Setup Instructions for 31390

Unmanned Autonomous Systems 31390
June 2020
Department of Electrical Engineering • Technical University of Denmark

Introduction:

The purpose of this guide is to setup the system to run the drone simulator and software tools needed to be able to complete the assignments for the course.

The course focuses on Autonomous planning and control of Unmanned Systems, and we will particularly work using the Ardrone and Hummingbird simulator. In order for you to being able to work with the drones, you need to setup your computer and environment. Precisely, what you need is:

- Ubuntu 18.04
- Matlab 2019b with Simulink plus some additional packages.
- ROS Melodic
- Gazebo9
- RotorS simulation package for ROS and Gazebo

You will now have 2 options: Install Ubuntu 18.04 on your own machine together with everything needed (in a partion or whatever you chose) or use the Virtual Box image provided which contains Ubuntu 18.04 with ROS and Gazebo + packages. If you choose to install everything on your own go to Chapter 1 and if you choose to use the virtual machine go to Chapter 2.

NOTE:
Some of the bash commands might miss a space at the line breaks if they are copied directly so check when copying and if they cause errors this might be the cause.
NOTE:
You will need 30 GB of free space on your computer to get everything set up.

Chapter 1

Full install

This instruction assumes you have an installation of Ubuntu 18.04 Bionic Beaver ready to go.

1.1 Installing ROS Melodic

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms citeROSpage.

We will use the ROS Melodic Morenia release, which is the 12th official ROS release. The following instructions are taken from the official ROS installation guide:

http://wiki.ros.org/melodic/Installation/Ubuntu

First we need to add the ros packages to the apt source.list:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release
↪  -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Next we add the needed keys for getting ROS packages:

```
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
↪  C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

First, make sure your Debian package index is up-to-date and then install the full ros melodic version:

```
1 $ sudo apt update
2 $ sudo apt install ros-melodic-desktop-full
```

After the install we need to source the ros location to the `bashrc` script:

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Lastly we need to install some dependecies:

```
$ sudo apt install python-rosdep python-rosinstall
↪  python-rosinstall-generator python-wstool build-essential
```

After installing we need to initialize and update the rosdep:

```
$ sudo apt install python-rosdep
$ sudo rosdep init
$ rosdep update
```

1.2 Installing rotorS

For this course we will use the simulation environment rotorS:
`https://github.com/ethz-asl/rotors_simulator`

First we will install some build tools and other ros packages needed:

```
$ sudo apt install ros-melodic-joy ros-melodic-octomap-ros
 ↪  ros-melodic-mavlink
$ sudo apt install python-wstool python-catkin-tools protobuf-compiler
 ↪  libgoogle-glog-dev ros-melodic-control-toolbox
$ rosdep update
$ sudo apt install python-rosinstall python-rosinstall-generator
 ↪  build-essential
```

Once we have installed the needed packages we first need to setup a workspace and download the needed source material from github:

```
$ mkdir -p ~/ROS/31390_ws/src
$ cd ~/ROS/31390_ws/src
$ catkin_init_workspace
$ cd ~/ROS/31390_ws/src
$ git clone https://github.com/AnandaNN/rotors_simulator.git
$ git clone -b med18_gazebo9 https://github.com/gsilano/mav_comm.git
$ cd ~/ROS/31390_ws
```

Next we need to install some dependencies. While still in the `31390_ws` run

```
$ rosdep install --from-paths src -i
$ sudo apt install ros-melodic-rqt-rotors ros-melodic-rotors-comm
 ↪  ros-melodic-mav-msgs ros-melodic-rotors-control
$ sudo apt install ros-melodic-rotors-gazebo ros-melodic-rotors-evaluation
 ↪  ros-melodic-rotors-joy-interface
$ sudo apt install ros-melodic-rotors-gazebo-plugins
 ↪  ros-melodic-mav-planning-msgs ros-melodic-rotors-description
 ↪  ros-melodic-rotors-hil-interface
```

With everything installed we update the rosdep and finally build the CrazyS packages:

```
$ rosdep update
$ catkin_make
```

If everything build without problems we need to source out workspace to `bashrc`:

```
$ echo "source ~/ROS/31390_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

The last step before testing rotorS we need to update the Gazebo simulator. To do so run the following commands:

```
$ sudo sh -c 'echo "deb
 ↪  http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release
 ↪  -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
$ wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add
 ↪  -
$ sudo apt update
$ sudo apt install gazebo9 gazebo9-* ros-melodic-gazebo-*
$ sudo apt upgrade
```

1.2.1 Testing rotorS

To test if rotorS is installed correctly we run the hovering test example:

```
$ roslaunch rotors_gazebo mav_hovering_example.launch
```

If everything is working Gazebo should open and and spawn the Hummingbird. After 5 seconds (simulation time) the Hummingbird should start hovering and go to 1 meters up. To easy see what position the drone is at open a new terminal and run:

```
$ rostopic echo /hummingbird/odometry_sensor1/position
```

1.3 Installing Matlab with ROS

First install MatLab R2019b from:
`https://www.mathworks.com/academia/tah-portal/danmarks-tekniske-universitet-870549.html`

You will need to have a Mathworks user with your DTU email. This should download a `.zip` file with the MatLab installer. Unzip/extract it:

```
$ mkdir ~/Downloads/matlab_install
$ sudo apt install unzip
$ unzip -x Downloads/matlab_R2019b_glnxa64.zip -d Downloads/matlab_instal
```

Now the MatLab installer can be started with:

```
$ sudo ~/Downloads/matlab_install/install
```

When installing just install it to the default suggested location like in Figure 2.4.

