**~~Applications of Semiconductor Technologies and Devices:~~ Building an Autonomous 2WD Robot for Testing Various Navigation and AI Concepts**

**Overview:**

This project requires students to design, build, and test a two-wheel drive (2WD) autonomous robot. The robot will be constructed using specific hardware components and will demonstrate autonomous navigation with loop closure on a simple indoor track. The task will culminate in the submission of a 10-page IEEE style double column formatted paper that details the implementation, testing, and a deep dive into one of the key subsystems.

**Objectives:**

1. **Build the Robot**: Using the provided components, assemble the robot.
2. **Programming and Integration**: Run and become familiar with the software to control the robot using ROS2, integrate sensors and actuators, and implement navigation and mapping algorithms.
3. **Testing and Optimization**: Test the robot on a predefined indoor track to ensure it can autonomously navigate and achieve loop closure.
4. **Technical Writing**: Document the entire process, focusing on the details of the implementation of a subsystem in a well-structured IEEE formatted paper.
5. **~~Semiconductor Technologies:~~** ~~Evaluate the application of specific semiconductor technologies and devices in various circuits and subsystems.~~

**Required Components:**

* Raspberry Pi 4: Acts as the central processing unit.
* Waveshare General Driver for Robots: To control the motors and IMU.
* DFRobot TT Motor with Encoders: Provides motion and feedback on distance and speed.
* 2S 18650 LiPo Battery Bank: Power source.
* 2S 2A Multi-Cell 2S Type-C to 8.4V Step-Up Boost LiPo Polymer Li-Ion Charger: Manages battery charging.
* FHL-LD19 DTOF Lidar: For environmental scanning and mapping.
* Acrylic 2WD Robot Chassis: Structural framework.

**Assignment Details:**

**Part 1: Design and Construction**

* **Chassis Assembly**: Assemble the 2WD chassis integrating the motor, battery bank, and driver board.
* **Circuit Integration**: Implement the circuit connections between the Raspberry Pi, Waveshare driver board, motors, and sensors.

**Part 2: Software**

* **ROS Setup**: Install and configure ROS on the Raspberry Pi for robot programming.
* **Sensor Integration**: Run ROS nodes for handling inputs from the encoders, IMU, and Lidar.
* **Control Algorithms**: Execute the software used to process sensor data, control motor actions, and handle navigation and path planning.

**Part 3: Navigation and Mapping**

* **SLAM Implementation**: Utilize Lidar data to implement SLAM for real-time mapping and navigation.
* **Loop Closure**: Ensure the robot can navigate the track and return to the starting point accurately.
* **Path Planning**: Implement and tune algorithms for optimal path finding within the mapped environment.

**Part 4: Testing and Validation**

* **Performance Testing**: Conduct tests to validate the functionality of all systems.

**Part 5: Documentation**

* **IEEE Paper**: Document the entire project process including detailed diagrams of the robot’s design, code snippets, and explanations of the decision-making process.
* **Subsystem Deep Dive**: Focus on starting at the Waveshare General Driver level, illustrating the schematic, semiconductor technology, standards, and circuit implementation. Use the provided templates as guides for your deep dive.
* **Results and Discussion**: Analyze the performance and discuss any challenges and solutions.

**Deliverables:**

* **Functional 2WD Autonomous Robot**.
* **10-page IEEE formatted report**.
* **Demonstration**: Demonstrate the functioning robot on or before the scheduled final exam day 5/15.

**Evaluation Criteria:**

* **Mastery of Software, Network and Operating Systems**. (10%)
* **Navigation and Loop Closure success**. (25%)
* **Depth and Technical Accuracy in Documentation**. (50%)
* **Overall Aesthetics, Functionality and Reliability of the robot**. (15%)

This project will challenge students to apply their engineering knowledge to solve a real-world problem and will hone their technical writing, design, and analytical skills. Most importantly, the student will connect the various technologies covered in EEGR215 with a real-world application.

**Part 1: Design and Construction**

**Chassis Assembly**

Download [this image file.](https://drive.google.com/file/d/1z108ZDqcy3cuDBdIRIqlCUSFCn6ONv9a/view?usp=drive_link) and burn it onto an SDXC card. Be sure the SDXC card is at least 64Gb. The image was created on a Microcenter generic SDXC card be sure that the card size is large enough to accommodate the image transfer process. Balena Etcher or Windisk32 Imager are good programs to burn the card with.

User: Student215

Password: eegr215

Review this tutorial:

[Introduction to Linux system.](http://www.yahboom.net/study/YDLIDAR-4ROS)

The [Waveshare General Driver for Robots.](https://www.waveshare.com/wiki/General_Driver_for_Robots" \t "_blank) is a great resource for building a small robot. It has an integrated MPU and connectivity options for interfacing with sensors and networks. The schematic diagram can be found [here](https://files.waveshare.com/upload/3/37/General_Driver_for_Robots.pdf).

We will begin this project by building the power management system from a 2S LiPo battery bank.

A green circuit board with many different components

Description automatically generated

Next you will wire the power management to the general driver board. Use 2 JST 6-pin connectors and 4 3-pin low voltage splicers to make 2 motor wiring harness cables. Note the JST connectors on the board are flipped on each side for MA and MB motor interfaces.

+

A diagram of a circuit

Description automatically generated

Typical motor encoder circuit

The figure below only shows connections for one motor. You will need to connect both motors. The motor pinouts can be found on [this website](https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-SJ01_SKU__FIT0450).

