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
Meta - 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.

Chapter 5
Accounts

5.1. 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.

5.2. 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.

Part 2
Meta - How to contribute

Chapter 6
Contributing to the documentation

6.1. 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.

6.2. 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.

6.3. 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.)

6.4. Installing the documentation system

In the following, we are going to assume that the documentation system is installed in `~/duckuments`. However, it can be installed anywhere.

We are also going to assume that you have setup a Github account with working public keys.

1) Dependencies

On Ubuntu 16.04, these are the dependencies to install:

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

2) Download the `duckuments` repo

Download the `duckietown/duckuments` repository in that directory:

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

3) Setup the virtual environment

Next, we will create a virtual environment using inside the `~/duckuments` directory.

Change into that directory:

```
$ cd ~/duckuments
```

Create the virtual environment using `virtualenv`:

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

Other distributions: In other distributions you might need to use `venv` instead of `virtualenv`.

Activate the virtual environment:

```
$ source ~/duckuments/deploy/bin/activate
```

4) Setup the `mcdp` external repository

Make sure you are in the directory:

```
$ cd ~/duckuments
```

Clone the `mcdp` external repository, with the branch `duckuments`.

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

Install it and its dependencies:

```
$ cd ~/duckuments/mcdp
$ python setup.py develop
```

> Note: If you get a permission error here, it means you have not properly activated the virtual environment.

Other distributions: If you are not on Ubuntu 16, depending on your system, you might need to install these other dependencies:

```
$ pip install numpy matplotlib
```

6.5. Compiling the documentation

Check before you continue

Make sure you have deployed and activated the virtual environment. You can check this by checking which `python` is active:

```
$ which python
/home/user/duckuments/deploy/bin/python
```

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.

> **Check before you continue**
> At this point, please make sure that you have these two `.git` folders:
>
> ```
> ~/duckuments/.git
> ~/duckuments/duckuments-dist/.git
> ```

To compile the docs, run `make clean compile`:

```
$ 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`.

6.6. The workflow to edit documentation.

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

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

6.7. *Deploying the documentation

> Note: 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.

6.8. *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`.

1) Installing `nodejs`

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
```

2) Troubleshooting `nodejs` installation problems

The only pain point in the installation procedure has been the installation of `nodejs` packages using `npm`. For some reason, they cannot be installed 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.

3) Installing Prince

Install PrinceXML from this page.

4) Installing fonts

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
```

5) Compiling the PDF

To compile the PDF, use:

```
$ make compile-pdf
```

This creates the file:

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

CHAPTER 7

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.

7.1. 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 Listing 1.

```
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}
```

Listing 1. Use of LaTeX code.

7.2. LaTeX symbols

The LaTeX symbols definitions are in a file called `docs/symbols.tex`.

Put all definitions there; if they are centralized it is easier to check that they are coherent.

7.3. Variables in command lines and command output

Use the syntax "`![name]`" for describing the variables in the code.

Example 1.

For example, to obtain:

```
$ ssh robot name.local
```

Use the following:

```
For example, to obtain:

    $ ssh ![robot name].local
```

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

7.4. 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 are shown in Table 1.

TABLE 1. SYMBOLS TO ESCAPE

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

7.5. Keyboard keys

Use the `kbd` element for keystrokes.

Example 1.

For example, to obtain:

Press a then Ctrl-C.

use the following:

```
Press <kbd>a</kbd> then <kbd>Ctrl</kbd>-<kbd>C</kbd>.
```

7.6. Figures

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

For example:

```
<div figure-id="fig:figure ID">
    figure content
</div>
```

It will create HMTL of the form:

```
<div id='fig:code-wrap' class='generated-figure-wrap'>
    <figure id='fig:figure ID' class='generated-figure'>
        <div>
            figure content
        </div>
    </figure>
</div>
```

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

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

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

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

<figcaption id='fig:figure ID:caption'>
    This the caption figure.
</figcaption>
```

To refer to the figure, use an empty link:

```
Please see [](#fig:figure ID).
```

The code will put a reference to “Figure XX”.

7.7. Subfigures

You can also create subfigures, using the following syntax.

```
<div figure-id="fig:big">
    <figcaption>Caption of big figure</figcaption>

    <div figure-id="subfig:first">
        <figcaption>Caption 1</figcaption>
        <p>Content of first subfig</p>
    </div>

    <div figure-id="subfig:second">
        <figcaption>Caption 2</figcaption>
        <p>Content of second subfig</p>
    </div>
</div>
```

Content of first subfig

(a) Caption 1

Content of second subfig

(b) Caption 2

Figure 1. Caption of big figure

7.8. Shortcut for tables

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

```
<col2 figure-id="tab:mytable" figure-caption="My table">
    <span>A</span>
    <span>B</span>
    <span>C</span>
    <span>D</span>
</col2>
```

gives the following result:

TABLE 1. MY TABLE

A	B
C	D

1) `labels-row1` and `labels-row1`

Use the classes `labels-row1` and `labels-row1` to make pretty tables like the following.

`labels-row1`: the first row is the headers.

`labels-col1`: the first column is the headers.

TABLE 1. USING `CLASS="LABELS-COL1"`

header A	B	C
header D	E	F
header G	H	I

TABLE 1. USING `CLASS="LABELS-ROW1"`

header A	header B	header C
D	E	F
G	H	I

7.9. 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 8
Documentation style guide

This chapter describes the conventions for writing the technical documentation.

8.1. 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.

8.2. 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.

8.3. 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
```

It will become:

 `$ cd ~/duckietown`

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

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

It will become:

 `$ cd ~/duckietown`

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

```
$ cd ~/duckietown
```

8.4. Frequently misspelled words

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

8.5. 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.)

8.6. 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.

Chapter 9
Knowledge graph

9.1. Formalization

1) Atoms

Definition 1. (Atom) An *atom* is a concrete resource (text, video) that is the smallest unit that is individually addressable. It is indivisible.

Each atom as a type, as follows:

