v2023.02

THE UNIVERSITY OF SYDNEY

MTRX3760
Turtlebot User's Guide

For this unit of study your group will be granted exclusive access to a Turtlebot3 mobile robotic platform: https://tribotix.com/product/turtlebot3-burger/.

This is a sophisticated piece of equipment, and you are responsible for keeping the hardware in good working order. On return day at the end of semester, the state of your Turtlebot will be evaluated. Loose boards and connections may result in lost marks, damaged or incomplete hardware will result in your marks being withheld until the turtlebot is repaired and returned. Because you are responsible for your turtlebot, it is very important to inspect the one you receive and signal any damaged or missing hardware immediately to your tutor.

If you have constructed your Turtlebot from scratch, please follow sections marked with 🤖. Feel free to skip these sections otherwise.

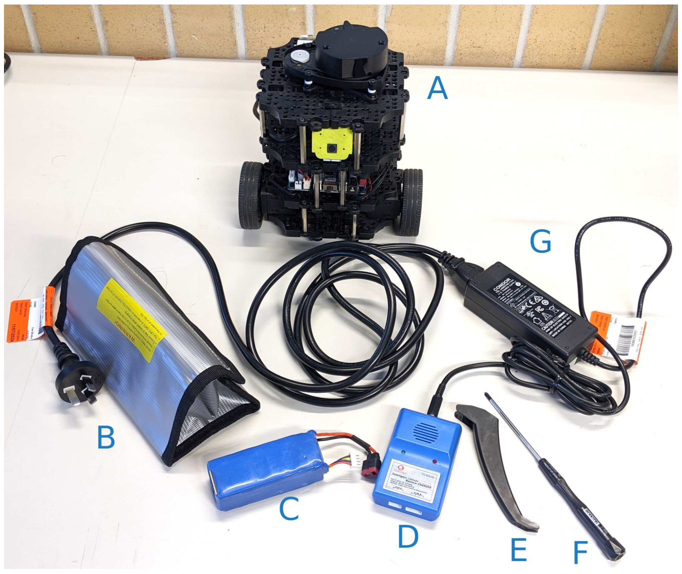
If you received an already-assembled Turtlebot, you may need to disassemble and reassemble parts of it, especially if any of the PCBs, LiDAR, or connectors are loose. If you damage your turtlebot, you should carry out repairs. Replacement parts can be obtained from the Tribotix website linked to above. They sell both complete kits and replacement parts.

On boot, your Turtlebot may not automatically connect to the KIRBYBOTS WiFi in the lab with the fixed IP address. We have installed additional hardware in the form of a small “hat” which allows you to switch the turtlebot into an access point mode. Consult with the instructions below and seek your tutor’s assistance for correctly configuring your Turtlebot’s network connection on KIRBYBOTS. Do not change your turtlebot’s password.

Should you have issues with running ROS programs, you may need to update the OpenCR board firmware. Please see the relevant section below.

Kit Contents

When you receive your turtlebot you will receive the parts depicted below. It is your responsibility to confirm receipt of all parts. You will be liable for any missing parts at the end of the semester.



A. Turtlebot 3 with LiDAR and camera module
B. LiPo battery fire safety bag
C. LiPo battery
D. Balanced battery charger
E. Rivet pulling tool
F. Screwdriver
G. Switched Mode Power Supply

Battery Care

The Turtlebot battery is a lithium-polymer (LiPo) battery and must be treated with care. LiPo batteries are dangerous. Please follow these guidelines to avoid risk of fire or damage to the battery or Turtlebot :

- The provided balanced charger can safely charge the battery, but only if the battery is first disconnected from the Turtlebot. Do NOT attempt to charge the battery while it is still connected to the Turtlebot. Only one of the battery's two connectors should ever be connected at one time.
- To power a Turtlebot from the wall, rather than battery: 1) disconnect the battery from everything; 2) disconnect the balanced charger from the Switched Mode Power Supply (SMPS), and plug the SMPS directly into the OpenCR board on the Turtlebot. Please note the motors will not be able to actuate when attached to the wall.
- Do not leave a battery connected to a Turtlebot while not in use. The battery will slowly discharge and eventually run completely flat. A completely flat LiPo battery cannot be charged again, and it will be your responsibility to safely dispose of the old one and obtain a replacement.
- Do not leave a battery charging unattended. Charging batteries can and have swollen, heated, and caught fire. This process takes time, and if caught early can be stopped by removing power.
- Be careful to never scratch or puncture a battery.
- Store batteries in LiPo fire safety bags when not in use and while charging.
- If the Turtlebot starts beeping while on battery power, this indicates the battery is low and should be recharged.

