Duckiebot operation manual

- **Part A - Preliminaries: Software**: The software basics to know before you start.
- **Part B - Preliminaries: Hardware**: The hardware basics to know before you start.
- **Part C - Duckiebot assembly and setup**: Setting up your Duckiebot.
- **Part D - Demos and Exercises**: Running the demos.
PART A
Preliminaries: Software

Before starting this adventure, cover the software basics.
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.

### 1.1. SSH aliases

Instead of using

```bash
$ ssh duckie@ROBOT.local
```

You can set up SSH so that you can use:

```bash
$ ssh ROBOT
```

During your init_sd_card process described later in the book, the command will automatically setup `~/.ssh/config`. If you are having trouble using it, you can follow the instructions below.

To manually create an SSH alias, create a host section in `~/.ssh/config` on your laptop with the following contents:

```ini
Host ROBOT
    User duckie
    Hostname ROBOT.local
```

Note that this does not let you do
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.
Version Control with Git

2.1. Background reading
- Github tutorial
- Github workflow

2.2. 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, which comes in handy when you have files that you don’t actually want to push to the remote branch (such as temporary files).
2.3. Setting up global configurations for Git

Use these commands to tell Git who you are:

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

2.4. Git tips

1) Fork a repository

To fork (creating a copy of a repository, that does not belong to you), you simply have to go to the repository’s webpage dashboard and click fork on the upper right corner.

2) Clone a repository

To clone a repository, copy either the HTTPS or SSH link from the repository’s webpage. The following command will download the git repository in a new directory on the local computer (starting from the current working directory).

```
$ git clone git@github.com:USERNAME/REPOSITORY
```

If you have SSH setup properly, you can directly download it. If you are using the HTTPS then github will ask for your credentials.

3) Move between branches

You can move to a different branch using the command,

```
$ git checkout destination-branch
```

4) Create a new branch

After you successfully cloned a repository, you may want to work on your own branch. Move to the branch you want to start from and run the following command,

```
$ git checkout -b branch-name
```

To see which branch you are working on you can either use both of these commands...
The latter provides more information on which files you might have changed, which are staged for a new commit or that you are up-to-date (everything is ok).

5) Commit and Push changes

After you edited some files, you want to push your changes from the local to the remote location. Check the changes that need to be committed/pushed with the command,

$ git status

Use the following command to mark a `file` as ready to be committed,

$ git add file

Once you marked all the files you want to include in the next commit, complete the commit with a commit message to let collaborators know what you have changed,

$ git commit -m "commit-message"

If everything went well, you are now ready to push your changes to your remote with,

$ git push origin branch-name

6) Fetch new branches

If new branches have been pushed recently to the repository and you don’t have them you can invoke a

$ git fetch --all

to see all new branches and checkout to those.
7) Delete branches

To delete a local branch execute (you cannot be on the branch that you are going to delete!):

```bash
$ git branch -d branch-name
```

To delete a remote branch you need to push the delete command:

```bash
$ git push origin --delete branch-name
```

8) Open a pull request

If you are working on another branch than the master or if you forked a repository and want to propose changes you made into the master, you can open a so-called pull request. In order to do so, press the corresponding tab in the dashboard of a repository and then press the green button New pull request. You will be asked which branch from which fork you want to merge.

9) Keep your password stored locally

If you are setting up Github on your own personal computer, and you use two factor authentication, it might be time consuming to configure that every time you need to provide git credentials. Instead, you can have the computer to remember your password. To do that, you can:

```bash
$ git config --global credential.helper store
```

Please note you should only do that if this is your personal computer!

2.5. Submitting issues

If you are experiencing issues with any code or content of a repository (such as this operating manual you are reading right now), you can submit issues. For doing so go to the dashboard of the corresponding repository and press the Issues tab where you can open a new request.

