Robot Usage Guide [*]

EECS/ME/BIOE C106A/206A Spring 2023

Contents

[*]Developed by Riddhi Bagadiaa, Fall 2022. Adapted for Spring 2023 by Han Nguyen

1 Logistics and Lab Safety

We expect that all students in this class are mature and experienced at working with hardware, so we give you much more leeway than in 106A. Please live up to our expectations.

1.1 Robot Reservations

Robots should be reserved on the robot calendars:

- Sawyer Robot Calendar
- TurtleBot Calendar

The rules are:

1. **Groups with a reservation have priority.** You can go in and use (or continue using) the hardware whenever you like if no one has a reservation. However, you must pass along the hardware once the next reservation starts. Please be respectful and try to plan ahead; it's not cool to take the first twenty minutes of another group's section winding down.
2. **Do not reserve more than two hours at a time per group.** This is shared hardware and we want to ensure that all groups have enough time to complete the projects.
3. **One computer per group (if needed).** Please try to limit yourself to one computer per group. In addition, groups with robot reservations have priority on the computers next to their respective robots. Groups can accomplish work in parallel by using personal computers, file-sharing software like Git, and/or using simulators like Gazebo or RViz.

1.2 Lab Safety

Remember to follow the lab safety rules:

1. **Never work on hardware alone.** Robots are potentially dangerous and very expensive, so you should always have someone on hand to help if something goes wrong. This person should be another 106B student, though in cases when group schedules are highly incompatible, you may talk to a TA to arrange an alternate solution, such as bringing a friend (note that you would be entirely responsible for this person's conduct).
2. **Do not leave the room while the robot is running.** Running robots must be under constant observation.
3. **No food or drink (except water) in the lab.** We already have an ant problem, so do not bring anything into the lab that may attract more. Keep water in a container with a lid and drink it away from the computers and robots.
4. **Keep the walkways clear.** Make sure your fellow classmates can safely move around the lab and access the robots with ease. Do not place anything on the floor that may trip or block someone.
5. **Always check the trajectory before executing MoveIt! trajectories.** You can view a trajectory in RViz. Sometimes MoveIt! plans extremely strange trajectories, and checking them before running them can prevent damage to the robot. If a trajectory looks strange in RViz, do not execute it on the robot. Try giving the robot a different start or end position and then replan the trajectory.
6. **Always operate the robot with the E-Stop within reach.** Be ready to press the E-Stop button at a moment's notice to prevent the robot from crashing into anything. If a robot is going to hit a person, the wall, or a table, press the E-Stop. Luna wears a parka when it is cold, but not everyone has such protective padding around them for safety. Be particularly careful of people who may be walking through the aisle between the Rethink robots and computers.
7. **Terminate all processes before logging out.** Because the lab workstations are shared, we sometimes run into strange bugs. The most common is that someone leaves a process running when they leave the lab, sometimes by forgetting to log out properly. When the next person uses the same workstation, ROS can get confused by the additional processes running from the prior user. A particularly insidious example is leaving a `roscore` master node running in the background; the next person will not be able to run a master node with proper communications. To avoid issues like this in the lab, please do the following before leaving:

- Ctrl+C out of every terminal before closing it. It is critical that you Ctrl+C. **Do not Ctrl+Z**. Ctrl+Z may look like it stops your process, but it really only pauses it. The process will continue to run in the background.
- Instead of simply logging out of your account, you should type

  ```
  pkill -u [username]
  ```

  into your terminal, where `[username]` is your instructional account username (`ee106b-xyz`). This will close out every process on your account before logging you out.

  You can also use

  ```
  ps -ef | grep ros
  ```

  to check your currently running processes.

8. **Do not modify robots without consulting lab staff.** In previous semesters we had problems with students losing gripper pieces and messing with TurtleBots. This is inconsiderate and makes things much more difficult for everyone.
9. **Tell the course staff if something is broken.** This is an instructional lab, and we expect a certain amount of wear and tear to occur. However, if we don't know something is broken, we cannot fix it.
10. **If you are unsure of how to do something, ask someone on course staff.** We are all here to learn, so don't be afraid to ask for help. Operating the hardware without knowing exactly what you are doing can be extremely dangerous, so please ask for help whenever you have questions about the hardware.

