UR@B Software-Control Setup & Intro Project

Andy Tang, 8/2022

Welcome to UnderWater Robotics at Berkeley, Controls Subteam.
This Tutorial will:

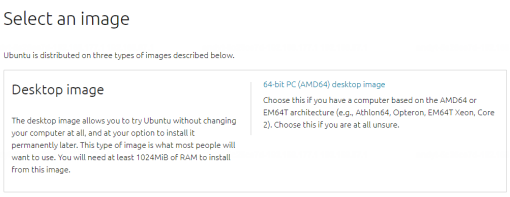
1. Help you set up your development environment
2. Provide you basic insights into ROS2
3. Challenge you with an intro task which will make you an official team member once completed (we will help you if you get stuck)
4.

No prior experience with robotics needed. If you encounter problems, ask in our slack work group.

1. Software Setup

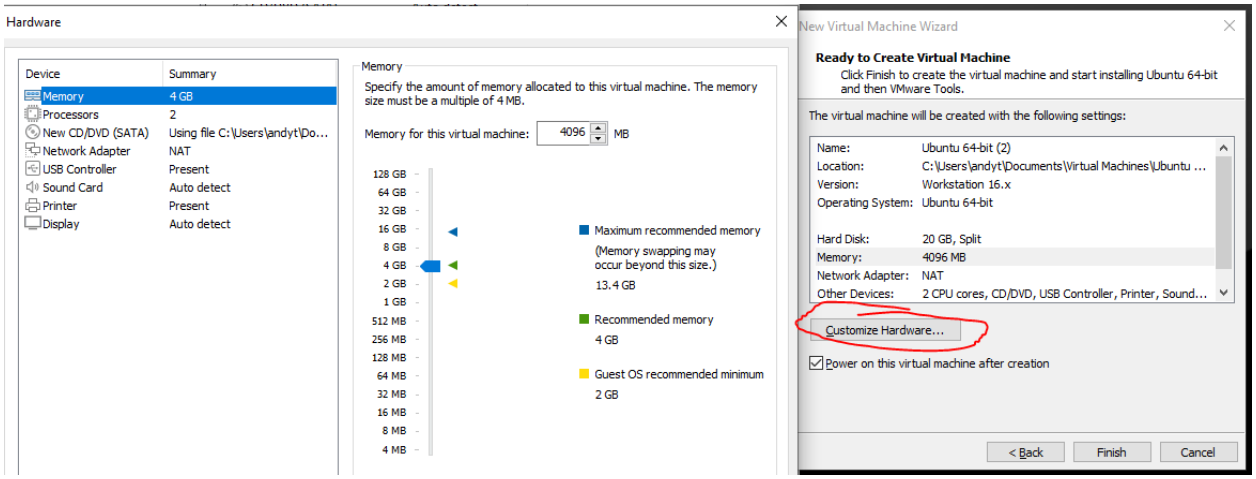
We perform all our simulations on linux, and we prefer Ubuntu 20.
If you are a Windows/Mac user, you will set up an Ubuntu virtual machine using VMware, which is free for all students at Cal. (It may take a few days for IT to send you the product key via email.)

- Download VMware
https://software.berkeley.edu/vmware
- Download Ubuntu 20 Image
https://releases.ubuntu.com/20.04.6/?_ga=2.152804730.251475035.1693974820-351722476.1693974820



Once you have both items downloaded, open VMware and go to file->New Virtual Machine->typical->select the Ubuntu.iso you just downloaded->create your login credentials->allocate disk size(default option of 20G is good enough)->Now don't click finish yet

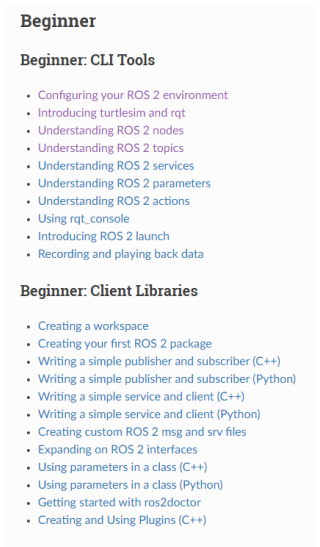
We want to allocate at least 4G of RAM, (8 GB preferred if your machine has 16GB or more installed). This is because the Gazebo simulations are resource intensive.