```
text
  text/theory
  text/setup
  text/demo
  text/exercise
  text/reference
  text/instructor-guide
  text/quiz
video
  video/lecture
  video/instructable
  video/screencast
  video/demo
```

2) Semantic graph of atoms

Atoms form a directed graph, called "semantic graph".

Each node is an atom.

The graph has four different types of edges:

- "**Requires**" edges describe a strong dependency: "You need to have done this. Otherwise it will not work."
- "**Recommended**" edges describe a weaker dependency; it is not strictly necessary to have done that other thing, but it will significantly improve the result of this.
- "**Reference**" edges describe background information. "If you don't know / don't remember, you might want to see this"
- "**See also**" edges describe interesting materials for the interested reader. Completely optional; it will not impact the result of the current procedure.

3) Modules

A "module" is an abstraction from the point of view of the teacher.

Definition 1. (Module) A *module* is a directed graph, where the nodes are either atoms or other modules, and the edges can be of the four types described in Subsection 9.1.2.

Because modules can contain other modules, they allow to describe hierarchical contents. For example, a class module is a module that contains other modules; a "degree" is a module that contains "class" modules, etc.

Modules can overlap. For example, a "Basic Object Detection" and an "Advanced Ob-

ject Detection" module might have a few atoms in common.

9.2. Atoms properties

Each atom has the following **properties**:

- An **ID** (alphanumeric + - and '_'). The ID is used for cross-referencing. It is the same in all languages.
- A **type**, as above.

There might be different **versions** of each atom. This is used primarily for dealing with translations of texts, different representations of the same image, Powerpoint vs Keynote, etc.

A version is a tuple of attributes.

The attributes are:

- **Language**: A language code, such as `en-US` (default), `zh-CN`, etc.
- **Mime type**: a MIME type.

Each atom version has:

- A **status**: one of `draft`, `ready`.
- A human-readable **title**.
- A human-readable **summary** (1 short paragraph).

9.3. Markdown format for text-like atoms

For the text-like resources, they are described in Markdown files.

The name of the file does not matter.

All files are encoded in UTF-8.

Each file starts with a `H1` header. The contents is the title.

The header has the following attributes:

1. The ID. (`{#ID}`)
2. The language is given by an attribute `lang` (`{lang=en-US}`).
3. The type is given by an attribute `type` (`{type=demo}`).
4. The status is given by an attribute `status` (`{status=draft}`).

Here is an example of a header with all the attributes:

```
# Odometry calibration {#odometry-calibration  lang=en-US type='text/theory' status=ready}

This first paragraph will be used as the "summary" for this text.
```

Listing 1. `calibration.en.md`

And this is how the Italian translation would look like:

```
# Calibrazione dell'odometria {#odometry-calibration  lang=it type='text/theory' status=draft}

Questo paragrafo sarà usato come un sommario del testo.
```

Listing 1. `calibration.it.md`

9.4. How to describe the semantic graphs of atoms

In the text, you describe the semantic graph using tags and IDs.

In Markdown, you can give IDs to sections using the syntax:

```
# Setup step 1  {#setup-step1}

This is the first setup step.
```

Then, when you write the second step, you can add a a semantic edge using the following.

```
# Setup step 2  {#setup-step2}

This is the second setup step.

