Meeto your Neato!

Overview -

In module 3 we will be programming the Neato BotVac. The Neato is a powerful, low cost robot platform that we have customized for QEA (it's also technically a vacuum cleaner, but we'll be ignoring that for this module!). We have engineered the platform to abstract away a lot of the frustrating bits, allowing you to focus on learning the really fun robotics, physics, math, and computing content.

The Neato in All Its Glory!

The robots have each been outfitted with a Raspberry Pi. The Raspberry Pi is a low cost, Linux computer that will serve as a bridge between your laptop and the robot. To use the robot you will initiate a connection from your laptop, via the Olin network, to the Raspberry Pi. Once the connection has been made, the Raspberry Pi will start talking to the robot. All sensor data will then be streamed from the Raspberry Pi to your laptop over Olin's wireless network. Your laptop will process this sensor data and then send motor commands to the robot over the same network. In this way, all important computation will be done on your laptop. This architecture simplifies the sharing of robots and makes debugging and editing code as easy as possible.

Since you are not modifying the code running on the Raspberry Pi, each robot will function identically (i.e. there is no software you need to modify on the robot). While we do have enough robots (18) to give each table their own robot, that wouldn't be optimal. For example, sometimes your robot will run out of batteries, and you'll want to grab another robot while yours charges.

We'll be programming the robots using ROS. ROS is a powerful, open-source framework for robotics that has been widely adopted in both industry and academia. ROS runs natively on Linux, however, we are supporting Windows via Docker. Therefore, you can connect to the robots through either Windows or Linux (instructions for both are available here). We are not able to support Mac OSX at this time (see Target Environment for more details).

Environment Setup

The purpose of this document is to get you up and running with your Neato robot. After following these instructions you will be able to connect to your robot, examine its sensor data, and drive it around.

Target Environment

We will be using ROS (Robot Operating System) to program our robots. ROS is a powerful platform for advanced robotics work in both industry and academia. ROS supports several programming languages out of the box, including C++, Python, and Java. Additionally, MATLAB's Robotics System Toolbox includes support for ROS as well. In this document and in this module, we will walk you through setting up your environment to connect to the robots using Windows, and we will show you how to program the robots using MATLAB. Some folks, especially those in SoftDes, may want to use Ubuntu instead. This is totally fine, and we have instructions on how to setup your environment for Ubuntu. Unfortunately, due to limitations in some of the software we will be using, we cannot support MacOSX. You may be tempted to use Python to program the robots, however, we discourage you from doing so. The reason is that we will have a lot of scaffolding / tutorials written in MATLAB, and the extra effort in mapping things over to Python is a hurdle that we don't want you to have to contend with. Additionally, there are some things we will be doing with the robots that will work *better* in MATLAB (e.g., the operations may be faster, visualization / debugging may be easier).

Docker Setup

The only operating systems officially supported by ROS are Ubuntu and Debian. However, using a virtual machine you will be able to run ROS in Windows 10. The tool we will be using to accomplish this is Docker. To install Docker, create an account and download and run the most recent version of docker from the docker installer page. **When you see two checkboxes, leave both as is.** You want to run Docker with Linux containers, not Windows containers, so that we can use Docker to run Linux code. There are some additional instructions on the Docker installation page that walk you through the install process. Once the install is complete you may be asked to restart your computer.

Downloading the QEA Docker Image

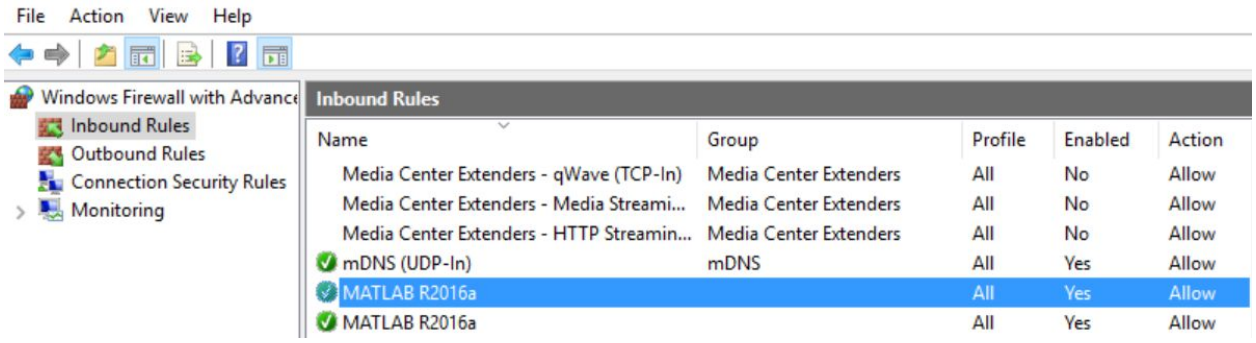
Once the Docker installation has completed (and you have possibly restarted your computer), you will see a message pop up that says something like “Docker is Running. Open PowerShell to start hacking” (it takes a bit of time for this to show up upon reboot, so be patient). Once you see this pop up, open the windows “Run” dialog box by hitting the Windows key and ‘r’ simultaneously. Type the command **cmd /c docker pull paulruvolo/neato_docker:qea** (Do not cut-and-paste, adds an erroneous space after the command) into the textbox that pops up, and then hit enter. We recommend doing this on the Olin network. The resulting series of downloads should take less than 10min, if it gets completely stuck for longer than that, just close the window and retry.

