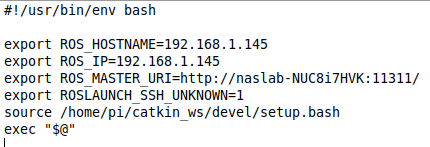
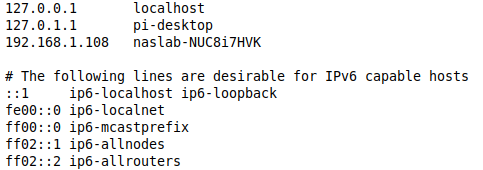
**Uctronics Car Setup – ROS Remote Machine**

The goal of this document is to detail how I was able to successfully connect to the Raspberry Pi’s on several Uctronics cars from a separate computer through ROS. This can be done by ssh’ing into every single car in a new terminal and individually typing in the lines to execute and launch the necessary lines of codes and nodes. However, this is a lengthy process and would be hard to do in conjunction with all the existing Crazyflie scripts I’ve written. Therefore, these steps explain what needs to be done to have several remote machines launch automatically from a single launch file on the main computer. This will detail all of the specific steps for setting up the mobile chargers in the Crazyflie mission planning system but can be easily adapted for other cases.

1. Flash the car with ‘spring2020’ image file
2. Use ifconfig to find the Raspberry Pi’s IP address in wlan0 (inet)
3. Type ‘nmtui’ and edit ‘naslab’ connection so that the pi is available to all users (hit space)
4. SSH into the pi for the first time using the ssh-rsa algorithm that can be specified in this way: “ssh -oHostKeyAlgorithms=’ssh-rsa’ pi@...”. Type yes to the question and type in the password to set up continue with the ssh
5. Setup a ROS workspace (already done if flashing the SD card with the ‘spring2020’ image file)
6. Create a package (autonomy) in that workspace (already done with ‘spring2020’)
   1. Make sure that the package is dependent on all of the packages that you may need (i.e. message\_runtime/message\_generation for custom messages, rospy, std\_msgs, geometry\_msgs, etc)
7. Create src, msg, and launch directories in that package (already done with ‘spring2020’)
8. Copy all necessary files (mobileCharger\_move.py, mobileCharger\_motors.py, etc) to the Raspberry Pi and put them in their respective folders (.launch -> launch, .msg -> msg, and .py/etc -> src)
9. Create or copy an environment file (custom\_env.sh) in my case, and put it in the devel folder for the workspace. The ROS\_IP should be changed to the Raspberry Pi’s IP address (use ifconfig/wlp6so/inet number). The environment file should look like this:
10. Change the /etc/hosts file to recognize the main computer’s hostname. It should look like this when the main computer’s hostname (Qualisys desktop computer) is ‘naslab-NUC8i7HVK’, and its network IP address is ‘192.168.1.108’ or ‘192.168.1.162’:
11. Add the pid.msg to the list of messages in the CMakeLists.txt file
12. Build the catkin\_ws using ‘catkin\_make’
13. If already ssh’ed into the pi without the oHostKeyAlgorithms tag, delete the last line in .ssh/known\_hosts on the main computer and regenerate it by ssh’ing into the pi using “ssh -oHostKeyAlgorithms=’ssh-rsa’ pi@...”
14. Either in a bash script that runs the launch file or before launching the launch file on the main computer, export the main computer’s IP address as so: ‘export ROS\_IP=192.168.1.108’.
15. In the launch file, use the XML <machine/> tag to specify the Raspberry Pi’s machine parameters and the <node/> tag to create a node on that machine. (The username and password for the pi with the ‘spring2020’ image file are ‘pi’ and ‘spring2020’ respectively.)
16. Launch the launch file!

Extra notes:

* PID gains can be tuned for angular/linear control in lin\_ang\_vel.py on the main computer
* Remember that any changes made in files on the Raspbery Pi must be done on the Pi itself or copied to it using secure copy (scp)’
* If you receive an error from the car node, the log says that it cannot import name pid, and the catkin\_make shows an error of the sorts ‘make -j4 -l4 failed’, do this:
  + Delete ‘build’ directory under ‘catkin\_ws’
  + Copy files from a working build directory (except ‘catkin\_make.cache’)
  + Rebuild the workspace using ‘catkin\_make’
  + Try recopying pid.msg and rebuilding if above steps don’t work