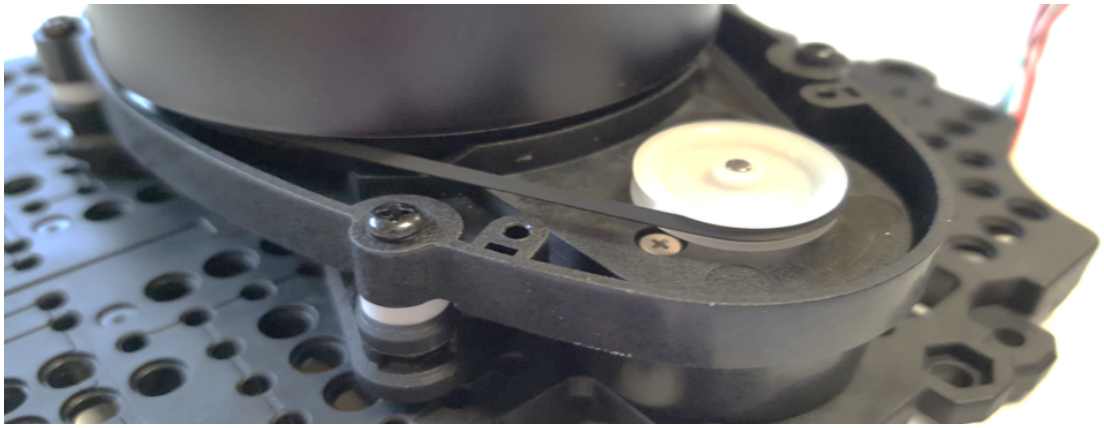
1. Do not remove the power supply for the Raspberry Pi 3B+ (Rpi). Ensure that the wire is placed at the correct pins as shown. Other configurations may short and destroy the Rpi.



2. Do not remove any existing USB cables from the Rpi.
3. Do not remove the SD card from the Rpi.



4. Make sure the LIDAR rubber belt is wedged inside its white wheel.



4.5 Prerequisite Hardware for Project

Make sure you have these items before proceeding to the next section.

Hardware	Purpose
Remote PC	Runs the ROS server and nodes to control the turtle.
Turtle	Runs a ROS client that receives commands to move, and sends sensor data to remote PC. Provided in the kit, including charger and adapter.
Non-NUS Wi-Fi / Hotspot	Connect both remote PC and turtle to this. The NUS wi-fi will not work. VirtualBox users will not be able to connect to the hotspot broadcasted from the remote PC. This guide will treat this as a separate device, like your mobile phone or another PC.
USB Keyboard	For first-time setup or connectivity troubleshoots. USB wired ones preferred.
Monitor	For first-time setup or connectivity troubleshoots. Rpi has a HDMI port. Find a cable that can connect to your monitor. A VGA cable and a HDMI to VGA converter is provided in the kit.

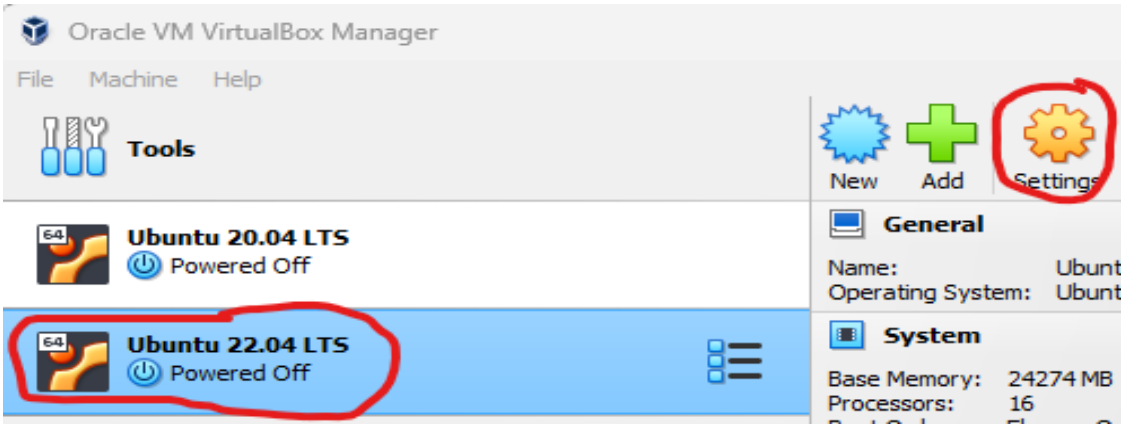
5 First-time Remote PC Setup

The remote PC is the one you have used so far to code your labs.