Click Finish, and your vm should be ready.
(On Ubuntu, you may be asked for your password for every little thing, note that if you type your password in the command prompt, the characters won't show but they are actually recorded, so just type it as normal. If you find it annoying, consider disabling password requirements: [https://askubuntu.com/questions/235084/how-do-i-remove-ubuntus-password-requirement)](https://askubuntu.com/questions/235084/how-do-i-remove-ubuntus-password-requirement)

2. ROS2 Basics

It is vital to understand the basics of the Robotic Operating System, or ROS for short. We are currently using ROS2 Foxy (ubuntu 20), but will move on to the latest LTS ROS2 Humble (ubuntu 22).
You should install ROS2 Humble from its official wiki. Proceed to the website and follow through all the beginner tutorials shown to the right. Note that the instructions come in both python and C++. You choose whichever you are more familiar with, no need to go over both.

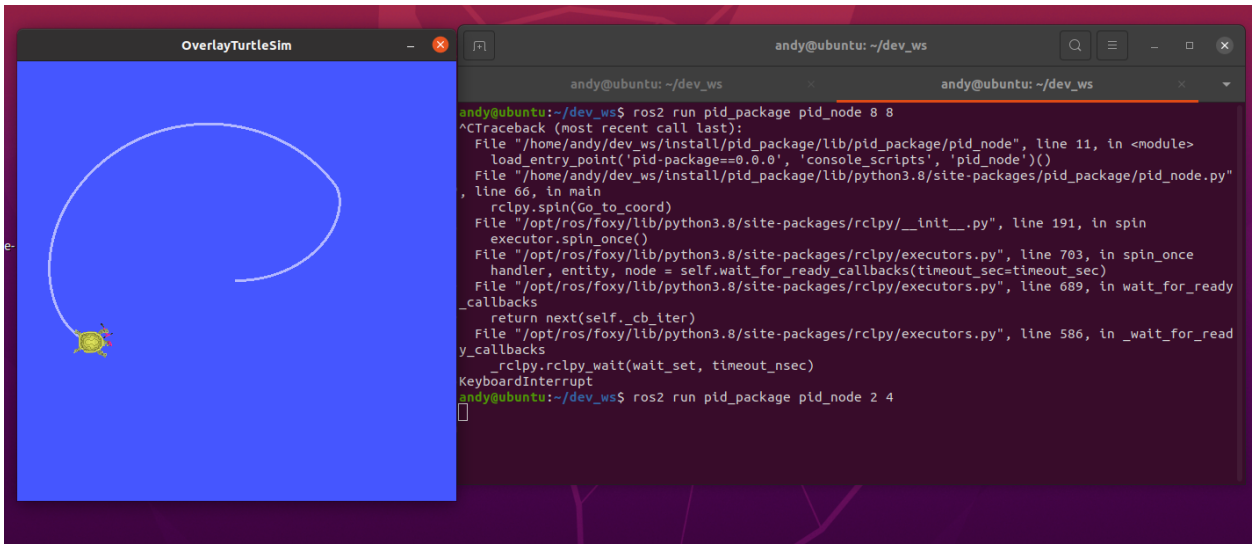


Play around with Turtlesim, use rqt, topic list, and the echo command to see how nodes & topics communicate, and see what data is passed between them.
If you run into issues during this stage, you can likely find it on Google.

3. Intro Task

   Now that you understand ROS2 basics and have familiarized yourself with turtlesim, let's try something more exciting!

   You will write a PID algorithm for turtlesim so that the turtle will go to the specified coordinates in a nice smooth path. The result should look something like this:



(here the turtle spawned in the center of the screen, went to (8,8), then to (2,4))

Tips and Resources:

- What is PID?

  https://www.youtube.com/watch?v=wkfEZmsQqiA

  (based on the video examples, you might realize that for this task, you only need to implement the proportional

component, but it's helpful to also understand the integral and derivative components since they are implemented on the actual robot)

- How does my program know where the turtle is, and how do I tell it where to go?

  Your program should subscribe to the "pose" topic, which gives you the turtle's current position and direction. It should then process the data and publish to "cmd_vel" topic so that the turtle moves accordingly.

  Remark: "cmd_vel" interprets coordinates as relative to the turtle instead of being relative to the 2d plane. For example, if the turtle is at (4,4) facing down, then passing (0,1) to cmd_vel will make it end up at (4,3). Therefore, you will have to do some transformation so that the goal of every movement is converted from absolute coordinate to that relative to the turtle.

- How does my program accept command-line input as arguments?

  Use sys.argv[]

- Lastly, here is a screenshot of the dependencies you might need. Remember to ask for help if stuck.

  Good Luck!

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
import math
import sys
```

- On Completion:

  Congratulations!

  You will demonstrate your working turtlesim during our software sub-team meetings.