The Duckietown Book

The last version of this book and other documents are available at the URL
http://book.duckietown.org/

TABLE OF CONTENTS

Part 1
The Duckietown project

Chapter 1
What is Duckietown?

1.1. Goals and objectives

Duckietown is a robotics educations and outreach effort.

The most tangible goal of the project is to provide a low-cost educational platform for learning autonomy, consisting of the Duckiebots, an autonomous robot, and the Duckietowns, the infrastructure in which the Duckiebots navigates.

However, we focus on the *learning experience* as a whole, by providing a set of modules teaching plans and other guides, as well as a curated role-play experience.

We have two targets:

1. For **instructors**, we want to create a "class-in-a-box" that allows to offer a modern and engaging learning experience. Currently, this is feasible at the advanced undergraduate and graduate level, though in the future we would like to present the platform as multi-grade experiences.
2. For **self-guided learners**, we want to create a "self-learning experience", that allows to go from zero knowledge of robotics to graduate-level understanding.

In addition, the Duckietown platform has been used as a research platform.

1.2. Results obtained so far

While we are at the early phases of the project, many people have been used the materials in the past year.

1.3. Learn about the platform

The best way to get a sense of how the platform looks is to watch these videos. They show off the capabilities of the platform.

This video is part of the Red Hat documentary:

1.4. Learn about the educational experience

These papers present a more formal description of the technical side of the projecrt as well as the educational side.

This paper [1] describes the course design for Duckietown: learning objectives, teaching methods, etc.

This video is a Duckumentary about the first version of the class, during Spring 2016. The Duckumentary was shot by Chris Welch.

1.5. Learn about the platform

The paper [2] describes the Duckiebot and its software. With 29 authors, we made the record for a robotics conference.

Chapter 2
Duckietown history and future

Assigned to: Liam

2.1. The beginnings of Duckietown

Duckietown started as an MIT class during Spring 2016.

2.2. Duckietown around the world

1) Duckietown High School

2.3. Coming up

In 2017, the class will be offered contemporaneously at:

- ETH Zurich
- University of Montreal
- University of Chicago

as well as:

Chapter 3
First steps

3.1. How to get started

If you are an instructor, please jump to Section 3.2.

If you are a self-guided learner, please jump to Section 3.3.

If you are a company, and interested in working with Duckietown, please jump to Section 3.4.

3.2. Duckietown for instructors

3.3. Duckietown for self-guided learners

3.4. Introduction for companies

3.5. How to keep in touch

3.6. How to contribute

Chapter 4
Frequently Asked Questions

4.1. General questions

Q: What is Duckietown?

Duckietown is a low-cost educational and research platform.

Q: Is Duckietown free to use?

Yes. All materials are released according to an open source license.

Q: Is everything ready?

Not quite! Please sign up to our mailing list to get notified when things are a bit more ready.

Q: How can I start?

See the next section, Getting started.

Q: How can I help?

If you would like to help actively, please email duckietown@mit.edu.

4.2. FAQ by students / independent learners

Q: I want to build my own Duckiebot. How do I get started?

4.3. FAQ by instructors

Q: How large a class can it be? I teach large classes.

Q: What is the budget for the robot?

Q: I want to teach a Duckietown class. How do I get started?

Please get in touch with us at duckietown@mit.edu. We will be happy to get you started and sign you up to the Duckietown instructors mailing list.

Q: Why the duckies?

Compared to other educational robotics projects, the presence of the duckies is what makes this project stand out. Why the duckies?

We want to present robotics in an accessible and friendly way.

Part 2
Software carpentry

This part describes things that you should know about UNIX/Linux environments.

Please read the “background reading” section before you start, while the rest can be used as a reference.

Documentation writers: please make sure that every command used has a section in these chapters.

Chapter 5
Linux

Assigned to: Andrea

5.1. Background reading

- UNIX
- Linux
- free software; open source software.

5.2. Ubuntu packaging

1) `apt install`

2) `apt update`

3) `apt dist-upgrade`

5.3. Measuring resource usage

1) `pgrep`

2) Testing SD Card and disk speed

Test SD Card (or any disk) speed using the following commands, which write to a file called `[filename]`.

```
$ dd if=/dev/zero of=[filename] bs=500K count=1024
$ sync
$ echo 3 | sudo tee /proc/sys/vm/drop_caches
$ dd if=[filename] of=/dev/null bs=500K count=1024
$ rm [filename]
```

Note the `sync` and the `echo` command are very important.

Example results:

```
524288000 bytes (524 MB, 500 MiB) copied, 30.2087 s, 17.4 MB/s
524288000 bytes (524 MB, 500 MiB) copied, 23.3568 s, 22.4 MB/s
```

That is write 17.4 MB/s, read 22 MB/s.

3) Measuring CPU usage using `htop`

You can use `htop` to monitor CPU usage.

```
$ sudo apt install htop
```

4) Measuring I/O usage using `iotop`

```
$ sudo apt install iotop
```

5.4. How to burn an image to an SD card

> Requires:
>
> - A blank SD card.
> - An image file to burn.
> - An Ubuntu computer with an SD reader.
>
> Results:
>
> - A burned image.

1) Finding your device name for the SD card

First, find out what is the device name for the SD card.

Insert the SD Card in the slot.

Run the command:

```
$ sudo fdisk -l
```

Find your device name, by looking at the sizes.

For example, the output might contain:

```
Disk /dev/mmcblk0: 14.9 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

In this case, the device is `/dev/mmcblk0`. That will be the `[device]` in the next commands.

You may see `/dev/mmcblk0pX` or a couple of similar entries for each partition on the card, where `X` is the partition number. If you don't see anything like that, take out the SD card and run the command again and see what disappeared.

2) Unmount partitions

Before proceeding, unmount all partitions.

Run `df -h`. If there are partitions like `/dev/mmcblk0p[n]`, then unmount each of them. For example:

```
laptop $ sudo umount /dev/mmcblk0p1
laptop $ sudo umount /dev/mmcblk0p2
```

3) Burn the image

Now that you know that the device is `[device]`, you can burn the image to disk.
Let the image file be `[image file]`.
Burn the image using the command `dd`:

```
laptop $ sudo dd of=[device] if=[image file] status=progress bs=4M
```

> **Note:** Use the name of the device, without partitions. i.e., `/dev/mmcblk0`, not `/dev/mmcblk0pX`.

CHAPTER 6
Networking

Assigned to: Andrea

6.1. Background reading

Here are some background reading about networking and networking in Linux:

Make sure that you know:

- what are IP addresses
- what are subnets
- how DNS works
- how `.local` names work
- ...

A short reference follows.

6.2. Visualizing information about the network

1) `ping`

2) `ifconfig`

```
$ ifconfig
```

Chapter 7
Wireless networks

1) iwlist

What wireless networks do I have around?

```
$ sudo iwlist [interface] scan | grep SSID
```

Does the interface support 5 GHz channels?

```
$ sudo iwlist [interface] freq
```

Example output:

```
wlx74da38c9caa0  20 channels in total; available frequencies :
  Channel 01 : 2.412 GHz
  Channel 02 : 2.417 GHz
  Channel 03 : 2.422 GHz
  Channel 04 : 2.427 GHz
  Channel 05 : 2.432 GHz
  Channel 06 : 2.437 GHz
  Channel 07 : 2.442 GHz
  Channel 08 : 2.447 GHz
  Channel 09 : 2.452 GHz
  Channel 10 : 2.457 GHz
  Channel 11 : 2.462 GHz
  Channel 36 : 5.18 GHz
  Channel 40 : 5.2 GHz
  Channel 44 : 5.22 GHz
  Channel 48 : 5.24 GHz
  Channel 149 : 5.745 GHz
  Channel 153 : 5.765 GHz
  Channel 157 : 5.785 GHz
  Channel 161 : 5.805 GHz
  Channel 165 : 5.825 GHz
  Current Frequency:2.437 GHz (Channel 6)
```

Note that in this example only *some* 5Ghz channels are supported (36, 40, 44, 48, 149, 153, 157, 161, 165); for example, channel 38, 42, 50 are not supported. This means that you need to set up the router not to use those channels.

Chapter 8
Compilers

Assigned to: Andrea

Chapter 9
Accessing computers using SSH

Assigned to: Andrea

9.1. Background reading

- Encryption
- Public key authentication

9.2. Installation of SSH

This installs the client:

```
$ sudo apt install ssh
```

This installs the server:

This enables the server:

9.3. Local configuration

The SSH configuration as a client is in the file

`~/.ssh/config`

Create the directory with the right permissions:

```
$ mkdir ~/.ssh
$ chmod 0700 ~/.ssh
```

Then add the following lines:

```
HostKeyAlgorithms ssh-rsa
```

The reason is that Paramiko, used by `roslaunch`, does not support the ECSDA keys.

9.4. How to login with SSH and a password

To log in to a remote computer `[remote]` with user `[remote-user]`, use:

```
$ ssh [remote-user]@[remote]
```

1) Troubleshooting

Symptom: “Offending key error”.

If you get something like this:

```
Warning: the ECDSA host key for [...] differs from the key for the IP address '[...] '
Offending key for IP in /home/[user]/.ssh/known_hosts:[line]
```

then remove line `[line]` in `~/.ssh/known_hosts`.

9.5. Creating an SSH keypair

This is a step that you will repeat twice: once on the Duckiebot, and once on your laptop.

The program will prompt you for the filename on which to save the file.

Use the convention

```
/home/[username]/.ssh/[username]@[host name]
/home/[username]/.ssh/[username]@[host name].pub
```

where:

- `[username]` is the current user name that you are using (`ubuntu` or your chosen one);
- `[host name]` is the name of the host (the Duckiebot or laptop);

An SSH key can be generated with the command:

```
$ ssh-keygen -h
```

The session output will look something like this:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/[username]/.ssh/id_rsa):
```

At this point, tell it to choose this file:

```
/home/[username]/.ssh/[username]@[host name]
```

Then:

```
Enter passphrase (empty for no passphrase):
```

Press enter; you want an empty passphrase.

```
Enter same passphrase again:
```

Press enter.

```
Your identification has been saved in /home/[username]/.ssh/[username]@[host name]
Your public key has been saved in /home/[username]/.ssh/[username]@[host name].pub
The key fingerprint is:
[XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX] [username]@[host name]
The key's randomart image is:
+--[ RSA 2048]----+
|             .   |
|       o   o  .  |
|      o = o  . o |
|       B . .  * o|
|        S o    O |
|         o o  . E|
|          o o  o |
|           o  +  |
|            .. . |
+-----------------+
```

Note that the program created two files.