A circuit board with many wires

Description automatically generated

Wiring diagram for power and motor connections

**Test the motors and encoders**

[Github repository for the microcontroller firmware.](https://github.com/hippo5329/linorobot2_hardware/wiki#test-the-motors-and-encoders)

The test\_motors utility will spin the motors forward and backward alternately every 10 seconds without user intervention. The motors will run one by one after power on. Then it will display the motors linear velocity, angular velocity and distance to full stop. Make sure the motors are running at the correct direction and the encoders get the correct speed reading, which is the maximum speed of the motors. You should calibrate the rpm reading with a digital tachometer and adjust the COUNTS\_PER\_REV for encoders 1-4. The test\_motors utility will include the firmware/platformio.ini. There is no need to edit the test\_motors/platformio.ini for your custom configuration.

Connect the bottom USB 3.1 port on the raspberry Pi to the USB C port on the general driver board. To run commands use must open a terminal window by right clicking anywhere on the desktop and selecting “Open in Terminal”.

Each time we need to upload the firmware, we have to kill the robot bring up script that automatically runs when we boot or reboot the system.

sudo systemctl stop robot-boot.service

Then, build and upload with

cd ~/linorobot2\_hardware/test\_motors

pio run -e myrobot -t upload

pio device monitor -e myrobot

**Test the other sensors**

The test\_sensors utility will display IMU, MAG, battery and range sensors data every second, in x y z sequence. Again, don’t forget to stop the robot-boot.service before trying to upload the test.

Build and upload with

cd ~/linorobot2\_hardware/test\_sensors

pio run -e myrobot -t upload

pio device monitor -e myrobot

**Connecting the Lidar**

This LD19 LiDAR sensor offers a ranging distance of up to 12m and 360° environmental scanning, which can realize autonomous mapping, and obstacle detection. Connect the LiDAR to your RPI4 via the USB-to-serial CP2102 module. Test the lidar by launching a node to visualize the laser scans with

sudo systemctl stop robot-boot.service

cd ~/linorobot2\_ws

ros2 launch ldlidar\_stl\_ros2 viewer\_ld19.launch.py

You should see a new Rviz2 window similar to the one below.

A screenshot of a computer

Description automatically generated

**Bring up the robot**

If you haven’t done so, build and upload the firmware that allows the ESP32S MCU to communicate with with the raspberry PI.

cd firmware

pio run -e myrobot -t upload

This will allow you to interact with the robot using ROS2 commands. These commands can be issued from a terminal on the robot’s computer or on a remote machine. Using the keyboard and mouse connected to the robot machine allows you to test the ros2 installation. Your RPI4 images have been already configured to publish the topics that ROS needs to do navigation. For informational purposes this command would be:

ros2 launch linorobot2\_bringup bringup.launch.py base\_serial\_port:=/dev/esp32s lidar\_serial\_port:=/dev/ldlidar

Again this command is already programmed to run on boot-up.

**Drive the robot with a keyboard**

At this point you can move the robot but cables are still required to read the screen and communicate with the mouse and keyboard. The robot listens to topic /cmdvel and move. You may run the teleop\_twist\_keyboard package and follow the instructions on the terminal to drive the robot:

ros2 run teleop\_twist\_keyboard teleop\_twist\_keyboard

You will see the keyboard command list.

This node takes keypresses from the keyboard and publishes them

as Twist messages. It works best with a US keyboard layout.

---------------------------

Moving around:

u i o

j k l

m , .

For Holonomic mode (strafing), hold down the shift key:

---------------------------

U I O

J K L

M < >

t : up (+z)

b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%

w/x : increase/decrease only linear speed by 10%

e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently: speed 0.5 turn 1.0

Press 'i' key, and the robot should move forward.

**Part 2: Software**

**Remote Computer Setup**

In order for you to achieve wireless operation you must set up a host machine to connect with your robot and issue commands from there. The process on this machine will be to:

1. Install a Virtual Machine (VM)
2. Install Ubuntu 22.04
3. Install ROS2
4. Install Parts of Linorobot2

**Install a Virtual Machine**

In this section you will go through the in-depth process to install Ubuntu 22.04 with ROS 2 on a Windows 10/11 system using the free [Oracle VirtualBox software](https://www.virtualbox.org/wiki/Downloads). If you have a Mac [see this link.](https://www.theroboticsspace.com/blog/How-To-Install-ROS-2-in-Ubuntu-22-04-On-M1-Mac/)