For example you encounter a bug or a mistake in this operating manual, please visit this repository to open a new issue.
2.6. 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 2: `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
```
3.1. What is Docker?

Docker is used to perform operating-system-level virtualization, something often referred to as “containerization”. While Docker is not the only software that does this, it is by far the most popular one.

Containerization is a process that allows partitioning the hardware and the kernel of an operating systems in such a way that different containers can co-exist on the same system independently from one-another. Programs running in such a container have access only to the resources they are allow to and are completely independent of libraries and configurations of the other containers and the host machine. Because of this feature Docker containers are extremely portable.

Containers are often compared to virtual machines (VMs). The main difference is that VMs require a host operating system (OS) with a hypervisor and a number of guest OS, each with their own libraries and application code. This can result in a significant overhead. Imagine running a simple Ubuntu server in a VM on Ubuntu: you will have most of the kernel libraries and binaries twice and a lot of the processes will be duplicated on the host and on the guest. Containerization, on the other hand, leverages the existing kernel and OS and adds only the additional binaries, libraries and code necessary to run a given application. See the illustration bellow.
Because containers don’t need a separate OS to run they are much more lightweight than VMs. This makes them perfect to use in cases where one needs to deploy a lot of independent services on the same hardware or to deploy on not-especially powerful platforms, such as Raspberry Pi - the platform Duckiebots use.

Containers allow for reuse of resources and code, but are also very easy to work with in the context of version control. If one uses a VM, they would need to get into the VM and update all the code they are using there. With a Docker container, the same process is as easy as pulling the container image again.

### 3.2. How does Docker work?

You can think that Docker containers are build from Docker images which in turn are build up of Docker layers. So what are these?

Docker images are build-time constructs while Docker containers are run-time constructs. That means that a Docker image is static, like a `.zip` or `.iso` file. A container is like a running VM instance: it starts from a static image but as you use it, files and configurations might change.

Docker images are build up from layers. The initial layer is the base layer, typically an official stripped-down version of an OS. For example, a lot of the Docker images we run on the Duckiebots have `rpi-ros-kinetic-base` as a base.

Each layer on top of the base layer constitutes a change to the layers below. The Docker internal mechanisms translate this sequence of changes to a file system that the container can then use. If one makes a small change to a file, then typically only a single layer will be changed and when Docker attempts to pull the new version, it will need to download and store only the changed layer, saving space, time and bandwidth.

In the Docker world images get organized by their repository name, image name and
tags. As with Git and GitHub, Docker images are stored in image registers. The most popular Docker register is called DockerHub and it is what we use in Duckietown.

An image stored on DockerHub has a name of the form:

```
duckietown/IMAGE_NAME:VERSION-NAME-ARCH-NAME
```

All Duckietown-related images are in the `duckietown` repository. The images themselves can be very different for various applications.

Sometimes a certain image might need to have several different versions. These can be designated with `tags`. For example, the `daffy` tag means that this is the image to be used with the `daffy` version of Duckietown.

It is not necessary to specify a tag. If you don’t, Docker assumes you are interested in the image with `latest` tag, should such an image exist.

### 3.3. Working with images

If you want to get a new image from a Docker register (e.g. DockerHub) on your local machine then you have to **pull** it. For example, you can get an Ubuntu 18.04 image by running the following command:

```
$ docker pull library/ubuntu:18.04
```

You will now be able to see the new image you pulled if you run:

```
$ docker image list
```

If you don’t need it, or if you’re running down on storage space, you can remove an image by simply running:

```
$ docker image rm ubuntu:18.04
```

You can also remove images by their `IMAGE ID` as printed by the `list` command.

Sometimes you might have a lot of images you are not using. You can easily remove them all with:
However, be careful not to delete something you might actually need. Keep in mind that you can’t remove images that a container is using. To do that, you will have to stop the container, remove it, and then you can remove the related images.

If you want to look into the heart and soul of your images, you can use the commands `docker image history` and `docker image inspect` to get a detailed view.

### 3.4. Working with containers

Containers are the run-time equivalent of images. When you want to start a container, Docker picks up the image you specify, creates a file system from its layers, attaches all devices and directories you want, “boots” it up, sets up the environment up and starts a pre-determined process in this container. All that magic happens with you running a single command: `docker run`. You don’t even need to have pulled the image beforehand, if Docker can’t find it locally, it will look for it on DockerHub.

Here’s a simple example:

```bash
$ docker run ubuntu
```

This will take the `ubuntu` image with `latest` tag and will start a container from it.

The above won’t do much. In fact, the container will immediately exit as it has nothing to execute. When the main process of a container exits, the container exits as well. By default this `ubuntu` image runs `bash` and as you don’t pass any commands to it, it exits immediately. This is no fun, though.

Let’s try to keep this container alive for some time by using the `-it` switch. This tells Docker to create an interactive session.

```bash
$ docker run -it ubuntu
```

Now you should see something like:

`root@73335ebd3355:/#`
Keep in mind that the part after @ will be different—that is your container ID. In this manual, we will use the following icon to show that the command should be run in the container:

$ command to be run in the container

You are now in your new `ubuntu` container! Try to play around, you can try to use some basic `bash` commands like `ls`, `cd`, `cat` to make sure that you are not in your host machine.

**Note:** If you are sure about the difference between the host and the container, you might want to see what happens when you do `rm -rf /` IN THE CONTAINER. You will destroy the OS inside the container—but you can just exit and start another one. If instead you have confused host and container, at this point you probably need to re-install from scratch.

You can check which containers you are running using the `docker ps` command — analogous to the `ps` command. Open a new terminal window (don’t close the other one yet) and type:

$ docker ps

An alternative syntax is

$ docker container list

These commands list all running containers.

Now you can go back to your `ubuntu` container and type `exit`. This will bring you back to your host shell and will stop the container. If you again run the `docker ps` command you will see nothing running. So does this mean that this container and all changes you might have made in it are gone? Not at all, `docker ps` and `docker container list` only list the currently running containers.

You can see all containers, including the stopped ones with:

$ docker container list -a
Here `-a` stands for all. You will see you have two `ubuntu` containers here. There are two containers because every time you use `docker run`, a new container is created. Note that their names seem strangely random. We could have added custom, more descriptive names—more on this later.

We don’t really need these containers, so let’s get rid of them:

```
$ docker container rm container name 1 container name 2
```

You need to put your container names after `rm`. Using the container IDs instead is also possible. Note that if the container you are trying to remove is still running you will have to first stop it.

You might need to do some other operations with containers. For example, sometimes you want to start or stop an existing container. You can simply do that with:

```
$ docker container start container name
$ docker container stop container name
$ docker container restart container name
```

Imagine you are running a container in the background. The main process is running but you have no shell attached. How can you interact with the container? You can open a terminal in the container with:

```
$ docker attach container name
```

### 3.5. Running images

There are many command line arguments that can be passed to the `docker run` command.

Table 3.1 shows a summary of the options we use most often in Duckietown. Below, we give some examples
Table 3.1. Docker Run Options

<table>
<thead>
<tr>
<th>Short command</th>
<th>Full command</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>--interactive</td>
<td>Keep STDIN open even if not attached, typically used together with -t.</td>
</tr>
<tr>
<td>-t</td>
<td>--tty</td>
<td>Allocate a pseudo-TTY, gives you terminal access to the container, typically used together with -i.</td>
</tr>
<tr>
<td>-d</td>
<td>--detach</td>
<td>Run container in background and print container ID.</td>
</tr>
<tr>
<td>--name</td>
<td></td>
<td>Sets a name for the container. If you don’t specify one, a random name will be generated.</td>
</tr>
<tr>
<td>-v</td>
<td>--volume</td>
<td>Bind mount a volume, exposes a folder on your host as a folder in your container. Be very careful when using this.</td>
</tr>
<tr>
<td>-p</td>
<td>--publish</td>
<td>Publish a container’s port(s) to the host, necessary when you need a port to communicate with a program in your container.</td>
</tr>
<tr>
<td>-d</td>
<td>--device</td>
<td>Similar to -v but for devices. This grants the container access to a device you specify. Be very careful when using this.</td>
</tr>
<tr>
<td>--privileged</td>
<td></td>
<td>Give extended privileges to this container. That includes access to all devices. Be extremely careful when using this.</td>
</tr>
<tr>
<td>--rm</td>
<td></td>
<td>Automatically remove the container when it exits.</td>
</tr>
<tr>
<td>-H</td>
<td>--hostname</td>
<td>Specifies remote host name, for example when you want to execute the command on your Duckiebot, not on your computer.</td>
</tr>
<tr>
<td>--help</td>
<td></td>
<td>Prints information about these and other options.</td>
</tr>
</tbody>
</table>

Note that most of this is hidden from the Duckietown user because it is contained within the Duckietown Shell.

Examples

Set the container name to joystick:

```bash
--name joystick
```
Mount the host’s path `/home/myuser/data` to `/data` inside the container:

```
-v /home/myuser/data:/data
```

Publish port 8080 in the container as 8082 on the host:

```
-p 8082:8080
```

Allow the container to use the device `/dev/mmcblk0`:

```
-d /dev/mmcblk0
```

Run a container on the Duckiebot:

```
-H duckiebot.local
```

### 3.6. Other useful commands

1) **Pruning images**

Sometimes your docker system will be clogged with images, containers and what not. You can use `docker system prune` to clean it up.

```
$ docker system prune
```

Keep in mind that this command will delete all containers that are not currently running and all images not used by running containers. So be extremely careful when using it.

2) **Portainer**

Often, for simple operations and basic commands, one can use Portainer. Portainer is itself a Docker container that allows you to control the Docker daemon through your web browser. You can install it by running:
Note that Portainer comes pre-installed on your Duckiebot, so you don’t need to run the above command to access the images and containers on your robot. You still might want to set it up for your laptop.

3.7. Further resources

There is much more that you can learn to do with Docker.
Here are some resources you can look up:

• Duckietown Introduction to Docker for Robotics and Machine Learning;
• Docker official Get Started tutorial;
• Docker Curriculum;
• Docker Deep Dive, by Nigel Poulton.

3.8. Docker common troubleshooting

1) docker: Got permission denied while trying to connect to the Docker daemon socket

If this is on your laptop, that means when you setup your enviornment you did not grant your user account right to do certain things. You can fix this by running:

```
$ sudo adduser `whoami` docker
```

Log out and in again and it should be fixed.

2) Container does not start

**Symptom:** docker: Error response from daemon: Conflict. The container name "/container_name" is already in use by container "container_hash". You have to remove (or rename) that container to be able to reuse that name.

**Resolution:** Stop the container (docker stop container_name) if running and then remove (docker rm container_name) the container with the
3) Docker exits with \textbf{tls: oversized record received}

If Docker exits with the above error when running remote commands, the most likely reason is different versions of Docker on your computer and Duckiebot. You can check that by running \texttt{docker version} on both devices. If that is indeed the case, you need to upgrade the Docker binaries on your computer. To do that, follow the official instructions \url{here}.

4) I can’t run a container because I get \texttt{exec user process caused "exec format error"}

An error like this:

\begin{verbatim}
standard_init_linux.go:190: exec user process caused "exec format error"
\end{verbatim}

despite not being very descriptive typically means that there is a mismatch between the container’s processor architecture and the one on your computer. Different processor architectures have different instruction sets and hence binaries compiled for one are generally not executable on another. Raspberry Pis use ARM processors, while most of the laptops use x86 architecture which makes them incompatible. Still, there's hope. Most of the Duckietown Raspberry Pi containers have a piece of magic inside called Qemu which allows emulation of an ARM processor on a x86 machine. You can activate this emulator if you change the default entrypoint of the container by adding \texttt{--entrypoint=qemu3-arm-static} to options when running it.
PART B

Preliminaries: Hardware

Before starting this adventure, let’s cover the hardware basics.
UNIT B-1

Handling circuits and batteries

KNOWLEDGE AND ACTIVITY GRAPH

Requires: Nothing
Results: Preliminary knowledge on circuits and power source properties useful in Duckietown

Duckiebots support several power bank models, although not any power will work. Here, we list properties of the supported models.

1.1. The Duckiebattery (DB-C-DBatt)

This battery is the standard battery for the Duckiebot Founder’s edition (DB21M).

Figure 1.1. The Duckiebattery.

1) Technical specification
• Capacity: 10Ah at 3.7V
• Charging: Micro USB 5V at up to 2A
• Output 2 x USB type A 5V at up to 4A (combined)
• Charge time: 0-100% takes about 5h and 0-90% about 4h with a 2A power supply
• Weight: 189g (fully charged)

**Warning:** Lithium Ion batteries, like the Duckiebattery, are potentially dangerous and must be handled with care. Here are some do’s and don’t. Please do not skip this section.

2) **Handling: DO’s**

- If a battery has been subject to moisture and/or the case has been damaged eminently, dispose of the battery pack.
- In case of fire use a CO2 extinguisher.
- Storage preferably in cool, dry and ventilated area which is subject to little temperature change.
- Storage at high temperatures should be avoided.

3) **DONT’s**

- Do not connect a charge voltage greater than 5V.
- Do not connect an external voltage source to the USB output ports.
- The battery must not be opened, destroyed or incinerated, since it may leak or rupture, releasing in the environment it’s hermetically sealed chemicals.
- Do not short circuit terminals.
- Do not crush or puncture the battery, or immerse it in liquid.
- Do not place the battery near heating equipment, nor expose to direct sunlight for long periods.

4) **LED description**

The battery has five LEDs on the top, used for indicating state of charge.
Figure 1.2. LEDs indicate the state of charge of the Duckie battery.

**Note:** To see the battery state of charge, click *once* on the button. The state of charge LEDs will stay on for 10 seconds and the battery set in *idle* state, ”waking up” the battery.
5) Charge the battery

After setting the battery in idle mode, we can charge it by connecting the charger. The LEDs will be flashing at 1Hz, showing the battery is receiving charge.
6) Battery protection mode

The battery is equipped with safety features to prevent damage to others and itself. In particular, it has dedicated hardware to protect its cell from low voltage discharge. When a certain low cell voltage level is detected, the battery micro-controller, together with all other active components will be turned off, except the charger. When a Duckiebattery enters protection mode, it will look unresponsive. Nonetheless, the charger will “trickle” charge the battery cell until it has reached a safe voltage level, exiting the battery protection mode.