Background and Manual

Before using the turtlebot, we strongly recommend completing the ROS tutorials up to and including Beginner Level Tutorial 20 at http://wiki.ros.org/ROS/Tutorials.

Acquaint yourself with the Turtlebot 3 platform characteristics here:
https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Turtlebot Version: Burger
Turtlebot OS: Botbuntu custom Ubuntu image
ROS (OS) Version: Noetic (Native for Ubuntu 20.04)

2023 SDCard Image Update

Reflash SDCard

This year there is a new SDCard image available. Students should all reflash their turtlebot's SDCard following the instructions in Appendix A (search for "SDCard Burning"), making sure to select the most recent image from AARnet, `turtlebot3_image_2023.img`.

Changes to Packages

With the SDCard image update there is a change to how we use the camera on the physical turtlebot platform. The camera should now be launched with the command

```
roslaunch turtlebot3_bringup turtlebot3_camera.launch
```

This is a change from the previous launch file which was called `turtlebot3_rpicamera.launch`. The node being launched is documented here: http://wiki.ros.org/cv_camera. By default the `cv_camera` node only publishes raw images. Compressed images are automatically added by the `image_transport_plugins` http://wiki.ros.org/image_transport_plugins which are installed as part of the new image.

The new launch file launches a camera node under the namespace `/camera` and images are published under the topics `/camera/image_raw` and `/camera/image_raw/compressed`. Raw images are published as message type `sensor_msgs/Image` while the compressed images are published as message type `sensor_msgs/CompressedImage`. Subscribing to the raw images is slower because the images are larger, yielding a maximum frame rate of about 6 Hz. Subscribing to the compressed images yields a maximum frame rate close to 30 Hz.

THE UNIVERSITY OF SYDNEY

Network Configuration

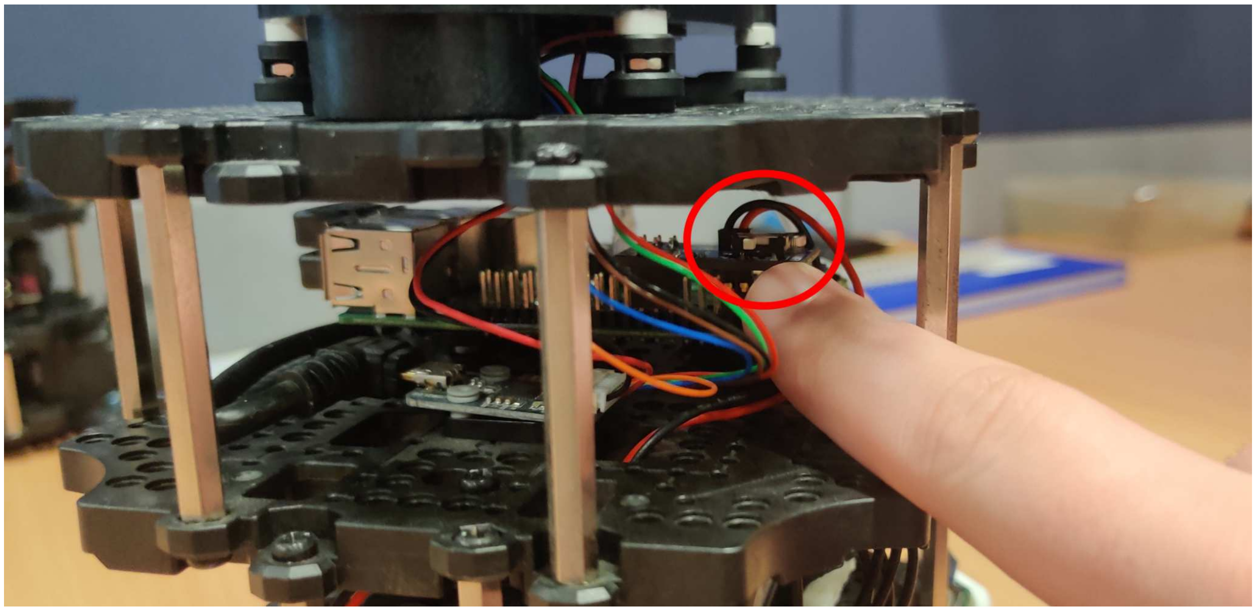
The turtlebot has two modes of network connection, selected using a switch on the “AP Hat” connected to the Raspberry Pi:

- **WIFI MODE:** The turtlebot connects to a known WiFi signal, with the turtlebot taking on a fixed IP address, allowing you to remotely connect to the turtlebot at that IP address, or

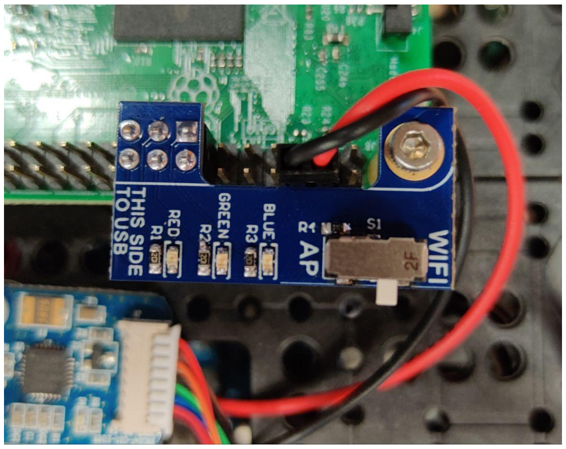
- **ACCESS POINT MODE:** The turtlebot acts as an access point (WiFi hotspot) with an automatically generated access point name. This allows you to connect directly to the turtlebot. The disadvantage of this method is that you will not have internet access while connected to the turtlebot WiFi signal.

You can select the appropriate mode using the AP Hat switch, shown below. In Access Point Mode, the green LED will illuminate once the Access Point is ready to connect. A flashing green LED is displayed in WiFi mode when connected. If no connection is available the green LED remains off.

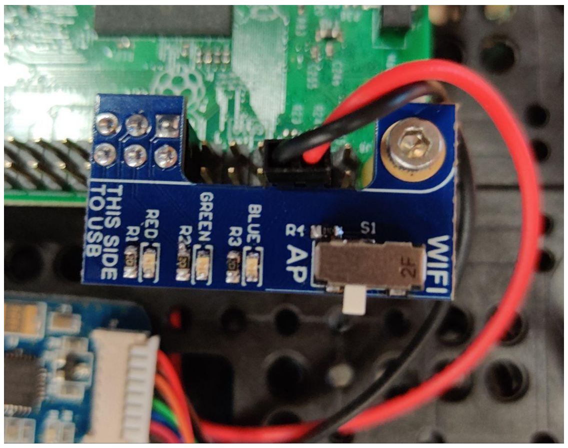
When first connecting to the turtlebot you need to use the Access Point method. This will allow you to configure the WiFi Mode connection.



Switch location on-board Turtlebot to configure the network. Note the switch is on the opposite end of the Raspberry Pi to the USB ports, and is mounted onto the GPIO pin header.



Turtle Bot in Wifi Mode



Turtlebot in Access Point Mode

Access Point Mode

When in Access Point mode, the turtlebot appears as a WiFi network that you can connect to from any WiFi-enabled device. The AP Hat's green LED will illuminate when the access point is ready. The access point will take on a name of the form `<MAC>pi`, where `<MAC>` is the last 8 characters of your turtlebot's MAC address, e.g.

```
SSID: eb09ba97pi
Password: turtlebot
```

The relevant information is printed on the top of your turtlebot. Once connected you can remotely connect to the turtlebot via SSH. The Turtlebot IP in AP mode is always set to 10.42.0.1, so for example:

```
$ ssh ubuntu@10.42.0.1
Password: turtlebot
```

Configuring for WiFi Mode

Follow these instructions to connect the turtlebot to either the lab's WiFi KIRBYBOTS network or your home network. The turtlebot comes preloaded with a WiFi configuration script that will prompt you to enter the connection information for your network.

