Robot Usage Guide [*]

EECS/ME/BIOE C106A/206A Spring 2024

Contents

1 Lab Safety

General lab safety guidelines apply to our lab, including:

- Do not do any lab(s) alone
- Keep walkways clear
- No food or drink (except water) in the lab

[*]Modified by Kirthi Kumar, Spring 2024. Developed by Riddhi Bagadiaa, Fall 2022.

- If you are unsure of how to do something, ask someone on course staff
- If you become aware of a hardware issue, notify someone on the course staff immediately

Whenever you are using the Rethink robots, **be ready to press the E-Stop button at a moment's notice to prevent the robot from crashing into anything**. Be particularly careful of people who may be walking through the aisle between the Rethink robots and computers.

1.1 Health Safety Measures

You should not have to sacrifice your health for your education, so we are implementing a few safety measures:

- If you feel unwell, stay home and email your lab TA and the Head TAs. We will arrange accommodations for you to complete the labs at a later date.
- Students who are assigned to a lab section will always be allowed to work during that section, but otherwise you may be asked to leave if there are too many people in the lab.

If you do not comply with these safety measures, you may be asked to leave the lab.

2 Setting Up Your Environment

Your environment is automatically set to use ROS and interface with the robots, in your `~/.bashrc` file. The `~/.bashrc` file is a script that is run when a new terminal window is started. Depending on where you sit, you will get the packages for the robot. You will get Sawyer packages if you are sitting on Workstation 6 - 10; otherwise, you will get the Turtlebot packages if you are sitting on Workstation 1 - 5 and 11.

2.1 Turtlebot

2.1.1 Setting up ROS

The TurtleBots are assigned to specific workstations. Workstations 1 - 5 and 11 will automatically run this command line whenever you open a terminal or a new terminal tab.

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

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 ROS Hostname

This node environment variable sets the declared network address of a ROS node or tool. For our lab, the ROS hostname is set to the specific computer that you are working on.

2.1.3 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). 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. The workstations from 1 - 5 and 11 will automatically set it up for you.

2.2 Rethink Robot (Sawyer)

2.2.1 Setting up ROS

The Sawyers are assigned to specific workstations. Workstations 6 - 10 will automatically have packages for the Sawyer once you open the terminal. You can immediately run the simulations for the Sawyers; however, **you will not be controlling the actual Sawyer robots yet.**

2.2.2 ROS Hostname

Like the Turtlebot, this node environment variable sets the declared network address of a ROS node or tool. For our lab, the ROS hostname of the workstation computer would be the specific computer number that you are working on.

2.2.3 ROS Master URI

It is a hostname:port combination that tells nodes where they can locate the master node. By default, it will be set to the workstation computer when you run the simulations for the Sawyer. When you are ready to actually control the robot, you must run the following command on the terminal:

```
source ~ee106a/sawyer_setup.bash
```

Please note that **you must run this command first if you open a new terminal or terminal tab!** If you fail to run the command, you will switch back to simulations. It is a precaution to prevent damage to our robot during your planning stage.

2.3 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 $(whoami)`. 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.**

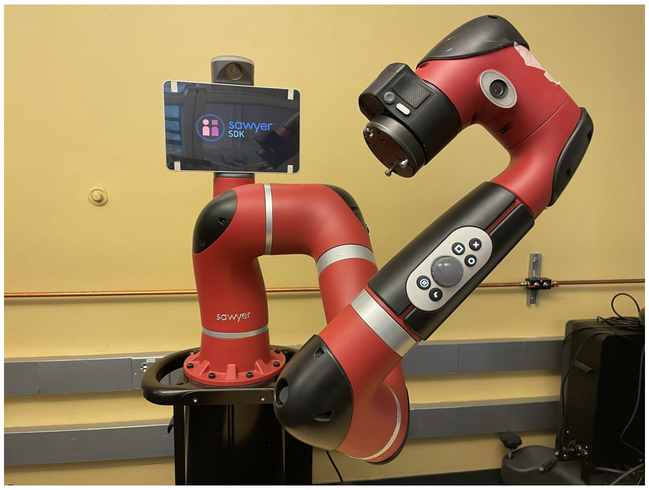


Figure 1: Sawyer robot.

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.