2 Setting Up Your Environment

To set up your environment to automatically use ROS and interface with the robots, you need to edit your `~/.bashrc` file. The `~/.bashrc` file is a script that is run when a new terminal window is started. Adding the commands in this section to the end of your `~/.bashrc` file will automatically set up ROS and the environment variables for connecting to the robots in every terminal window you open. Be careful with which commands you include and in what order; some may override or interfere with one another. This is why we suggest you add the commands in this section at the end of your default `~/.bashrc` file. A separate set of commands are required to set up Turtlebots and Sawyers. Make sure you comment out the Turtlebot commands when using the Sawyers and comment out the Sawyer commands when using the Turtlebots. As a reference, your `~/.bashrc` file should look like this. You can directly copy this file and replace the contents of your own, **but you must make sure the proper edits are made.**

2.1 Turtlebot

2.1.1 Setting up ROS

The first additional command you should add to the end of your `~/.bashrc` file is for setting up ROS.

```
source /opt/ros/noetic/setup.bash
```

to set up only ROS. More specifically, this line tells Ubuntu to run a ROS-specific configuration script every time you open a new terminal window. This script sets several environment variables that tell the system where the ROS installation is located. It also adds the directories for all of ROS's built-in packages to the package path.

2.1.2 Defining the Turtlebot model

Turtlebot 3 is available in two types of models: Burger and Waffle Pi. We will be using the Burger model, so we must specify it to the environment by adding this line

```
export TURTLEBOT3_MODEL=burger
```

2.1.3 Setting the ROS hostname

After the command to set up ROS, you can set the ROS hostname for your workstation in the lab. This node environment variable sets the declared network address of a ROS node or tool.

The ROS hostname of the workstation computer's `bashrc` file will be the the hostname of the computer you are using. Adding the following line will set the ROS hostname to the current computer you are using

```
export ROS_HOSTNAME=192.168.1.[COMPUTER_NUMBER]
```

where `[COMPUTER_NUMBER]` is your computer number (1 through 12). This sets the ROS hostname to the specific computer that you are working on.

The ROS hostname for each Turtlebot is already set correctly to its value. You shouldn't need to change this, but you can confirm that it is true as a debugging point if you run into an issue.

Not Required: You can `ssh` into the Turtlebot's environment and view the ROS hostname of the Turtlebot in the `bashrc` file. Run the following commands on the terminal to do so

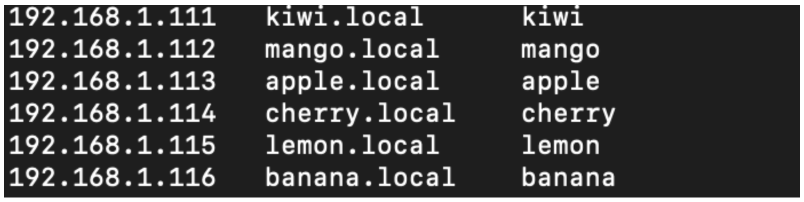
```
ssh [FRUITNAME@FRUITNAME]
cat ~/.bashrc
```

where [FRUITNAME] is the name of the Turtlebot and the password is [FRUITNAME]2022.

In the `bashrc`, **DO NOT CHANGE ANYTHING** but confirm this line is true

```
export ROS_HOSTNAME=192.168.1.[T3_IP]
```

where [T3_IP] can be found in the list below



```
192.168.1.111   kiwi.local      kiwi
192.168.1.112   mango.local     mango
192.168.1.113   apple.local     apple
192.168.1.114   cherry.local    cherry
192.168.1.115   lemon.local     lemon
192.168.1.116   banana.local    banana
```

You can view this list by running

```
cat /etc/hosts
```

in the terminal.

2.1.4 Setting the ROS master URI

If you are running ROS nodes on a robot, you may also need to change the ROS master URI (Uniform Resource Identifier) **on your workstation only**. This is a hostname:port combination that tells nodes where they can locate the master node. In the case of Turtlebots, we set the ROS master as the workstation computer. To do this, add the following line to your `~/.bashrc` file

```
export ROS_MASTER_URI=http://192.168.1.[COMPUTER_NUMBER]:11311
```

2.2 Rethink Robot (Sawyer)

2.2.1 Setting up ROS

The first command you need to add to your `~/.bashrc` file to set up ROS for Sawyer is

```
source /opt/ros/eecsbot_ws/devel/setup.bash
```

which sets up ROS and its built-in packages like the `source /opt/ros/noetic/setup.bash` command does for Turtlebots, but additionally edits the `ROS_PACKAGE_PATH` environment variable which tells ROS which directories to search for software packages. Any code you want to run with ROS must be located beneath one of the directories specified in the list. You want to be able to use the Robot SDK packages in addition to ROS's built-in packages, and using this line in the `~/.bashrc` file enables you to do so.