The battery protection mode can last up to 30 minutes, during which the battery might not indicate a state of charge nor that it is actually being charged. This does not mean the battery is dead, just “hibernating”.

7) USB outputs

The battery have two separate 5V 2A USB type A output, namely USB OUT-1 (a.k.a.
the muscles) and USB OUT-2 (a.k.a. the brains).

![Diagrams of USB OUT-1 and USB OUT-2]

Figure 1.5. Duckiebattery outputs behave differently.

- **USB OUT-1**: Connect this output to a non-sensitive power load, i.e., motor or LEDs. This output will experience short power drops when plugging and unplugging the charger cable.

- **USB OUT-2**: This is a 5V 2A USB output, uninterrupted by the charging process or the status of USB OUT-1. This port should be connected to the computing unit (i.e., NVIDIA Jetson Nano or Raspberry Pi) to allow the unit not to restart when plugging or unplugging the charger of the battery.

8) **Troubleshooting**

The most common fault is not related to the battery pack itself but the connection between the pack and the charger and/or the load.

**Note:** Always make sure the USB cable is not damaged and of good quality. Do not use a cable longer than 30cm. A faulty cable can cause excessive voltage drops between the battery pack and load, leading to low voltage issues.

**Symptom:** My battery does not charge.

**Resolution:** There can be several reasons why a charge is not being accepted. Below are the most common issue.
• The input voltage is too low or too high. Make sure you are apply 5V via the micro USB connector
• The battery is in battery protection mode and does not look like it’s charging, but it is. Come back in >30 minutes and press the button once to enter idle mode.
• The battery is in a fault state. This can be caused by over temperature on the battery cell and/or its internal PCB. Leave the battery to cool down for 1h then attempt to charge it again.

**Symptom:** One or both USB output are not working  
**Resolution:** There can be several reasons why the USB output is not working. Below are the most common issue.

• The battery is not on idle mode. Press the battery button once.
• The battery is in battery protection mode. Remove all loads, put in charge and wait >30mins to have the battery exit protection mode. Then enter wake up the battery by pressing the button once.
• The USB output is in over current/temperature mode. Disconnect all loads, enter idle mode and let the battery rest for 30 minutes.
• A external voltage was applied to the USB (output) port(s). This is a big no no (refer to DO's and DONT's above). Disconnect all loads and enter idle mode.

1.2. **The Duckie-power-bank**

The Duckie-power-bank (or Duckiebattery version 1) is the standard power source for Duckiebots in DB18 and DB19 configurations. Duckiebatteries are easily recognizable.
Figure 1.6. The Duckie Power Bank

1) Overview

The Duckie battery is equipped with 2 USB type A outputs (port A and B) and 1 Micro USB connector for charging.
It also has 4 LEDs representing the state of charge. Push the button on the side of the battery pack to turn on the LEDs. The LEDs indicate the residual charge according to:

<table>
<thead>
<tr>
<th>LED</th>
<th>Charge Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>3-25%</td>
</tr>
<tr>
<td>D2</td>
<td>25-50%</td>
</tr>
<tr>
<td>D3</td>
<td>50-75%</td>
</tr>
<tr>
<td>D4</td>
<td>75-100%</td>
</tr>
</tbody>
</table>

If D1 is flashing (0.5Hz) while not being charged, the battery pack is at a critical low charge (less than 3%).

2) Charging

The battery pack is charged via the Micro USB port with a 5V supply. While charging, one of the LEDs will be flashing (0.5Hz) indicating where in the charge cycle it is.

---

**Note:** When the battery pack is connected to the charger, the output voltage of port A and B will turn off for around 280ms:
This is an unwanted effect which will cause the Raspberry Pi, if on, to reboot.

**Note:** likewise, when disconnecting the charger, the outputs will turn OFF for 20ms causing the Raspberry Pi to reboot as well.

Furthermore, while the Duckiebattery supports pass-through (both outputs A and B will function while the Duckiebattery is being charged), during charge the output voltage of port A and B will drop around 300mV which might cause an under voltage warning of the Raspberry Pi. This will put the Raspberry Pi in a throttling mode limiting its performance.