First connect to the turtlebot in AP mode and SSH into it. Then run the script to configure a network. The following is an example of using the script to configure the KIRBYBOTS network:

```
$ python3 /home/ubuntu/add_wifi.py
---------------
Enter SSID: KIRBYBOTS
Enter Password: room330!
Static IP Address: 172.17.33.xx  (Use the IP address printed on your turtlebot)
---------------
Network configured:
      wlan0:
            dhcp4: yes
            dhcp6: yes
            addresses: [172.17.33.xx/32]
            optional: true
            access-points:
                  KIRBYBOTS:
                        password: room330!
---------------
```

This will add KIRBYBOTS to the list of available networks. When configuring for the KIRBYBOTS network it is important to use the IP address printed on your turtlebot.

To configure your turtlebot for use on a home network, use the same script and assign a static ip address of your choice. Note that it needs to lie in the correct domain for your home network, e.g. 172.17.yy.xx, or 192.168.yy.xx, or 10.42.0.xx. You can check this by checking your computer's IP address when connected to the same network, e.g. by running ifconfig from a terminal.

Running the configuration script adds network connections to a persistent list of available connections on the turtlebot. When adding multiple configurations, the turtlebot will attempt to connect all of them when in WiFi mode.

In case something goes wrong and you wish to purge the list of known WiFi networks and return to a base working state, you can execute the below script to clear the network configuration

```
$ python3 /home/ubuntu/reset_wifi.py
```

If you accidentally enter incorrect network details, you can also edit the list of known connections without having to clear the whole list by editing the network configuration file using the command

```
$ nano /home/ubuntu/.mtrx/wifi.yaml
```

The netplan examples will help you to understand what is found in this file: https://netplan.io/examples.

Once a WiFi configuration is entered you can switch the mode from access point to WiFi using the AP Hat switch. The turtlebot will then attempt to connect to the most recently added WiFi connection. If this does not work, try reconnecting in AP mode and running “python3 /home/ubuntu/reset_wifi.py”, then re-attempt the WiFi connection.

Once the turtlebot is connected to a WiFi network, the green LED should blink indicating successful connection. With your own device connected to the same network as the turtlebot, you can ssh into your turtlebot using

```
$ ssh ubuntu@172.17.33.xx
password: turtlebot
```

You should now have set up the turtlebot network and remotely connected to it.

Note: If using multiple turtlebots you will need to use the WiFi mode, not the Access Point mode.

ROS / Hardware Bringup and Connecting via Remote PC

Once the turtlebot is set up and connected to a network, SSH into your Turtlebot from one of the lab PCs using the command

```
ssh ubuntu@<ip address>
```
e.g.
```
ssh ubuntu@172.17.33.250
```
password: `turtlebot`

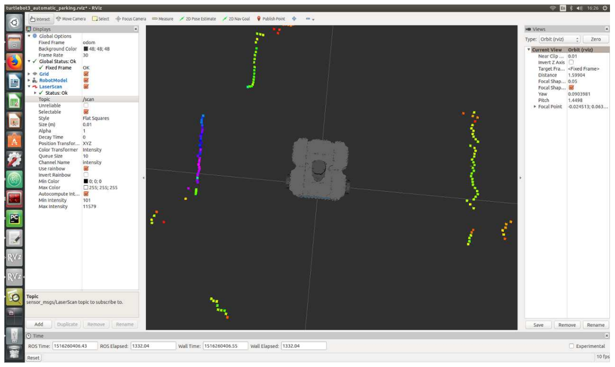
You may avoid having to type the SSH password every time by issuing the command

```
ssh-copy-id ubuntu@<ip address>
```

and entering the password. Subsequent ssh attempts should succeed without a password.

