Cd Step 2: From SD image to RC control

Prerequisites:

- A complete Duckiebot in configuration C0.
- A joystick.
- An SD card with image v1.1.

Result:

- Remote RC control: you can drive the Duckiebot with the joystick.

Color code:

- In purple, things that will need to be moved to Step 0. (e.g. package installation). That is, these things are only necessary now, but should not be necessary later.
- In red, please note things that are broken.
- In blue, things that are specific to Feb 16 lab (and will be erased later)

Setting up a PI with an HDMI monitor

Put the SD card in your PI.

Attach an HDMI cable to the PI.
Note: It's important that you connect the HDMI cable before powering up the PI.

Attach a USB keyboard.

Plug in the WiFi USB dongle.

Power up. You should see a login screen.

Login and update base system

Login. (username: ubuntu, password: ubuntu)

Learn to love Byobu

Run "byobu":

```
duckiebot $ byobu
```

Yes, you need to learn to use byobu, unless you know an equivalent program. You will save much time later.

Byobu is “GNU screen” with fancy configuration. Please learn about Byobu here: http://byobu.co/

Quick commands reference, using function keys:

F2: open a new terminal
F3/F4: switch among the terminals
Ctrl-F6: close current terminal

Using control sequences:

ctrl-A then C: creates new terminal
ctrl-A then number: switches to terminal

To quit a terminal:

```
duckiebot $ exit
```

Update basic system

Use:

```
duckiebot $ sudo apt-get update
duckiebot $ sudo apt-get dist-upgrade
```

Feb 17 note: the command “dist-upgrade” currently breaks WiFi but it is inevitable eventually, so we are leaving it here for the moment and fix the real problem, rather than curing the symptom.

Camera Test with a monitor

You can test the camera right away, without setting up ROS.

Use the command:

```
duckiebot $ raspistill -t 100000 -o out.jpg
```

You should see an image on the screen. If it’s black, remove the lens cap.

Rotate the lens until the image is in focus.

Press ‘Ctrl + c’ to exit.

Camera Test without a monitor

If you don’t have a monitor, you can still take a picture and see it on your computer. Use this:

```
raspistill -t 1 -o out.jpg
```

Then, **after all network setup is done**, download out.jpg using scp:

```
$ scp ubuntu@<robot name>.local:~/out.jpg out.jpg]
```

Do not change the default shell

If you know what you are doing, you are welcome to install and use additional shells (such as zsh), but please **keep bash as be the default shell**. This is important for some scripts.

(For the record, our favorite shell is `zsh` with `oh-my-zsh`.)

Set hostname

Choose a name for your robot. This is a simple string that will always appear lowercase.

Suppose that the name is “duckiebot”.

Edit /etc/hostname and put “duckiebot” instead of “ubuntu”.

```
duckiebot $ sudo nano /etc/hostname
```

Also edit /etc/hosts and replace “ubuntu” with “duckiebot”:

```
duckiebot $ sudo nano /etc/hosts
```

Note: the command “sudo hostname duckiebot” is not enough. The change will not persist. You need to go through the steps above.

NEVER ADD HOSTNAMES IN /etc/hosts (e.g. duckiebot.local)

Note: When switching Wifi adapters, or putting an SD card in a different body, remove the file:

```
$ sudo rm /etc/udev/rules.d/70-persistent-net.rules
```

Just do it, just in case, if we forgot to remove it.

Then reboot:

```
$ sudo reboot
```

When you reboot, you should see your new hostname:

```
Ubuntu 14.04.3 LTS duckiebot tty1:
duckiebot login:
```

Setting up the network

First make sure that your PI can see the WiFi dongle by using

```
$ lsusb
```

You should see something like this:

```
Bus 001 Device 005: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter
Bus 001 Device 004: ID 258a:0001
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast
Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsyistems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Note the “Ralink Technology, Corp. RT5370 Wireless Adapter” .

If it doesn’t then your adapter is malfunctioning. Get a new one.

Use ifconfig to check the status of wlan0, you should see something like

```
wlan0   Link encap:Ethernet  HWaddr 00:0f:60:06:92:d5
        inet addr:192.168.0.109  Bcast:192.168.0.255  Mask:255.255.255.0
        inet6 addr: fe80::20f:60ff:fe06:92d5/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:214 errors:0 dropped:0 overruns:0 frame:0
        TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:40528 (40.5 KB)  TX bytes:15571 (15.5 KB)
```

Network troubleshooting
This means that the wlan is not configured:

```
duckiebot $ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:0f:60:05:ff:e8
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