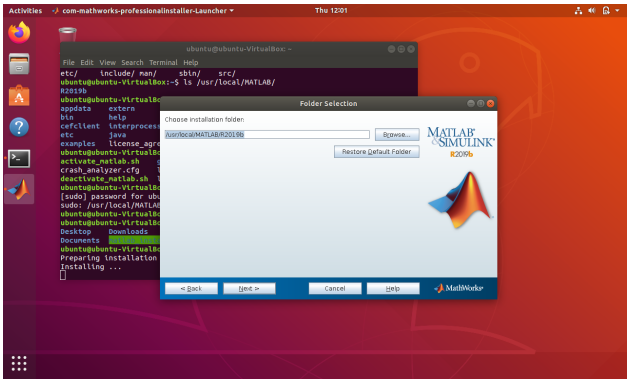

Figure 1.1: The default MatLab installation folder when installing

Follow the installer and if it ask if you wish to install even if there is not enough space say 'Yes'. You will need a Mathworks account with your DTU credentials.

When you get to the list of products to install, see Figure 2.5, chose the following products:

1. MatLab
2. Simulink
3. MATLAB Coder
4. Robotics System Toolbox
5. ROS Toolbox
6. Simulink Coder

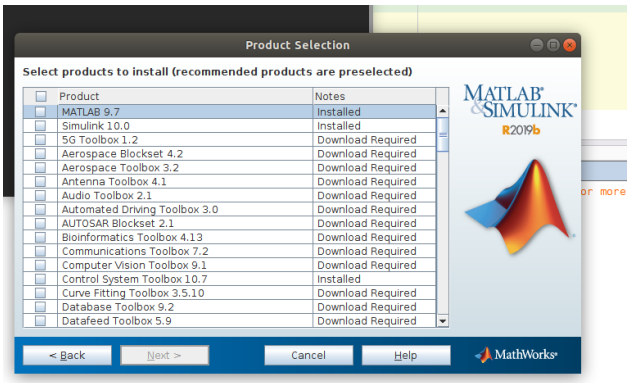

Figure 1.2: List of available MatLab products for installation

When asked if a symbolic link should be created, check that box, like on Figure 2.6. When activating MatLab remember when choosing the user name(Login name) of the computer to user your Ubuntu username(See Figure 2.7).

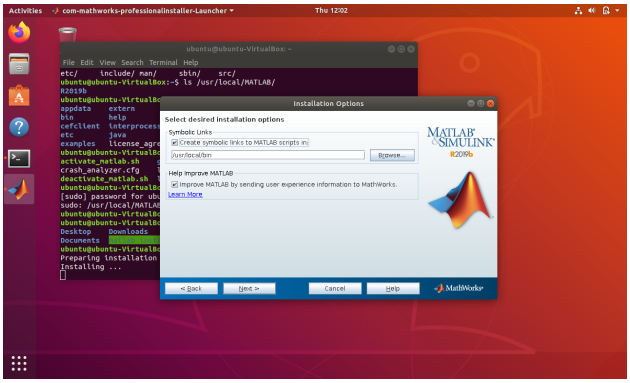

Figure 1.3: Installer option to create a symbolic link for MatLab

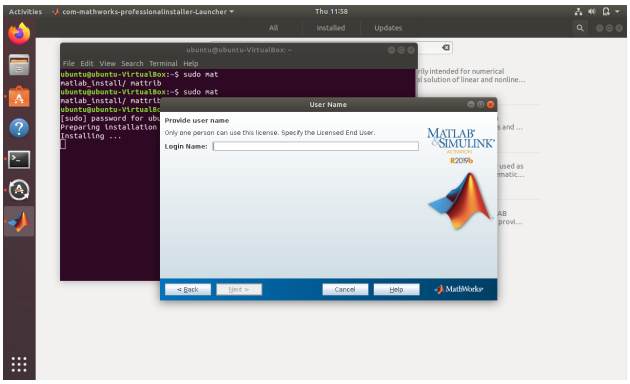

Figure 1.4: Choosing the user name or login name for the ubuntu account using MatLab

After the installer completes and activation is doneyou have 2 options. To run Matlab you can simple open a terminal and run: `matlab`. This will require you to keep that terminal

open as long as MatLab runs.

Another option is to install the MatLab header for Ubuntu. First open 'Ubuntu Software' (the Orange bag with an 'A' on it in the left menu). See Figure 2.8. In 'Ubuntu Software' search for and install MatLab, See Figure 2.8.