Checking Your Firewall Settings

In order to allow MATLAB to properly communicate with Docker, you need to make sure you have the right firewall settings. When you run MATLAB for the first time, you were asked whether or not to allow incoming connections to this application. If you selected “yes”, you are good. If you selected “no”, or if you don’t remember what you selected (which is probably most people!), then we need to make sure you have the right settings.

1. Type **wf.msc** into the run dialog (remember, hit the Windows key and “r” simultaneously to bring up the dialog). This will bring up a dialog that shows your firewall settings.
2. Click on Inbound Rules
3. Browse the list for two rules that say “MATLAB R2017a” (assuming you are using this version of MATLAB. If you are using a different version of MATLAB, you should look for rules labeled with that version.). If you do not see MATLAB, go on and come back to this later once you have connected to the robots.
4. Click on one of these two rules (you will repeat this process for both), and click the properties button on the right panel of the window.
5. Make sure the following settings are chosen:
    a. Under the “General” tab, make sure “Enabled” is checked and that “Action” is set to “Allow the connection”.
    b. Under the “Advanced” tab, make sure that “Domain”, “Private”, and “Public” are all checked and make sure that “Edge traversal” is set to “Defer to user”.
6. Repeat step 5 for the second firewall rule.

If you have done this correctly, your “Inbound Rules” list should look like the ones below (note: on this computer the version of MATLAB was R2016a).



Connecting to the Neatos

Step 1: Grab a battery for the raspberry Pi

Checklist before performing this step:

1. The battery indicator light should be at least level 2 (preferably full)

Step 2: Choose your Neato

Checklist before performing this step:

1. Make sure the Neato's batteries are charged. To test this, pull the Neato away from it's charging station and for the newer Neato's hit the button near the front bumper of the Neato that has the home icon on it. For the older Neato's hit the larger orange power button. The display should illuminate revealing a battery capacity indicator. Sometimes you will have to click the button below the display to dismiss any errors that show up on the Neato's screen before the battery level is displayed.

Step 3: Connect the USB battery pack to the Raspberry Pi's USB cable.

It should take about 1 minute for the robot to be ready to use (see step 4 for a final checklist).

Step 4: Connecting to the Neato from Your Laptop

Checklist before performing this step:

1. Raspberry pi display backlight is illuminated and not flashing on and off (see troubleshooting section for what to do if this is not the case)
2. Raspberry pi display shows that the Neato is connected to the OLIN-ROBOTICS network and has an IP address assigned to it.
3. Raspberry pi display shows that the signal strength of the Neato's connection is at least 70 (the max is 99) for the wifi dongles with antennas, and at least 50 (max is 65) for the dongles without antennas. **Note: If the signal strength is too low see troubleshooting section for more information.**
4. Your laptop is either connected to the ethernet or the OLIN wifi network **(Note: this will not work if you are on OLIN-GUEST).**

Open the run dialog (by hitting Windows key and "r"). Paste in the following command, and hit enter. Replace the part that says HOST=192.168.16.74 with the IP address of your robot (the IP address can be found from looking at the display of the Raspberry Pi on your Neato).

cmd /c docker run --net=host -e HOST=192.168.16.68 -it paulruvolo/neato_docker:qea

You can verify this worked because the robot will start making a quiet whirring sound and the laser (visible from the side) will start rotating. You should also see in the command window that you are "connected" and the packet loss should be 0%.