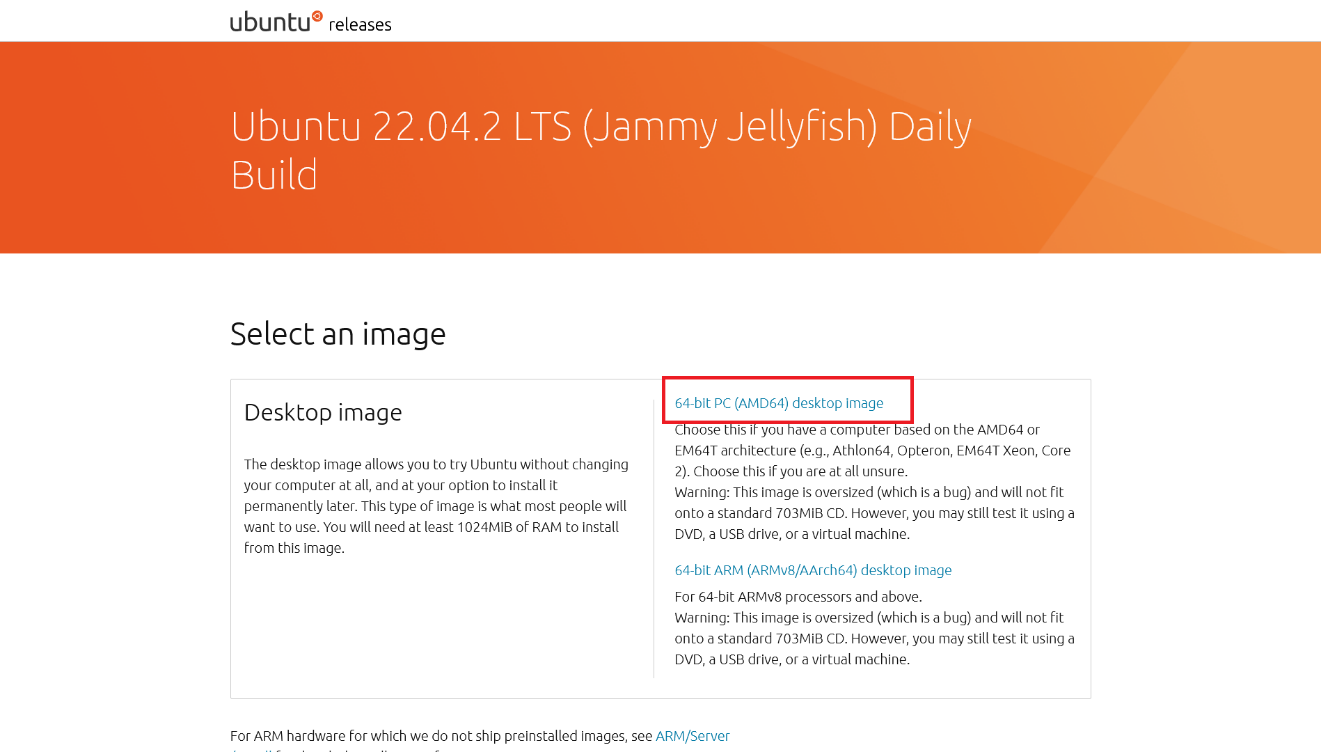
**Pre Requisites**

Make sure you satisfy the below prerequisites before installing Ubuntu 22.04 Virtual Machine and ROS 2 Humble on your Windows system.