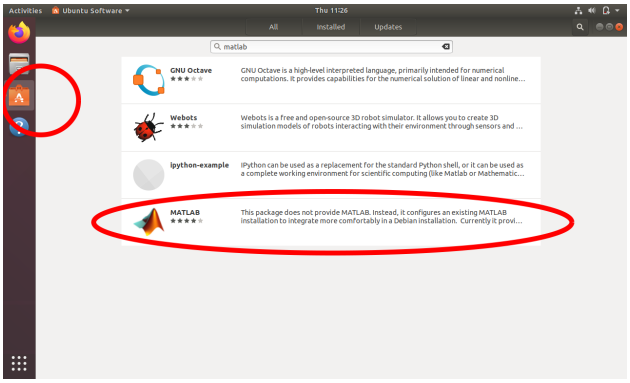

Figure 1.5: Ubuntu Software for installing MatLab wrapper

When installing the wrapper you will be asked for the MatLab location, which is should found on its own unless you changed it when installing MatLab'. The default location will be:

```
1 /usr/local/MATLAB/R2019b
```

When asked to rename libraries don't do anything and everything should install without problems.

Once the wrapper is installed you can open MatLab as a regular program by just searching for it in Ubuntu and if you want pin it to your task bar on the side.
Next we need to install the needed MatLab packages from the the Add-On manager. Open the Matlab Addon explorer by clicking on Add-on on the Home tab of Matlab, see Figure 2.9. In the Add-On manager search for and install the following 2 packages:

1. ROS Toolbox Interface for ROS Custom Messages
2. Robotics System Toolbox UAV Library

NOTE: If your resolution is not high enough you may need to press on 'Environment' in the top bar to find the 'Add-on' button.

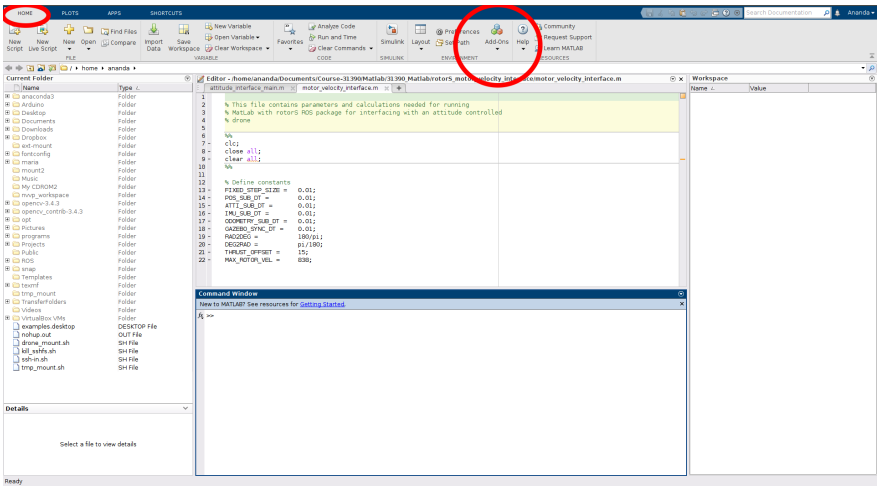

Figure 1.6: Open the Add-On manager

If you for some reason forgot to install any of the packages this can always be done in the Add-On explorer.

1.3.1 Configuring ROS messages

First download the following folder with ROS messages:
`https://github.com/gsilano/CrazyS/wiki/files/customMessages.rar`
Create a folder for the messages and unrar the folder there:

```
1 $ mkdir -p ~/ROS/customMessages
2 $ sudo apt install unrar
3 $ unrar x ~/Downloads/customMessages.rar ~/ROS/customMessages
```

When downloaded and extracted open MatLab and run the MatLab-commands below in the MatLab console. In line 2 change the `userFolder`-path to the location of the custom messages extracted above (if you ran the above lines change `ananda` to your Ubuntu username)

```
1 examplePackages = fullfile(fileparts(which('rosgenmsg')), 'customMessages')
2 userFolder = '/home/ananda/ROS/customMessages/'
3 copyfile(userFolder, examplePackages)
4 folderpath = userFolder;
5 rosgenmsg(examplePackages)
```

When the commands are run you should see the following message in the MatLab console. Follow the 3 steps. You will probably be asked to create the file `javaclasspath.txt`, just press yes:

```
1 To use the custom messages, follow these steps:
2
3 1. Edit javaclasspath.txt, add ...
```

```

...
```

If the last step fails and you get a 'Could not save path error' save the path to the startup folder of MatLab instead (the folder you get by typing `cd` in MatLab right after launch). If that folder is e.g. `'/home/ananda/'` then run the following command in MatLab, where you again should change `ananda` to your username:

```
savepath /home/ananda/pathdef.m
```

Follow the 3 steps to finish the installation of the ROS messages. Restart MatLab after finishing. When running `'rosmsg list'` you should see messages under `mav_msgs/` and `trajectory_msgs/`.