Programming the Robot in MATLAB

Next, fire up MATLAB. In order to connect MATLAB to the robot, type the following into the MATLAB command window.

rosinit('10.0.75.2',11311, 'NodeHost','10.0.75.1')

If all goes well you should see output similar to this.
Initializing global node /matlab_global_node_38893 with NodeURI http://10.0.75.1:57999/

Now that you are connected, you can see the list of topics by typing the following command into the command window.

rostopic list

Each of these topics is either a sensor channel (e.g., laser scanner, bump sensor, wheel encoder) or a motor control channel (e.g., cmd_vel, raw_vel, etc.). Go ahead and display the data flowing across the /bump topic by typing the following command in the command window.

rostopic echo /bump

You should see output like this.

```
  Data   :  [0, 0, 0, 0]
  Layout
    DataOffset :  0
    Dim        :  []
```

The interpretation of each of these numbers is dependent on the particular topic you are examining, however, in the case of the /bump topic the four numbers in Data correspond to the output of each of the four bump sensors on the robot. Go ahead and push the front bumper of the robot to see which bump sensor corresponds to which number.

As always, you can use Ctrl-C to terminate the execution of any Matlab script so you can keep programming.

Your First Robotics Program

Let's go ahead and create a program to drive the robot forward until it rams into something. To do this we'll need to first learn how to control the robot's wheels. In order to send a velocity to each of the robot's wheels we will need to create a publisher for the /raw_vel topic.

```
pub = rospublisher('/raw_vel');
```

Once we have a publisher, we can create a message suitable for sending on that topic.

```
msg = rosmessage(pub);
msg.Data = [.1, .1];
send(pub, msg);
```

This message corresponds to telling the robot's wheels to each move forward at a velocity of 0.1 m/s.

If your robot is not moving, return to your firewall settings and make sure **ALL** MATLAB inbound rules are as described above. (There could be new inbound rules now that you've connected to the robot.)

We can create subscribers to topics using the rossubscriber command.

sub_bump = rossubscriber('/bump');

We can put these two together to create a simple program where the robot will move forward with a constant velocity (in this case 0.1 m/s) until one of the bump sensors is triggered.

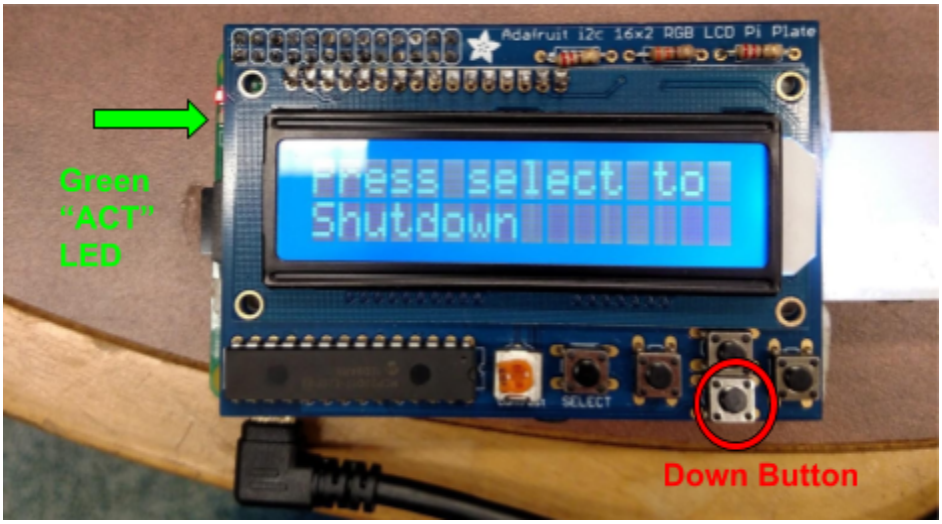
```
pub = rospublisher('/raw_vel');
sub_bump = rossubscriber('/bump');
msg = rosmessage(pub);

% get the robot moving
msg.Data = [0.1, 0.1];
send(pub, msg);

while 1
    % wait for the next bump message
    bumpMessage = receive(sub_bump);
    % check if any of the bump sensors are set to 1 (meaning triggered)
    if any(bumpMessage.Data)
        msg.Data = [0.0, 0.0];
        send(pub, msg);
        break;
    end
end
```

Step 5: Shutting Down the Raspberry Pi

When you are done working with the robot it is important to properly shutdown the raspberry pi. DO NOT just unplug the battery. To shutdown the pi, push the "down" button until you see the message "Press select to Shutdown". Press select and wait for the green "ACT" LED on the left side of the Pi to flash steadily ten times then stay off. It is then safe to unplug the battery.