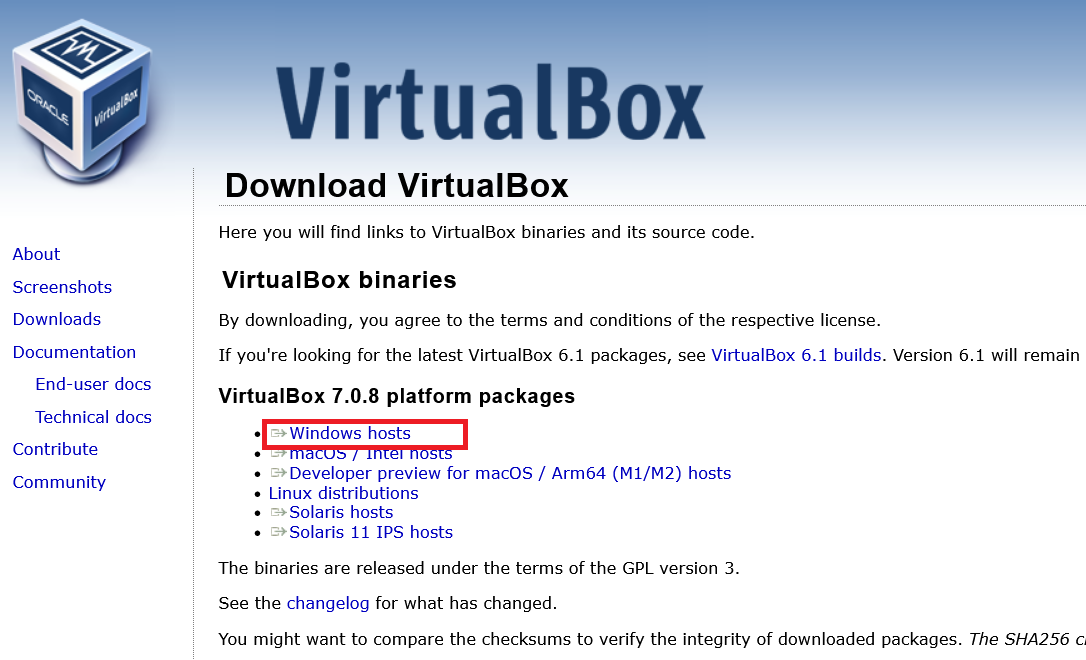
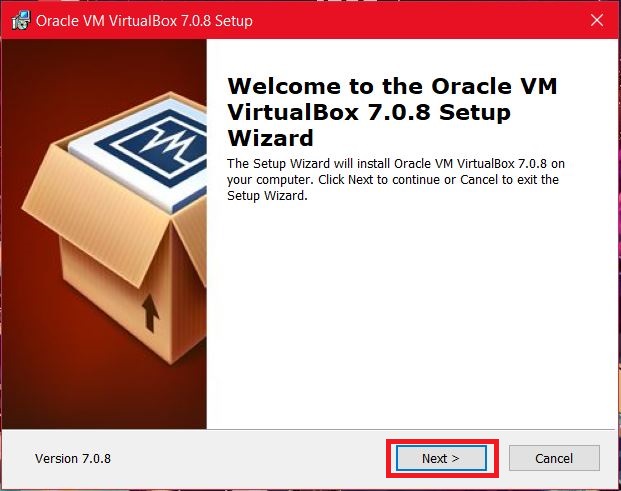
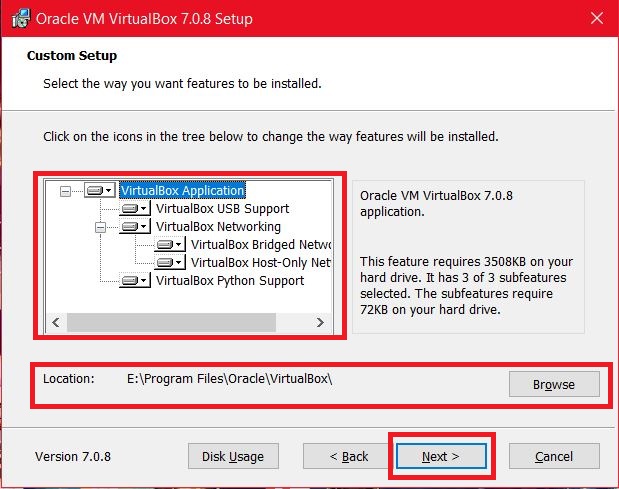
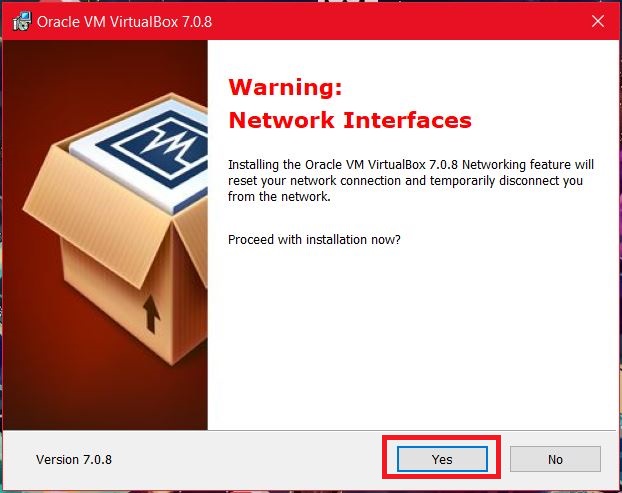
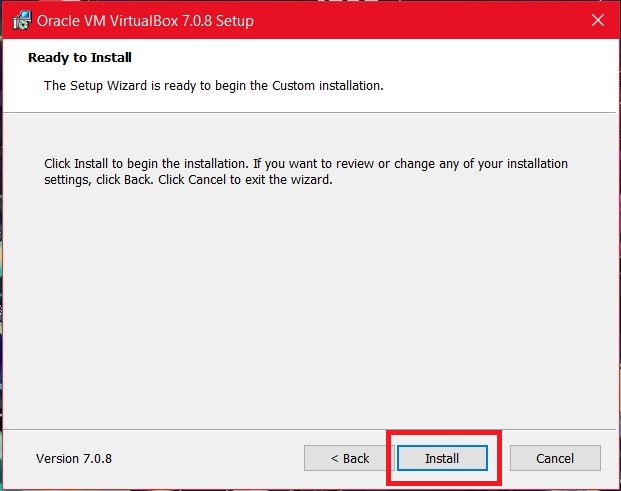
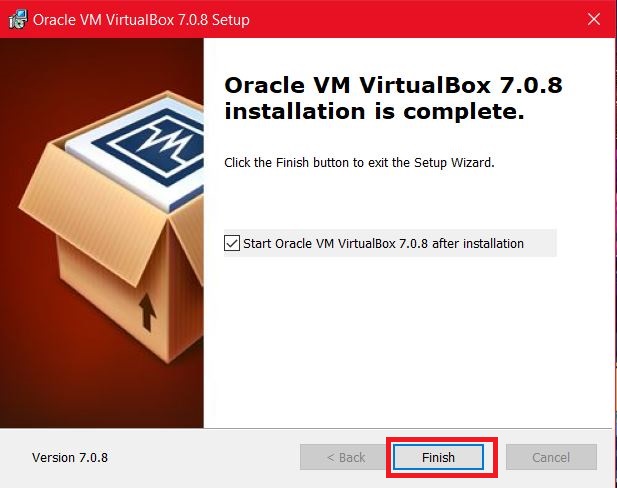
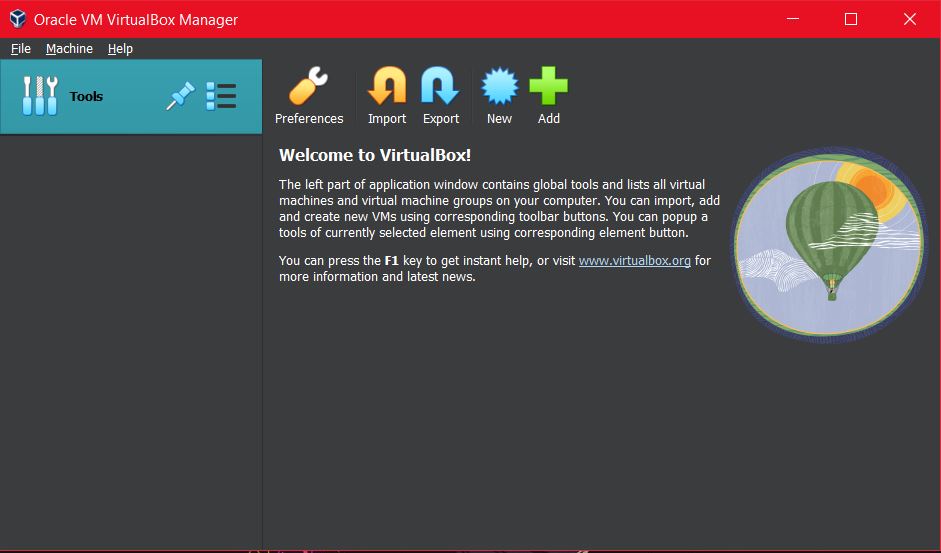
* A computer running Windows 10 or 11. This guide uses Windows 10 64-bit. Note: VirtualBox should work fine on any recent version of Windows 10 or 11.
* Ubuntu: Desktop 22.04 ISO Image
* Oracle VirtualBox 7.1 software
* Minimum 50 GB of Storage memory
* Minimum of 8 GB of RAM

Let’s dive right into the process to set up Ubuntu 22.04 Virtual machine to start working with ROS 2.