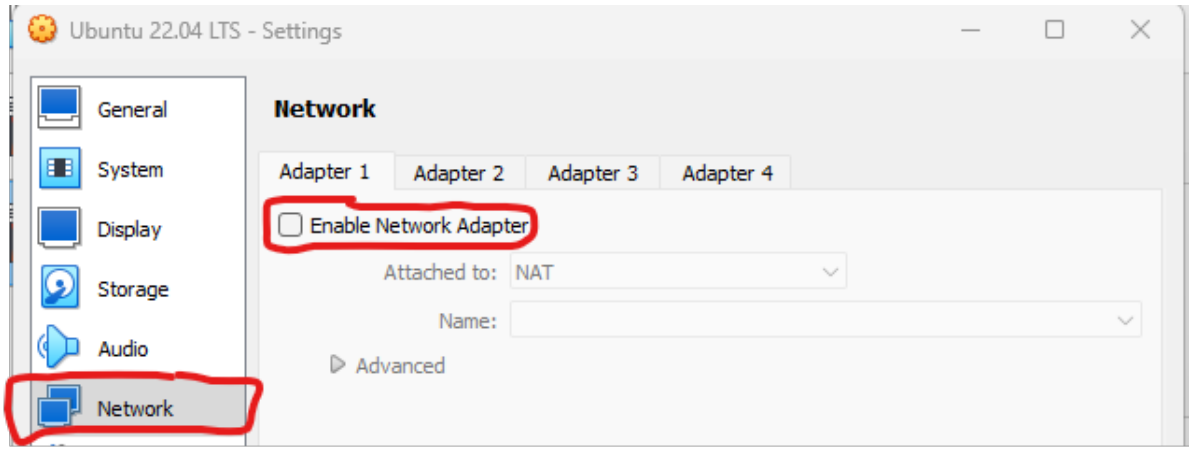
5.1 Remote PC Setup on VirtualBox

The NAT network adapter has to be deactivated. Only the bridged adapter can be used when communicating with the robot.

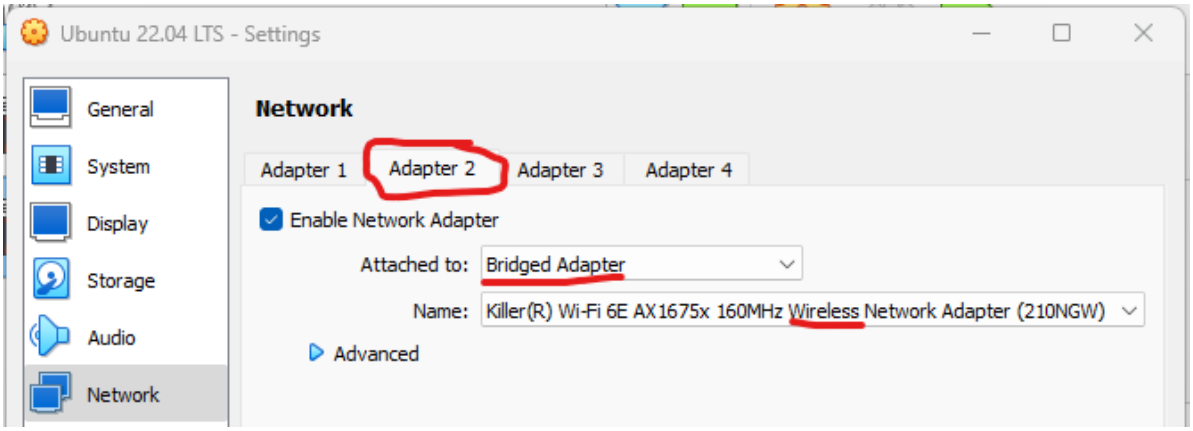
1. Before starting Ubuntu on VirtualBox, go to `Settings`.



2. Click on `Network`. In `Adapter 1`, uncheck `Enable Network Adapter` to disable the NAT network adapter.



3. Click `Adapter 2`.
   a) Ensure that `Enable Network Adapter` is checked.
   b) Ensure that it is attached to `Bridged Adapter`.
   c) Ensure that your wireless adapter is selected.



4. Click `OK` and you can start virtual machine.

5.2 Connect to Non-NUS Wi-Fi or Hotspot

ROS2 communication cannot occur over the NUS Wi-Fi due to the institution's firewall rules. This requires you to use your own Wi-Fi or phone hotspot. Make sure that the remote PC can be connected to the selected Wi-Fi or hotspot. If using hotspot, the hotspot is preferably broadcasting from an Android phone.