Requires: You have completed the first step in [](#setup-step1).
```

The following table describes the syntax for the different types of semantic links:

TABLE 1. SEMANTIC LINKS

Requires	`Requires: You need to have done [](#setup-step).`
Recommended	`Recommended: It is better if you have setup Wifi as in [](#setup-wifi).`
Reference	`Reference: For more information about rostopic, see [](#rostopic).`
See also	`See also: If you are interested in feature detection, you might want to learn about [SIFT](#SIFT).`

9.5. How to describe modules

Chapter 10
Basic Markdown Reference

> **Assigned to:** Andrea

PART 3

Operation manual - Duckiebot

Chapter 11
Duckiebot configurations

11.1. Configuration list

Configuration `D17-0`: Only camera and motors.
Configuration `D17-0+w`: Previous one + an additional WiFi card (Edimax).
Configuration `D17-0+j`: Previous one + joystick.
Configuration `D17-1`: LED lights and bumpers

11.2. Configuration functionality

Chapter 12
Acquiring the parts for the Duckiebot C0

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

In general, keep in mind that:

- The links might expire, or the prices might vary.
- Shipping times and fees vary, and are not included in the prices shown below.
- Substitutions are OK for the mechanical components, and not OK for all the electronics, unless you are OK in writing some software.
- Buying the parts for more than one Duckiebot makes each one cheaper than buying only one.

> Resources necessaries:
>
> - Cost: USD 193.50 + Shipping Fees (configuration D17-0)
> - Time: 15 days (average shipping for cheapest choice of components)
>
> Results:
>
> - A kit of parts ready to be assembled.

12.1. Bill of materials

TABLE 1. BILL OF MATERIALS

Chassis	USD 20
Camera with 160-FOV Fisheye Lens	USD 22
Camera Mount	USD 8.50
300mm Camera Cable	USD 2
Raspberry Pi 3 - Model B	USD 35
Heat Sinks	USD 5
Power supply	USD 7.50
16 GB Class 10 MicroSD Card	USD 20
Mirco SD card reader	USD 6
Tiny 32GB USB Flash Drive	USD 12.50
Stepper Motor HAT	USD 22.50
Stacking Headers 2 for D17-1, 1 otherwise	USD 2.50/piece
Battery	USD 20
16 Nylon Standoffs (M2.5 12mm F 6mm M	USD 0.05/piece
4 Nylon Hex Nuts (M2.5)	USD 0.02/piece
4 Nylon Screws (M2.5x10)	USD 0.05/piece
2 Zip Ties (300x5mm)	USD 8.99
Wifi Augmenter (D17-0+w)	USD 20
Joypad (D17-0+j)	USD 10.50
20 Female-Female Jumper Wires (300mm) (D17-1)	USD 8
Male-Male Jumper Wire (150mm) (D17-1)	USD 1.95
LEDs (D17-1)	USD 10
LED HAT (D17-1)	USD 28.20 for 3 pieces
PWM/Servo HAT (D17-1)	USD 17.50
40 pin female header (D17-1)	USD 1.50
Bumpers (D17-1)	TBD (custom made)
5 4 pin female header (D17-1)	USD 0.60/piece
2 16 pin male header (D17-1)	USD 0.61/piece
12 pin male header (D17-1)	USD 0.48/piece
3 pin male header (D17-1)	USD 0.10/piece
2 pin female shunt jumper (D17-1)	USD 2/piece
5 200 Ohm resistors (D17-1)	USD 0.10/piece
10 130 Ohm resistors (D17-1)	USD 0.10/piece
Total for D17-0 configuration	USD 191.50
Total for D17-0+w configuration	USD 211.50
Total for D17-0+j configuration	USD 222
Total for D17-1 configuration	USD 281+Bumpers

12.2. Chassis

We selected the Magician Chassis as the basic chassis for the robot (Figure 2).

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

The chassis pack includes the motors and wheels as well as the structural part.

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

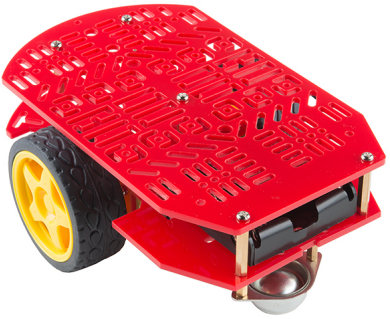


Figure 2. The Magician Chassis

12.3. Raspberry Pi 3 - Model B (RPI-3)

The RPI-3 is the central computer of the Duckiebot. Duckiebot version `D17` uses Model B (Figure 3) (A1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM), a small but powerful computer.

Figure 3. The Raspberry Pi 3 Model B

The price for this in the US is about USD 35.

1) Power Supply

We want a hard-wired power source (5VDC, 2.4A, Micro USB) to supply the RPI-3 (Figure 4).

Figure 4. The Power Supply

The price for this in the US is about USD 5-10.

2) Heat Sinks

The RPI-3 will heat up significantly during use. It is warmly recommended to add heat sinks, as in Figure 5. Since we will be stacking HATs on top of the RPI-3 with 15 mm standoffs, the maximum height of the heat sinks should be well below 15 mm. The chip dimensions are 15x15 mm and 10x10 mm.

Figure 5. The Heat Sinks

3) Class 10 MicroSD Card

The MicroSD card (Figure 6) is the hard disk of the RPI-3. 16 Gigabytes of capacity are sufficient for the system image.

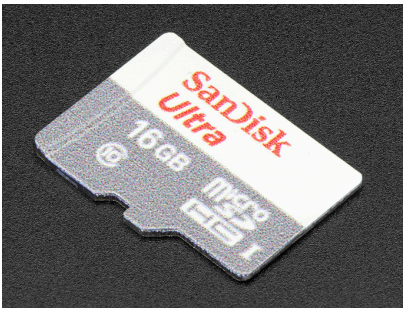


Figure 6. The MicroSD card

4) Mirco SD card reader

A MicroSD card reader (Figure 7) is useful to copy the system image to a Duckiebot from a computer to the RPI-3 microSD card, when the computer does not have a native SD card slot.

Figure 7. The Mirco SD card reader

5) Tiny 32GB USB Flash Drive

This “external” hard drive (Figure 8) is very convenient to store logs during experiments and later port them to a workstation for analysis. It provides storage capacity and faster data transfer than the MicroSD card.

Figure 8. The Tiny 32GB USB Flash Drive

12.4. Camera

The Camera is the main sensor of the Duckiebot. Version D17 equips a 5 Mega Pixels 1080p camera with wide field of view (**160°**) fisheye lens (Figure 9).

Figure 9. The Camera with Fisheye Lens

1) Camera Mount

The camera mount (Figure 10) serves to keep the camera looking forward at the right angle to the road (looking slightly down). The front cover is not essential.

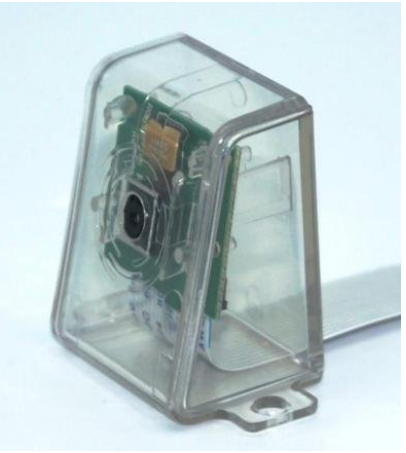


Figure 10. The Camera Mount

2) 300mm Camera Cable

A longer (300 mm) camera cable Figure 11 make assembling the Duckiebot easier, allowing for more freedom in the relative positioning of camera and computational stack.

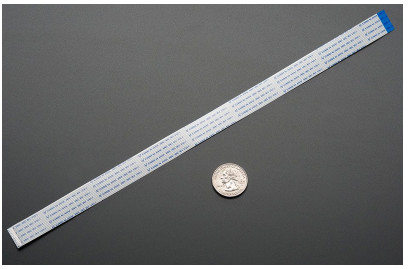


Figure 11. A 300 mm camera cable for the RPI-3

12.5. Wifi Augmenter

The Edimax AC1200 EW-7822ULC wifi adpater (Figure 12) boosts the connectivity of the Duckiebot, especially useful in busy Duckietowns (e.g., classroom).

Figure 12. The Edimax AC1200 EW-7822ULC wifi adapter

12.6. Joypad

The joypad is used to manually remote control the Duckiebot. Any 2.4 GHz wireless controller (with a *tiny* USB dongle) will do.

The model link in the table (Figure 13) does not include batteries (2 AA 1.5V)!

Figure 13. A Wireless Joypad

12.7. DC Stepper Motor HAT

We use the DC Stepper motor HAT (Figure 13) to control the DC motors that drive the wheels. This item will require soldering to be functional.

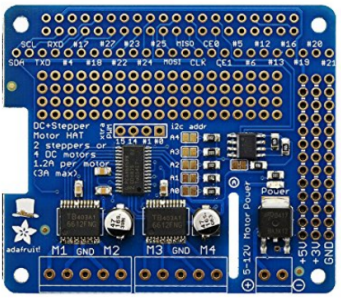


Figure 14. The Stepper Motor HAT

1) Stacking Headers

We use long 20x2 stacking headers (Figure 15) to connect the RPI-3 with the other HATs, creating a stack. This item will require soldering to be functional.

In configuration `D17-1`, we need 2 stacking headers.

In all configurations, we use only 1 stacking header.

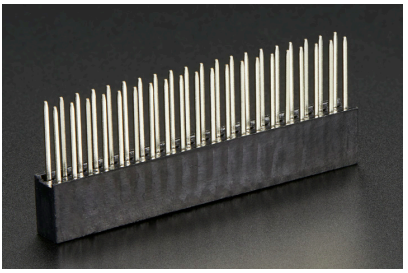


Figure 15. The Stacking Headers

12.8. Battery

The battery (Figure 16) provides power to the Duckiebot.

We choose this battery because it has a good combination of size (to fit in the lower deck of the Magician Chassis), high output amperage (2.4A and 2.1A at 5V DC) over two USB outputs, a good capacity (10400 mAh) at an affordable price (USD 20).

Figure 16. The Battery

12.9. Standoffs, Nuts and Screws

We use non electrically conductive standoffs (M2.5 12mm F 6mm M), nuts (M2.5), and screws (M2.5x10mm) to hold the RPI-3 to the chassis and the HATs stacked on top of the RPI-3.

In configuration `D17-0` and `D17-0+w` or `D17-0+j`, the Duckiebot requires 8 standoffs, 4 nuts and 4 screws.

In configuration `D17-1`, the Duckiebot requires 16 standoffs, 4 nuts and 4 screws.

Figure 17. Standoffs, Nuts and Screws

12.10. Zip Tie

Two 300x5mm zip ties are going to be useful to keep the battery at the lower deck from moving around.

Figure 18. The zip ties

12.11. LEDs

In configuration D17-1, the Duckiebot is equipped with 5 RGB LEDs. LEDs can be used to signal to other Duckiebots, or just make cool patterns!

The pack of LEDs linked in the table above holds 10 LEDs, enough for two Duckiebots.

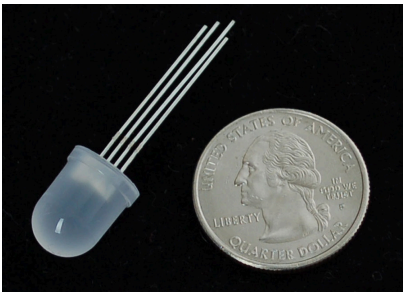


Figure 19. The RGB LEDs

1) LED HAT

In configuration D17-1, the LED HAT (Figure 20) provides an interface for our RGB LEDs and the computational stack. This board is a daughterboard for the Adafruit 16-Channel PWM/Servo HAT, and enables connection with additional gadgets such as ADS1015 12 Bit 4 Channel ADC, Monochrome 128x32 I2C OLED graphic display, and Adafruit 9-DOF IMU Breakout - L3GD20H+LSM303. This item will require soldering to be functional.

This board is custom degined and can only be ordered in minimum runs of 3 pieces. The price scales down quickly with quantity, and lead times may be significant, so it is better to buy these boards in bulk.

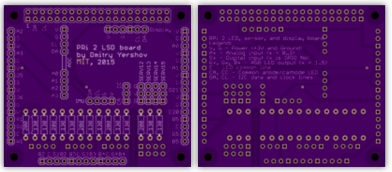


Figure 20. The LED HAT

2) PWM/Servo HAT

In configuration D17-1, the PWM/Servo HAT HAT (Figure 21) mates to the LED HAT and provides the signals to control the LEDs, without taking computational resources away from the Rasperry Pi itself. This item will require soldering to be functional.

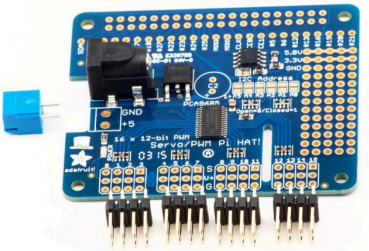


Figure 21. The PWM-Servo HAT

3) Male-Male Jumper Wires

In configuration D17-1, the Duckiebot requires one male-male jumper wire (Figure 22) to power the DC Stepper Motor HAT from the PWM/Servo HAT.

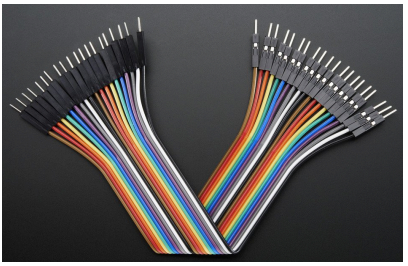


Figure 22. Premier Male-Male Jumper Wires

4) Female-Female Jumper Wires

In configuration D17-1, 20 Female-Female Jumper Wires (Figure 23) are necessary to connect 5 LEDs to the LED HAT.

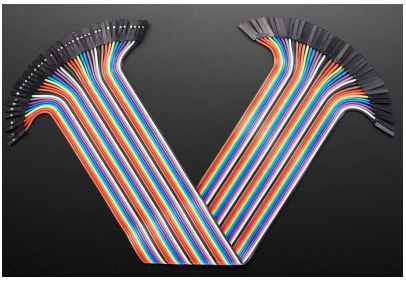


Figure 23. Premier Female-Female Jumper Wires

12.12. Bumpers

These bumpers are designed to keep the LEDs in place and are therefore used only in configuration D17-1. They are custom designed parts, so they must be produced and cannot be bought. We used laser cutting facilities. Our design files are available [here].

Figure 24. The Bumpers

12.13. Passive Electric Components

5 4 pin female header

2 16 pin male header

1 12 pin male header

1 3 pin male header

1 2 pin female shunt jumper

5 200 Ohm resistors

10 130 Ohm resistors

These items will require soldering to be functional.

Figure 25. The Headers

Figure 26. The Resistors

Chapter 13
Soldering boards for C0

Assigned to: Shiying

Resources necessaries:

- ...
- Time: ??? minutes

Results:

- ...

Chapter 14
Assembling the Duckiebot `C0`

Assigned to: Shiying

Resources necessaries:

- Duckiebot `D17-C0` parts.

Requires: The acquisition process is explained in Chapter 12.

- Time: about ??? minutes.

Results:

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

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

Chapter 15
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.

15.1. 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:

```
$ 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`:

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

15.2. Burn the image to an SD card

Next, burn the image on to the SD card.

➡ This procedure is explained in Section 79.1.

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:

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

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

15.3. Raspberry Pi Config

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

```
$ 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

15.4. Install packages

Install these packages.
Etckeeper:

```
$ sudo apt install etckeeper
```

Editors / shells:

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

Git:

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

Other:

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

Development:

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

Python:

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

I2C:

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

15.5. 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.

15.6. Install ROS

Install ROS.

➡ The procedure is given in Section 94.1.

15.7. 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
}
```

15.8. 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.

15.9. SSH server config

This enables the SSH server:

```
$ sudo systemctl enable ssh
```

15.10. Create swap Space

Do the following:

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

```
$ 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:

```
$ sudo mkswap /swap0
```

Add the swap file to the system configuration:

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

Add this line to the bottom:

```
/swap0 swap swap
```

Activate the swap space:

```
$ sudo swapon -a
```

15.11. 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
```

15.12. Clean up

You can use the command `dpigs` to find out which packages take lots of space.

```
sudo apt install wajig  debian-goodies
```

Either:

```
$ wajig large
$ dpigs -H -n 20
```

Stuff to remove:

```
$ sudo apt remove thunderbird
$ sudo apt remove libreoffice-\*
$ sudo apt remove openjdk-8-jre-headless
$ sudo apt remove fonts-noto-cjk
$ sudo apt remove brasero
```

At the end, remove extra dependencies:

```
$ sudo apt autoremove
```

And remove the `apt` cache using:

```
$ sudo apt clean
```

The total size should be around 6.6GB.

15.13. Ubuntu user configuration

1) Groups

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

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

: forgot to add to aug20 image:

```
$ sudo adduser ubuntu video
```

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

```
$ sudo udevadm trigger
```

2) Basic SSH config

Do the basic SSH config.

➡ The procedure is documented in Section 81.3.

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`:

```
$ 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
```

5) Creating the image

You may now want to create an image that you can share with your friends. They will think you are cool because they won't have to duplicate all of the work that you just did. Luckily this is easy. Just power down the duckiebot with:

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

and put the SD card back in your laptop.

➡ The procedure of how to burn an image is explained in Section 79.1

except you will invert the `if` and `of` destinations

You may want to subsequently shrink the image, for example if your friends have smaller SD cards than you.

➡ The procedure of how to shrink an image is explained in Section 79.2

15.14. Additions since last image

Note here the additions since the last image was created.

- Install Git LFS

Create a file

```
/etc/duckietown-image.yaml
```

Containing these lines

```
base: Ubuntu 16.04.2
date: DATE
comments: |
    any comments you have
```

So that we know which image is currently in used

CHAPTER 16

Installing Ubuntu on laptops

Assigned to: Andrea

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

Requirements:

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

Results:

- A laptop ready to be used for Duckietown.

16.1. 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.