If you don't see the wifi interface at all, you switched wifi usb keys or it's possible that your SD card remembers the MAC address of another body (wifi dongle, ethernet chip, etc). In that case you can make it forget the MAC addresses by removing the 70-persistent-net.rules file and reboot to trigger the regeneration of the file with the current wifi and ethernet hardware.

```
duckiebot $ sudo rm /etc/udev/rules.d/70-persistent-net.rules
duckiebot $ sudo reboot
```

If you do not get an inet4 address on wlan0:
If you only get a inet6 address when you run

```
$ ifconfig wlan0
```

and if you run

```
$ sudo dhclient wlan0
```

and then you run again

```
$ ifconfig wlan0
```

and then you get an inet4 address, then your dhclient is not invoked automatically at a reboot. The following script will fix your problem. Just run it on your duckiebot, it will fix a problem with the "ifupdown" package, that fails to execute dhclient after a reboot:

```
$ cd ~/
$ wget https://raw.githubusercontent.com/duckietown/Software/devel/fix_wifi.sh
$ source fix_wifi.sh
```

make sure you reboot after downloading and running the script.

If you do not get an inet 4 address after running dhclient manually, please check your wifi password and ssid (see next troubleshooting sections).

WPA configuration
Try:

```
duckiebot $ grep wpa /var/log/syslog | tail
```

if it says:

```
Failed to read or parse configuration ‘/etc/wpa_supplicant/wpa_supplicant’
```

ensure that the `wpa_supplicant` file looks like below/

Verify Network Setting

Open the file /etc/wpa_supplicant/wpa_supplicant.conf, it should contain the following:

```
duckiebot $ cat /etc/wpa_supplicant/wpa_supplicant.conf

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
 ssid="duckietown"
 scan_ssid=1
 psk="quackquack"
 priority=10
}
```

(Note that space matters. Don’t leave spaces.)

If you are on a different wifi network, change the ssid and psk to reflect your wifi setting using

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Note: could connect to campus/building-wide network such as MIT or StataCenter, but the avahi function won’t work properly. In short, you should connect to a network where you have access to the actual router.

Reset the wireless interface by

```
$ sudo ifdown wlan0
$ sudo ifup wlan0
```

When the wifi dongle is working, you should see the blue light on it blinking.

This is the output of iwconfig in nominal conditions:

```
$ iwconfig
wlan0     IEEE 802.11bgn  ESSID:"duckietown"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 90:84:0D:DA:06:63
          Bit Rate=43.3 Mb/s   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=47/70  Signal level=-63 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
```

```
Tx excessive retries:770  Invalid misc:1448   Missed beacon:0
```

Check that the following command gives the correct response (when you're in duckietown):

```
$ iwgetid
wlan0    ESSID:"duckietown"
```

Try

```
$ ping google.com
```

You should see something like:

```
PING google.com (4.53.56.94) 56(84) bytes of data.
64 bytes from 4.53.56.94: icmp_seq=1 ttl=55 time=3.28 ms
64 bytes from 4.53.56.94: icmp_seq=2 ttl=55 time=10.5 ms
64 bytes from 4.53.56.94: icmp_seq=3 ttl=55 time=5.34 ms
64 bytes from 4.53.56.94: icmp_seq=4 ttl=55 time=9.98 ms
```

which means that the PI is now connected to the internet. Do not continue if you don't see this.

Setup an Ubuntu laptop to ssh to the duckiebot

Connect your laptop to the same network as the robot.

From your laptop, now you should be able to ping the duckiebot:

```
laptop $ ping duckiebot.local
```

Do not continue if you cannot do this successfully.

> ***Protip**: In general, if you find yourself:*
> - *typing an IP*
> - *typing a password*
> - *typing "ssh" more than once*
> - *using a screen / USB keyboard*
>
> *it means you should learn more about Linux and networks, and you are setting yourself up for failure. Yes, you "can do without", but with an additional 30 seconds of your time. The 30 seconds you are not saving every time are the difference between being*

productive roboticists and going crazy. Really, it is impossible to do robotics when you have to think about IPs and passwords...

Verify that you can ssh to the PI:

```
laptop $ ssh ubuntu@duckiebot.local
```

Say “yes” if you get asked whether you want to add it to a list of known hosts.

Offending key error: If you get something like this:

```
Warning: the ECDSA host key for ... differs from the key for the IP address '10.0.1.17'
Offending key for IP in /Users/<user>/.ssh/known_hosts:<line>
```

then remove line <line> in ~/.ssh/known_hosts

Now, let’s set up **passwordless ssh.**

On the laptop, create the .ssh directory:

```
laptop $ mkdir -p ~/.ssh
```

Install the duckietown_key1 by downloading it:

```
laptop $ wget -O ~/.ssh/duckietown_key1
"https://www.dropbox.com/s/q23qptu01u7ur3y/duckietown_key1?dl=1"
```

Changing Key Permission

Edit the permission of the file to make ssh happy. The key file must not be readable or writable from other users or groups.