The file that contains the private key is

```
/home/[username]/.ssh/[username]@[host name]
```

The file that contains the public key has extension `.pub`:

```
/home/[username]/.ssh/[username]@[host name].pub
```

Next, tell SSH that you want to use this key.

Make sure that the file `~/.ssh/config` exists:

```
$ touch ~/.ssh/config
```

Add a line containing

```
IdentityFile [PRIVATE_KEY_FILE]
```

(using the filename for the private key).

Check that the config file is correct:

```
$ cat ~/.ssh/config
[...]
IdentityFile [PRIVATE_KEY_FILE]
[...]
```

9.6. How to login without a password

Assumptions:

- You have two computers, called "[local]" and "[remote]", with users "[local-user]" and "[remote-user]".
- The two computers are on the same network.
- You have created a keypair for [local-user] on [local].

➡ This procedure is described in Section 9.5.

Results:

- From the [local] computer, [local-user] will be able to log in to [remote] computer without a password.

First, connect the two computers to the same network, and make sure that you can ping [remote] from [local]:

```
[local] $ ping [remote].local
```

Do not continue if you cannot do this successfully.

If you have created a keypair for [local-user], you will have a public key in this file on the [local] computer:

```
/home/[local-user]/.ssh/[local-user]@[local].pub
```

This file is in the form:

```
ssh-rsa [long list of letters and numbers] [local-user]@[local]
```

You will have to copy the contents of this file on the [remote] computer, to tell it that this key is authorized.

On the [remote] computer, edit or create the file:

`/home/[remote-user]/.ssh/authorized_keys`

and add the entire line as above containing the public key.

Now, from the `[local]` computer, try to log in into the `[remote]` one:

```
[local] $ ssh [remote-user]@[remote]
```

This should succeed, and you should not be asked for a password.

9.7. Fixing SSH Permissions

Sometimes, SSH does not work because you have the wrong permissions on some files.

In doubt, these lines fix the permissions for your `.ssh` directory.

```
$ chmod 0700 ~/.ssh
$ chmod 0700 ~/.ssh/*
```

9.8. SCP

1) Download a file with SCP

9.9. RSync

Chapter 10
Editors

> **Assigned to:** Andrea

10.1. VIM

The editor to choose is `vi`, or more precisely, `vim` (improved vi).
Install like this:

```
$ sudo apt install vim
```

Documentation:

- A VIM tutorial

Chapter 11
Source code control with Git

> Assigned to: Andrea

11.1. Background reading

- Git
- GitFlow

11.2. Installation

The basic Git program is installed using

```
$ sudo apt install git
```

Additional utilities for `git` are installed using:

```
$ sudo apt install git-extras
```

This include the `git-ignore` utility.

11.3. Setting up global configurations for Git

This should be done twice, once on the laptop, and later, on the robot.

These options tell Git who you are:

```
$ git config --global user.email "[email]"
$ git config --global user.name  "[full name]"
```

Also do this, and it doesn't matter if you don't know what it is:

```
$ git config --global push.default simple
```

11.4. Git tips

1) Shallow clone

You can clone without history with the command:

```
$ git clone --depth 1 [repository URL]
```

11.5. Git troubleshooting

1) Problem 1: https instead of ssh:

The symptom is:

```
$ git push
Username for 'https://github.com':
```

Diagnosis: the `remote` is not correct.

If you do `git remote` you get:

```
$ git remote -v
origin  https://github.com/duckietown/Software.git (fetch)
origin  https://github.com/duckietown/Software.git (push)
```

Solution:

```
$ git remote remove origin
$ git remote add origin git@github.com:duckietown/Software.git
```

Expectation:

```
$ git remote -v
origin  git@github.com:duckietown/Software.git (fetch)
origin  git@github.com:duckietown/Software.git (push)
```

2) Problem 1: `git push` complains about upstream

The symptom is:

```
fatal: The current branch [branch name] has no upstream branch.
```

Solution:

```
$ git push --set-upstream origin [branch name]
```

Chapter 12
Shells

> Assigned to: Andrea

12.1. Byobu

You need to learn to use `byobu`. It will save much time later.

Byobu is "GNU `screen`" with fancy configuration; if you know `screen`, that's fine as well.

Please learn about Byobu here:

- http://byobu.co/

Install using:

```
$ sudo apt install byobu
```

1) Advantages of using Byobu

2) Quick command reference

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 a number: switches to that terminal.
- `ctrl-A` then `D`: detaches the terminal.

To quit a terminal, just use `exit`.

Chapter 13
Other things to know

Assigned to: Andrea

13.1. Markdown

Part 3
Preliminaries

Chapter 14
Linear algebra

Assigned to: Jacopo

Chapter 15
Probability basics

Assigned to: Liam?

Chapter 16
Dynamics

Assigned to: Jacopo

Part 4
Duckiebot setup

Chapter 17
Acquiring the parts for the Duckiebot

Assigned to: Jacopo

The trip begins with acquiring the parts. Here, we provide a link to all bits and pieces that are needed to build a Duckiebot.

In general, keep in mind that:

- The links might expire, or the prices might vary.
- In general, substitutions are OK for the mechanical components, and not OK for all the electronics, unless you are OK in writing some software.

Resources necessaries:

- Cost: USD ???
- Time: ??? days (average shipping)

Results:

- A kit of parts ready to be assembled.

17.1. Duckiebot configurations

Configuration `D17-0`: Only camera and motors.

Configuration `D17-0+W`: Previous one + an additional WiFi card (Edimax).

Configuration `D17-1`: LED lights and bumpers

17.2. Bill of materials

Chassis	USD xxx
Camera	USD xxx
Raspberry PI 3	USD 35
Total for minimum configuration	USD ??
Total for fancy configuration	USD ??

17.3. Chassis

We selected the Magician Chassis as the basic chassis for the robot (Figure 1).

We chose it because it has a double-decker configuration, and so we can put the battery in the lower part.

The price for this in the US is about USD 15-30.

Figure 1. The Magician Chassis

17.4. Raspberry PI 3

...

17.5. Camera

...

Chapter 18
Soldering boards

Assigned to: Shiying

Resources necessaries:

- ...
- Time: ??? minutes

Results:

- ...

Chapter 19
Assembling the Duckiebot

> Assigned to: Shiying

Resources necessaries:

- Duckiebot `D17-C0` parts.

  ➡ The acquisition process is explained in Chapter 17.

- Time: about ??? minutes.

Results:

- An assembled Duckiebot in configuration `D17-C0`.

Shiying: here will be the instruction about assembling the Duckiebot. :-)

Chapter 20
Reproducing the image

Assigned to: Andrea

These are the instructions to reproduce the Ubuntu image that we use.

Please note that the image is already available, so you don't need to do this manually.

However, this documentation might be useful if you would like to port the software to a different distribution.

> Resources necessaries:
>
> - Internet connection to download the packages.
> - A PC running any Linux with an SD card reader.
> - Time: about 20 minutes.
>
> Results:
>
> - A baseline Ubuntu Mate 16.04.2 image with updated software.

20.1. Download and uncompress the Ubuntu Mate image

Download the image from the page

https://ubuntu-mate.org/download/

The file we are looking for is:

```
filename: ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img.xz
    size: 1.2 GB
  SHA256: dc3afcad68a5de3ba683dc30d2093a3b5b3cd6b2c16c0b5de8d50fede78f75c2
```

After download, run the command `sha256sum` to make sure you have the right version:

```
laptop $ sha256sum ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img.xz
dc3afcad68a5de3ba683dc30d2093a3b5b3cd6b2c16c0b5de8d50fede78f75c2
```

If the string does not correspond exactly, your download was corrupted. Delete the file and try again.

Then decompress using the command `xz`:

```
laptop $ xz -d ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img.xz
```

20.2. Burn the image to an SD card

Next, burn the image on to the SD card.

➡ This procedure is explained in Section 5.4.

1) Verify that the SD card was created correctly

Remove the SD card and plug it in again in the laptop.

Ubuntu will mount two partitions, by the name of `PI_ROOT` and `PI_BOOT`.

2) Installation

Boot the disk in the Raspberry PI.
Choose the following options:

```
language: English
username: ubuntu
password: ubuntu
hostname: duckiebot
```

Choose the option to log in automatically.
Reboot.

3) Update installed software

The WiFi was connected to airport network `duckietown` with password `quackquack`.
Afterwards I upgraded all the software preinstalled with these commands:

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

Expect `dist-upgrade` to take quite a long time (up to 2 hours).