**Download Ubuntu 22.04 ISO Image**

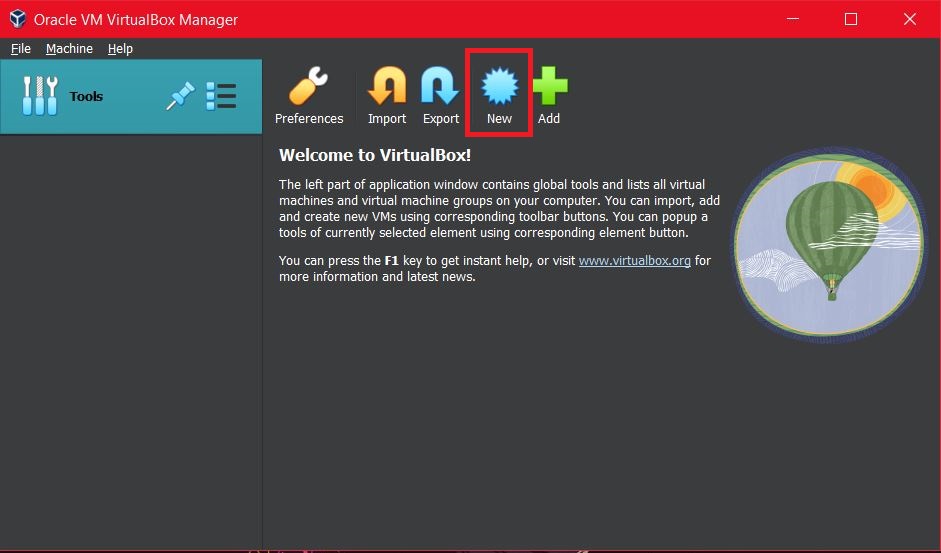
1. This section will walk you through the process of downloading and installing the latest LTS version of the Ubuntu distribution of Linux, i.e., Ubuntu 22.04 Jammy Jellyfish.
2. Go to [Ubuntu 22.04 Download Page](https://cdimage.ubuntu.com/jammy/daily-live/current/) and Download the [64-bit PC (AMD64) desktop image](https://cdimage.ubuntu.com/jammy/daily-live/current/jammy-desktop-amd64.iso). A .iso desktop image will start downloading.  
   NOTE : Windows 10/11 systems are based on the AMD64/Intel-64 architecture. Make sure you use the AMD image for a Windows 10/11 system and not an ARMv8 image. This step might take time depending on your internet speed.