5.3 Download and Merge Code

1. Download `proj1.zip` from `Canvas` → `Files` → `Project 1` onto the remote PC.
2. Open it and extract the `ee4308` workspace folder onto the `~` folder.
3. Merge your code in Lab1 to the `estimator.hpp` header file in the `ee4308_turtle` package.
4. Merge your code in Lab2 to the `controller.hpp` header file in the `ee4308_turtle` package.
5. Merge your parameters in Lab1 and Lab2 to the `proj1.yaml` in the `ee4308_bringup` package.
6. Open a terminal, navigate to the `ee4308` folder, and build using

```
. bd.sh
```

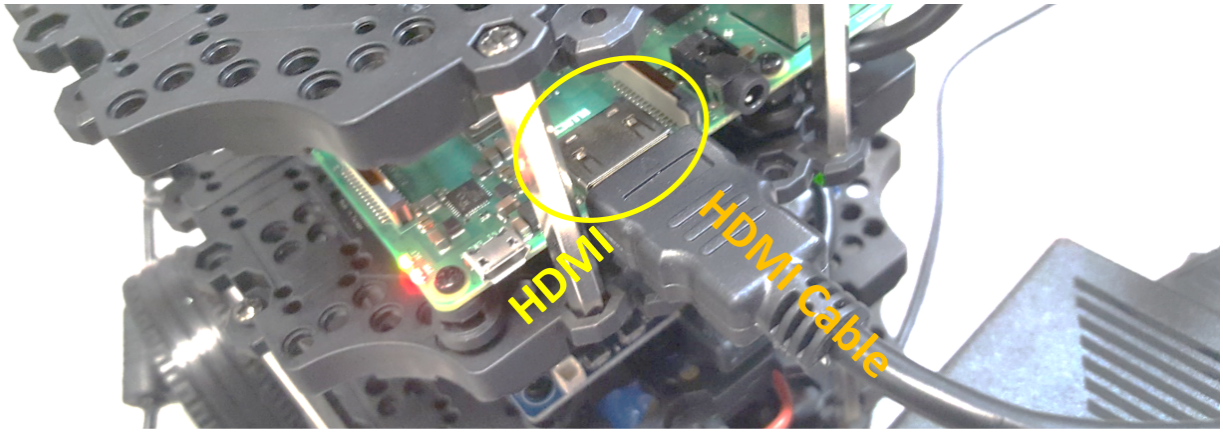
Please resolve any errors before proceeding.

6 First-time Turtle Setup

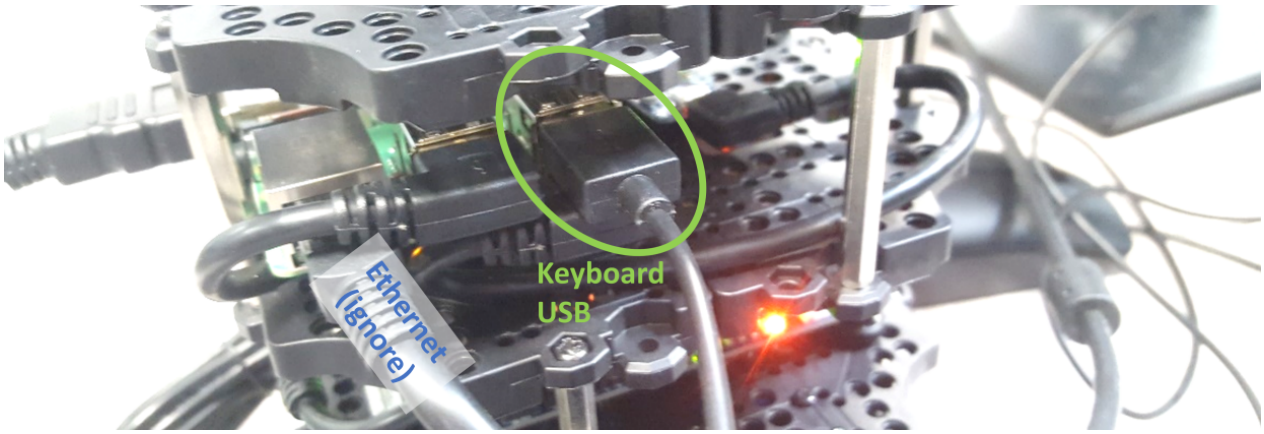
This section has to be done if connecting to a new Wi-Fi or hotspot for the first time, and you need to know the IP address of the turtlebot.