20.3. Raspberry PI Config

The Raspberry PI is not accessible by SSH by default.
Run `raspi-config`:

```
duckiebot $ sudo raspi-config
```

choose “3. Interfacing Options”, and enable SSH,
We need to enable the camera and the I2C bus.
choose “3. Interfacing Options”, and enable camera, and I2C.
Also disable the graphical boot

20.4. Install packages

Install these packages.
Etckeeper:

```
duckiebot $ sudo apt install etckeeper
```

Editors / shells:

```
duckiebot $ sudo apt install -y vim emacs byobu zsh
```

Git:

```
duckiebot $ sudo apt install -y git git-extras
```

Other:

```
duckiebot $ sudo apt install htop atop nethogs iftop
duckiebot $ sudo apt install aptitude apt-file
```

Development:

```
duckiebot $ sudo apt install -y build-essential libblas-dev liblapack-dev libatlas-base-dev
gfortran libyaml-cpp-dev
```

Python:

```
duckiebot $ sudo apt install -y python-dev ipython python-sklearn python-smbus
duckiebot $ sudo pip install scipy --upgrade
```

I2C:

```
duckiebot $ sudo apt install -y i2c-tools
```

20.5. Install Edimax driver

First, mark the kernel packages as not upgradeable:

```
$ sudo apt-mark hold raspberrypi-kernel raspberrypi-kernel-headers
raspberrypi-kernel set on hold.
raspberrypi-kernel-headers set on hold
```

Then, download and install the Edimax driver from this repository.

20.6. Install ROS

Install ROS.

➡ The procedure is given in Section 40.1.

20.7. Wireless configuration (old version)

This is the old version.

There are two files that are important to edit.

The file `/etc/network/interfaces` should look like this:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
#source-directory /etc/network/interfaces.d

auto wlan0

# The loopback network interface
auto lo
iface lo inet loopback

# Wireless network interface
allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

The file `/etc/wpa_supplicant/wpa_supplicant.conf` should look like this:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
ssid="duckietown"
psk="quackquack"
proto=RSN
key_mgmt=WPA-PSK
pairwise=CCMP
auth_alg=OPEN
}
network={
    key_mgmt=NONE
}
```

20.8. Wireless configuration

The files that describe the network configuration are in the directory

```
/etc/NetworkManager/system-connections/
```

This is the contents of the connection file `duckietown`, which describes how to connect to the `duckietown` wireless network:

```
[connection]
id=duckietown
uuid=e9cef1bd-f6fb-4c5b-93cf-cca837ec35f2
type=wifi
permissions=
secondaries=
timestamp=1502254646

[wifi]
mac-address-blacklist=
mac-address-randomization=0
mode=infrastructure
ssid=duckietown

[wifi-security]
group=
key-mgmt=wpa-psk
pairwise=
proto=
psk=quackquack

[ipv4]
dns-search=
method=auto

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
ip6-privacy=0
method=auto
```

This is the file

```
/etc/NetworkManager/system-connections/create-5ghz-network
```

Contents:

```
[connection]
id=create-5ghz-network
uuid=7331d1e7-2cdf-4047-b426-c170ecc16f51
type=wifi
# Put the Edimax interface name here:
interface-name=[wlx74da38c9caa0 - to change]
permissions=
secondaries=
timestamp=1502023843

[wifi]
band=a
# Put the Edimax MAC address here
mac-address=[74:DA:38:C9:CA:A0 - to change]
mac-address-blacklist=
mac-address-randomization=0
mode=ap
seen-bssids=
ssid=duckiebot-not-configured

[ipv4]
dns-search=
method=shared

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
ip6-privacy=0
method=ignore
```

Note that there is an interface name and MAC address that need to be changed on each PI.

20.9. SSH server config

This enables the SSH server:

```
$ sudo systemctl enable ssh
```

20.10. Create swap Space

Do the following:

Create an empty file using the `dd` (device-to-device copy) command:

```
duckiebot $ sudo dd if=/dev/zero of=/swap0 bs=1M count=512
```

This is for a 512 MB swap space.

Format the file for use as swap:

```
duckiebot $ sudo mkswap /swap0
```

Add the swap file to the system configuration:

```
duckiebot $ sudo vi /etc/fstab
```

Add this line to the bottom:

```
/swap0 swap swap
```

Activate the swap space:

```
duckiebot $ sudo swapon -a
```

20.11. Passwordless sudo

First, make `vi` the default editor, using

```
sudo update-alternatives --config editor
```

and then choose `vim.basic`.

Then run:

```
$ sudo visudo
```

And then change this line:

```
%sudo   ALL=(ALL:ALL) ALL
```

into this line:

```
%sudo   ALL=(ALL:ALL) NOPASSWD:ALL
```

20.12. Ubuntu user configuration

1) Groups

You should make the `ubuntu` user belong to the `i2c` and `input` groups:

```
duckiebot $ sudo adduser ubuntu i2c
duckiebot $ sudo adduser ubuntu input
```

: forgot to add to aug20 image:

```
duckiebot $ sudo adduser ubuntu video
```

You may need to do the following (but might be done already through `raspi-config`):

```
duckiebot $ sudo udevadm trigger
```

2) Basic SSH config

Do the basic SSH config.

➡ The procedure is documented in Section 9.3.

Note: this is not in the aug10 image.

3) Passwordless SSH config

Add `.authorized_keys` so that we can all do passwordless SSH.

The key is at the URL

```
https://www.dropbox.com/s/pxyou3qy1p8m4d0/duckietown_key1.pub?dl=1
```

Download to `.ssh/authorized_keys`:

```
duckiebot $ curl -o .ssh/authorized_keys [URL above]
```

4) Shell prompt

Add the following lines to `~ubuntu/.bashrc`:

```
echo ""
echo "Welcome to a duckiebot!"
echo ""
echo "Reminders:"
echo ""
echo "1) Do not use the user 'ubuntu' for development - create your own user."
echo "2) Change the name of the robot from 'duckiebot' to something else."
echo ""

export EDITOR=vim
```

Chapter 21
Installing Ubuntu on laptops

Assigned to: Andrea

Before you prepare the Duckiebot, you need to have a laptop with Ubuntu installed.

Requires:

- A laptop with free disk space.
- Internet connection to download the Ubuntu image.
- About ??? minutes.

Results:

- A laptop ready to be used for Duckietown.

21.1. Install Ubuntu

Install Ubuntu 16.04.2.

➡ For instructions, see for example this online tutorial.

On the choice of username: During the installation, create a user for yourself with a username different from `ubuntu`, which is the default. Otherwise, you may get confused later.

21.2. Install useful software

Use `etckeeper` to keep track of the configuration in `/etc`:

```
laptop $ sudo apt install etckeeper
```

Install `ssh` to login remotely and the server:

```
laptop $ sudo apt install ssh
```

Use `byobu`:

```
laptop $ sudo apt install byobu
```

Use `vim`:

```
laptop $ sudo apt install vim
```

Use `htop` to monitor CPU usage:

```
laptop $ sudo apt install htop
```

Additional utilities for `git`:

```
laptop $ sudo apt install git git-extras
```

Other utilities:

```
laptop $ sudo apt install avahi-utils ecryptfs-utils
```

21.3. Install ROS

Install ROS on your laptop.

➡ The procedure is given in Section 40.1.

21.4. Other suggested software

1) Redshift

This is Flux for Linux. It is an accessibility/lab safety issue: bright screens damage eyes and perturb sleep [3].

Install redshift and run it.

```
laptop $ sudo apt install redshift-gtk
```

Set to “autostart” from the icon.

2) Installation of the duckuments system

Optional but very encouraged: install the duckuments system. This will allow you to have a local copy of the documentation and easily submit questions and changes.

➡ The procedure is documented in Section 81.4.

21.5. Passwordless `sudo`

Set up passwordless `sudo`.

➡ This procedure is described in Section 20.11.

21.6. SSH and Git setup

1) Basic SSH config

Do the basic SSH config.

➡ The procedure is documented in Section 9.3.

2) Create key pair for `[username]`

Next, create a private/public key pair for the user; call it `[username]@[robot name]`.

➡ The procedure is documented in Section 9.5.

3) Add [username]'s public key to Github

Add the public key to your Github account.

➡ The procedure is documented in Section 22.3.

If the step is done correctly, this command should succeed:

```
duckiebot $ ssh -T git@github.com
```

4) Local Git setup

Set up Git locally.

➡ The procedure is described in Section 11.3.

Chapter 22
Setup Github access

Assigned to: Andrea

This chapter describes how to create a Github account and setup SSH on the robot and on the laptop.

22.1. Create a Github account

Our example account is the following:

```
Github name: greta-p
E-mail: greta-p@duckietown.com
```

Create a Github account (Figure 2).

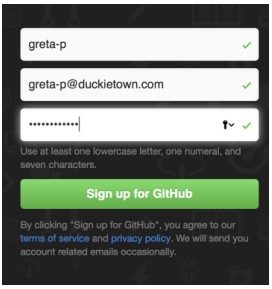

Figure 2

Go to your inbox and verify the email.

22.2. Become a member of the Duckietown organization

Give the administrators your account name. They will invite you.

Accept the invitation to join the organization that you will find in your email.

22.3. Add a public key to Github

You will do this procedure twice: once for the public key created on the laptop, and later with the public key created on the robot.

Requires:

- A public/private keypair already created and configured.

  ➡ This procedure is explained in Section 9.5.

Result:

- You can access Github using the key provided.

Go to settings (Figure 3).

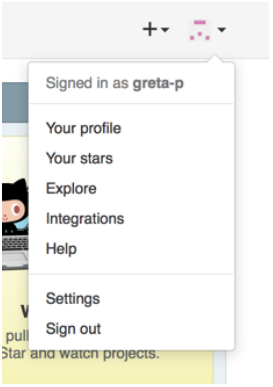

Figure 3

Add the public key that you created:

Figure 4

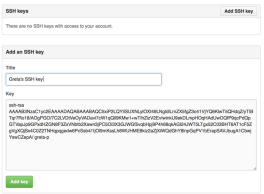

Figure 5

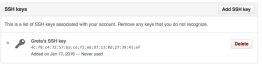

Figure 6

To check that all of this works, use the command

```
$ ssh -T git@github.com
```

The command tries to connect to Github using the private keys that you specified. This is the expected output:

```
Warning: Permanently added the RSA host key for IP address '[ip address]' to the list of known
hosts.
Hi [username]! You've successfully authenticated, but GitHub does not provide shell access.
```

If you don't see the greeting, stop.

Repeat what you just did for the Duckiebot on the laptop as well, making sure to change the name of the file containing the private key.

Chapter 23
Duckiebot Initialization

Assigned to: Andrea

Prerequisites:

- An SD card of dimensions at least 32 GB.
- A computer with an internet connection, an SD card reader, and 35 GB of free space.
- A mounted Duckiebot in configuration `D17-C0`.

➡ This is the result of Chapter 19.

Result:

- A Duckiebot that is ready to use.

What does it mean "ready to use"?.

23.1. Acquire and burn the image

On the laptop, download the compressed image at this URL:

https://www.dropbox.com/s/1p4am7erdd9e53r/duckiebot-RPI3-AC-aug10.img.xz?dl=1

The size is 2.5 GB.

You can use:

```
$ curl -o duckiebot-RPI3-AC-aug10.img.xz [URL above]
```

Uncompress the file:

```
$ xz -d -k duckiebot-RPI3-AC-aug10.img.xz
```

This will create a file of 32 GB in size.

To make sure that the image is downloaded correctly, compute its hash using the program `sha256sum`:

```
$ sha256sum duckiebot-RPI3-AC-aug10.img
2ea79b0fc6353361063c89977417fc5e8fde70611e8afa5cbf2d3a166d57e8cf  duckiebot-ac-aug10.img
```

Compare the hash that you obtain with the hash above. If they are different, there was some problem in downloading the image.

Next, burn the image on disk.

➡ The procedure of how to burn an image is explained in Section 5.4.

23.2. Turn on the Duckiebot

Put the SD Card in the Duckiebot.

Turn on the Duckiebot by connecting the power cable to the battery.

23.3. Connect the Duckiebot to a network

You can login to the Duckiebot in two ways:

1. Through an Ethernet cable.
2. Through a `duckietown` WiFi network.

In the worst case, you can use an HDMI monitor and a USB keyboard.

1) Option 1: Ethernet cable

Connect the Duckiebot and your laptop to the same network switch.

Allow 30 s - 1 minute for the DHCP to work.

2) Option 2: Duckietown network

The Duckiebot connects automatically to a 2.4 GHz network called "`duckietown`" and password "`quackquack`".

Connect your laptop to the same wireless network.

23.4. Ping the Duckiebot

To test that the Duckiebot is connected, try to ping it.

The hostname of a freshly-installed duckiebot is `duckiebot-not-configured`:

```
laptop $ ping duckiebot-not-configured.local
```

You should see output similar to the following:

```
PING duckiebot-not-configured.local ([X.X.X.X]): 56 data bytes
64 bytes from [X.X.X.X]: icmp_seq=0 ttl=64 time=2.164 ms
64 bytes from [X.X.X.X]: icmp_seq=1 ttl=64 time=2.303 ms
[...]
```

23.5. SSH to the Duckiebot

Next, try to log in using SSH, with account `ubuntu`:

```
laptop $ ssh ubuntu@duckiebot-not-configured.local
```

The password is `ubuntu`.

By default, the robot boots into Byobu.

Please see Section 12.1 for an introduction to Byobu.

Not sure it's a good idea to boot into Byobu.

23.6. (For D17-C1) Configure the robot-generated network

The Duckiebot in configuration `D17-C0+w` can create a WiFi network.

It is a 5 GHz network; this means that you need to have a 5 GHz WiFi adapter in your laptop.

First, make sure that the Edimax is correctly installed. Using `iwconfig`, you should see four interfaces:

```
$ iwconfig
wlx[AABBCCDDEEFFGG]  unassociated  Nickname:"rtl8822bu"

[...]

lo        no wireless extensions.

enxb827eb1f81a4  no wireless extensions.

wlan1     IEEE 802.11bgn  ESSID:"duckietown"

[...]
```

Make note of the name `wlx[AABBCCDDEEFFGG]`.

Look up the MAC address using the command:

```
$ ifconfig wlx[AABBCCDDEEFFGG]
wlx74da38c9caa0 Link encap:Ethernet  HWaddr [AA:BB:CC:DD:EE:FF:GG]
```

Then, edit the connection file

`/etc/NetworkManager/system-connections/create-5ghz-network`

Make the following changes:

- Where it says `interface-name=[...]`, put "`wlx[AABBCCDDEEFFGG]`".
- Where it says `mac-address=[...]`, put "`[AA:BB:CC:DD:EE:FF:GG]`".
- Where it says `ssid=duckiebot-not-configured`, put "`ssid=[robot name]`".

Reboot.

At this point you should see a new network being created named "`[robot name]`".

You can connect with the laptop to that network.

If the Raspberry PI's network interface is connected to the `duckietown` network and to the internet, the Raspberry PI will act as a bridge to the internet.

23.7. Setting up wireless network configuration

This part should not be necessary anymore

The Duckiebot is configured by default to connect to a wireless network with SSID `duckietown`. If that is not your SSID then you will need to change the configuration.

You can add a new network by editing the file:

`/etc/wpa_supplicant/wpa_supplicant.conf`

You will see a block like the following:

```
network={
 ssid="duckietown"
 scan_ssid=1
 psk="quackquack"
 priority=10
}
```

Add a new one with your SSS and password.

This assumes you have a roughly similar wireless network setup - if not then you might need to change some of the other attributes.

23.8. Update the system

Next, we need to update to bring the system up to date.

Use these commands

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

23.9. Give a name to the Duckiebot

It is now time to give a name to the Duckiebot.

These are the criteria:

- It should be a simple alphabetic string (no numbers or other characters like "-", "_", etc.) .
- It will always appear lowercase.
- It cannot be a generic name like "`duckiebot`", "`robot`" or similar.

From here on, we will refer to this string as "`[robot name]`". Every time you see `[robot name]`, you should substitute the name that you chose.

23.10. Change the hostname

We will put the robot name in configuration files.

> Note: Files in `/etc` are only writable by `root`, so you need to use `sudo` to edit them. For example:

```
duckiebot $ sudo vi [filename]
```

Edit the file

`/etc/hostname`

and put "`[robot name]`" instead of `duckiebot-not-configured`.

Also edit the file

`/etc/hosts`

and put "`[robot name]`" where `duckiebot-not-configured` appears.

The first two lines of `/etc/hosts` should be:

```
127.0.0.1   localhost
127.0.1.1   [robot name]
```

> Note: there is a command `hostname` that promises to change the hostname. However, the change given by that command does not persist across reboots. You need to edit the files above for the changes to persist.

> Note: Never add other hostnames in `/etc/hosts`. It is a tempting fix when DNS does not work, but it will cause other problems subsequently.

Then reboot the Raspberry PI using the command

```
$ sudo reboot
```

After reboot, log in again, and run the command `hostname` to check that the change has

persisted:

```
$ hostname
[robot name]
```

23.11. Create your user

You must not use the `ubuntu` user for development. Instead, you need to create a new user.

Choose a user name, which we will refer to as `[username]`.

To create a new user:

```
duckiebot $ sudo useradd -m [username]
```

Make the user an administrator by adding it to the group `sudo`:

```
duckiebot $ sudo adduser [username] sudo
```

Make the user a member of the group `input` and `i2c`

```
duckiebot $ sudo adduser [username] input
duckiebot $ sudo adduser [username] video
duckiebot $ sudo adduser [username] i2c
```

Set the shell `bash`:

```
duckiebot $ sudo chsh -s /bin/bash andrea
```

To set a password, use:

```
duckiebot $ sudo passwd [username]
```

At this point, you should be able to login to the new user from the laptop using the password:

```
laptop $ ssh [username]@[robot name]
```

Next, you should repeat some steps that we already described.

1) Basic SSH config

Do the basic SSH config.

➡ The procedure is documented in Section 9.3.

2) Create key pair for `[username]`

Next, create a private/public key pair for the user; call it `[username]@[robot name]`.

➡ The procedure is documented in Section 9.5.

3) Add `[username]`'s public key to Github

Add the public key to your Github account.

➡ The procedure is documented in Section 22.3.

If the step is done correctly, this command should succeed:

```
duckiebot $ ssh -T git@github.com
```

4) Local Git configuration

➡ This procedure is in Section 11.3.

5) Set up the laptop-Duckiebot connection

Make sure that you can login passwordlessly to your user from the laptop.

➡ The procedure is explained in Section 9.6. In this case, we have: `[local]` = laptop, `[local-user]` = your local user on the laptop, `[remote]` = `[robot name]`, `[remote-user]` = `[username]`.

If the step is done correctly, you should be able to login from the laptop to the robot, without typing a password:

```
laptop $ ssh [username]@[robot name]
```

6) Some advice on the importance of passwordless access

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...

23.12. Other customizations

If you know what you are doing, you are welcome to install and use additional shells, but please keep Bash as be the default shell. This is important for ROS installation.

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

23.13. Hardware check: camera

Check that the camera is connected using this command:

```
duckiebot $ vcgencmd get_camera
supported=1 detected=1
```

If you see `detected=0`, it means that the hardware connection is not working.

You can test the camera right away using a command-line utility called `raspistill`.

Use the `raspistill` command to capture the file `out.jpg`:

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

Then download `out.jpg` to your computer using `scp` for inspection.

➡ For instructions on how to use `scp`, see Subsection 9.8.1.

1) Troubleshooting

Symptom: `detected=0`

Resolution: If you see `detected=0`, it is likely that the camera is not connected correctly. If you see an error that starts like this:

```
mmal: Cannot read camera info, keeping the defaults for OV5647
[...]
mmal: Camera is not detected. Please check carefully the camera module is installed correctly.
```

then, just like it says: "Please check carefully the camera module is installed correctly.".

Chapter 24
Software setup and RC remote control

Assigned to: Andrea

Prerequisites:

- You have configured the laptop.

  ➡ The procedure is documented in Chapter 21.

- You have configured the Duckiebot.

  ➡ The procedure is documented in Chapter 23.

- You have created a Github account and configured public keys, both for the laptop and for the Duckiebot.

  ➡ The procedure is documented in Chapter 22.

Results:

- You can run the joystick demo.

24.1. Clone the Duckietown repository

All of the following should be

Clone the repository in the directory `~/duckietown`:

```
duckiebot $ git clone git@github.com:duckietown/Software.git ~/duckietown
```

For the above to succeed you should have a Github account already set up.

It should not ask for a password.

1) Troubleshooting

Symptom: It asks for a password.

Resolution: You missed some of the steps described in Chapter 22.

Symptom: Other weird errors.

Resolution: Probably the time is not set up correctly. Use `ntpdate` as above:

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

24.2. Set up ROS environment on the Duckiebot

All the following commands should be run in the `~/duckietown` directory:

```
duckiebot $ cd ~/duckietown
```

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

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

Then, build the workspace using:

```
duckiebot $ catkin_make -C catkin_ws/
```

☀ *for more information about* `catkin_make`*, see Subsection 40.5.3.*

AC: I had to run it twice. The first time it complained:

```
In file included from /home/andrea/duckietown/catkin_ws/src/apriltags_ros/apriltags_ros/src/
apriltag_detector.cpp:1:0:
/home/andrea/duckietown/catkin_ws/src/apriltags_ros/apriltags_ros/include/apriltags_ros/
apriltag_detector.h:6:41: fatal error: duckietown_msgs/BoolStamped.h: No such file or directory
```

24.3. Add your vehicle to the machines file

On the robot edit the file

```
~/duckietown/catkin_ws/src/duckietown/machines
```

You will see something like this:

```
<launch>
    <arg name="env_script_path" default="~/duckietown/environment.sh"/>

    <machine name="[robot name]" address="[robot name].local" user="[username]"
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.

For example, for Andrea, `[robot name]` = `emma` and `[username]` = `andrea`. So, he would add this line:

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

Commit and push the new machines file. (No, don't commit the machines file.)

24.4. Test that the joystick is detected

Make sure that your user is in the group `input` and `i2c`:

```
duckiebot $ groups
[username] sudo input i2c
```

If `input` and `i2c` are not in the list, you missed a step. Ohi ohi! You are not following the instructions carefully!

➡ Consult again Section 23.11.

Plug the joystick receiver in one of the USB port on the Raspberry PI.

To make sure that the joystick is detected, run:

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

and check if there is a device called `js0` on the list.

To test whether or not the joystick itself is working properly, run:

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

Move the joysticks and push the buttons. You should see the data displayed change according to your actions.

24.5. Run the joystick demo

SSH into the Raspberry PI and run the following from the `duckietown` directory:

```
duckiebot $ cd ~/duckietown
duckiebot $ source environment.sh
```

The `environment.sh` setups the ROS environment at the terminal (so you can use commands like `rosrun` and `roslaunch`).

Now make sure the motor shield is connected.

Run the command:

```
duckiebot $ roslaunch duckietown joystick.launch veh:=[robot name]
```

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".

Is all of the above valid with the new joystick?

Close the program using Ctrl-C.

1) Troubleshooting

Symptom: The robot moves weirdly (e.g. forward instead of backward).

Resolution: The cables are not correctly inserted. Please refer to the assembly guide for pictures of the correct connections. Try swapping cables until you obtain the expected behavior.

Resolution: Check that the joystick has the switch set to the position "x".And the mode light should be off.

Symptom: The left joystick does not work.

Resolution: 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.

Symptom: The robot does not move at all.

Resolution: The cables are disconnected.

Resolution: The program assumes that the joystick is at `/dev/input/js0`. In doubt, see Section 24.4.

24.6. The proper shutdown procedure for the Raspberry PI

Generally speaking, you can terminate any `roslaunch` command with `Ctrl-C`.

To completely shutdown the robot, issue the following command:

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

Then wait 30 seconds.

Warning If you disconnect the power before shutting down properly using `shutdown`, the system might get corrupted.

Then, disconnect the power cable, at the **battery end**.

Warning If you disconnect frequently the cable at the Raspberry PI's end, you might damage the port.

Chapter 25
Reading from the camera

Prerequisites:

- You have configured the Duckiebot.

  ➡ The procedure is documented in Chapter 23.

- You know the basics of ROS (launch files, `roslaunch`, topics, `rostopic`).

Results:

- You know that the camera works under ROS.

25.1. Check the camera hardware

It might be useful to do a quick camera hardware check.

➡ The procedure is documented in Section 23.13.

25.2. Create two windows

On the laptop, create two Byobu windows.

➡ A quick reference about Byobu commands is in Section 12.1.

You will use the two windows as follows:

- In the first window, you will launch the nodes that control the camera.
- In the second window, you will launch programs to monitor the data flow.

Note: You could also use multiple *terminals* instead of one terminal with multiple Byobu windows. However, using Byobu is the best practice to learn.

25.3. First window: launch the camera nodes

In the first window, we will launch the nodes that control the camera.

Activate ROS:

```
duckiebot $ source environment.sh
```

Run the launch file called `camera.launch`:

```
duckiebot $ roslaunch duckietown camera.launch veh:=[robot name]
```

At this point, you should see the red LED on the camera light up continuously.

In the terminal you should not see any red message, but only happy messages like the following:

```
[...]
[INFO] [1502539383.948237]: [/[robot name]/camera_node] Initialized.
[INFO] [1502539383.951123]: [/[robot name]/camera_node] Start capturing.
[INFO] [1502539384.040615]: [/[robot name]/camera_node] Published the first image.
```

☀ *For more information about* `roslaunch` *and "launch files", see Section 40.3.*

25.4. Second window: view published topics

Switch to the second window.

Activate the ROS environment:

```
duckiebot $ source environment.sh
```

1) List topics

You can see a list of published topics with the command:

```
duckiebot $ rostopic list
```

☀ *For more information about* `rostopic`*, see Section 40.5.*

You should see the following topics:

```
/[robot name]/camera_node/camera_info
/[robot name]/camera_node/image/compressed
/[robot name]/camera_node/image/raw
/rosout
/rosout_agg
```

2) Show topics frequency

You can use `rostopic hz` to see the statistics about the publishing frequency:

```
laptop $ rostopic hz /[robot name]/camera_node/image/compressed
```

On a Raspberry PI 3, you should see a number close to 30 Hz:

```
average rate: 30.016
    min: 0.026s max: 0.045s std dev: 0.00190s window: 841
```

3) Show topics data

You can view the messages in real time with the command `rostopic echo`:

```
laptop $ rostopic echo /[robot name]/camera_node/image/compressed
```

You should see a large sequence of numbers being printed to your terminal.

That's the "image" — as seen by a machine.

If you are Neo, then this already makes sense.

If you are not Neo, use `Ctrl-C` to stop `rostopic`.

Later, in Chapter 27, you will learn how to visualize the image stream on the laptop using `rviz`.

Chapter 26
RC control launched remotely

> Assigned to: Andrea
>
> Prerequisites:
>
> - You can run the joystick demo from the Raspberry PI.
>
> ➡ The procedure is documented in Chapter 24.
>
> Results:
>
> - You can run the joystick demo from your laptop.

26.1. Two ways to launch a program

ROS nodes can be launched in two ways:

1. "local launch": log in to the Raspberry PI using SSH and run the program from there.
2. "remote launch": run the program directly from a laptop.

Which is better when is a long discussion that will be done later. Here we set up the "remote launch".

26.2. Make sure that you can log in from the laptop

Make sure that you can login with SSH *without a password*. From the laptop, run:

```
laptop $ ssh [username]@[robot name].local
```

If this doesn't work, you missed some previous steps.

26.3. Download and setup `Software` repository on the laptop

As you did on the Duckiebot, you should clone the `Software` repository in the `~/duckietown` directory.

➡ The procedure is documented in Section 24.1.

Then, you should build the repository.

➡ This procedure is documented in Section 24.2.

26.4. Edit the `machines` files on your laptop

You have to edit the `machines` files on your laptop, as you did on the Duckiebot.

➡ The procedure is documented in Section 24.3.

26.5. Start the demo

Now you are ready to launch the joystick demo remotely.

Run this *on the laptop*:

```
laptop $ source environment.sh
laptop $ roslaunch duckietown joystick.launch veh:=[robot name]
```

You should be able to drive the vehicle with joystick just like the last example. Note that remotely launching nodes from your laptop doesn't mean that the nodes are running on your laptop. They are still running on the Raspberry PI in this case.

☀ *For more information about* `roslaunch`*, see Section 40.3.*

26.6. Watch the program output using `rqt_console`

Also, you might have notice that the terminal where you launch the launch file is not printing all the printouts like the previous example. This is one of the limitation of remote launch.

Don't worry though, we can still see the printouts using `rqt_console`.

On the laptop, open a new terminal window, and run:

```
laptop $ export ROS_MASTER_URI=http://[robot name].local:11311/
laptop $ rqt_console
```

AC: I could not see any messages in `rqt_console` - not sure what is wrong.

You should see a nice interface listing all the printouts in real time, completed with filters that can help you find that message you are looking for in a sea of messages.

You can use `Ctrl-C` at the terminal where `roslaunch` was executed to stop all the nodes launched by the launch file.

☀ *For more information about* `rqt_console`*, see Section 40.2.*

26.7. Troubleshooting

Symptom: `roslaunch` fails with an error similar to the following:

```
remote[[robot name].local-0]: failed to launch on [robot name]:

Unable to establish ssh connection to [[username]@[robot name].local:22]:
Server u'[robot name].local' not found in known_hosts.
```

Resolution: You have not followed the instructions that told you to add the `HostKeyAlgorithms` option. Delete `~/.ssh/known_hosts` and fix your configuration.

➡ The procedure is documented in Section 9.3.