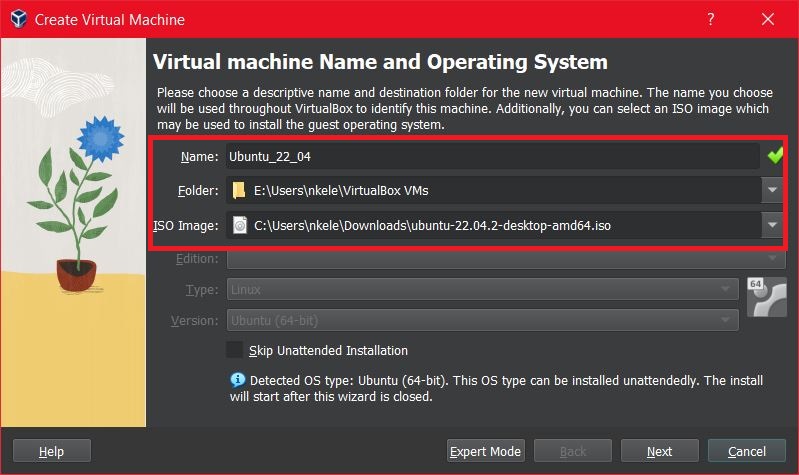
**Download and Install VirtualBox**

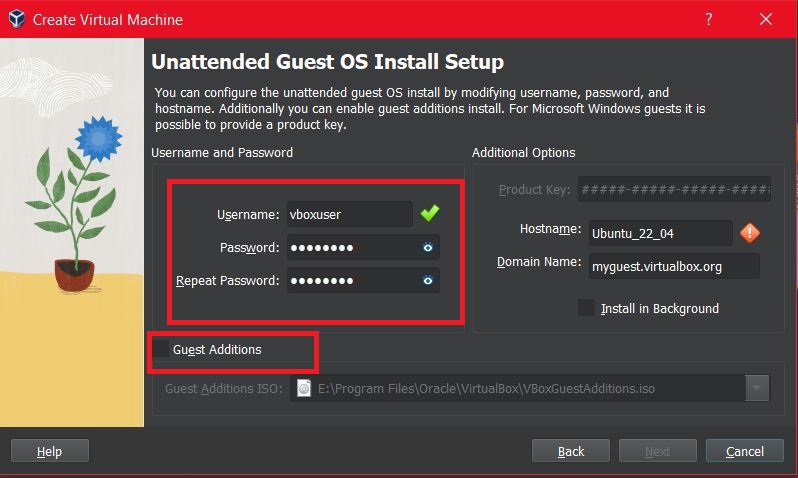
1. In this section, you will use the Oracle VirtualBox software. VirtualBox is a free general-purpose virtualizer available across Linux, Mac OS, and Windows.
2. Go to the VirtualBox Downloads page [Oracle VM VirtualBox Downloads page](https://www.virtualbox.org/wiki/Downloads) and click on the **Windows hosts** link to download the VirtualBox software. It will download a “.exe” installer file into your Downloads folder.
3. Locate and double-click the VirtualBox installer file from the Downloads folder in your File Explorer and it will launch the VirtualBox 7.0.8 Setup wizard.
4. Click **Next** on the welcome screen in the Setup Wizard to begin the process.
5. On the **Custom Setup** screen, you can leave the default selections for now so that the installer can set up Wizard to install the features. You can change the installation location by clicking on **Browse** or else leave the default location. Click **Next** when ready to continue.
6. The next screen will show a Warning about the network interfaces. The Setup Wizard will install a virtual network adapter that may reset and temporarily disconnect the network connection. Click on **Yes** to process the installation.
7. The next screen will ask you to confirm the installation. Click on **Install** to begin the installation of VirtualBox 7.0.8. The installation process will begin and it might take several minutes.
8. Once the installation is done, click on the **Finish** button to exit the Setup Wizard.
9. Launch the VirtualBox Manager from the **Start** Menu.

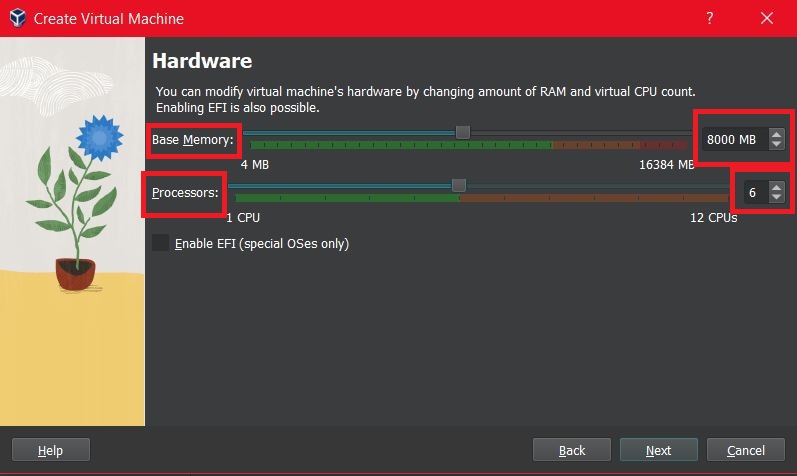
**Creating a Virtual Machine**

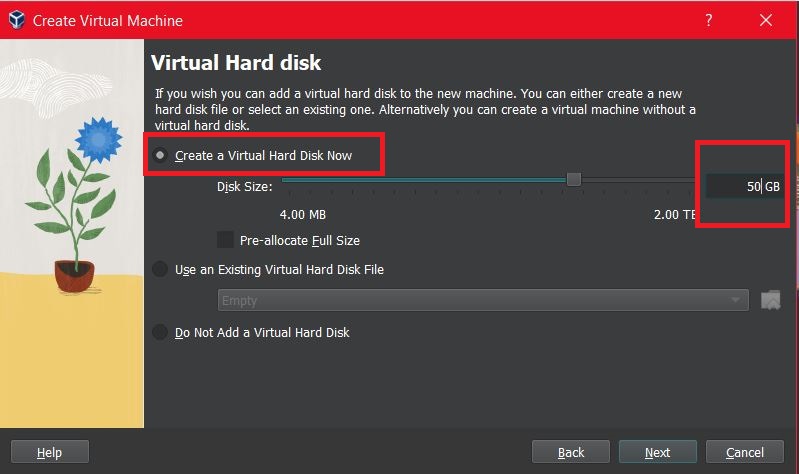
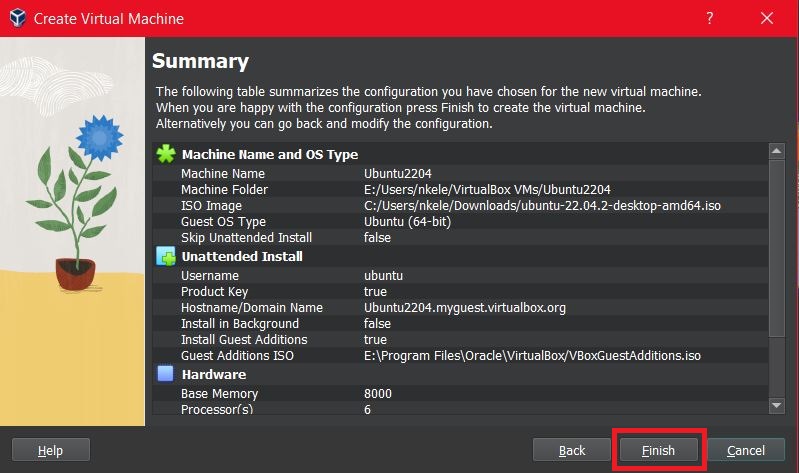
Once the VirtualBox Software is installed properly, you can begin creating your Virtual Machine.