1.3.2 Installing the Gazebo Simulink plugin

To install the gazebo-simulink-plugin run the following command in the MatLab console:

```
packageGazeboPlugin
```

This will download a folder, `packageGazeboPlugin`, with the needed files for installing the plugin. Copy the folder into the ROS directory. Next we need to compile it. In a terminal run:

```
cd ~/ROS/GazeboPlugin
mkdir build
cd build
cmake ..
make
```

Now everything needed for the course should be installed and ready to run.

1.3.3 Testing MatLab with ROS

Download the `31390_matlab` folder from `https://github.com/AnandaNN/31390_matlab.git` or clone the repository into a folder with `git clone`.

To test if everything is set up and working run the test example. First open MatLab and go to the following folder called `initial_test` in the folder you just downloaded.

Open and run the `.m`-file to declare needed variables and open the Simulink model: (`initlial_test.slx`).
Before running the Simulink model it is necessary to start Gazebo and ROS. Start this by opening a terminal and running:

```
roslaunch rotors_gazebo mav_with_position_controller.launch
```

After running this command a Gazebo window should open with a Hummingbird quadcopter spawned. Once you see this go to your simulink model and click on 'RUN' in the 'Simulation' tab, see Figure 1.7. Gazebo you should now open and you see the drone fly up and fly in a small square.

When running the simulation it may freeze up. This does not have any effect and you just need to wait until it continues.

Stop the simulation by clicking on 'stop' in Simulink and run a `Ctrl+c` in the terminal with Gazebo. Always stop Simulink before Gazebo.

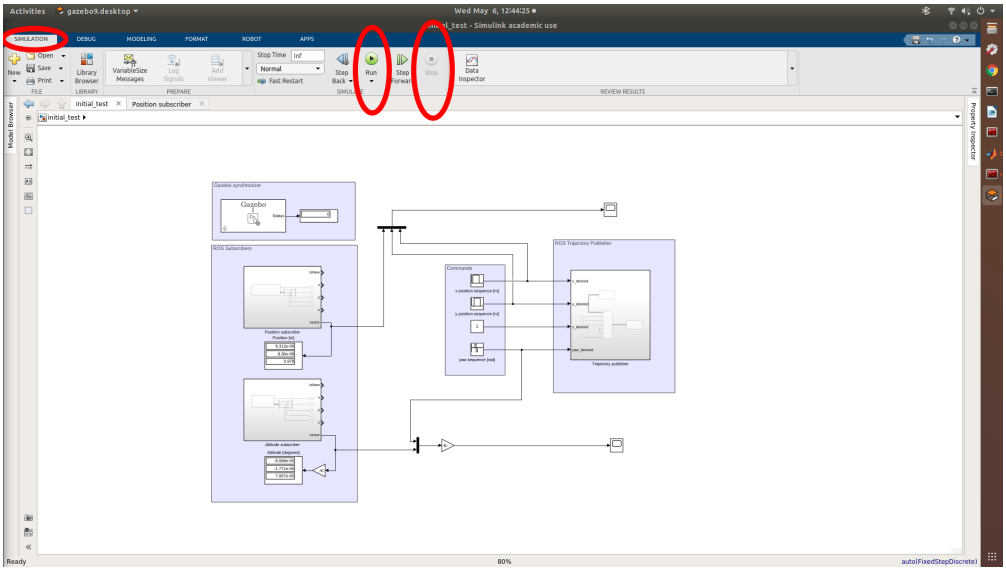

Figure 1.7: Your simulink window should look like this when opening the initial_test.slx

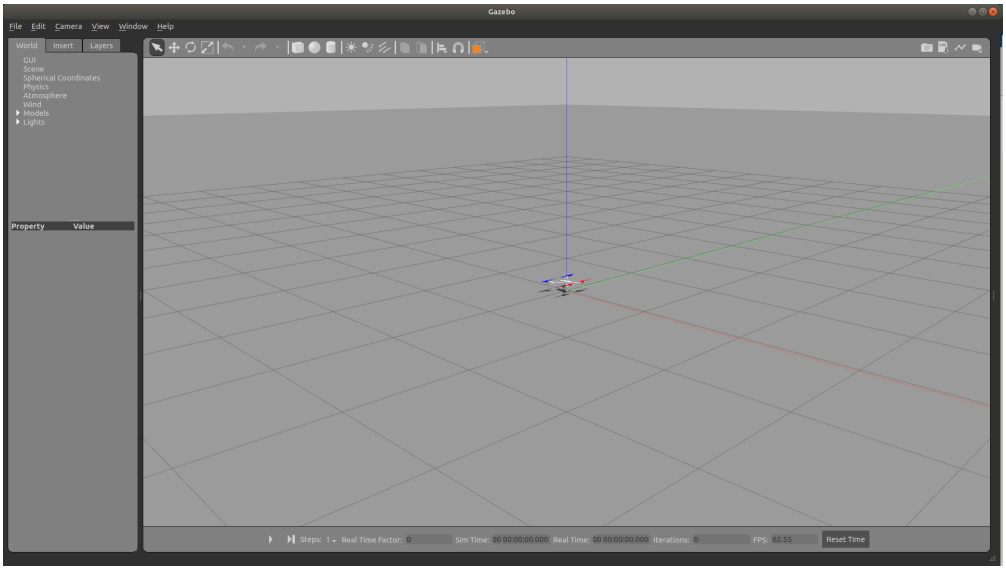

Figure 1.8: The Gazebo window

NOTE: Because of time synchronization we need to do a full reset of Gazebo and ROS before the simulation can be run again.