It is good practice to only import the ROS packages that you need, so you should only use this line if you are using the Robot SDK packages as to not clutter your workspace with the things associated with the Robot SDK.

2.2.2 Setting the ROS hostname

After the command to set up ROS, you can set the ROS hostname for your workstation in the lab. This node environment variable sets the declared network address of a ROS node or tool, and you usually want this to be the hostname of the computer you are using. Adding the following line will set the ROS hostname to the current computer you are using

```
export ROS_HOSTNAME=192.168.1.[COMPUTER_NUMBER]
```

where `[COMPUTER_NUMBER]` is your computer number (1 through 11). This sets the ROS hostname to the specific computer that you are working on.

2.2.3 Setting the ROS master URI

If you are running ROS nodes on a robot, you may also need to change the ROS master URI (Uniform Resource Identifier) **on your workstation only**. This is a hostname:port combination that tells nodes where they can locate the master node. Setting this environment variable to point to a robot will allow you to run ROS nodes on the robot remotely by setting the ROS master as the robot's computer rather than your workstation computer. To do this, add the following line to your `~/.bashrc` file

```
export ROS_MASTER_URI=http://[RobotName].local:11311
```

2.3 Sourcing the .bashrc file

When you're done editing the `~/.bashrc` file, save and close the `~/.bashrc` file and re-run it by executing the command

```
source ~/.bashrc
```

in each existing terminal that you want to be updated with the new settings. The `~/.bashrc` file is only automatically run when a new terminal window is started, so you have to run this command to manually "source" (run) the `~/.bashrc` file again in order to update an existing terminal. This will only update the terminal window you run the command in, so make sure to run it in all terminal windows that you want to be updated. After saving the `~/.bashrc` file, every new terminal window you open will automatically be updated with the new settings and you won't have to source it manually.

2.4 Additional Lab Etiquette

Because the lab workstations are shared, we sometimes run into strange bugs. The most common is that someone leaves a process running when they leave the lab, sometimes by forgetting to log out properly. When the next person uses the same workstation, ROS can get confused by the additional processes running from the prior user. A particularly insidious example is leaving a `roscore` master node running in the background; the next person will not be able to run a master node with proper communications. To avoid issues like this in the lab, please do the following before leaving:

- Ctrl+C out of every terminal before closing it. It is critical that you Ctrl+C. **Do not Ctrl+Z**. Ctrl+Z may look like it stops your process, but it really only pauses it. The process will continue to run in the background.
- To log out, use the command `pkill -u [username]`. where `[username]` is your login credential. This will close out every process on your account before logging you out.

3 Rethink (Sawyer) Robots

In the lab, there are Sawyer robots (Ada, Alan, Amir, Azula and Alice) robots. They are from Rethink Robotics are manufacturing robots designed to operate around people. They are unfortunately no longer supported by the company, so **please be especially careful when using these robots as they are difficult to fix and replace.**

Figure 1: Sawyer robot.

3.1 Hardware

Note: Much of this section is borrowed from the Rethink Sawyer Wiki. We picked out the material you will find most useful in this class, but feel free to explore other resources if you are interested in learning more.

- Arms
    - Sawyer: One arm with seven joints. Because the Sawyer only has one arm, we default to calling it to right arm.

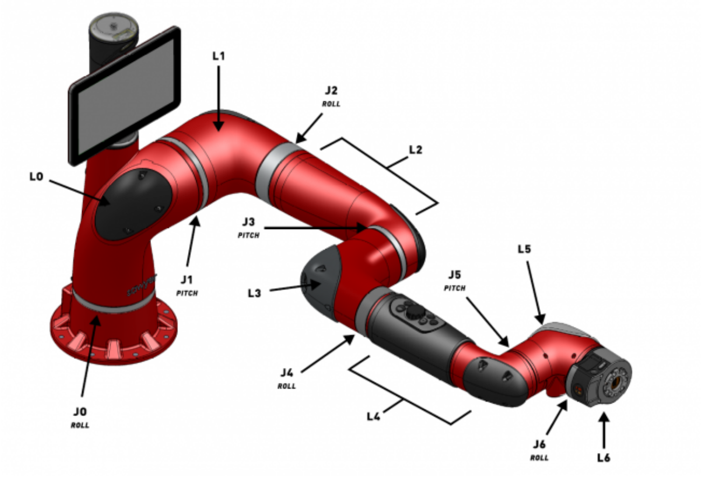


Figure 2: Labeled Sawyer joints and links

- Grippers
    - Please ask someone on course staff for assistance when installing or removing a gripper.
    - Electric parallel-jaw grippers: 44 mm throw and attachment points for a variety of finger configurations, meant for lifting payloads up to five pounds