Follow the instructions in Section 3.1 of the Turtlebot e-manual to configure the ROS network. Follow the example in the e-manual so that the ROS master runs on the remote PC (the Lab PC). Note that there are required settings on both the Turtlebot and the remote PC.

Follow the instructions in Sections 3.5 and 3.6 to bring up your Turtlebot, establish basic functionality, visualise Turtlebot topics and control the Turtlebot from a remote PC. Section 10 shows an example of using RViz.



Global Options	
Fixed Frame	odom
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	✓
LaserScan	✓
✓ Status: Ok	
Topic	/scan
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfor...	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0
Max Color	☐ 255; 255; 255
Autocompute Int...	✓
Min Intensity	101
Max Intensity	11579

Topic
sensor_msgs/LaserScan topic to subscribe to.

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	1.59904
Focal Shap...	0.05
Focal Shap...	✓
Yaw	0.0903981
Pitch	1.4498
Focal Point	-0.024513; 0.063...

ROS Time: 1516260406.43 ROS Elapsed: 1332.04 Wall Time: 1516260406.55 Wall Elapsed: 1332.04

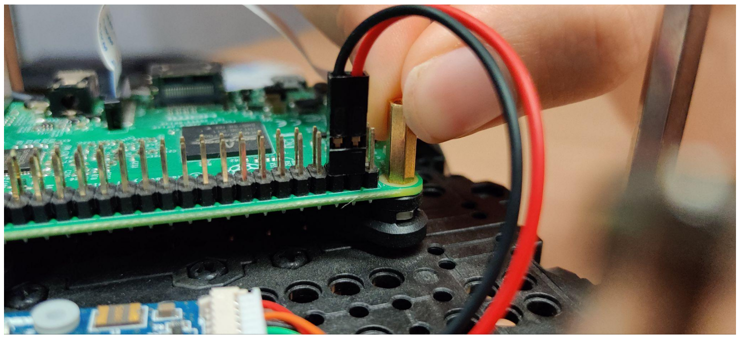
Appendix A: Configuring a New Turtlebot

The following instructions are for those building their own turtlebots.

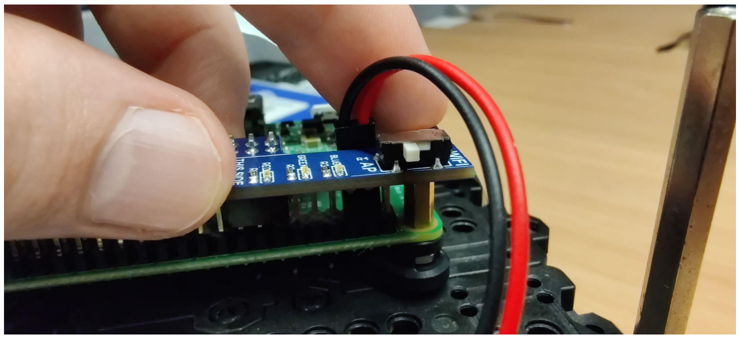
🤖 AP Hat Installation

If you have constructed your Turtlebot from scratch, we have additional custom hardware that you will need to install to enable the additional networking capabilities built especially for this unit of study. Please ask your tutor for a new AP Hat circuit board and required parts.

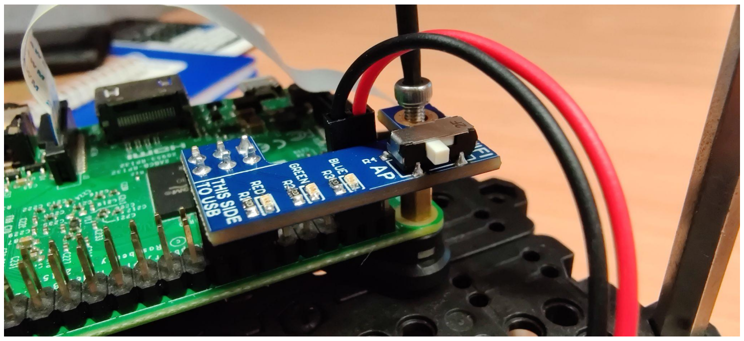
1) Screw in a hex standoff as shown here:



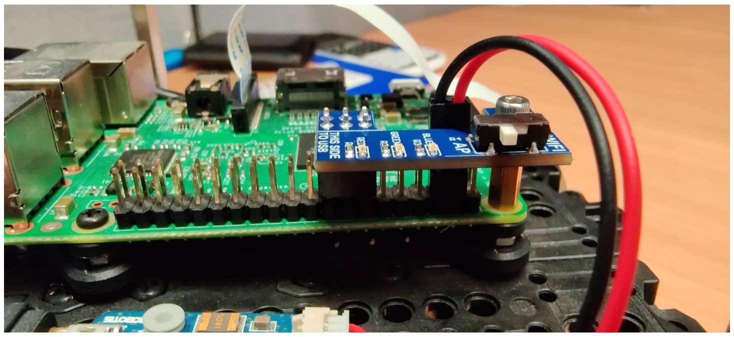
2) Insert the AP Hat onto the GPIO pins, ensuring the mounting hole lines up with the standoff. If these do not line up, the AP Hat will connect the wrong pins and you risk shorting the board.



3) Fix AP Hat in place with screw onto standoff. Careful not to over-tighten the screw.



4) Ensure the board is seated correctly and securely but not over-tensioned or warped.



🤖 Raspberry Pi and OpenCR Setup

If you have constructed your Turtlebot from scratch, there are additional setup steps. You will find the required Raspberry Pi Linux image and OpenCR firmware update here: https://cloudstor.aarnet.edu.au/plus/s/RXxLdJeoddnACd0. Please note we are using a custom Linux image for the Raspberry Pi and so you must use this link to acquire the image for your turtlebot. For the OpenCR firmware, the AArnet link will be faster than using the links in the manual.

Linux Install

For getting the Raspberry Pi single-board computer (SBC) and OpenCR robot control board running, follow the instructions in Section 3.2 of the e-manual. Note that we are running a custom Ubuntu 20.04 Linux based OS “Botbuntu”, use the image provided at the link above, not a Linux version sourced from the Robotis website. You can still follow the instructions in Section 3.2.1.2 while using the custom image.

SDCard Burning

For burning the RPi SDCard, "etcher" is recommended in the instructions, but this is slow to install. Use `dd` instead following https://elinux.org/RPi_Easy_SD_Card_Setup#Using_the_Linux_command_line .

Do be careful with `dd`, it can wipe your hard drive if misused. Note that `dd` can appear to have stalled at the end of the burn process, sitting idle for up to 5 minutes, but it is still active and is simply flushing cached writes. Be patient. You may install and use `iotop` to establish whether there is still activity writing to the SDCard. Establish if there is activity before removing from the machine, otherwise you will need to begin the process over again.

OpenCR Firmware
You will need to write firmware to your OpenCR board. Follow instructions in Section 3.3 of the e-manual. This step is flashing a microcontroller with code and any interruption will cause issues - do not remove cables or power during this process. Once the OpenCR board is flashed, use the buttons installed on the OpenCR board to test the motor functionality following the instructions in the e-manual.

sudo access
Some of the steps above require the superuser (sudo) password, e.g. burning the sdcard and running the OpenCR firmware upgrade. If you are running on the lab machines, please ask your instructor for sudo access at each of these steps. Alternatively, complete these steps on your own computer. Some steps like the OpenCR firmware upgrade can be completed from the raspberry pi computer on the turtlebot once you've got the SDCard running. Remember, the raspberry pi is a linux computer, and you have sudo access. Username: ubuntu, password: turtlebot.