Chapter 27
RC+camera remotely

Assigned to: Andrea

Prerequisites:

- You can run the joystick demo remotely.

  ➡ The procedure is documented in Chapter 26.

- You can read the camera data from ROS.

  ➡ The procedure is documented in Chapter 25.

- You know how to get around in Byobu.

  ➡ You can find the Byobu tutorial in Section 12.1.

Results:

- You can run the joystick demo from your laptop and see the camera image on the laptop.

27.1. Assumptions

We are assuming that the joystick demo in Chapter 26 worked.

We are assuming that the procedure in Chapter 25 succeeded.

We also assume that you terminated all instances of `roslaunch` with `Ctrl-C`, so that currently there is nothing running in any window.

27.2. Terminal setup

On the laptop, this time create **four** Byobu windows.

➡ A quick reference about Byobu commands is in Section 12.1.

You will use the four windows as follows:

- In the first window, you will run the joystick demo, as before.
- In the second window, you will launch the nodes that control the camera.
- In the third window, you will launch programs to monitor the data flow.
- In the fourth window, you will use `rviz` to see the camera image.

27.3. First window: launch the joystick demo

In the first window, launch the joystick remotely using the same procedure in Section 26.5.

```
laptop $ source environment.sh
laptop $ roslaunch duckietown joystick.launch veh:=[robot name]
```

You should be able to drive the robot with the joystick at this point.

27.4. Second window: launch the camera nodes

In the second window, we will launch the nodes that control the camera.

The launch file is called `camera.launch`:

```
laptop $ source environment.sh
laptop $ roslaunch duckietown camera.launch veh:=[robot name]
```

You should see the red led on the camera light up.

27.5. Third window: view data flow

Open a third terminal on the laptop.

You can see a list of topics currently on the `ROS_MASTER` with the commands:

```
laptop $ source environment.sh
laptop $ export ROS_MASTER_URI=http://[robot name].local:11311/
laptop $ rostopic list
```

You should see the following:

```
/diagnostics
/[robot name]/camera_node/camera_info
/[robot name]/camera_node/image/compressed
/[robot name]/camera_node/image/raw
/[robot name]/joy
/[robot name]/wheels_driver_node/wheels_cmd
/rosout
/rosout_agg
```

27.6. Fourth window: visualize the image using `rviz`

Launch `rviz` by using these commands:

```
laptop $ source environment.sh
laptop $ source set_ros_master.sh [robot name]
laptop $ rviz
```

☀ *For more information about `rviz`, see Section 40.4.*

In the `rviz` interface, click "Add" on the lower left, then the "By topic" tag, then select the "Image" topic by the name

```
/[robot name]/camera_node/image/compressed
```

Then click "ok". You should be able to see a live stream of the image from the camera.

27.7. Proper shutdown procedure

To stop the nodes: You can stop the node by pressing `Ctrl-C` on the terminal where `roslaunch` was executed. In this case, you can use `Ctrl-C` in the terminal where you launched the `camera.launch`.

You should see the red light on the camera turn off in a few seconds.

Note that the `joystick.launch` is still up and running, so you can still drive the vehicle with the joystick.

Chapter 28
Interlude: Ergonomics

Assigned to: Andrea

So far, we have been spelling out all commands for you, to make sure that you understand what is going on.

Now, we will tell you about some shortcuts that you can use to save some time.

Note: in the future you will have to debug problems, and these problems might be harder to understand if you rely blindly on the shortcuts.

Results:

- You will know about some useful shortcuts.

28.1. `set_ros_master.sh`

Instead of using:

```
$ export ROS_MASTER_URI=http://[robot name].local:11311/
```

You can use the "`set_ros_master.sh`" script in the repo:

```
$ source set_ros_master.sh [robot name]
```

Note that you need to use `source`; without that, it will not work.

28.2. SSH aliases

Instead of using

```
$ ssh [username]@[robot name].local
```

You can set up SSH so that you can use:

```
$ ssh my-robot
```

To do this, create a host section in `~/.ssh/config` with the following contents:

```
Host my-robot
    User [username]
    Hostname [robot name].local
```

Here, you can choose any other string in place of "`my-robot`".

Note that you **cannot** do

```
$ ping my-robot
```

You haven't created another hostname, just an alias for SSH.

However, you can use the alias with all the tools that rely on SSH, including `rsync` and `scp`.

Chapter 29
Wheel calibration

Assigned to: Andrea

Chapter 30
Camera calibration

Chapter 31
Taking a log

Assigned to: Andrea

Chapter 32
D17-1 (LEDs) parts

Assigned to: Jacopo

Chapter 33
D17–1 (LEDs) assembly

Assigned to: Shiying

Chapter 34
D17-1 (LEDs) setup

Assigned to: Andrea

Part 5
Duckietowns

Chapter 35
Duckietown parts

Assigned to: Jacopo

Chapter 36
Duckietown Assembly

Assigned to: Shiying

Chapter 37
The Duckietown specification

Assigned to: Liam?

37.1. Topology

1) Topology constraints

37.2. Signs placement

Chapter 38
Traffic lights

PART 6
Developing software

This part is about how to develop software for the Duckiebot.

Chapter 39
Python

39.1. Background reading

- Python
- Python tutorial

39.2. Python virtual environments

Install using:

```
$ sudo apt install virtualenv
```

39.3. Useful libraries

```
matplotlib
seaborne
numpy
panda
scipy
opencv
...
```

Chapter 40
Introduction to ROS

Assigned to: Liam

40.1. Install ROS

This part installs ROS. You will run this twice, once on the laptop, once on the robot.

The first commands are copied from this page.

Tell Ubuntu where to find ROS:

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

Tell Ubuntu that you trust the ROS people (they are nice folks):

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

Fetch the ROS repo:

```
$ sudo apt update
```

Now install the mega-package `ros-kinetic-desktop-full`.

```
$ sudo apt install ros-kinetic-desktop-full
```

There's more to install:

```
$ sudo apt install
ros-kinetic-{tf-conversions,cv-bridge,image-transport,camera-info-manager,theora-image-transport,joy,image-
```

Note: Do not install packages by the name of `ros-X`, only those by the name of `ros-kinetic-X`. The packages `ros-X` are from another version of ROS.

: not done in aug20 image:

Initialize ROS:

```
$ sudo rosdep init
$ rosdep update
```

40.2. `rqt_console`

40.3. `roslaunch`

40.4. `rviz`

40.5. rostopic

1) rostopic hz

2) rostopic echo

3) catkin_make

4) Troubleshooting

Symptom: [computer] is not in your SSH known_hosts file

See this thread. Remove the known_hosts file and make sure you have followed the instructions in Section 9.3.

Chapter 41
What the duck!

`what-the-duck` is a program that tests *dozens* of configuration inconsistencies that can happen on a Duckiebot.

To use it, first compile the repository, and then run:

```
$ ./what-the-duck
```

41.1. Adding more tests to `what-the-duck`

The idea is to add to `what-the-duck` all the tests that can be automated.

The documentation about to do that is not ready yet.

The current tests are available in the file:

```
./catkin_ws/src/f23-LED/led_detection/include/what_the_duck/list_of_checks.py
```

41.2. Tests already added

Here's the list of tests already added:

```
✓  Camera is detected
✓  Scipy is installed
✓  sklearn is installed
✓  Date is set correctly
✓  Not running as root
✓  Not running as ubuntu
✓  Member of group sudo
✓  Member of group input
✓  Member of group video
✓  Member of group i2c
✓  ~/.ssh exists
✓  ~/.ssh permissions
✓  ~/.ssh/config exists
✓  SSH option HostKeyAlgorithms is set
✓  At least one key is configured.
✓  ~/.ssh/authorized_keys exists
✓  Git configured
✓  Git email set
✓  Git name set
✓  Git push policy set
✓  Edimax detected
✓  The hostname is configured
✓  /etc/hosts is sane
✓  Correct kernel version
✓  Messages are compiled
✓  Shell is bash
✓  Working internet connection
✓  Github configured
✓  Joystick detected
✓  Environment variable DUCKIETOWN_ROOT
✓  ${DUCKIETOWN_ROOT} exists
✓  Wifi network configured
```

41.3. List of tests to add

Please add below any configuration test that can be automated:

- Check that all the `rosX` command resolve to a file`/opt/ros/kinetic/bin/rosX`.
- Make sure that packages such as `python-roslaunch` are not installed. (The user is invited to install it when `roslaunch` is not found!)

Chapter 42
How to create a ROS package

42.1. Conforming ROS package checklist

□ The name of the package is `package_[handle]`

□ The directory is in ...

□ The messages are called ....

□ there is a readme file

□ there is the first launch file

Chapter 43
Integrate package in the architecture

Chapter 44
Creating unit tests

Part 7
Modules

These are the learning modules.