1

```
laptop $ chmod 600 ~/.ssh/duckietown_key1
```

Regenerate the public key according to:

```
laptop $ ssh-keygen -f ~/.ssh/duckietown_key1 -y > ~/.ssh/duckietown_key1.pub
```

Troubleshooting

If there are issues such as “scheme missing” and the file duckietown_key1 does not exist in ~/.ssh/ folder but instead downloaded a file named duckietown_key1?dl=1 in the current folder simply rename duckietown_key1?dl=1 to duckietown_key1 and copy it over to the directory ~/.ssh/.

To move the mislabeled file:
im ~

```
laptop $ mv “duckietown_key1?dl=1”  ~/.ssh/duckietown_key1
```

On the laptop, now edit ~/.ssh/config:

```
laptop $ nano ~/.ssh/config
```

and add the following lines:

```
Host duckiebot
      Hostname duckiebot.local
      User ubuntu
      IdentityFile ~/.ssh/duckietown_key1
      HostKeyAlgorithms ssh-rsa
```

Now you should be able to ssh passwordlessly from your laptop:

```
laptop $ ssh duckiebot
```

This should succeed. Do not continue unless it does.

Note that you can actually auto-complete with tab after ssh.

(optional/best practices) Using your own certificate for connecting

(Robert Katzschmann)

If you do not want to allow others to get your ssh certificate that allows them to log into your github account, I recommend you to do the following steps.

Ssh into your bot first:

```
laptop $ ssh ubuntu@duckiebot.local
```

Make a backup of the authorized_keys file:

```
duckiebot $ cp ~/.ssh/authorized_keys ~/.ssh/authorized_keys.orig
```

Open authorized_keys file on duckiebot:

```
duckiebot $ nano ~/.ssh/authorized_keys
```

and delete the old content (if it was only the key that was placed there originally) and paste in the public certificate content from your laptop.

Copy your public certificate content from your laptop to your bot (replace `<own_certificatename>` to something like “id_rsa”):

```
laptop $ cat ~/.ssh/<own_certificatename>.pub | (ssh ubuntu@duckiebot.local "cat >> ~/.ssh/authorized_keys")
```

~~laptop $ sudo apt-get install xclip~~
~~laptop $ xclip -sel clip < ~/.ssh/<own_certificatename>.pub~~

On the laptop, now edit ~/.ssh/config by changing this section from the old content:

```
Host duckiebot
      Hostname duckiebot.local
      User ubuntu
      IdentityFile ~/.ssh/duckietown_key1
```

to this changed content (basically just replace the IdentityFile name):

```
Host duckiebot
      Hostname duckiebot.local
      User ubuntu
      IdentityFile ~/.ssh/<own_certificatename>
```

To test if it all worked, close any open ssh session to your bot and rerun

```
laptop $ ssh duckiebot
```

You should be able to connect without typing in a password.

Last step on duckiebot, change your password from the standard “ubuntu” to something secret by using:

```
duckiebot $ passwd ubuntu
```

(optional/best practices) Add further Laptops as authorized hosts

(Robert Katzschmann)

Follow step **Step # 2: Generate next/multiple ssh key** of this guide:
http://www.cyberciti.biz/tips/linux-multiple-ssh-key-based-authentication.html

Go wireless

Now you can access the PI remotely.

Disconnect the monitor and keyboard.

You will do everything that follows in a remote console.

Again, trust us, it is worth making sure that everything above works.

Fix date issues

This step is very important for RPi to accept SSL certificates for git and wget using ssh and https protocols.

```
duckiebot $ sudo ntpdate -u us.pool.ntp.org
```

This is to be repeated every time the system is restarted.

Set up Github keys

After completing the steps above, SSH into the duckiebot:

```
laptop $ ssh duckiebot
```

(It should not ask you for a password; if it does, fix that before continuing.)

Now follow Setup Step 1.9 - Github basics to configure git on the Raspberry PI.
You already did this step once for the laptop. Now you have to do it again.

Clone the Duckietown repository

You should have already followed the steps in [Setup Step 1.9 - Github basics].