Chapter 2

Installing 31390 Virtual machine image

This section will explain how to install the needed virtual box image and get it ready for the course. First download and install Virtual Box to your system from:

`https://www.virtualbox.org/wiki/Downloads`

See Figure 2.1, and choose your operating system and install Virtual Box.

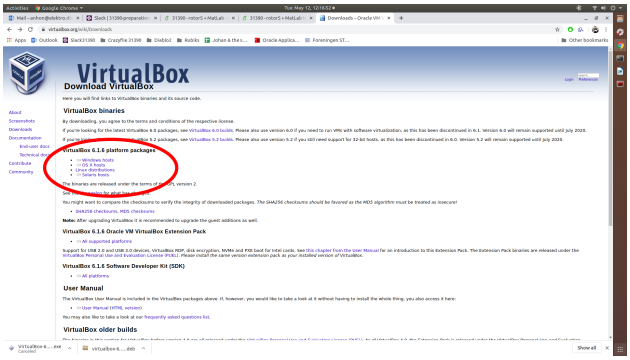

Figure 2.1: Choose your operating systems and download the correct installer

Once you have installed the virtual box download the virtual box image created for the course. Download all the file from this folder (31390_vm.z01, 31390_vm.z02, 31390_vm.z03, 31390_vm.zip) and have them placed in the same location.

`https://mega.nz/folder/rkwFDLhI#pgiJuBLMOplrDrTPDGKL7Q`

Unzip the folders by starting an unzip of the file with `.zip`-extension. Place the `31390-Ubuntu` folder to your desired location for VirtualBoxes.
Open Virtual Box and under the 'Tools'-tab click 'Add'. Navigate to the folder and select `31390-Ubuntu.vbox`.

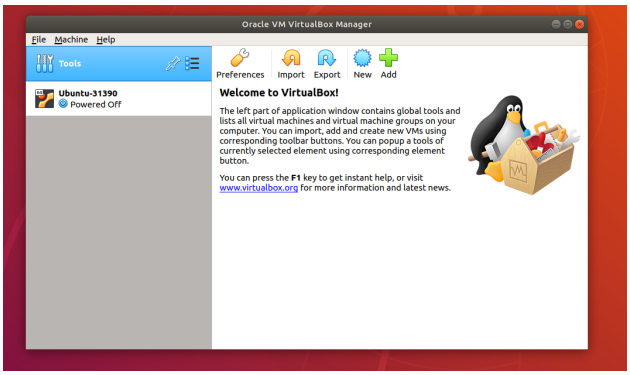

Figure 2.2: Virtual Box window. New images can be created and added from the tools menu and installed machines are accessed on the left.

Next select the now added virtual Box and click settings. In the settings windows go to the 'system' tab. Under 'Motherboard' set the 'Base memory' to how much RAM you want to assign to the image. Under 'Processor' set the 'Processor(s)' to the number of cores the image can use. Next go to the Display tab and increase the 'Video Memory' and check the 'Enable 3D Acceleration' box. If you place the sliders in the green range you should be fine. If you feel it is running slow you can increase it later. For reference my settings are 3 out of 4 cores, 7 GB out of 16 GB RAM and 128 MB graphics memory, but it has also run on much less hardware.

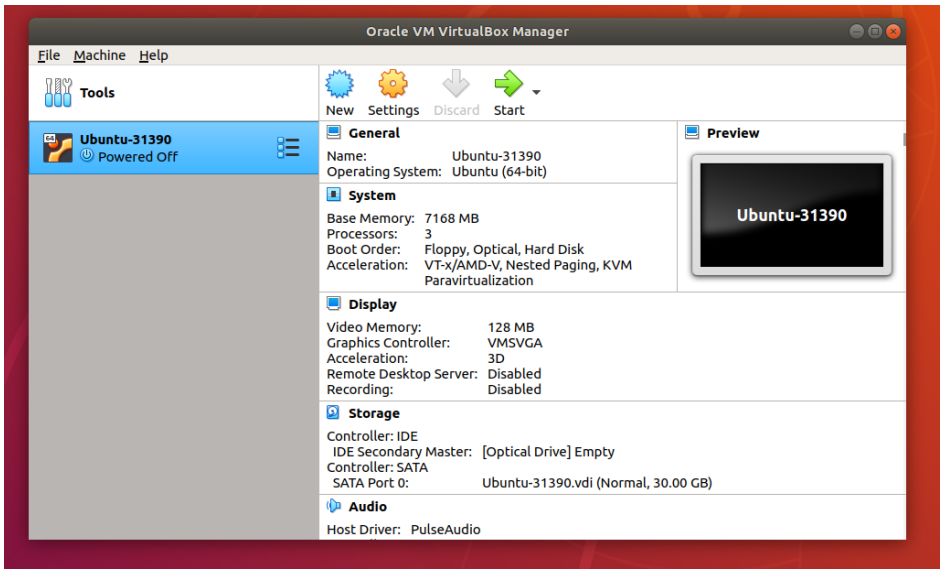

Figure 2.3: Virtual box window with a virtual box selected. Press settings(yellow cogwheel) to adjust hardware use. Start the machine by pressing start(green arrow).