    - Pneumatic suction grippers: can attach either a single vacuum cup or a multi-cup vacuum manifold

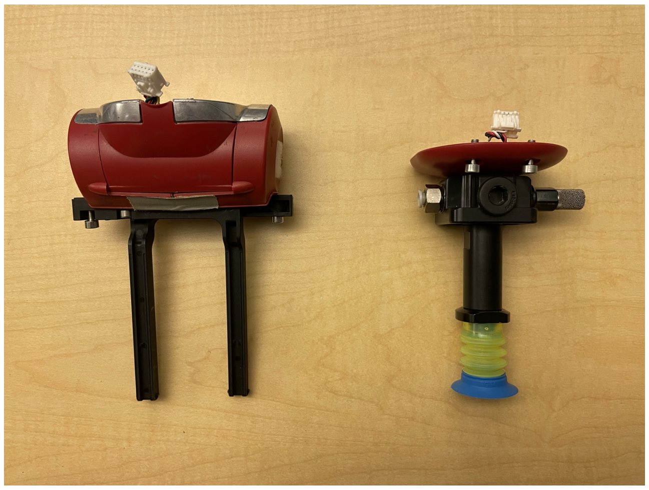


Figure 3: Electric parallel-jaw gripper (left) and pneumatic suction gripper (right).

- Cameras
    - One camera per arm (Figure 4)
    - One camera in the head

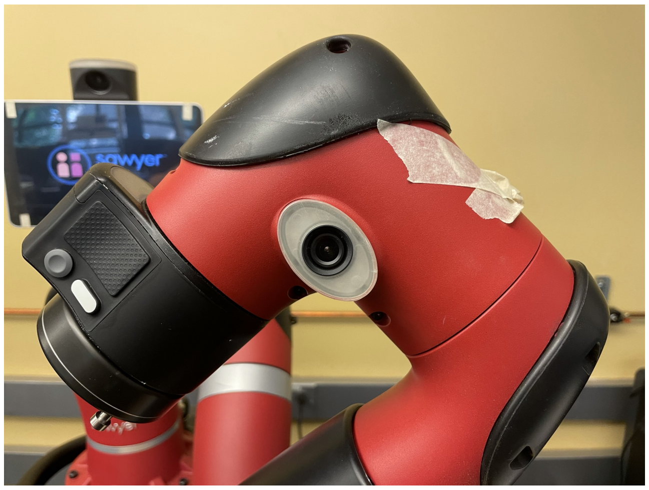


Figure 4: Sawyer wrist camera.

- Head
    - Panning and nodding abilities
    - Camera
    - Sonar ring
    - Display
- Control
    - On-board computer running Linux and ROS Noetic
    - Can run ROS nodes remotely or on-board
    - Low-level controllers prevent self-collisions and enforce limits on acceleration, torque, and position

- Zero-g mode: If you wish to move the robot arms manually, enable the robot and grasp the cuff of the arm over its grooves (Figure 5). This will enable the zero-g mode where the controllers are disabled and so the arm can be freely moved across without much resistance. **Do not try to push or pull on the arms without enabling zero-g mode.**

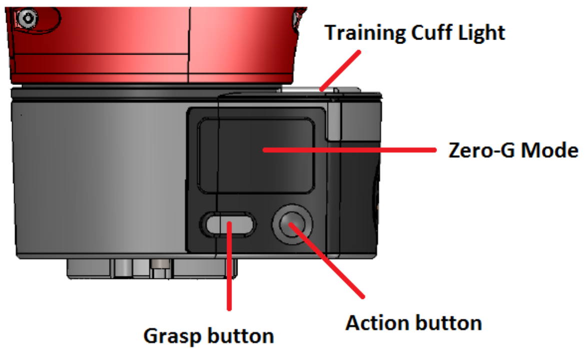


Figure 5: Diagram of zero-g mode indentations on Sawyer cuff.