Notes from working in Linux

First, make sure you are running Matlab r2016b or later. The installer can be found on Public, as described here (http://wikis.olin.edu/it/doku.php?id=matlab&s[]=matlab#linux). Two tips:

- Copy the .deb file from Public to your computer before installing to make the process go much faster
- If the version doesn't get updated properly, try using the command line **sudo dpkg -i Matlab_R2016b_9.1.0.441655-1.deb** instead of the GUI

Linux Docker install instructions can be found here:
https://docs.docker.com/engine/installation/linux/ubuntu/#install-docker
You want Docker CE, not Docker EE

Post installation steps;
https://docs.docker.com/engine/installation/linux/linux-postinstall/#manage-docker-as-a-non-root-user
Then restart computer

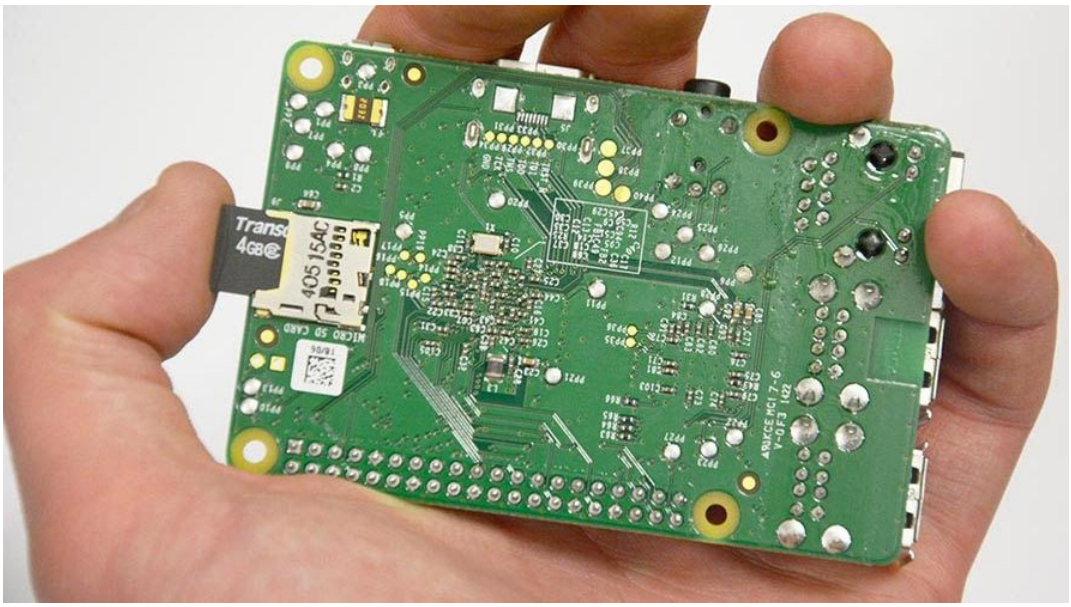
Docker pull the QEA image:
docker pull paulruvolo/neato_docker:qea

When you start the docker container, use the following slightly edited command:
docker run --net=host -e ROS_HOSTNAME=localhost -e HOST=192.168.16.68 -it paulruvolo/neato_docker:qea
remembering to replace the 192.168.16.68 with the actual IP of your robot.

In MATLAB, simply call **rosinit()** with no arguments. Everything else should be the same.

Troubleshooting

Symptom: Both the red and green LEDs on the raspberry pi are illuminated and not flashing.
Problem: the Pi was unable to boot from its SD card.
Solution 1: the first thing to check is that the Raspberry Pi's SD card is fully inserted into the Raspberry Pi. See the image below for the location of the SD card. You will know it is fully inserted if you push on the card and it clicks into place.



Solution 2: if the card is fully inserted, the SD card may have become corrupted (possibly because some people didn't properly shutdown the Raspberry Pi!). Please send me (Paul.Ruvolo@olin.edu) an e-mail and tell me which robot is having the problem. I'll fix it ASAP, but in the meantime just use another robot.

Symptom: the raspberry Pi display's backlight is flashing on and off.
Problem: the Pi cannot connect to the robot via the USB cable.
Solution: sometimes the Neato will turn off due to inactivity. Press button near the front of the Neato's bumper labeled with the home icon to wake your Neato up. If that doesn't work, shutdown and then reboot the Pi. If none of this works, the robot battery might be dead. Try recharging the robot. While the robot is recharging, switch to another robot.

Symptom: the Wifi signal strength indicator on the Raspberry Pi is below 60 even though you are right near an access point.
Problem: The Pi has connected to an access point that is not the closest one (this will sometimes happen).
Solution: Assuming the Pi display is at the screen showing the IP address, press right to enter the network setup menu. OLIN-ROBOTICS should be highlighted with an asterisk. Press right

again to reconnect the Pi to the Wifi. If it doesn't work the first time, try one more time. If it doesn't work then, switch to a new robot.