6.1 Plugging in Monitor and Keyboard

1. If the robot is already turned on, turn it off using the switch on the OpenCR board. Avoid turning it off this way in the future.
2. Plug in the monitor via the HDMI port on the Rpi. If this step is skipped, nothing will be displayed on the monitor, even after plugging in when the Rpi is turned on.



3. Plug in the USB keyboard into the USB port on the Rpi.



4. Turn on the robot in the **tethered** mode (see Sec. 4.1).

6.2 Logging In

Once the flurry of activity has stopped on the monitor, you can proceed to log in:

1. Type `ubuntu` as the username, on the USB keyboard that is connected to the Rpi. Press `Enter`.
2. Type password as `!Control1` as password. No characters will appear as you type the password. Press `Enter`.
3. There may be occasional `under voltage` warnings that will be shown next to the cursor. Even while shown, the warning is not typed into the terminal. The warning can be safely ignored.
4. FYI. If you require additional terminals on the turtle, which you do not for this guide, you can open them by using `Alt+F2`, `Alt+F3`, all the way to `Alt+F6`. The default is `Alt+F1`.

6.3 Connecting to Non-NUS Wi-Fi or Hotspot

1. Make sure you have logged into the turtle in the previous section.
2. Edit the *netplan* configuration file:

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

   Key in the password `!Control1` when prompted.
3. For the next step, please pay attention to the 4-space indentations.
4. Under `access-points`,

```
access_points:
    "WIFI_SSID":
        password: "WIFI_PASSWORD"
```

a) Replace `WIFI_SSID` with the name of the Wi-Fi or hotspot you are using. The double quotes have to be kept.

b) Replace `WIFI_PASSWORD` with the password. The double quotes have to be kept.

c) You can add another SSID-password pair if you would like to fall back to a secondary Wi-Fi or hotspot when the first one is not available. Simply add the pair below the current pair in the YAML file, keeping the same indentation as the first pair.

5. `Ctrl+S` and `Ctrl+X` to save and exit.

6. Update the configuration using:

```
sudo netplan generate && sudo netplan apply
```

7. Run the following to see if the connection succeeds for `wlan0`:

```
networkctl
```

You should see `routable` (green) and `configured` (green) for `wlan0` when the connection succeeds. You can run the command repeatedly until it connects.

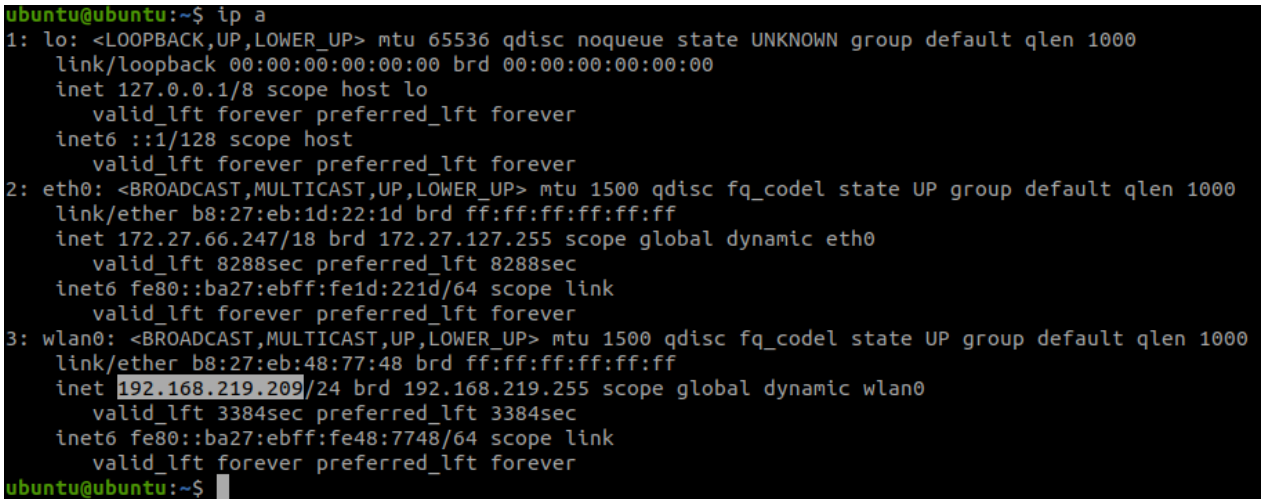
6.4 Turtle IP Address

Find the IP address of the turtle to allow it to communicate with the ROS daemon on the remote PC.