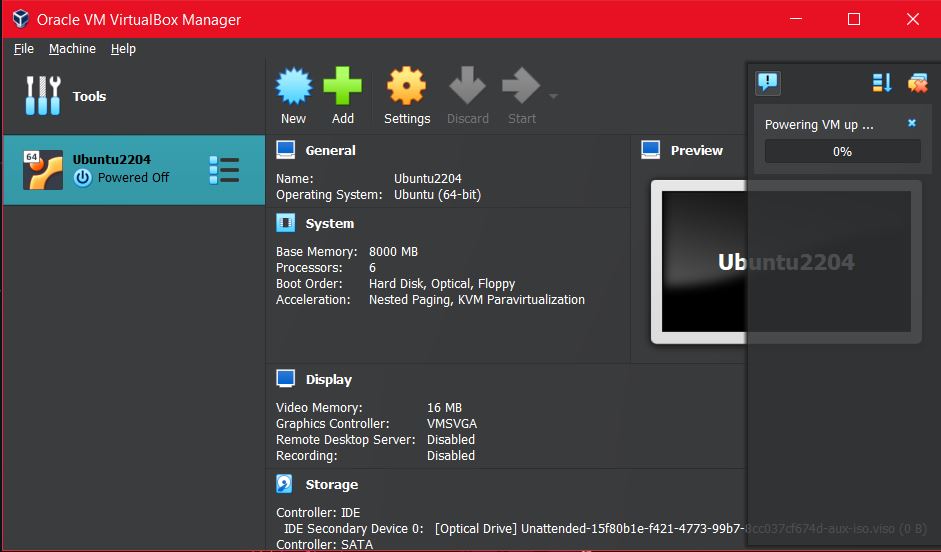
1. Click the **New** button from the top Menu on the **VirtualBox Manager** window. This will open the **Create Virtual Machine** wizard which will let you create and configure your new Virtual Machine with the desired settings.
2. On the **Virtual Machine Name and Operating System** section, fill in the name, Folder (location to store the Virtual machine files), and select the **Ubuntu 22.04 ISO Image** (downloaded in the previous section) in the ISO Image option. Click on the **Next** button to process.



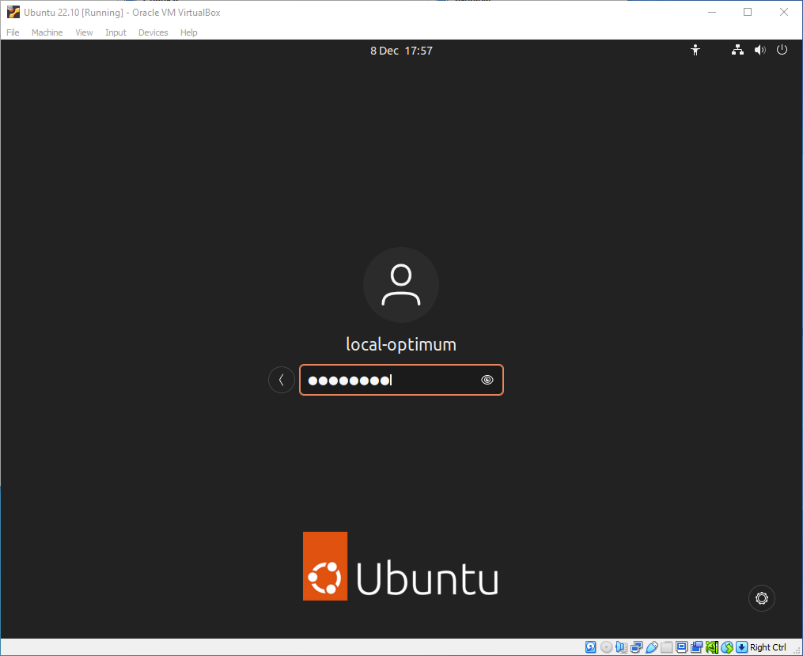
1. On the **Unattended Guest OS Install Setup** section, you need to enter your username and password in addition to your machine name so that it can be configured automatically during the first boot.
2. Also, check the **Guest Additions** box to install the default Guest Additions ISO that is downloaded as part of VirtualBox.
3. In the next **Hardware** section, specify how much of your host machine’s RAM and processors the virtual machine can use. Change the slider position to allocate the correct values.  
   **NOTE:** For using ROS 2 application, it is recommended to use a minimum of 8 GB RAM and at least 6 CPUs.



1. Next in the **Virtual Hard Disk section**, specify the size of the hard disk for the virtual machine. For ROS 2, it is recommended around 50 GB as a minimum.
2. Click **Next** to continue and view a summary of your virtual machine setting. After this click **Finish** to initialize the machine.
3. You will see a message saying Powering VM up … in the VirtualBox manager and a new desktop Window will appear for your Ubuntu 22.04 Virtual machine’s first boot.



1. On the first boot, it will begin the unattended installation.
2. Once the installation completes, the machine will automatically reboot to complete the installation.
3. Finally, you will be greeted with the Ubuntu **log-in screen** where you can enter your username and password defined during the previous setup.



1. You may need to add your username to the sudo group.

Note: If you chose to have VirtualBox automatically create your first user for you when setting up Ubuntu, you may get an error when using the sudo command. This is because the first user is not added to sudoers group as one would expect.

To fix this, go into terminal and

1. type su -

This will log you in as root. Your prompt should change to a "#" explanation: VirtualBox automatically set the root password as the same as your user password. It's not great behavior, but that is what it does.

1. type sudo adduser [username] sudo

Where [username] is your user name "student215". This adds your user to the sudo users list.