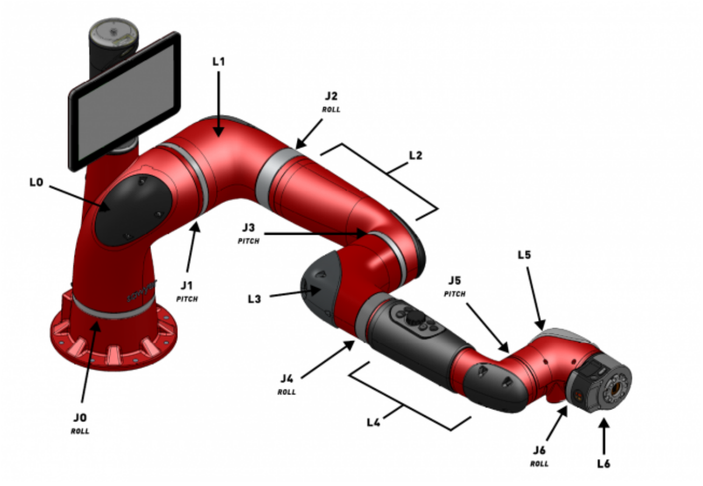


Figure 2: Labeled Sawyer joints and links

- 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

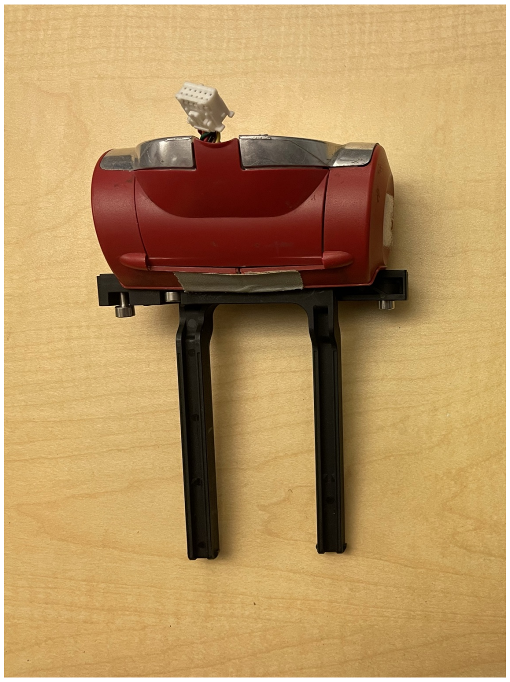


Figure 3: Electric parallel-jaw gripper.

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

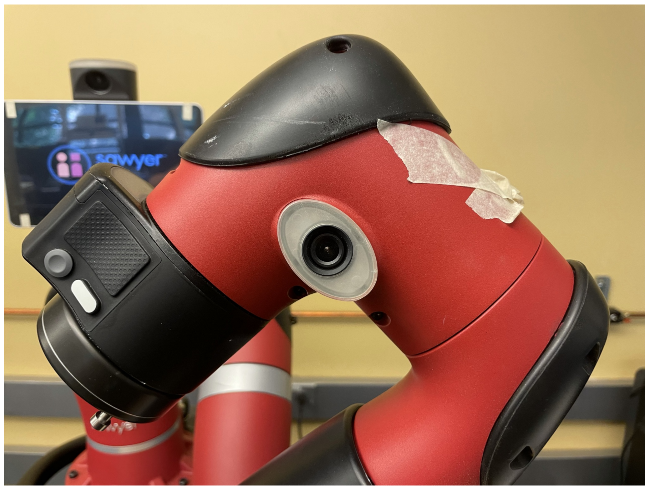


Figure 4: Sawyer wrist camera.

- 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.**

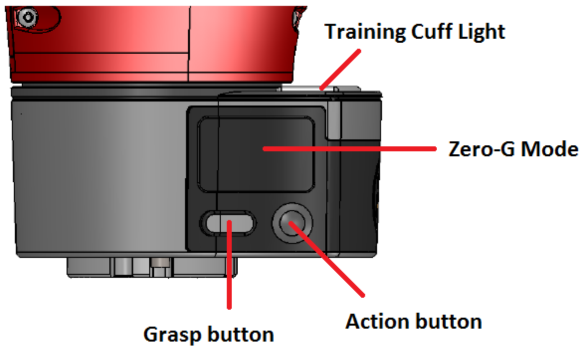


Figure 5: Diagram of zero-g mode indentations on Sawyer cuff.

- 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.

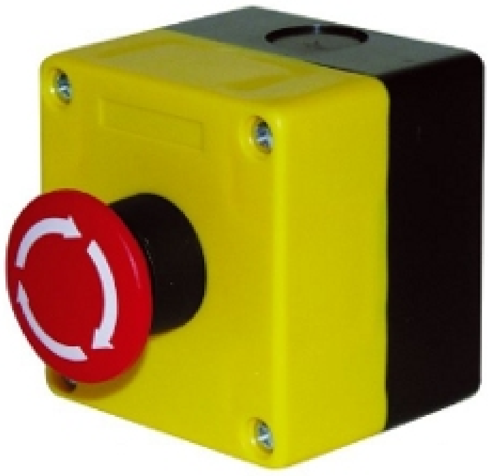


Figure 6: E-Stop button with arrows indicating how to turn the button to release it.

    - 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 actually use the Sawyer Robot instead of the simulation, you run the following command on your terminal.

```
source ~ee106a/sawyer_setup.bash
```

You must always run the command line above if you open a new terminal or a new tab; otherwise, you will working in simulation.

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 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.

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 11 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

All TurtleBot packages are automatically on the PC if you are working on Workstations 1 - 5 and 11. You will need to ssh into the TurtleBots for your lab by running this on your terminal

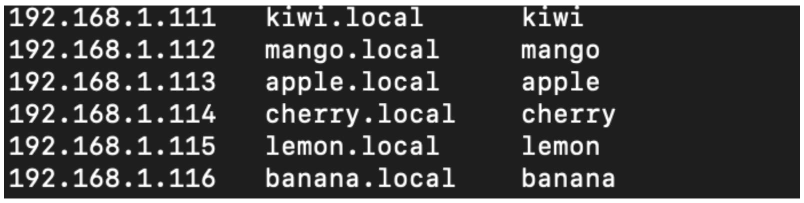
```
ssh [FRUITNAME]@[FRUITNAME]
```

where [FRUITNAME] is `kiwi, cherry, apple, mango, banana or lemon`. The password is `[FRUITNAME]2022`.
Fun Fact: Turtlebot 3 was officially deployed to the EECS106 lab in 2022 by ESG.

You can view the list of robots by running this command in the terminal.

```
cat /etc/hosts
```

Here is a snippet of the TurtleBots' information:



```
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
```

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 (dont 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
```