1. Ensure that the turtle is connected to the Non-NUS Wi-Fi or hotspot, as done in the previous section.
2. An alternative to this section is to check the IP address from the chosen Wi-Fi or hotspot. For example, the IP address can be found if an Android phone's hotspot is used.
3. Type:

```
ip a
```

4. Find the IP by locating the interface `wlan0`. The IP address is right beside `inet`, excluding `/24`. Let this be `<TURTLEIP>`.



In the example above, the IP address is `192.168.219.209` and highlighted.

5. You can now unplug the keyboard and the monitor.

7 Running Programs

7.1 Subsequent Startups

If you connect the remote PC and turtle to the same router, `<TURTLEIP>` is unlikely to change. As such, the following procedure is usually valid for subsequent operations:

1. Make sure that the chosen Non-NUS Wi-Fi and hotspot can be connected to.
2. Connect remote PC to the Wi-Fi or hotspot.
3. Turn on turtle. Wait for around 2 minutes to let it connect to the router. You can verify this on your Wi-Fi or hotspot – the turtle's SSID is `ubuntu`, so it will show up as `ubuntu` on the router. The monitor and keyboard does not need to be plugged into the turtle.

7.2 SSH from Remote PC to Turtle

SSH enables you to control the turtle from the remote PC.

1. Make sure that the remote PC and turtle are connected to the same non-NUS Wi-Fi or hotspot.
2. FYI, on a remote PC's terminal and before SSH, you can verify if the robot is on the network by using

   ```
   1 ping <TURTLEIP>
   ```

   If the data packets are received, the turtle is on the network and can be SSH.

3. In a terminal on the remote PC, SSH into the turtle:

```
ssh ubuntu@<TURTLEIP>
```

   Wait a moment, and you should see `ubuntu@ubuntu:~` when the SSH is successful. Let this be called the **SSH terminal**.

4. If the SSH or ping fails with `<TURTLEIP>`,
   a) Check that the robot is turned on.
   b) Check that your Wi-Fi or hotspot is turned on. The Wi-Fi cannot be the NUS Wi-Fi.
   c) Check that the IP address is correct by checking from the Wi-Fi or hotspot, or repeating Sec. 6.
   d) Make sure that the `export ROS_DOMAIN_ID=30` is near the bottom of `~/.bashrc` by using `nano ~/.bashrc`. Add the line if it is not there, save and exit, and run `source ~/.bashrc`.

7.3 Turtle Bringup

1. Prepare the robot in a stationary position. Do not touch the robot for the next step as the Gyroscope in the IMU has to be calibrated.
2. In the SSH terminal (see previous section), bring up the the robot's sensors and peripherals using:

```
. robot.sh
```

   which wraps a launch file. Do not touch the robot.
3. Wait between 5s to 10s for the calibration to finish. You will see two `Run!` on the output when completed.

7.4 Teleoperate Turtle

1. Make sure that the turtle is brought up in the previous section.
2. Robot should be in portable mode. See Sec. 4.1.
3. Open a new terminal on the remote PC, navigate to the `ee4308` folder, and execute

```
. teleop_turtle.sh
```

4. Make sure that the robot is turning left or right as expected and not turning in the wrong direction. If the robot is turning in the wrong direction, you need to inform the teaching assistant.
5. `Ctrl+C` when you finish teleoperating.
6. When the robot stops moving, set it to tethered mode. See Sec. 4.1.

7.5 Running Simulation Mode

The simulation mode enables you to test your code against the estimated motion in simulation. Simulation allows you to validate your code more quickly without going to the lab to test on the playing field.

1. The hardware bring up in Sec. 7.3 **should not be running**.
2. Open `params.sh` in the `ee4308` workspace folder. Find and set `EE4308_TASK` to

```
export EE4308_TASK=proj1sim
```

   There is no space before and after `=`.
3. Make sure that you code is already built with `. bd.sh`.
4. Run with `. run.sh`.

7.6 Running Hardware Mode

Use the hardware mode to test your code with the robot hardware.

1. The hardware bring up in Sec. 7.3 is **required**.
2. Open `params.sh` in the `ee4308` folder. Find and set `EE4308_TASK` to