**Install ROS 2 Humble Hawksbill**

To install the latest LTS version of ROS 2, i.e., Humble Hawksbill, You can use an installer script that has been tested to work on x86 and ARM based dev boards ie. Raspberry Pi4/Nvidia Jetson Series. Run these commands from within the virtual machine.

cd ~/

git clone <https://github.com/linorobot/ros2me.git>

cd ros2me

./install

At this point you will be able to establish a remote connection with your robot and take a test drive

**Controlling the robot remotely from the host machine**

**Data Distribution Service (DDS) Networking Configuration**

Connect your robot and the host computer to the same network. **You cannot do this on the University WiFi Network.** Use the access point provided. The network must support multicast routing and both machines must be on the same domain.

Use the command:

ifconfig

to get the network interface and the wireless IP address information for each of the two machines you want to connect together.

A screenshot of a computer

Description automatically generated

Network interface

IP address

modify the cyclonedds.xml file to reflect this information for each machine. Be sure to include the peer IP addresses for both machines in the cyclonedds.xml file.

A screen shot of a computer

Description automatically generated

IP addresses for all machines

Network interface

Copy it to the directory by typing:

cp cyclonedds.xml $HOME/linorobot2\_ws/src/cyclonedds/cyclonedds.xml

Add this line to your .bashrc file by typing:

echo "export CYCLONEDDS\_URI=$HOME/linorobot2\_ws/src/cyclonedds/cyclonedds.xml">> ~/.bashrc

To test this connection examine the available ros2 topics now appearing on the host machine by typing:

ros2 topic list

ros2 topic echo /battery

**Run Keyboard Teleop on the host machine and take a drive around**

Run [teleop\_twist\_keyboard](https://index.ros.org/r/teleop_twist_keyboard/) to control the robot using your keyboard:

ros2 run teleop\_twist\_keyboard teleop\_twist\_keyboard

Press:

* **i** - To drive the robot forward.
* **,** - To reverse the robot.
* **j** - To rotate the robot CCW.
* **l** - To rotate the robot CW.
* **shift + j** - To strafe the robot to the left (for mecanum robots).
* **shift + l** - To strafe the robot to the right (for mecanum robots).
* **u / o / m / .** - Used for turning the robot, combining linear velocity x and angular velocity z.

**Install Parts of Linorobot2 on the host machine:**

mkdir ~/linorobot2\_ws

cd ~/linorobot2\_ws

git clone -b humble https://github.com/linorobot/linorobot2 src/linorobot2

rosdep update && rosdep install --from-path src --ignore-src -y --skip-keys microxrcedds\_agent --skip-keys micro\_ros\_agent

colcon build

source install/setup.bash

**Install RVIZ Configuration Files on the host machine:**

Install [linorobot2\_viz](https://github.com/linorobot/linorobot2_viz) package to visualize the robot remotely specifically when creating a map or initializing/sending goal poses to the robot. The package has been separated to minimize the installation required if you're not using the simulation tools on the host machine.

cd ~/linorobot2\_ws

git clone https://github.com/linorobot/linorobot2\_viz src/linorobot2\_viz

rosdep update && rosdep install --from-path src --ignore-src -y

colcon build

source install/setup.bash

At this point you are ready to navigate your robot autonomously! Great job!

**Part 3: Navigation and Mapping**

**Creating a map**

**Run rviz2 to visualize the robot from host machine:**

The rviz argument on slam.launch.py won't work on headless setup but you can visualize the robot remotely from the host machine:

ros2 launch linorobot2\_navigation slam.launch.py rviz:=true

**Move the robot to start mapping**

Drive the robot manually until the robot has fully covered its area of operation. Alternatively, you can use the 2D Goal Pose tool in RVIZ to set an autonomous goal while mapping. More info [here](https://navigation.ros.org/tutorials/docs/navigation2_with_slam.html).

**Save the map**

cd ~/linorobot2\_ws/src/linorobot2/linorobot2\_navigation/maps

ros2 run nav2\_map\_server map\_saver\_cli -f <map\_name> --ros-args -p save\_map\_timeout:=10000.

Change <map\_name> to the name of the map you wish to use without an extension.

**Autonomous Navigation**

**Load the map you created:**

Open linorobot2/linorobot2\_navigation/launch/navigation.launch.py and **change <*map\_name>* to the name of the newly created map.** The map argument can be used when launching Nav2 to dynamically load map files. For example **this is all one line**:

ros2 launch linorobot2\_navigation navigation.launch.py map:= ~/linorobot2\_ws/src/linorobot2/linorobot2\_navigation/maps/<map\_name>.yaml rviz:=true

**Run rviz2 to visualize the robot from host machine:**

The rviz argument for navigation.launch.py won't work on headless setup but you can visualize the robot remotely from the host machine using the rviz:=true argument.

**Initialize the Location of your robot**[**¶**](https://navigation.ros.org/tutorials/docs/navigation2_on_real_turtlebot3.html#initialize-the-location-of-turtlebot-3)

First, find where the robot is on the map. Check where your robot is in the room. Set the pose of the robot in RViz. Click on the 2D Pose Estimate button and point the location of the robot on the map. The direction of the green arrow is the orientation of your robot.

**Send a Goal Pose**[**¶**](https://navigation.ros.org/tutorials/docs/navigation2_on_real_turtlebot3.html#send-a-goal-pose)

Pick a target location for your on the map. You can send a goal position and a goal orientation by using the **Nav2 Goal** button.

Note: Nav2 Goal button uses a ROS 2 Action to send the goal and the GoalTool publishes the goal to a topic.

Check out Nav2 [tutorial](https://navigation.ros.org/tutorials/docs/navigation2_on_real_turtlebot3.html#initialize-the-location-of-turtlebot-3) for more details on how to initialize and send goal pose.