Troubleshooting:

- *For this to succeed you should have a Github account sudo ntpdate -u us.pool.ntp.org already set up.*
- *if it fails with weird errors, probably the time is not set up correctly. Use ntpdate as above.*

Set up ROS environment on the Duckiebot

Install some libraries that are not in the current version of the image by running the duckietown_install.sh script

```
duckiebot $ cd ~/duckietown
duckiebot $ ./duckietown_install_car.sh
```

Now we are ready to make the workspace. First you need to source the baseline ROS indigo environment:

```
duckiebot $ source /opt/ros/indigo/setup.bash
```

Then, build the workspace (you have to be under the catkin_ws folder to invoke catkin_make)

```
duckiebot $ cd ~/duckietown/catkin_ws
duckiebot $ catkin_make
duckiebot $ catkin_make   (again)
duckiebot $ catkin_make   (again)
```

Note: **the catkin_make command might fail** with a message about “exhausted virtual memory” and invoking "make -j4 -l4" failed. The output output is similar to this:

```
[  2%] [  4%] [  4%] Building CXX object
slam/CMakeFiles/sum_and_average_node.dir/src/sum_and_average_node.cpp.o
Building CXX object slam/CMakeFiles/slamNode.dir/src/slamNode.cpp.o
```

```
Building CXX object slam/CMakeFiles/listener.dir/src/listener.cpp.o
virtual memory exhausted: Cannot allocate memory
make[2]: *** [slam/CMakeFiles/listener.dir/src/listener.cpp.o] Error 1
make[1]: *** [slam/CMakeFiles/listener.dir/all] Error 2
make[1]: *** Waiting for unfinished jobs....

Invoking "make -j4 -l4" failed
```

Just run the command catkin_make again. It **may** take up to **three** runs until everything is built successful.

Add your vehicle to the machines file

On the robot edit the file

```
duckiebot $ nano ~/duckietown/catkin_ws/src/duckietown/machines
```

You should see something like

```
<launch>
    <arg name="env_script_path" default="~/duckietown/environment.sh"/>
    <machine name="megaman" address="megaman.local" user="ubuntu" env-loader="$(arg env_script_path)"/>
    ...
</launch>
```

Now, duplicate a <machine ... /> line between <launch> and </launch>, and replace the name and address string with the name of your vehicle. In this example, the additional line to add is

```
<machine name="duckiebot" address="duckiebot.local" user="ubuntu"
env-loader="$(arg env_script_path)"/>
```

```
commit and push the new machines file.
```

Run the joystick demo

Plug the joystick to one of the usb port on the RasPi.

SSH into your PI. Go to the duckietown folder and invoke the following scripts:

```
duckiebot:~/duckietown$ source environment.sh
duckiebot:~/duckietown$ source set_ros_master.sh
```

The environment.sh setup the ROS environment at the terminal (so you can use commands like rosrun and roslaunch). The set_ros_master.sh by default sets the PI as its own rosmaster.

Now make sure the motor shield is connected.

Run the command:

```
duckiebot $ roslaunch duckietown joystick.launch veh:=duckiebot
```

If there is no “red” output in the command line then pushing the left joystick knob controls throttle - right controls steering.

This is the expected result of the commands:

left joystick up = forward
left joystick down = backward
right joystick left = turn left (positive theta)
right joystick right = turn right (negative theta)

It is possible you will have to unplug and replug the joystick or just push lots of buttons on your joystick until it wakes up. Also make sure that the mode switch on the top of your joystick is set to “X” not “D”.

Close the program using Ctrl-C.

Troubleshooting - robot moves weirdly (forward instead of backward): Either the cables or the motors are inverted. Please refer to the assembly guide for pictures of the correct connections.

Troubleshooting - left joystick does not work: If the green light on the right to the “mode” button is on, click the “mode” button to turn the light off. The “mode” button toggles between left joystick or the cross on the left.

Troubleshooting - robot does not move: The joy_mapper_test.launch assumes that the joystick is at /dev/input/js0. To make sure that the joystick is there, you can do

```
$ ls /dev/input/
```

and check if there is a js0 on the list.

To test whether or not the joystick itself is working properly (without ROS), you can do

```
$ jstest /dev/input/js0
```

Move the joysticks and push the buttons and check the printouts.

Troubleshooting - robot moves very weirdly: Check that the joystick has the switch set to the position “x”. And the mode light should be off.

shutting down the robot:
sudo shutdown -h now
then physically disconnect power cables

Proper shutdown procedure for the PI

Ctrl-C in the terminal of joystick.launch when you're done.

To shutdown: DO NOT DISCONNECT THE POWER - the system might get corrupted. Instead, issue the following command:

```
duckiebot $ sudo shutdown -h now
```

and then wait 30 seconds before disconnecting the power.

Also: Disconnect the battery end of the cable, not the one close to the PI, because you might damage it.