Once updated launch the virtual machine and change resolution if you like. The virtual machine is set up with all the ROS and Gazebo packages, but doesn't have MatLab installed, which we will also need.

The username on the virtual machine is `ubuntu` and the sudo password is `grenen`

2.1 Installing Matlab with ROS

First wee need to install MatLab. Start the installer by running in a terminal:

```
$ sudo ~/matlab_install/install
```

When installing just install it to the default suggested location like in Figure 2.4.

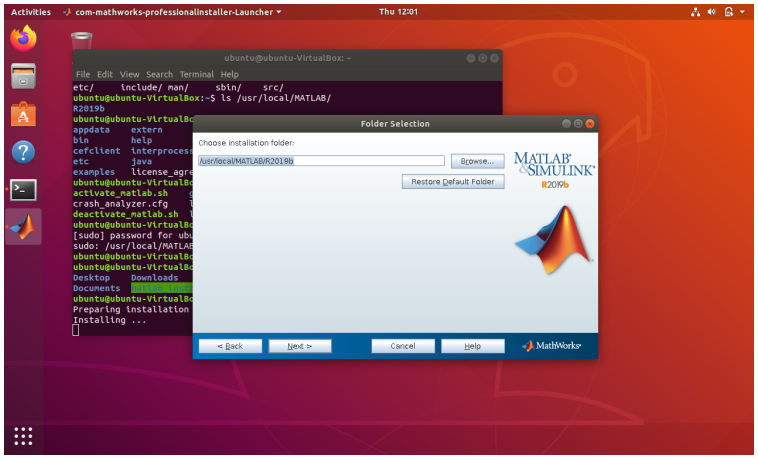

Figure 2.4: The default MatLab installation folder when installing

Follow the installer and if it ask if you wish to install even if there is not enough space say 'Yes'. You will need a Mathworks account with your DTU credentials.

When you get to the list of products to install, see Figure 2.5, chose the following products:

1. MatLab
2. Simulink
3. MATLAB Coder
4. Robotics System Toolbox
5. ROS Toolbox
6. Simulink Coder

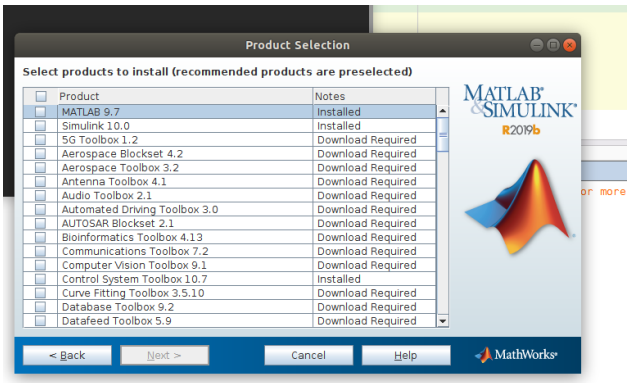

Figure 2.5: List of available MatLab products for installation

When asked if a symbolic link should be created, check that box, like on Figure 2.6. When activating MatLab remember when choosing the user name(Login name) of the computer to user the Ubuntu username(`ubuntu`)(See Figure 2.7).

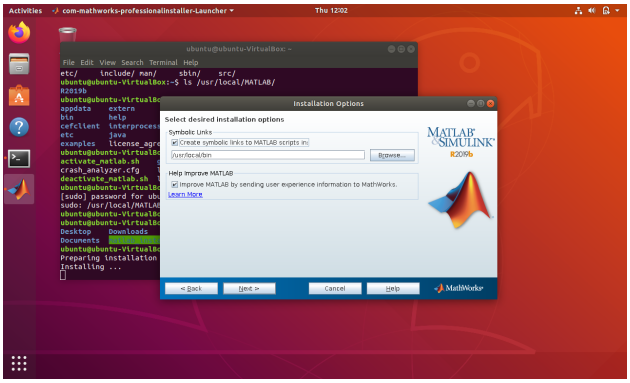

Figure 2.6: Installer option to create a symbolic link for MatLab

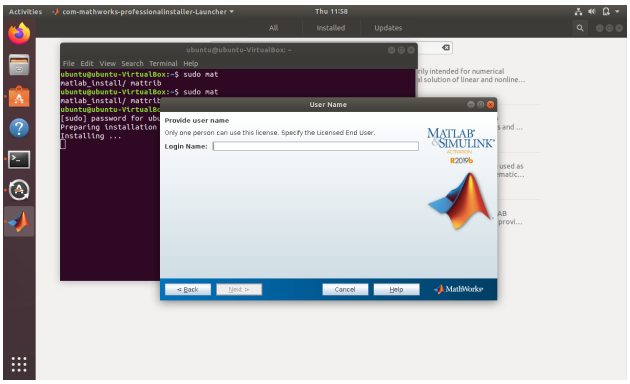

Figure 2.7: Choosing the user name or login name for the ubuntu account using MatLab

After the installer completes and activation is doneyou have 2 options. To run Matlab you can simple open a terminal and run: `matlab`. This will require you to keep that terminal

open as long as MatLab runs.