16.2. Install useful software

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

💻
```
$ sudo apt install etckeeper
```

Install `ssh` to login remotely and the server:

💻
```
$ sudo apt install ssh
```

Use `byobu`:

💻
```
$ sudo apt install byobu
```

Use `vim`:

💻
```
$ sudo apt install vim
```

Use `htop` to monitor CPU usage:

💻
```
$ sudo apt install htop
```

Additional utilities for `git`:

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

Other utilities:

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

16.3. Install ROS

Install ROS on your laptop.

➡ The procedure is given in Section 94.1.

16.4. 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.

```
$ 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 6.4.

16.5. Passwordless `sudo`

Set up passwordless `sudo`.

➡ This procedure is described in Section 15.11.

16.6. SSH and Git setup

1) Basic SSH config

Do the basic SSH config.

➡ The procedure is documented in Section 81.3.

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 81.5.

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

Add the public key to your Github account.

➡ The procedure is documented in Section 90.3.

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

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

4) Local Git setup

Set up Git locally.

➡ The procedure is described in Section 88.3.

Chapter 17
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`.

Requires: This is the result of Chapter 14.

Result:

- A Duckiebot that is ready to use.

What does it mean "ready to use"?.

17.1. 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 79.1.

17.2. Turn on the Duckiebot

Put the SD Card in the Duckiebot.

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

17.3. 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.

17.4. 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`:

```
$ 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
...
```

17.5. SSH to the Duckiebot

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

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

The password is `ubuntu`.

By default, the robot boots into Byobu.

Please see Chapter 87 for an introduction to Byobu.

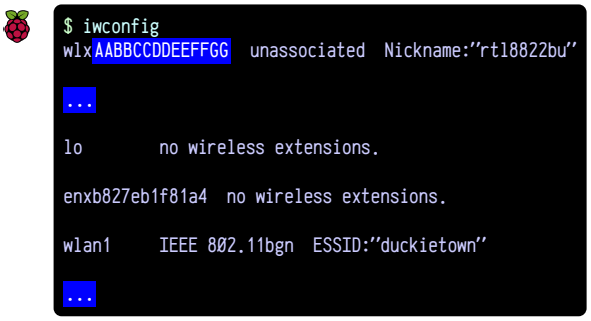
Not sure it's a good idea to boot into Byobu.

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

`D17-0+w` 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:



Make note of the name `wlxAABBCCDDEEFFGG`.

Look up the MAC address using the command:

Then, edit the connection file

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

Make the following changes:

- Where it says `interface-name=...`, put "`wlxAABBCCDDEEFFGG`".
- 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.

17.7. 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 SSID 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.

17.8. Update the system

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

Use these commands

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

17.9. 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.

17.10. 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:

```
$ 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
```

17.11. Expand your filesystem

If your SD card is larger than the image, you'll want to expand the filesystem on your robot so that you can use all of the space available. Achieve this with:

```
$ sudo raspi-config --expand-rootfs
```

and then reboot

```
$ sudo shutdown -r now
```

once rebooted you can test whether this was successful by doing

```
$ df -lh
```

the output should give you something like:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  7.8G   21G  28% /
devtmpfs        427M     0  427M   0% /dev
tmpfs           432M  316K  431M   1% /dev/shm
tmpfs           432M   12M  420M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           432M     0  432M   0% /sys/fs/cgroup
/dev/mmcblk0p1   63M   21M   43M  34% /boot
tmpfs            87M   24K   87M   1% /run/user/1000
/dev/sda1        29G  5.3G   24G  19% /media/ubuntu/44A7-9E91
```

You should see that the Size of your `/dev/sda1` partition is "close" to the side of your SD card.

17.12. 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:

```
$ sudo useradd -m username
```

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

```
$ sudo adduser username sudo
```

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

```
$ sudo adduser username input
$ sudo adduser username video
$ sudo adduser username i2c
```

Set the shell `bash`:

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

To set a password, use:

```
$ sudo passwd username
```

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

```
$ 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 81.3.

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 81.5.

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

Add the public key to your Github account.

➡ The procedure is documented in Section 90.3.

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

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

4) Local Git configuration

➡ This procedure is in Section 88.3.

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 81.6. 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:

```
$ 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...

17.13. 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`.

17.14. Hardware check: camera

Check that the camera is connected using this command:

```
$ 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`:

```
$ 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 83.1.1.

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 18
Software setup and RC remote control

Assigned to: Andrea

Prerequisites:

- You have configured the laptop.

Requires: The procedure is documented in Chapter 16.

- You have configured the Duckiebot.

Requires: The procedure is documented in Chapter 17.

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

Requires: The procedure is documented in Chapter 90.

Results:

- You can run the joystick demo.

18.1. Clone the Duckietown repository

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

```
$ 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 90.

Symptom: Other weird errors.

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

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

18.2. Set up ROS environment on the Duckiebot

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

```
$ cd ~/duckietown
```

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

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

Then, build the workspace using:

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

☀ *For more information about* `catkin_make`, *see Subsection 94.5.3.*

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
```

18.3. 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.)

18.4. Test that the joystick is detected

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

To make sure that the joystick is detected, run:

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

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

Check before you continue

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

```
$ 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 17.12.

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

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

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

18.5. Run the joystick demo

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

```
$ cd ~/duckietown
$ 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:

```
$ 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 yaw)
right joystick right	turn right (negative yaw)

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 18.4.

18.6. 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:

```
$ 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 19
Reading from the camera

Prerequisites:

- You have configured the Duckiebot.

Requires: The procedure is documented in Chapter 17.

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

Results:

- You know that the camera works under ROS.

19.1. Check the camera hardware

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

➡ The procedure is documented in Section 17.14.

19.2. Create two windows

On the laptop, create two Byobu windows.

➡ A quick reference about Byobu commands is in Chapter 87.

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.

19.3. First window: launch the camera nodes

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

Activate ROS:

```
$ source environment.sh
```

Run the launch file called `camera.launch`:

```
$ 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 94.3.*

19.4. Second window: view published topics

Switch to the second window.

Activate the ROS environment:

```
$ source environment.sh
```

1) List topics

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

```
$ rostopic list
```

✴ *For more information about* `rostopic`*, see Section 94.5.*

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:

```
$ 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`:

```
$ 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, in Chapter 21, you will learn how to visualize the image stream on the laptop using `rviz`.

use Ctrl-C to stop `rostopic`.

Chapter 20
RC control launched remotely

Assigned to: Andrea

Prerequisites:

- You can run the joystick demo from the Raspberry Pi.

Requires: The procedure is documented in Chapter 18.

Results:

- You can run the joystick demo from your laptop.

20.1. 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".

20.2. 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 18.1.

Then, you should build the repository.

➡ This procedure is documented in Section 18.2.

20.3. 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 18.3.

20.4. Start the demo

Now you are ready to launch the joystick demo remotely.

Check before you continue

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

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

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

Run this *on the laptop*:

```
$ source environment.sh
$ 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 94.3.*