   ```
   export EE4308_TASK=proj1
   ```

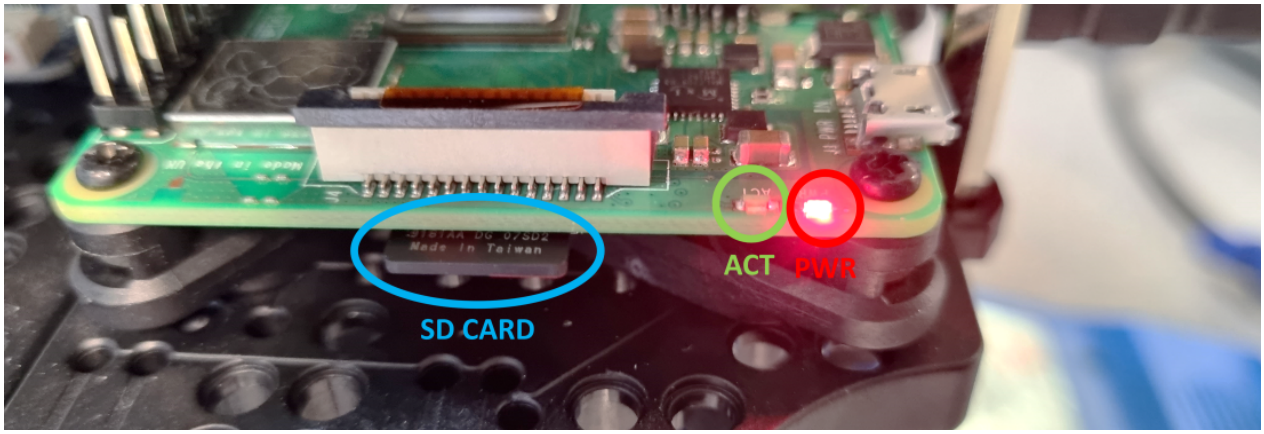
   There is no space before and after `=`.
3. Make sure that you code is already built with `. bd.sh`.
4. The default implementation does not estimate the pose of the robot, and will move perpetually. Make sure that you can interrupt with `Ctrl+C` in time before the robot falls off an edge. It may also take several seconds for the robot to stop moving after the interrupt is sent.
5. Robot should be in portable mode. See Sec. 4.1.
6. Run with `. run.sh`.
7. `Ctrl+C` when you are finished, or the final waypoint is reached. It may take several seconds for the robot to stop moving after sending `Ctrl+C`.
8. When the robot stops moving, set it to tethered mode. See Sec. 4.1.

7.7 Properly Turning Off Turtle

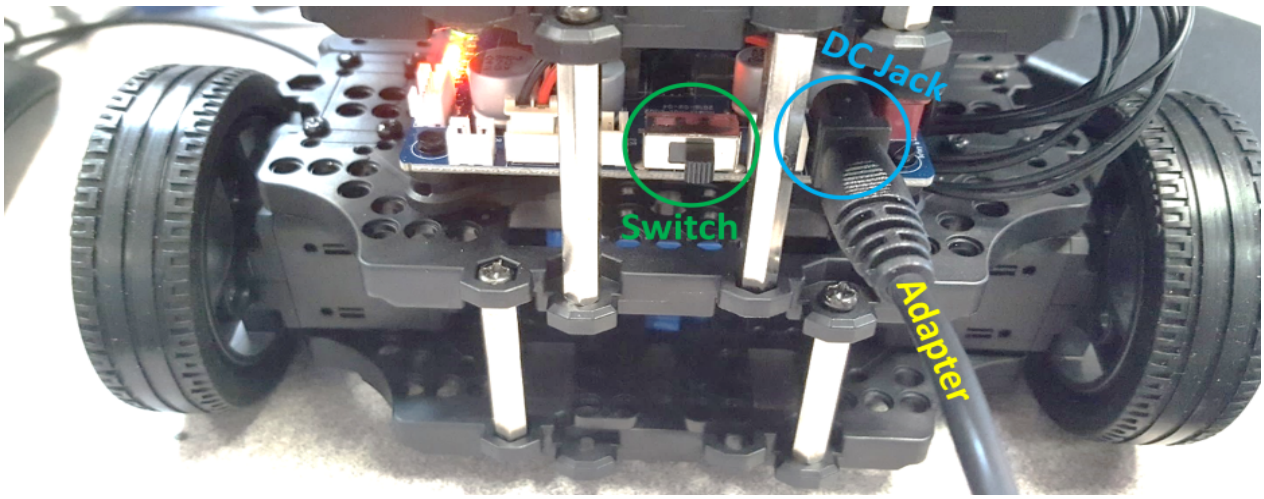
1. On the SSH terminal, shutdown the turtle with

```
sudo shutdown now
```

   Wait for about 30 seconds, or until the green LED `ACT` beside the red LED `PWR` stops flashing (the red LED is always turned on) on the Rpi. *If you are unable to access the turtle and push the command, just proceed to the next step, but try to avoid skipping this step in the future – it is meant to shutdown the turtle properly.*



2. You can proceed to turn off the robot by toggling the switch on the OpenCR board.



8 Tasks

8.1 Demonstration Tasks

1. The turtle has to go through three waypoints 1, 2, and 3, in order.
2. To reach a waypoint, the turtle has to lie $0.1i$ m in real distance from the waypoint $i \in 1, 2, 3$.
3. The final waypoint has to be reached in under 90s.