Chapter 45
Module template

Assigned to: Jacopo

Chapter 46
Autonomy overview

Assigned to: Liam

46.1. Perception, planning, control

CHAPTER 47
Autonomy architectures

Assigned to: Andrea

Chapter 48
Representations

Assigned to: Matt

Chapter 49
Software architectures and middlewares

Assigned to: Andrea

Chapter 50
Modern signal processing

Assigned to: Andrea

Chapter 51
Basic Kinematics

Assigned to: Jacopo

Chapter 52
Basic Dynamics

Assigned to: Jacopo

Chapter 53
Odometry Calibration

Assigned to: Jacopo

Chapter 54
Computer vision basics

Assigned to: Matt

Chapter 55
Illumination invariance

Assigned to: Matt

Chapter 56
Line Detection

Assigned to: Matt

Chapter 57
Feature extraction

Assigned to: Matt

Chapter 58
Place recognition

Assigned to: Matt

Chapter 59
Filtering 1

Assigned to: Liam

Chapter 60
Filtering 2

Assigned to: Liam

Chapter 61
Mission planning

Assigned to: ETH

Chapter 62
Planning in discrete domains

Assigned to: ETH

Chapter 63
Motion planning

Assigned to: ETH

Chapter 64
RRT

Assigned to: ETH

Chapter 65
Feedback control

Assigned to: Jacopo

Chapter 66
PID Control

Assigned to: Jacopo

Chapter 67
MPC Control

Assigned to: Jacopo

Chapter 68
Object detection

Assigned to: Nick and David

Chapter 69
Object classification

Assigned to: Nick and David

Chapter 70
Object tracking

Assigned to: Nick and David

Chapter 71
Reacting to obstacles

Assigned to: Jacopo

Chapter 72
Semantic segmentation

Assigned to: Nick and David

Chapter 73
Text recognition

Assigned to: Nick

Chapter 74
SLAM - Problem formulation

Assigned to: Liam

Chapter 75
SLAM - Broad categories

Assigned to: Liam

Chapter 76
VINS

Assigned to: Liam

Chapter 77
Advanced place recognition

Assigned to: Liam

Chapter 78
Fleet level planning (placeholder)

Assigned to: ETH

Chapter 79
Fleet level planning (placeholder)

Assigned to: ETH

Part 8
How to contribute

Chapter 80
Accounts

80.1. Complete list of accounts

Currently, Duckietown has the following accounts:

- Github: for source code, and issue tracking;
- Slack: a forum for wide communication;
- Twist: to be used for instructors coordination;
- Google Drive: to be used for instructors coordination, maintaining TODOs, etc;
- Dropbox Folders (part of Andrea's personal accounts): to be abandoned;
- Vimeo, for storing the videos;
- The `duckietown-teaching` mailing list, for low-rate communication with instructors;
- We also have a list of addresses, of people signed up on the website, that we didn't use yet;
- The Facebook page.

80.2. For Fall 2017

As a student in Fall 2017, these are the accounts that you need:

- A Github account and membership in the Duckietown organization.
- A Slack account, for team discussion and organization.

As an instructor/TA for the Fall 2017 class, in addition to the accounts above, these are the accounts that you need:

- Twist: for class organization (such as TAs, logistics);
- Google Docs, used to maintain TODOs.

80.3. For other contributors

If you are an international contributor:

- Sign up on Slack, to keep up with the project.
- (optional) Get Github permissions if you do frequent updates to the repositories.

CHAPTER 81
Contributing to the documentation

81.1. Where the documentation is

All the documentation is in the repository `duckietown/duckuments`.

The documentation is written as a series of small files in Markdown format.

It is then processed by a series of scripts to create this output:

- a publication-quality PDF;
- an online HTML version, split in multiple pages and with comments boxes.

81.2. Editing links

The simplest way to contribute to the documentation is to click any of the "✎" icons next to the headers.

They link to the "edit" page in Github. There, one can make and commit the edits in only a few seconds.

81.3. Comments

In the multiple-page version, each page also includes a comment box powered by a service called Disqus. This provides a way for people to write comments with a very low barrier. (We would periodically remove the comments.)

81.4. Installing dependencies for compiling the documentation

Let `DUCKUMENTS` be the base directory for the documentation.

Download the `duckuments` repo in that directory:

```
$ git clone git@github.com:duckietown/duckuments.git $DUCKUMENTS
```

Cd into directory:

```
$ cd $DUCKUMENTS
```

1) Setup a virtual environment

On Ubuntu 16.04, create a virtual environment usign `virtualenv` (`sudo apt install virtualenv` if needed):

```
$ virtualenv --system-site-packages deploy
```

In other distributions you might need to use `venv`:

```
$ venv deploy
```

Activate the virtual environment:

```
$ source $DUCKUMENTS/deploy/bin/activate
```

Install some dependencies:

```
$ sudo apt install libxml2-dev libxslt1-dev
$ sudo apt install libffi6 libffi-dev
$ sudo apt install python-dev python-numpy python-matplotlib
```

Clone the `mcdp` external repository:

```
$ cd $DUCKUMENTS
$ git clone -b duckuments git@github.com:AndreaCensi/mcdp.git
```

Install it and its dependencies:

```
$ cd $DUCKUMENTS/mcdp
$ python setup.py develop
```

(If you get a permission error here, it means you have not properly activated the virtualenv)

Depending on your system, you might need to install these other dependencies: (It should not be necessary on Ubuntu 16 given the `apt` commands above.)

```
$ cd $DUCKUMENTS
$ pip install numpy matplotlib
```

81.5. Extra dependencies for compiling the PDF version

> Note: The dependencies below are harder to install. If you don't manage to do it, then you only lose the ability to compile the PDF. You can do `make compile` to compile the HTML version, but you cannot do `make compile-pdf`.

Ensure the latest version (>6) of `nodejs` is installed.

Run:

```
$ nodejs --version
6.xx
```

If the version is 4 or less, remove `nodejs`:

```
$ sudo apt remove nodejs
```

Install `nodejs` using the instructions at this page.

Next, install the necessary Javascript libraries using `npm`:

```
$ cd $DUCKUMENTS
$ npm install MathJax-node jsdom@9.3 less
```

Install PrinceXML from this page.

Download STIX fonts from this site.

Unzip and copy the ttf to `~/.fonts`:

```
$ cp -R STIXv2.0.0 ~/.fonts
```

and then rebuild the font cache using:

```
$ fc-cache -fv
```

81.6. Troubleshooting installation problems

1) Installing nodejs packages

The only pain point in the installation procedure has been the installation of nodejs packages using npm. For some reason, they cannot be installated globally (npm install -g).

Do not use sudo for installation. It will cause problems.

If you use sudo, you probably have to delete a bunch of directories, such as: RBROOT/node_modules, ~/.npm, and ~/.node_modules, if they exist.

81.7. Compiling the documentation

Make sure you have deployed and activated the virtual environment. Then:

```
$ cd $DUCKUMENTS
$ make duckuments-dist
```

This creates the directory duckuments-dist, which contains another checked out copy of the repository, but with the branch gh-pages, which is the branch that is published by Github using the "Github Pages" mechanism.

At this point, please make sure that you have these two .git folders:

```
$DUCKUMENTS/.git
$DUCKUMENTS/duckuments-dist/.git
```

To compile the docs, go in the DUCKUMENTS directory and run make compile:

```
$ cd $DUCKUMENTS
$ make clean compile
```

To see the result, open the file

```
./duckuments-dist/master/duckiebook/index.html
```

1) Incremental compilation

If you want to do incremental compilation, you can omit the clean and just use:

```
$ make compile
```

This will be faster. However, sometimes it might get confused. At that point, do make clean.

2) Compiling the PDF

To compile the PDF, use:

```
$ make compile-pdf
```

This creates the file:

```
./duckuments-dist/master/duckiebook.pdf
```

81.8. Deploying the documentation

This part is now done by a bot, so you don't need to do it manually.

To deploy the documentation, jump into the `DUCKUMENTS/duckuments-dist` directory.

Run the command `git branch`. If the out does not say that you are on the branch `gh-pages`, then one of the steps before was done incorrectly.

```
$ cd $DUCKUMENTS/duckuments-dist
$ git branch
...
* gh-pages
...
```

Now, after triple checking that you are in the `gh-pages` branch, you can use `git status` to see the files that were added or modified, and simply use `git add`, `git commit` and `git push` to push the files to Github.

81.9. In summary: the workflow

This is the workflow:

1. Edit the Markdown in the `master` branch of the `duckuments` repository.
2. Run `make compile` to make sure it compiles.
3. Commit the Markdown and push on the `master` branch.

Done. The bot will redo the compilation and push the changes in the `gh-pages` branch.

Step 2 is done, so you know that the bot will not encounter errors.

Chapter 82
Features of the documentation writing system

The Duckiebook is written in a Markdown dialect. A subset of LaTeX is supported.

There are also some additional features that make it possible to create publication-worthy materials.

82.1. Embedded LaTeX

You can use $\LaTeX$ math, environment, and references. For example, take a look at

$$x^2 = \int_0^t f(\tau)\ \text{d}\tau$$

or refer to Proposition 1.

Proposition 1. (Proposition example) This is an example proposition: $2x = x + x$.

The above was written as in Figure 7.

```
You can use $\LaTeX$ math, environment, and references.
For example, take a look at

\[
    x^2 = \int_0^t f(\tau)\ \text{d}\tau
\]

or refer to [](#prop:example).

\begin{proposition}[Proposition example]\label{prop:example}
This is an example proposition: $2x = x + x$.
\end{proposition}
```

Figure 7. Use of LaTeX code.

82.2. Other interesting features

Make sure to quote (with 4 spaces) all command lines. Otherwise, the dollar symbol confuses the LaTeX interpreter.

1) Shortcut for tables

The shortcuts `col2`, `col3`, `col4`, `col5` are expanded in tables with 2, 3, 4 or 5 columns.

```
<div make-col2="">
    <span>A</span>
    <span>B</span>
    <span>C</span>
    <span>D</span>
</div>
```

2) Creating figures

For any element, adding an attribute called `figure-id` with contents `fig:[figure ID]` or `tab:[table ID]` will create a figure that wraps the element.