20.5. 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:

```
$ export ROS_MASTER_URI=http://robot name.local:11311/
$ 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 94.2.*

20.6. 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 81.3.

Chapter 21
RC+camera remotely

Assigned to: Andrea

Prerequisites:

- You can run the joystick demo remotely.

Requires: The procedure is documented in Chapter 20.

- You can read the camera data from ROS.

Requires: The procedure is documented in Chapter 19.

- You know how to get around in Byobu.

➡ You can find the Byobu tutorial in Chapter 87.

Results:

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

21.1. Assumptions

We are assuming that the joystick demo in Chapter 20 worked.

We are assuming that the procedure in Chapter 19 succeeded.

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

21.2. Terminal setup

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

➡ A quick reference about Byobu commands is in Chapter 87.

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.

21.3. First window: launch the joystick demo

In the first window, launch the joystick remotely using the same procedure in Section 20.4.

💻
```
$ source environment.sh
$ roslaunch duckietown joystick.launch veh:=robot name
```

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

21.4. 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`:

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

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

21.5. 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:

```
$ source environment.sh
 $ export ROS_MASTER_URI=http://robot name.local:11311/
$ 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
```

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

Launch `rviz` by using these commands:

```
$ source environment.sh
 $ source set_ros_master.sh robot name
$ rviz
```

☀ *For more information about `rviz`, see Section 94.4.*

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.

21.7. 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 22
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.

22.1. `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.

22.2. 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 23
Wheel calibration

Assigned to: Andrea

Chapter 24
Camera calibration

Chapter 25
Taking a log

Assigned to: Andrea

Part 4
Operation manual - Duckietowns

Chapter 26
Duckietown parts

Assigned to: Jacopo

Chapter 27
Duckietown Assembly

Assigned to: Shiying

Chapter 28
The Duckietown specification

Assigned to: Liam?

28.1. Topology

1) Topology constraints

28.2. Signs placement

Chapter 29
Traffic lights

PART 5

Operation manual - Duckiebot with LEDs

Chapter 30
D17-1 (LEDs) parts

Assigned to: Jacopo

Chapter 31
D17-1 (LEDs) assembly

Assigned to: Shiying

Chapter 32
D17-1 (LEDs) setup

Assigned to: Andrea

Part 6
Theory chapters

These are the theory chapters.

Chapter 33
Chapter template

Assigned to: Jacopo

Chapter 34
Symbols and conventions

Assigned to: Andrea

34.1. Conventions

If $\boldsymbol{x}$ is a function of time, use $\boldsymbol{x_t}$ rather than $\boldsymbol{x(t)}$.

✖ Consider the function $\boldsymbol{x(t)}$.

✔ Consider the function $\boldsymbol{x_t}$.

34.2. Table of symbols

Here are some useful symbols.

Table 1. Spaces

command	result	
`\SOthree`	$\mathbf{SO(3)}$	Rotation matrices
`\SEthree`	$\mathbf{SE(3)}$	Euclidean group
`\SEtwo`	$\mathbf{SE(2)}$	Euclidean group
`\setwo`	$\mathbf{se(2)}$	Euclidean group algebra

States and poses:

Table 1. Poses and states

command	result	
`\pose`	$\boldsymbol{q_t} \in \mathbf{SE(2)}$	Pose of the robot in the plane
`\state_t \in \statesp`	$\boldsymbol{x_t} \in \boldsymbol{\mathcal{X}}$	System state (includes the pose, and everything else)

Chapter 35
Linear algebra

Assigned to: Jacopo

Chapter 36
Probability basics

Assigned to: Liam?

Chapter 37
Dynamics

Assigned to: Jacopo

Chapter 38
Autonomy overview

Assigned to: Liam

38.1. Perception, planning, control

Chapter 39
Autonomy architectures

Assigned to: Andrea

Chapter 40
Representations

Assigned to: Matt

Chapter 41
Software architectures and middlewares

Assigned to: Andrea

Chapter 42
Modern signal processing

Assigned to: Andrea

Chapter 43
Basic Kinematics

Assigned to: Jacopo

Chapter 44
Basic Dynamics

Assigned to: Jacopo

Chapter 45
Odometry Calibration

Assigned to: Jacopo

Chapter 46
Computer vision basics

Assigned to: Matt

Chapter 47
Illumination invariance

Assigned to: Matt

Chapter 48
Line Detection

Assigned to: Matt

Chapter 49
Feature extraction

Assigned to: Matt

Chapter 50
Place recognition

Assigned to: Matt

Chapter 51
Filtering 1

Assigned to: Liam

Chapter 52
Filtering 2

Assigned to: Liam

Chapter 53
Mission planning

Assigned to: ETH

Chapter 54
Planning in discrete domains

Assigned to: ETH

Chapter 55
Motion planning

Assigned to: ETH

Chapter 56
RRT

Assigned to: ETH

Chapter 57
Feedback control

Assigned to: Jacopo

Chapter 58
PID Control

Assigned to: Jacopo

Chapter 59
MPC Control

Assigned to: Jacopo

Chapter 60
Object detection

Assigned to: Nick and David

Chapter 61
Object classification

Assigned to: Nick and David

Chapter 62
Object tracking

Assigned to: Nick and David

Chapter 63
Reacting to obstacles

Assigned to: Jacopo

Chapter 64
Semantic segmentation

Assigned to: Nick and David

Chapter 65
Text recognition

Assigned to: Nick

Chapter 66
SLAM - Problem formulation

Assigned to: Liam

Chapter 67
SLAM - Broad categories

Assigned to: Liam

Chapter 68
VINS

Assigned to: Liam

Chapter 69
Advanced place recognition

Assigned to: Liam

Chapter 70
Fleet level planning (placeholder)

Assigned to: ETH

Chapter 71
Fleet level planning (placeholder)

Assigned to: ETH

Chapter 72
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 7
Exercises

These are the exercises.

Chapter 73
ROS tutorial

Chapter 74
Line detection

Chapter 75
Data processing

Chapter 76
Git and conventions

Part 8
Software manuals

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.

76.1. Ubuntu packaging with APT

1) `apt install`

2) `apt update`

3) `apt dist-upgrade`

Chapter 77
GNU/Linux general notions

Assigned to: Andrea

77.1. Background reading

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

77.2. `pgrep`

Chapter 78
Linux resources usage

78.1. Measuring CPU usage using `htop`

You can use `htop` to monitor CPU usage.