8.2 Planner Task

1. Open the `planner_smoother.hpp` header file in the `ee4308_turtle` package.
2. Look for the function `run()` (around line 163) in the `PlannerSmoother` class.
3. Convert the Dijkstra implementation in the function to A*.
4. An optional task is to use Theta*.
   a) Use the `RayTracer` private member provided around line 100 (uncomment it).
   b) Initialize the ray tracer with `ray_tracer_.init()` method (see `raytracer.hpp`). The input parameters should be `expanded_node-` (you need to set the parent for the start node) and `neighbor_cell`. Function will return the first cell coordinates.
   c) For every other cell, call `ray_tracer_.next()`.

d) The provided cost map is the inflation layer, and will have multiple costs. For each cell along the line, you can simply add the cost of each cell to get the total cost at the end.

e) If the ray tracer passes through a lethal cost, that is, if it is too close to the obstacle, then the line drawn would not have line-of-sight, and the parent cannot be assigned to the expanded node. This would require the lethal cost to be properly implemented in the inflation zones. See Sec. 8.6.

8.3 Smoother Task

1. Open the `planner_smoother.hpp` header file in the `ee4308_turtle` package.
2. Look for the function `smooth()` (around line 249) in the `PlannerSmoother` class.
3. Implement cubic Hermite splines or the Savitsky-Golay (SG) moving average.

8.3.1 Cubic Hermite Splines

1. Find the turning points along the found path by A* or Theta*. This can be done by iteratively comparing two segments between three points. Let $\mathbf{v}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ for a point $\mathbf{x}_i$ along the found path.

   a) If $\mathbf{v}_i \times \mathbf{v}_{i+1} \neq 0$, a turning point is located at $\mathbf{x}_i$. $\times$ is the two-dimensional cross product.

   b) Otherwise, if the expression is 0, remove (prune) $\mathbf{x}_i$ from the path.

   c) Keep the start and goal points.

2. For each segment on the new path, generate the cubic spline.
    a) The velocity magnitude at each turning point should be provided by the YAML file. The velocity determines the curvature of the spline, and not the robot's velocity at a turning point. The robot's velocity is determined by the controller node.
    b) The velocity direction at each turning point should be the average of the unit directional vectors of both adjacent segments.

8.3.2 Savitsky-Golay Moving Average

The Savitsky-Golay moving average finds the least squares polynomial fit over a window of $2m + 1$ points, where m is the half window size. While the points have to occur at regular intervals, a good-enough smoothing is feasible, and we can assume the points returned from A* to be regular.

1. If using Theta*, interpolate the path so that points are regularly spaced. Some points can be irregularly spaced.
2. Let $\mathbf{J}$ be the Vandermonde matrix, which is a $(2m+1) \times (p+1)$ matrix, where m is the half-window size and p is the polynomial order to be fitted. For example, to fit a quadratic polynomial over seven points, $p = 2$ and $m = 3$.
3. Each element $j_{r,c} \in \mathbf{J}$ is $(-m + r - 1)^{c-1}$, where $r = 1$ is the first row, and $c = 1$ is the first column. For example, for a cubic polynomial fit over five points, the Vandermonde matrix is:

$$\mathbf{J} = \begin{bmatrix} 1 & -2 & (-2)^2 & (-2)^3 \\ 1 & -1 & (-1)^2 & (-1)^3 \\ 1 & 0 & 0 & 0 \\ 1 & -2 & (1)^2 & (1)^3 \\ 1 & -1 & (2)^2 & (2)^3 \end{bmatrix} \tag{1}$$

4. The weighted coefficients over the window is the first row of $\mathbf{a}$, where

$$\mathbf{a} = \left(\mathbf{J}^\top \mathbf{J}\right)^{-1} \mathbf{J}^\top, \tag{2}$$

and $\mathbf{a}$ is a $(p+1) \times (2m+1)$ matrix. Let the first row be $[a_{-m}, a_{-m+1}, \cdots, a_0, \cdots a_m]$