- Sensors
  - 3-axis accelerometer inside each cuff
- E-Stop and power buttons
  - E-Stop button: Sawyers have an emergency stop (E-Stop) button (Figure 6). **If you have even the slightest feeling that the robot is behaving or about to behave dangerously, push down on this button**; the robot will be immediately disabled and the arms will lower slowly. Dangerous behavior includes but is not limited to any of the following:
    - the arm is about to crash into someone/something
    - the arm is moving very quickly or jerkily
    - the arm is not behaving as expected

    In any case, keep a hand on the E-Stop button whenever you are running code on the robot and don't be afraid to press it if you think a situation is or is becoming unsafe. You do not need to hold down on the E-Stop button; after pressing it, it will be mechanically held down until you release it. To release the E-Stop button from the held-down state, turn the button clockwise until it pops up. Note that the robot can only be enabled after the E-Stop button is released, and you will need to re-enable the robot before moving it again. Refer to 3.3.1 on how to enable the robot.

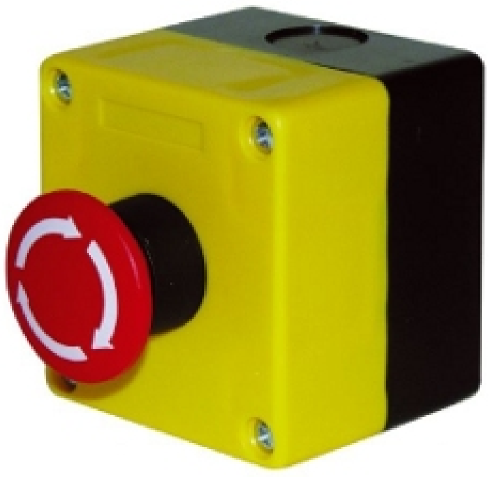


Figure 6: E-Stop button with arrows indicating how to turn the button to release it.

- Power button: The power button is located on the left side of each robot, but you should leave the robots on most of the time and should not need to use this button. **Do NOT hold down the power button;** holding down the power button can cause hard drive corruption on the next boot. Just press the power button once to turn on or once to turn off.

3.2 Setting up your environment for Rethink robots

To automatically set up the Sawyer packages in each new terminal, make sure that your `~/.bashrc` file includes the line

```
source /opt/ros/eecsbot_ws/devel/setup.bash
```

If you edited the `~/.bashrc` file, make sure to source it in any existing terminal windows that you plan on using. Refer to the Setting Up Your Environment section for more information about what this line does in the `~/.bashrc` file.

To set up your environment for interacting with Sawyer, make a shortcut (symbolic link) in the root of your catkin workspace to the Sawyer environment script `/scratch/shared/eecsbot_ws/intera.sh` using the command

```
ln -s /opt/ros/eecsbot_ws/intera.sh [path-to-workspace]
```

From the root of the catkin workspace, use the following line to connect to one of the Sawyer robots:

```
./intera.sh [name-of-robot].local
```

where `[name-of-robot]` is `ada`, `alan`, `amir`, `azula` or `alice`.

When you are done using any robot, make sure all of the terminal windows where you connected to the robot are no longer running any processes. Then in each terminal window that is connected to the robot, close the connection to the robot with the command

```
exit
```

3.3 Basic functions

3.3.1 Enabling the robot

In order to control Sawyer's arms, the robot must be put in the "Enabled" state. Enabling the robot provides power to the joint motors, which are initially in the "Disabled" state on start-up or after a serious error, such as an E-Stop. You can enable the Sawyer by running

```
rosrun intera_interface enable_robot.py -e
```

3.3.2 Controlling the joints from the keyboard

The `joint_position_keyboard.py` script allows you to move the robot's limbs from the keyboard. Start the `joint_position_keyboard.py` script by running

```
rosrun intera_examples joint_position_keyboard.py
```

3.3.3 Reading the transform

Read the rigid body transformation from the robot base to the hands by running

```
rosrun tf tf_echo base right_hand
```

3.4 Interfacing with the robot

Instead of publishing directly to a topic to control Sawyer's arms, the SDKs provide a library of functions that take care of the publishing and subscribing for you. Remember that whenever you try to move an arm on Sawyer, it must be the **right** one.