Another option is to install the MatLab header for Ubuntu. First open 'Ubuntu Software' (the Orange bag with an 'A' on it in the left menu). See Figure 2.8. In 'Ubuntu Software' search for and install MatLab, See Figure 2.8.

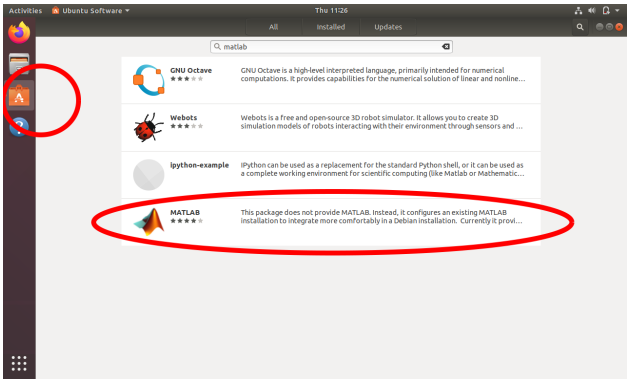

Figure 2.8: Ubuntu Software for installing MatLab wrapper

When installing the wrapper you will be asked for the MatLab location, which is should found on its own unless you changed it when installing MatLab'. The default location will be:

```
1 /usr/local/MATLAB/R2019b
```

When asked to rename libraries don't do anything and everything should install without problems.

Once the wrapper is installed you can open MatLab as a regular program by just searching for it in Ubuntu and if you want pin it to your task bar on the side.
Next we need to install the needed MatLab packages from the the Add-On manager. Open the Matlab Addon explorer by clicking on Add-on on the Home tab of Matlab, see Figure 2.9. In the Add-On manager search for and install the following 2 packages:

1. ROS Toolbox Interface for ROS Custom Messages
2. Robotics System Toolbox UAV Library

NOTE: If your resolution is not high enough you may need to press on 'Environment' in the top bar to find the 'Add-on' button.

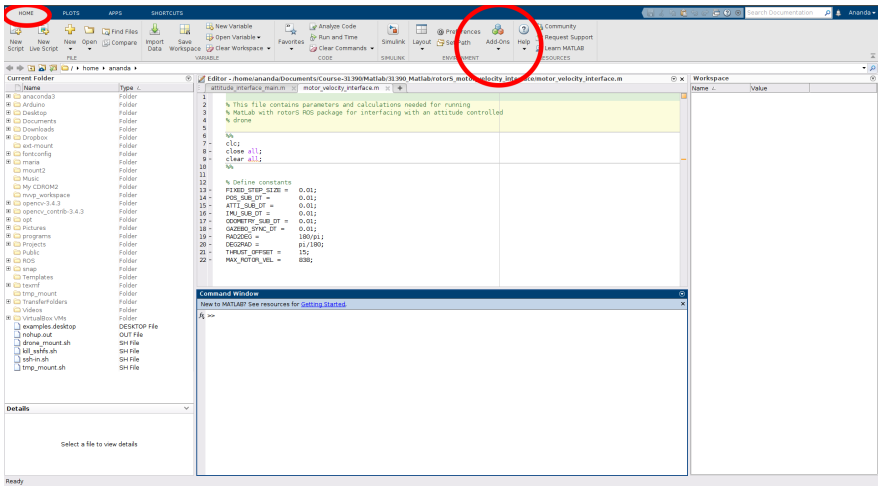

Figure 2.9: Open the Add-On manager

If you for some reason forgot to install any of the packages this can always be done in the Add-On explorer.

2.1.1 Configuring ROS messages

Open MatLab and run the following commands in the MatLab console:

```
examplePackages = fullfile(fileparts(which('rosgenmsg')), 'customMessages')
userFolder = '/home/ubuntu/ROS/customMessages/'
copyfile(userFolder, examplePackages)
folderpath = userFolder;
rosgenmsg(examplePackages)
```

When the commands are run you should see the following message in the MatLab console. Follow the 3 steps. You will probably be asked to create the file `javaclasspath.txt`, just press yes:

```
To use the custom messages, follow these steps:

1. Edit javaclasspath.txt, add ...

...
```

If the last step fails and you get a 'Could not save path error' save the path to the startup folder of MatLab instead by running the following command in the MatLab console:

```
savepath /home/ubuntu/pathdef.m
```

Follow the 3 steps to finish the installation of the ROS messages. Restart MatLab after finishing. When running `'rosmsg list'` you should see messages under `mav_msgs/` and `trajectory_msgs/`.