🤖 Hostname and Labelling

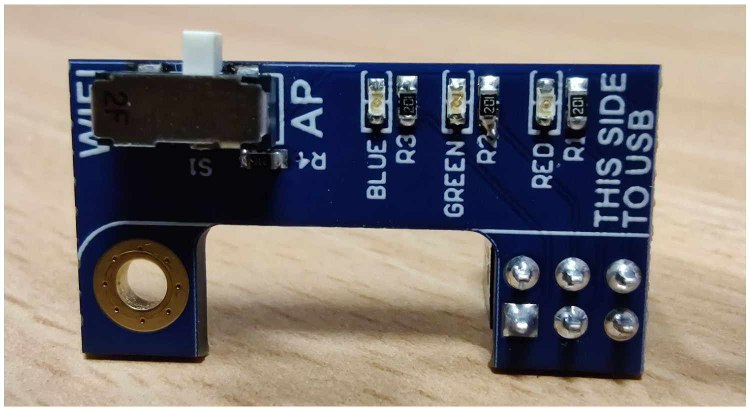
The hostname for the Raspberry Pi is automatically set at boot time, and consists of the last 8 characters of the WiFi adapter MAC address on-board the Raspberry Pi.

Please ask your teaching team for assistance in appropriately labelling your turtlebot with its SSID, etc.

You do not need to do anything to configure the hostname, and should not change it. Changes will not persist between reboots. Be aware that if you change the Raspberry Pi, the hostname of the Turtlebot also changes. If you believe the hostname is incorrect, try changing the hostname deliberately and rebooting to trigger an update. Failing this, reflash the image.

Appendix B: Using the AP Hat Status LEDs

The AP Hat is custom built and designed to ease network configuration, but it also offers two status LEDs that you can control as part of your project.



The board has 3 status LEDs (Red, Green and Blue) and a toggle switch. The Green LED is reserved to indicate network status, the other two are intended for use by you should you need additional status checks during your projects.

The red and blue status LEDs are connected to GPIOs 22 and 27 (or board pins 13 & 15). A summary of the pins is given below. You will note that two indices are given to each pin - there is a sequential board number of each pin from the top of the GPIO to the bottom, and a numbering used by the system which corresponds to the interface on the processor (referred to as **Broadcom SOC channel** or BCM numbering). Ensure nothing except the two power leads is installed in the GPIO if you are unsure of which pins are going to be actuated to avoid damaging any components or worse your Raspberry Pi. A comparison of the two labelling schemes can be found here: https://pinout.xyz/

We introduce this as C++ interfaces will often need the BCM pin number, while some Python packages allow you to use the board number.

Component	BCM Pin	Board Pin	Available for use?
Red LED	22	13	Yes
Green LED	23	16	No
Blue LED	27	15	Yes
Switch	24	18	No

Example Code to Drive LEDs

We provide a pre-compiled binary which flashes the LEDs like an ambulance (it is imperative you make audible "WEEE-AWWW-WEEE-AWWW…" noises as you drive around in this mode). You can use this to check the red and blue LEDs are operational. After connecting via SSH to the turtlebot, run the following:

```
$ sudo ~/.mtrx/led_test
```

Below is the source code for an example that you can modify for your own use. This makes `libgpiod-dev` on-board the turtlebot to control the GPIO pins. It will turn the on the blue LED and then the red LED for 2 seconds each. Save the below as `led_blink.cpp` and remember to link the C++ bindings of `libgpiod-dev` when compiling, as in

```
$ g++ led_blink.cpp -o led_blink -lgpiodcxx
```

```
#include <chrono>
#include <iostream>
#include <thread>

#include <gpiod.h>
#include <gpiod.hpp>

using namespace std::literals::chrono_literals;

const int bLedPin = 27;
const int rLedPin = 22;

int main (void)
{

  std::this_thread::sleep_for(2000ms);

  // Set-up the GPIO Chip
  gpiod::chip chip("gpiochip0");

  // Configure the blue "line"
  auto line1 = chip.get_line(bLedPin);
  line1.request({"blue_led",
             gpiod::line_request::DIRECTION_OUTPUT,
             0}, 0);
  // Now the red
  auto line2 = chip.get_line(rLedPin);
  line2.request({"red_led",
              gpiod::line_request::DIRECTION_OUTPUT,
              0}, 0);

  // Turn on the blue LED and turn off red LED for 2 seconds,
  line1.set_value(1);
  line2.set_value(0);

   std::this_thread::sleep_for(2000ms);

   // Turn on the red LED and turn off the blue LED for 2 seconds.
   line2.set_value(1);
   line1.set_value(0);

  std::this_thread::sleep_for(2000ms);

  return 0;
}
```