```
$ sudo apt install htop
```

78.2. Measuring I/O usage using `iotop`

Install using:

```
$ sudo apt install iotop
```

78.3. How fast is the SD card?

➡ Subsection 79.0.1.

CHAPTER 79

SD Cards tools

1) 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.

79.1. 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/mmcblk0pn`, then unmount each of them. For example:

```
$ sudo umount /dev/mmcblk0p1
$ 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`:

```
$ 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`.

79.2. How to shrink an image

Requires:

- An image file to burn.
- An Ubuntu computer.

Results:

- A shrunk image.

Note: Majority of content taken from here

We are going to use the tool `gparted` so make sure it's installed

```
$ sudo apt-get install gparted
```

Let the image file be `image file`. Run the command:

```
$ sudo fdisk -l image file
```

should give you something like:

```
Device                       Boot  Start      End  Sectors  Size Id Type
duckiebot-RPI3-LP-aug15.img1        2048   131071   129024   63M  c W95 FAT32 (LBA)
duckiebot-RPI3-LP-aug15.img2      131072 21219327 21088256 10.1G 83 Linux
```

Take note of the start of the Linux partition (in our case 131072), let's call it `start` Now we are going to mount the Linux partition from the image:

```
$ sudo losetup /dev/loop0 imagename.img -o $((start*512))
```

and then run `gparted`:

```
$ sudo gparted /dev/loop0
```

In `gparted` click on the partition and click “Resize” under the “Partition” menu. Resize drag the arrow or enter a size that is equal to the minimum size plus 20MB

> Note: This didn’t work well for me - I had to add much more than 20MB for it to work.

Click the “Apply” check mark. *Before* closing the final screen click through the arrows in the dialogue box to find a line such a *“resize2fs -p /dev/loop0 1410048K”*. Take note of the new size of your partition. Let’s call it `new size`.

Now remove the loopback on the 2nd partition and setup a loopback on the whole image and run `fdisk`:

```
$ sudo losetup -d /dev/loop0
 $ sudo losetup /dev/loop0 image file
$ sudo fdisk /dev/loop0

Command (m for help): *d*
Partition number (1,2, default 2): *2*
Command (m for help): *n*
Partition type
p   primary (1 primary, 0 extended, 3 free)
e   extended (container for logical partitions)
Select (default p): *p*
Partition number (2-4, default 2): *2*
First sector (131072-62521343, default 131072): *start*
Last sector, +sectors or +size{K,M,G,T,P} (131072-62521343, default 62521343): *+new size*
```

(Note: on the last line to include the `+` and the `K` as part of the size.)

```
Created a new partition 2 of type 'Linux' and of size 10.1 GiB.

Command (m for help): *w*
The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Invalid argument

The kernel still uses the old table. The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8).
```

Disregard the final error.

You partition has now been resized and the partition table has been updated. Now we will remove the loopback and then truncate the end of the image file:

```
$ fdisk -l /dev/loop0
```

```
Device       Boot  Start      End  Sectors  Size Id Type
/dev/loop0p1        2048   131071   129024   63M  c W95 FAT32 (LBA)
/dev/loop0p2      131072 21219327 21088256 10.1G 83 Linux
```

Note down the end of the second partition (in thise case 21219327). Call this `end`.

```
$ sudo losetup -d /dev/loop0
 $ sudo truncate -s $(((end+1)*512)) image file
```

You now have a shrunken image file. A further idea is to compress it:

```
$ xz image file
```

CHAPTER 80
Networking tools

> **Assigned to:** Andrea
>
> Preliminary reading:
>
> - Basics of networking, including
>   - what are IP addresses
>   - what are subnets
>   - how DNS works
>   - how `.local` names work
>   - ...
>
> ➡ (ref to find).

Make sure that you know:

80.1. Visualizing information about the network

1) `ping`: are you there?

2) `ifconfig`

```
$ ifconfig
```

Chapter 81
Accessing computers using SSH

Assigned to: Andrea

81.1. Background reading

- Encryption
- Public key authentication

81.2. Installation of SSH

This installs the client:

```
$ sudo apt install ssh
```

This installs the server:

This enables the server:

81.3. 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.

81.4. 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`.

81.5. 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
...
```

81.6. 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 81.5.

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.

81.7. 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/*
```

Chapter 82
Wireless networking in Linux

82.1. iwconfig

82.2. iwlist

1) Getting a list of WiFi networks

What wireless networks do I have around?

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

2) Do I have 5 GHz?

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 83
Moving files between computers

83.1. SCP

1) Download a file with SCP

83.2. RSync

Chapter 84
VIM

Assigned to: Andrea

To do quick changes to files, especially when logged remotely, we suggest you use the VI editor, or more precisely, VIM ("VI iMproved").

84.1. External documentation

➡ A VIM tutorial.

84.2. Installation

Install like this:

```
$ sudo apt install vim
```

84.3. Suggested configuration

Suggested `~/.vimrc`:

```
syntax on
set number
filetype plugin indent on
highlight Comment ctermfg=Gray
autocmd FileType python set complete isk+=.,(
```

84.4. Visual mode

84.5. Indenting using VIM

Use the `>` command to indent.

To indent 5 lines, use `5` `>` `>`.

To mark a block of lines and indent it, use `V`.

For example, use `V` `j` `j` `>` to indent 3 lines.

Use `<` to dedent.

Chapter 85
Atom

Chapter 86
Eclipse

86.1. Installing LiClipse

Chapter 87
Byobu

Assigned to: Andrea

You need to learn to use Byobu. It will save you much time later.

☀ *See the screencast on the website http://byobu.co/.*

87.1. Alternatives

GNU Screen is fine as well.

87.2. Advantages of using Byobu

87.3. Installation

On Ubuntu, install using:

```
$ sudo apt install byobu
```

87.4. Quick command reference

You can change the escape sequence from Ctrl-A to something else by using the configuration tool that appears when you type F9.

Commands to use windows:

Table 1. Windows

	Using function keys	Using escape sequences
Create new window	F2	Ctrl-A then C
Previous window	F3	
Next window	F4	
Switch to window		Ctrl-A then a number
Close window	F6	
Rename window		Ctrl-A then ,

Commands to use panes (windows split in two or more):

Table 1. Commands for panes

	Using function keys	Using escape sequences
Split horizontally	Shift-F2	Ctrl-A then |
Split vertically	Ctrl-F2	Ctrl-A then %
Switch focus among panes	Ctrl-↑↓←→	Ctrl-A then one of ↑↓←→
Break pane		Ctrl-A then !

Other commands:

TABLE 1. OTHER

	Using function keys	Using escape sequences
Help		Ctrl-A then ?
Detach		Ctrl-A then D

87.5. Commands on OS X

Scroll up and down using fn option ↑ and fn option ↓.

Highlight using alt

Chapter 88
Source code control with Git

Assigned to: Andrea

88.1. Background reading

- Git
- GitFlow

88.2. 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.

88.3. 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
```

88.4. Git tips

1) Shallow clone

You can clone without history with the command:

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

88.5. 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 entries with `https`::

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

Expectation:

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

Solution:

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

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 89
Git LFS

This describes Git LFS.

89.1. Installation

See instructions at:

https://git-lfs.github.com/

89.2. Ubuntu 16 installation

Following:

https://github.com/git-lfs/git-lfs/wiki/Installation

Run the following:

```
$ sudo add-apt-repository ppa:git-core/ppa
$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
$ sudo apt-get install git-lfs
```

1) Troubleshooting

Symptom: The binaries are not installed.