2.1.2 Testing Matlab with ROS

To test if everything is set up and working run the test example. First open MatLab and go to the following folder:

```
/home/ubuntu/Documents/31390_matlab/initial_test
```

Open and run the `.m`-file to declare needed variables and open the Simulink model: (`initlial_test.slx`). Opening the slx files can take a while.
Before running the Simulink model it is necessary to start Gazebo and ROS. Start this by opening a terminal and running:

```
roslaunch rotors_gazebo mav_with_position_controller.launch
```

After running this command a Gazebo window should open with a Hummingbird quadcopter spawned. Once you see this go to your simulink model and click on 'RUN' in the 'Simulation' tab, see Figure 2.10. Gazebo you should now open and you see the drone fly up and fly in a small square.

When the simulation is running it might freeze up. This does not affect the simulation you just need to wait a bit until it resumes.

Stop the simulation by clicking on 'stop' in Simulink and run a `Ctrl+c` in the terminal with Gazebo. Always stop the simulink model before Gazebo.

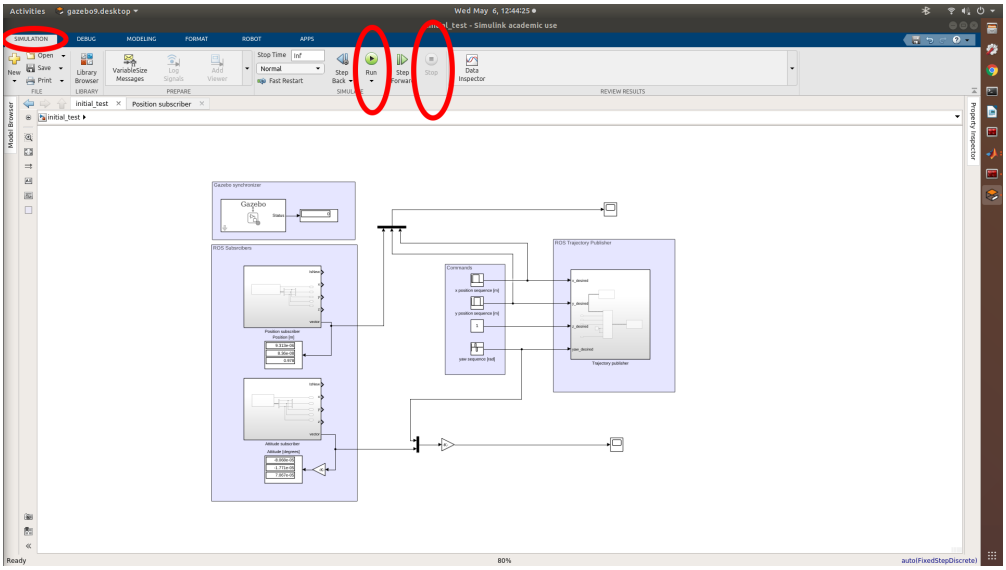

Figure 2.10: Your simulink window should look like this when opening the initial_test.slx

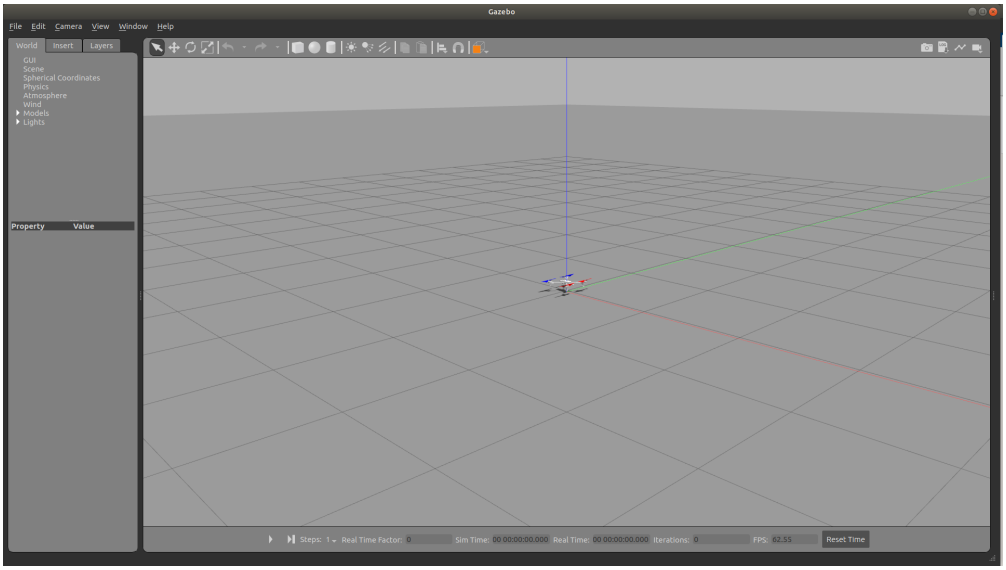

Figure 2.11: The Gazebo window

NOTE: Because of time synchronization we need to do a full reset of Gazebo and ROS before the simulation can be run again.