3.4.1 Joint Trajectory Action Server

The joint trajectory action server within Sawyer's SDK allows users to command the robot's arm through multiple waypoints and track the trajectory execution. The main benefit of this server is its ability to interpolate between supplied waypoints, command Sawyer's joints accordingly, and ensure the trajectory is being followed within a specified tolerance. You will need to run this before executing a trajectory on the robot.

You can start the joint trajectory action server by running

```
rosrun intera_interface joint_trajectory_action_server.py
```

3.4.2 Sawyer MoveIt!

MoveIt! is an open source robotics manipulation platform that runs on top of ROS and uses kinematics, motion planning, and collision checking to plan and execute trajectories for robot manipulators. You will need to run this before executing a trajectory on the robot. Start MoveIt by running

```
roslaunch sawyer_moveit_config sawyer_moveit.launch electric_gripper:=true
```

Omit the gripper argument if the robot does not have a gripper.

4 TurtleBots

TurtleBot is one of the classic platforms for mobile robotics research and teaching. We have upgraded to now TurtleBot 3 in this class, since it has the latest features. Turtlebot 3 is available in the `burger` and `waffle` type of model. We will be using the `burger` model in this class. Each Turtlebot has a name mentioned on it and has a different AR tag number.

Each TurtleBot is assigned to one workstation. You should not use any other TurtleBot besides the one assigned to your current workstation, but you may use the Turtlebot at station 12 if the TurtleBot assigned to your workstation is broken.

4.1 Hardware

Note: Much of this section is borrowed from the TurtleBot3 User Manual. We picked out the material you will find most useful in this class, but feel free to explore other resources if you are interested in learning more.

- Drive base
  - Pick up the TurtleBot from under the base on both sides.
  - "Unicycle model" tank drive (can rotate in place)
  - Encoders on 2 main drive wheels (can measure how far each wheel has turned)
  - 360° LiDAR, 9-Axis Inertial Measurement Unit for SLAM and navigation
  - Gyroscope and Accelerometer
  - Front bumper (can detect collisions)
  - AR (augmented reality) tag on top
- Control
  - Raspberry Pi running Linux and ROS Noetic and OpenCR
  - Can run ROS nodes remotely or on-board
- Power
  - Status LED
  - Charging port in the drive base
  - On-off switch: Please charge the TurtleBots when they are not in use. When charging, make sure that the power cord is plugged into the charging port in the drive base and the TurtleBot is switched OFF. **The TurtleBot must be switched off in order to charge.**

4.2 Setting up your environment for TurtleBots

To automatically set up the TurtleBot packages on the PC, make sure that your `~/.bashrc` file includes the lines

```
source /opt/ros/noetic/setup.bash
export ROS_MASTER_URI=http://192.168.1.[COMPUTER_NUMBER]:11311
```

where `[COMPUTER_NUMBER]` is the workstation computer number. If you edited the `~/.bashrc` file, make sure to source it in any existing terminal windows that you plan on using. Refer to the Setting Up Your Environment section for more information about what these lines do in the `~/.bashrc` file and how to confirm both `~/.bashrc` files on the workstation and the Turtlebot are correct.

Before you can listen for messages or give commands to the TurtleBot, you'll have to turn it on. Begin by turning on the power and checking that you can `ping` the onboard computer by running

```
ping [FRUITNAME]
```

and making sure that there are no errors when trying to connect to the TurtleBot. If pinging doesn't work, try turning the TurtleBot off and on again. Note that some of the TurtleBots take several minutes to wake up. When you are done using the robot, make sure all of the terminal windows where you connected to the robot are no longer running any processes. Then in each terminal window that is connected to the robot, close the connection to the robot with the command

```
exit
```

4.3 Basic functions

4.3.1 Start the base functionality

Start the TurtleBot basic nodes (including the master node) by ssh-ing into the robot and running

```
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

To launch the camera run

```
roslaunch turtlebot3_bringup turtlebot3_rpicamera.launch
```

The TurtleBot is now launched, along with all of its sensors, and it is ready to receive motion commands.

4.3.2 Controlling the TurtleBot from the keyboard

Keep the `turtlebot3_robot.launch` running and open a new terminal window. Open a new terminal window **(don't ssh in)**. Drive the TurtleBot from the keyboard by running

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

4.3.3 Using SLAM

SLAM is a useful library which stands for Simultaneous Locomotion and Mapping. It allows the robot to naviagte through unknown environments with the help of the LiDAR sensor on it. To start it we can run this command on the PC

```
roslaunch turtlebot3_slam turtlebot3_slam.launch
```