5. For a point $\mathbf{x}_j \in \{m, m+1, \cdots, n-m\}$ in a path containing n points, iteratively multiply the first row of $\mathbf{a}$ over the surrounding points $[\mathbf{x}_{j-m}, \mathbf{x}_{j-m+1}, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_{j+m}]$, such that the smoothed point $\mathbf{s}_j$ is

$$\mathbf{s}_j = a_{-m}\mathbf{x}_{j-m} + a_{-m+1}\mathbf{x}_{j-m+1} + \cdots + a_0\mathbf{x}_j + \cdots + a_m\mathbf{x}_{j+m} \tag{3}$$

6. The remaining points that lie on the side can be found by copying and appending $\mathbf{x}_0$ or $\mathbf{x}_n$. For example, if $m = 2$ and $j = 1$, $\mathbf{s}_1$ can be found as $a_{-2}\mathbf{x}_0 + a_{-1}\mathbf{x}_0 + a_0\mathbf{x}_1 + \cdots$. If $m = 3$, and $j = n-1$, $\mathbf{s}_{n-1}$ can be found as $\cdots + a_0\mathbf{x}_{n-1} + a_1\mathbf{x}_n + a_2\mathbf{x}_n + a_3\mathbf{x}_n$.

8.4 Controller Task

This section aims to enhance the Lab 2 pure pursuit task.

1. Open the `controller.hpp` header file in the `ee4308_turtle` package.
2. Look for the function `moveRobot()` (around line 317) in the `ROSNodeController` class.
3. Implement the pure pursuit from Lab 2 in the function.
4. Further enhance pure pursuit by allowing the robot to reverse if the waypoint lies at the back of the robot, such that x' is negative. Refer to the lecture slides to calculate x'.
5. Optionally enhance pure pursuit with the features mentioned in the lecture slides. You can read up https://arxiv.org/abs/2305.20026 (Regulated Pure Pursuit for Robot Path Tracking). There may be some errors in the paper that are corrected in the lecture slides.

8.5 Estimator Task

This section aims to implement the Lab 1's state estimator.

1. Open the `estimator.hpp` header file in the `ee4308_turtle` package.
2. Look for the function `run()` (around line 63) in the `ROSNodeEstimator` class.
3. Implement the state estimator from Lab 1 in the file.
4. Tune the IMU gains in `proj1.yaml` file by monitoring the average error at the waypoints in the hardware mode.

8.6 Optional Miscellaneous Tasks

1. **No points will be awarded to miscellaneous tasks if the main tasks are not completed.**
2. It is computationally expensive to keep requesting for a path by the controller. Using a new or existing ROS2 service, plan a path only if the existing path has crossed into a cell on the inflation layer that has a lethal inflation cost (i.e. is too close to an obstacle).
3. By treating the turtle as a cylinder, use a lethal cost inflation radius with exponential cost decay. See Fig. 13 in https://arxiv.org/abs/1706.09068 (ROS Navigation Tuning Guide). In `mapper.hpp` header file in the `ee4308_turtle` package, search for `inflation_mask_` to investigate how the task can be implemented.
4. You may refer to https://arxiv.org/abs/2307.15236 (From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2) for more ideas.

9 Future Announcements and Updates

1. Keep a lookout for an announcement to update some parameters related to the playing field in the `proj1.yaml` file.
2. If there are 3-man teams, an additional announcement will be made that includes any adjustments to the tasks for 3-man teams.
3. There may also be errors in the code or the manual. Keep a lookout for these on Canvas Discussions.

10 Submissions

10.1 Report (PDF)

1. Deadline: W7 Sun, 10 Mar, 23:59h
2. `.pdf` type.
3. Name the report as `team##.pdf`, where `##` is the team number in two digits (e.g. Team 4 is `team04.pdf`).
4. Submit the report to `Canvas` → `Assignments` → `P1R`.

10.2 Compiled Video (MP4)

1. Deadline: W7 Sun, 10 Mar, 23:59h
2. The compiled video must:

a) Be at most 7-min long.

b) Contain a 5-min presentation of the project, detailing the report succinctly.

c) Contain a 2-min maximum, demonstration video of the robot moving in the physical area, and played back side-by-side a screen recording of the terminal outputs and RViz. Put the video after the 5-min presentation.

d) You may put the demonstration video and screen recording together in PowerPoint, and record the playback together with Zoom for a maximum 7-min long video.

3. Name the video as `team##.mp4`.
4. Submit the video to `Canvas` → `Assignments` → `P1P`.
5. `.mp4` type.

10.3 Code (ZIP)

1. Deadline: W7 Sun, 10 Mar, 23:59h
2. Zip the `ee4308/src` folder.
3. Again, only the `src` folder, not the `ee4308` folder.
4. Name the zip file as `team##.zip`.
5. Submit the zip file to `Canvas` → `Assignments` → `P1C`.