For example:

```
<p figure-id="fig:code">
    I will be the content of a figure.
</p>
```

It will create HMTL of the form:

```
<div id='fig:code-wrap' class='generated-figure-wrap'>
    <figure id='fig:code' class='generated-figure'>
        <p figure-id="fig:code">
            I will be the content of a figure.
        </p>
    </figure>
</div>
```

To add a class to the figure, use `figure-class`:

```
<element figure-id="fig:code" figure-class="myclass">
    content
</element>
```

This will give it to the `<figure>` and the containing `<figure>`

To add a caption, add an attribute `figure-caption`:

```
<element figure-id="fig:code" figure-caption="This is my caption">
    content
</element>
```

Alternatively, you can put anywhere an element `figcaption` with ID `[figure id]:caption`:

```
<element figure-id="fig:code">
    content
</element>

<figcaption id='fig:code:caption'>
    This is my caption. Can contain <code>code</code>.
</figcaption>
```

82.3. Character escapes

Use the string `&#36;` to write the dollar symbol $, otherwise it gets confused with LaTeX math materials. Also notice that you should probably use "USD" to refer to U.S. dollars

Other symbols to escape:

- use `&#96;` instead of `
- use `&#36;` instead of $
- use `&lt;` instead of <
- use `&gt;` instead of >

82.4. Troubleshooting

Symptom: "Invalid XML"

Resolution: "Markdown" doesn't mean that you can put anything in a file. Except for the code blocks, it must be valid XML. For example, if you use ">" and "<" without quoting, it will likely cause a compile error.

Symptom: "Tabs are evil"

Resolution: Do not use tab characters. The error message in this case is quite helpful in telling you exactly where the tabs are.

Symptom: The error message contains `ValueError: Suspicious math fragment '$\LaTeX$'`

Resolution: You probably have forgotten to indent a command line by at least 4 spaces. The dollar in the command line is now being confused for a math formula.

CHAPTER 83

Documentation style guide

This chapter describes the conventions for writing the technical documentation.

83.1. General guidelines for technical writing

The following holds for all technical writing.

- The documentation is written in correct English.
- Do not say "should" when you mean "must". "Must" and "should" have precise meanings and they are not interchangeable. These meanings are explained in this document.
- "Please" is unnecessary in technical documentation.
  - ✖ "Please remove the SD card."
  - ✔ "Remove the SD card".
- Do not use colloquialisms or abbreviations.
  - ✖ "The pwd is `ubuntu`."
  - ✔ "The password is `ubuntu`."
- Do not use emojis.
- Do not use ALL CAPS.
- Make infrequent use of **bold statements**.
- Do not use exclamation points.

83.2. Style guide for the Duckietown documentation

- It's ok to use "it's" instead of "it is", "can't" instead of "cannot", etc.
- All the filenames and commands must be enclosed in code blocks using Markdown backticks.
  - ✖ "Edit the ~/.ssh/config file using vi."
  - ✔ "Edit the `~/.ssh/config` file using `vi`."
- `Ctrl-C`, `ssh` etc. are not verbs.
  - ✖ "Ctrl-C from the command line".
  - ✔ "Use Ctrl-C from the command line".
- Subtle humor and puns about duckies are encouraged.

83.3. Writing command lines

Use either "`laptop`" or "`duckiebot`" (not capitalized, as a hostname) as the prefix for the command line.

For example, for a command that is supposed to run on the laptop, use:

```
laptop $ cd ~/duckietown
```

For a command that must run on the Duckiebot, use:

```
duckiebot $ cd ~/duckietown
```

If the command is supposed to be run on both, omit the hostname:

```
$ cd ~/duckietown
```

83.4. Frequently misspelled words

- "Duckiebot" is always capitalized.
- Use "Raspberry PI", not "PI", "raspi", etc.
- These are other words frequently misspelled: 5 GHz WiFi

83.5. Other conventions

When the user must edit a file, just say: "edit `/this/file`".

Writing down the command line for editing, like the following:

```
$ vi /this/file
```

is too much detail.

(If people need to be told how to edit a file, Duckietown is too advanced for them.)

83.6. Troubleshooting sections

Write the documentation as if every step succeeds.

Then, at the end, make a "Troubleshooting" section.

Organize the troubleshooting section as a list of symptom/resolution.

The following is an example of a troubleshooting section.

1) Troubleshooting

Symptom: This strange thing happens.

Resolution: Maybe the camera is not inserted correctly. Remove and reconnect.

Symptom: This other strange thing happens.

Resolution: Maybe the plumbus is not working correctly. Try reformatting the plumbus.

Part 9
Fall 2017

This is the first time that a class is taught jointly across 3 continents!

There are 4 universities involved in the joint teaching for the term:

- ETH Zürich (ETHZ), with instructors Emilio Frazzoli, Andrea Censi, Jacopo Tani.
- University of Montreal (UdeM), with instructor Liam Paull.
- TTI Chicago (TTI), with instructor Matthew Walter.
- National C T University (NCTU), with instructor Nick Wang.

This part of the Duckiebook describes all the information that is needed by the students of the four institutions.

Chapter 84
General remarks

Assigned to: Andrea

84.1. The rules of Duckietown

The first rule of Duckietown

The first rule of Duckietown is: you don't talk about Duckietown, *using email*.

Instead, we use a communication platform called Slack.

There is one exception: inquiries about "meta" level issues, such as course enrollment and other official bureaucratic issues can be communicated via email.

The second rule of Duckietown

The second rule of Duckietown is: be kind and respectful, and have fun.

The third rule of Duckietown

The third rule of Duckietown is: read the instructions carefully.

Do not blindly copy and paste.

Only run a command if you know what it does.

84.2. Synchronization between classes

At ETHZ, UdeM, TTIC, the class will be more-or-less synchronized. The materials are the same; there is some slight variation in the ordering.

Moreover, there will be some common groups for the projects.

The NCTU class is undergraduate level. Students will learn slightly simplified materials. They will not collaborate directly with the classes.

84.3. Accounts for students

To participate in Duckietown, students must use two accounts: Slack and Github.

1) Slack

2) Github

84.4. Accounts for all instructors and TAs

There are two more accounts required for instructors and TAs

1) Twist

TODO:

2) Google docs

TODO:

In particular:

- This is the schedule: ...
- This is the calendar in which to annotate everything: ...

84.5. Other accounts for organizers

1) Duckietown-teaching

84.6. Additional information for ETH Zürich students

Assigned to: Andrea

This section describes information specific for ETH Zürich students.

1) Website

All really important information, such as deadlines, is in the authoritative website:

2) Duckiebox distribution

3) Lab access

4) The local TAs

84.7. Additional information for UdeM students

Assigned to: Liam

84.8. Additional information for TTIC students

Assigned to: Matt

84.9. Additional information for NCTU students

Assigned to: Nick

Chapter 85
Project proposals

Chapter 86
Template of a project

Part 10
Fall 2017 student progression

CHAPTER 87

Milestone: ROS node working

Chapter 88
Homework: Take and process a log

Chapter 89
Milestone: Calibrated robot

Chapter 90
Homework: Camera geometry

Chapter 91
Milestone: Illumination invariance

Chapter 92
Homework: Place recognition

CHAPTER 93
Milestone: Lane following

Chapter 94
Homework: localization

Chapter 95
Milestone: Navigation

Chapter 96
Homework: group forming

Chapter 97
Milestone: Ducks in a row

Chapter 98
Homework: Comparison of PID

Chapter 99
Homework: RRT

Chapter 100
Caffe tutorial

CHAPTER 101

Milestone: Object Detection

Chapter 102
Homework: Object Detection

Chapter 103
Milestone: Semantic perception

Chapter 104
Homework: Semantic perception

Chapter 105
Milestone: Reacting to obstacles

Chapter 106
Homework: Reacting to obstacles

CHAPTER 107

Milestone: SLAM demo

Chapter 108
Homework: SLAM

CHAPTER 109

Milestone: fleet demo

Chapter 110
Homework: fleet

110.1. Checklist for students

- Have a Github account. See Chapter 22. See name conventions (TODO).
- Be part of the Duckietown Github organization. You are sure only when you commit and push one change to one of our repositories.
- Be part of the Duckietown Slack. See name conventions (TODO).

110.2. Checklist for TAs

- Be signed up on

CHAPTER 111
Bibliography

[1] Jacopo Tani, Liam Paull, Maria Zuber, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. **Duckietown: an innovative way to teach autonomy.** In *EduRobotics 2016.* Athens, Greece, December 2016. pdf

[2] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Daniel Hoehener, Shih-Yuan Liu, Michael Novitzky, Igor Franzoni Okuyama, Jason Pazis, Guy Rosman, Valerio Varricchio, Hsueh-Cheng Wang, Dmitry Yershov, Hang Zhao, Michael Benjamin, Christopher Carr, Maria Zuber, Sertac Karaman, Emilio Frazzoli, Domitilla Del Vecchio, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. **Duckietown: an open, inexpensive and flexible platform for autonomy education and research.** In *IEEE International Conference on Robotics and Automation (ICRA).* Singapore, May 2017. pdf

[3] Tosini, G., Ferguson, I., Tsubota, K. *Effects of blue light on the circadian system and eye physiology.* Molecular Vision, 22, 61–72, 2016 (online).

Part 11
Drafts or pieces to remove

Chapter 112
Laptop setup

112.1. Setup passwordless SSH to log in using the `ubuntu` user

On each Duckiebot it is possible to log in as the `ubuntu` user using a common key.

Now, let's set up passwordless SSH, so that you don't need to type a password.

On the laptop, create the `.ssh` directory:

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

The key `duckietown_key1` is found at the URL:

```
https://www.dropbox.com/s/q23qptu01u7ur3y/duckietown_key1?dl=1
```

Download the file and call it `~/.ssh/duckietown_key1`

```
laptop $ curl -o ~/.ssh/duckietown_key1 [URL above]
```

Edit the permission of the file. SSH wants the key file to be not readable or writable from other users or groups.

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

Regenerate the public key according to:

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

On the laptop, now edit `~/.ssh/config` and add the following lines:

```
Host [robot name]
  Hostname [robot name].local
  User [user name]
  IdentityFile ~/.ssh/duckietown_key1
  HostKeyAlgorithms ssh-rsa
```

Now you should be able to connect without using a password.

The following command should connect without a password being asked:

```
laptop $ ssh ubuntu@[robot name]
```

1) Troubleshooting

Symptom: "Scheme missing"

Resolution: If there are issues such as "scheme missing" and the file `duckietown_key1` does not exist in the `~/.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/`.