If you have installed LFS after pulling the repository and you see only the pointer files, do:

```
$ git lfs pull --all
```

Chapter 90
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.

90.1. 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 27).

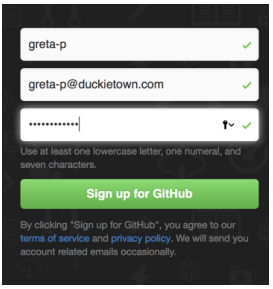


Figure 27

Go to your inbox and verify the email.

90.2. 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.

90.3. 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 81.5.

Result:

- You can access Github using the key provided.

Go to settings (Figure 28).

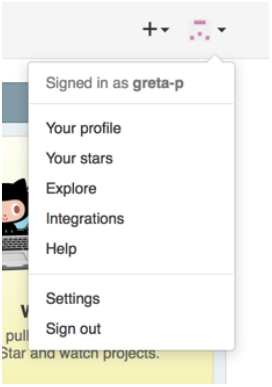


Figure 28

Add the public key that you created:

Figure 29

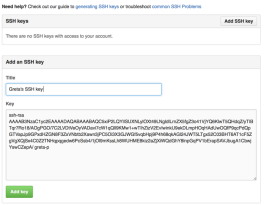


Figure 30

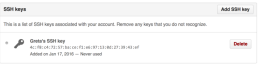


Figure 31

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.

Part 9
Duckietown development guide

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

Chapter 91
Configuration

This chapter explains what are the assumptions about the configuration.

While the "Setup" parts are "imperative" (do this, do that); this is the "declarative" part, which explains what are the properties of a correct configuration (but it does not explain how to get there).

☀ *The tool `what-the-duck` checks these conditions (Chapter 92). If you make a change from the existing conditions, make sure that it gets implemented in `what-the-duck` by filing an issue.*

91.1. Environment variables

You need to have set up the variables in Table 12.

Table 1. Environment variables used by the software

variable	reasonable value	contains
DUCKIETOWN_ROOT	~/duckietown	Software repository
DUCKIEFLEET_ROOT	~/duckiefleet	A repository that contains `scuderia.yaml` and other team-specific configuration.

1) Duckietown root directory `DUCKIETOWN_ROOT`

2) Duckiefleet directory `DUCKIEFLEET_ROOT`

For Fall 2017, this is the the repository `duckiefleet-fall2017`.

For self-guided learners, this is an arbitrary repository to create.

91.2. The scuderia file

In the `${DUCKIEFLEET_ROOT}` directory, there needs to exist a file called:

```
${DUCKIEFLEET_ROOT}/scuderia.yaml
```

The file must contain YAML entries of the type:

```
robot-name:
   username: username
   owner_duckietown_id: owner duckietown ID
```

A minimal example is in Listing 4

```
emma:
  username: andrea
  owner_duckietown_id: censi
```

Listing 1. Minimal scuderia file

Explanations of the fields:

- `robot name`: the name of the robot, also equal to the host name.
- `username`: the name of the Linux user on the robot, from which to run programs.
- `owner_duckietown_id`: the owner's globally-unique Duckietown ID.

91.3. The `machines` file

The `machines` file is created using:

```
$ rosrun duckietown create-machines-file
```

91.4. People database

Assigned to: Andrea

1) The globally-unique Duckietown ID

This is a globally-unique ID for people in the Duckietown project.

It is equal to the Slack username.

Chapter 92
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
```

92.1. 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.

92.2. 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
✓  Environment variable DUCKIETOWN_FLEET
✓  ${DUCKIETOWN_FLEET} exists
✓  ${DUCKIETOWN_FLEET}/scuderia.yaml exists
✓  ${DUCKIETOWN_FLEET}/scuderia.yaml is valid
✓  machines file is valid
✓  Wifi network configured
✓  Python: No CamelCase
✓  Python: No tab chars
✓  Python:  No half merges
```

92.3. 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!)
- Editor is set to `vim`.
- They put the right MAC address in the network configuration
- Ubuntu user is in group video, input, i2c (even if run from other user.)
- There is at least X.YGB of free disk space.
- If the SD is larger than 8GB, the disk has been resized.

Chapter 93
Python

93.1. Background reading

- Python
- Python tutorial

93.2. Python virtual environments

Install using:

```
$ sudo apt install virtualenv
```

93.3. Useful libraries

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

Chapter 94
Introduction to ROS

Assigned to: Liam

94.1. 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-proc,c
```

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
```

94.2. `rqt_console`

94.3. `roslaunch`

94.4. `rviz`

94.5. 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 81.3.

Chapter 95
How to create a ROS package

95.1. 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 96
Integrate package in the architecture

Chapter 97
Creating unit tests

CHAPTER 98

Duckietown Software architecture

Part 10
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 (TTIC), 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 99
General remarks

Assigned to: Andrea

99.1. 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.

99.2. 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.

99.3. Accounts for students

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

1) Slack

You need a Slack account, for team discussion and organization.

2) Github

- A Github account;
- Membership in the Duckietown organization.

99.4. Accounts for all instructors and TAs

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

1) Twist

Twist is used for class organization (such as TAs, logistics);
TODO:

2) Google docs

Google Docs is used to maintain TODOs and other coordination materials.
In particular:

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

Chapter 100
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

Chapter 101
Additional information for UdeM students

Assigned to: Liam

Chapter 102
Additional information for TTIC students

Assigned to: Matt

Chapter 103
Additional information for NCTU students

Assigned to: Nick

CHAPTER 104

Milestone: ROS node working

Chapter 105
Homework: Take and process a log

Chapter 106
Milestone: Calibrated robot

Chapter 107
Homework: Camera geometry

Chapter 108
Milestone: Illumination invariance

Chapter 109
Homework: Place recognition

CHAPTER 110
Milestone: Lane following

Chapter 111
Homework: localization

Chapter 112
Milestone: Navigation

Chapter 113
Homework: group forming

Chapter 114
Milestone: Ducks in a row

Chapter 115
Homework: Comparison of PID

Chapter 116
Homework: RRT

Chapter 117
Caffe tutorial

Chapter 118
Milestone: Object Detection

Chapter 119
Homework: Object Detection

Chapter 120
Milestone: Semantic perception

Chapter 121
Homework: Semantic perception

Chapter 122
Milestone: Reacting to obstacles

Chapter 123
Homework: Reacting to obstacles

CHAPTER 124

Milestone: SLAM demo

Chapter 125
Homework: SLAM

Chapter 126
Milestone: fleet demo

Chapter 127
Homework: fleet

Chapter 128
Project proposals

Chapter 129
Template of a project

129.1. Checklist for students

- Have a Github account. See Chapter 90. 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).

129.2. Checklist for TAs

- Be signed up on

Part 11
Drafts or pieces to remove

Page left blank