3) **Discharging**

The output ports A and B have an unloaded output voltage around 5.1V. To turn the outputs on simply attach a load on port A/B (e.g., plug in to the Raspberry PI and duckieboard) or push the button.

The output ports will automatically turn off if less than 100mA is being drawn.

To turn the outputs back on simply push the button or reconnect the USB connector.

The combined output current is limited to 2.8A.

The battery capacity is 7.4Ah at 5V with an efficiency as follows:
<table>
<thead>
<tr>
<th>Load</th>
<th>Efficiency</th>
<th>Autonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A</td>
<td>91%</td>
<td>6h 44m</td>
</tr>
<tr>
<td>1.5 A</td>
<td>88%</td>
<td>4h 33m</td>
</tr>
<tr>
<td>2 A</td>
<td>85%</td>
<td>3h 9m</td>
</tr>
<tr>
<td>2.5 A</td>
<td>79%</td>
<td>2h 21m</td>
</tr>
</tbody>
</table>
UNIT B-2
Duckiebot Configurations

KNOWLEDGE AND ACTIVITY GRAPH

**Requires:** Nothing

**Results:** Knowledge of Duckiebot configuration naming conventions and their respective functionalities.

We define the different Duckiebot configurations, from the first DB17 used in the MIT course 2.166 in 2017 to the latest available.

Duckiebots DB18 onwards can be obtained from the Duckietown project store.

2.1. Overview

<table>
<thead>
<tr>
<th>Model</th>
<th>Computation</th>
<th>Sensing</th>
<th>Actuation</th>
<th>Memory</th>
<th>Power</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB17</td>
<td>RPI3</td>
<td>Camera</td>
<td>2x DC motors, 5x RGB LEDs</td>
<td>32GB</td>
<td>Off-the-shelf</td>
<td></td>
</tr>
<tr>
<td>DB18</td>
<td>RPI3B+</td>
<td>Camera</td>
<td>2x DC motors, 5x RGB LEDs</td>
<td>32GB</td>
<td>Duckie-power bank</td>
<td>addressable LEDs</td>
</tr>
<tr>
<td>DB19</td>
<td>RPI3B+</td>
<td>Camera, Wheel Encoders</td>
<td>2x DC motors, 5x RGB LEDs</td>
<td>32GB</td>
<td>Duckie-power bank</td>
<td></td>
</tr>
<tr>
<td>DB21M</td>
<td>JN2GB</td>
<td>Camera, Wheel Encoders, ToF, IMU</td>
<td>2x DC motors, 4x RGB LEDs, Screen</td>
<td>32GB</td>
<td>Duckiebattery Chassis v1</td>
<td></td>
</tr>
<tr>
<td>DB21</td>
<td>JN2GB / JN4GB</td>
<td>Camera, Wheel Encoders, ToF, IMU</td>
<td>2x DC motors, 4x RGB LEDs, Screen</td>
<td>64GB</td>
<td>Duckiebattery Chassis v2</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- “JN”: NVIDIA Jetson Nano
- “RPI”: Raspberry Pi
2.2. Duckiebot version 2021, or DB21

The Duckiebot DB21 debuted with the “Self-Driving Cars with Duckietown” massive open online course, as version DB21M. Later revisions, referred to under the broader label DB21, improve the DB21M by:

- expanding the onboard memory from 32GB to 64GB;
- tweaking the chassis design (v1.0 -> v2.0) for reduced complexity and increased stiffness;
- introducing a newer version of the HUT (v3.15); which is backwards compatible and removes the need for an additional resistor on the top button;
- downgrades the IMU version from MPU-9250 to MPU-6050 due to global chip shortages (2021-2022 chip crisis).

To assemble a DB21 Duckiebot, follow these instructions.

You can obtain a DB21 Duckiebot from the Duckietown project shop.

2.3. Duckiebot MOOC Founder’s edition, or DB21M

The DB21M is the first Duckiebot equipped with a NVIDIA Jetson Nano 2 GB computational unit instead of a Raspberry Pi.

The DB21M debuts in 2021 with the first edition of the massive open online course, hosted on the edX platform.
The **DB21M** is readily recognized by its blazing blue chassis and triple-decker configuration. It is equipped with a sensor suite including: camera, time-of-flight sensor, inertial measurement unit (IMU) and wheel encoders. Moreover, the **DB21M** features new electronics (HUT v3.1, front and back bumpers), a screen, a button and a custom designed **Duckiebattery** (not to be confused with the **Duckie-power-bank**).

To assemble a **DB21M** Duckiebot, follow these instructions.

You can obtain a **DB21M** Duckiebot from the [Duckietown project shop](https://www.duckietown.org/).  

### 2.4. Duckiebot version 2019, or **DB19**

The **DB19** is the latest version of the Duckiebot. You have a **DB19** Duckiebot for sure if you have the blue motors shown in figure [Figure 2.2](#).
Figure 2.2. The motors for the version DB19.

Apart from the new motors and another HUT (v. 2.1), the DB19 is identical with the DB18. A complete version can be seen here:

Figure 2.3. The complete Duckiebot 19

To assemble a DB19 Duckiebot, follow these instructions.

You can obtain a DB19 Duckiebot from the Duckietown project shop.

2.5. Duckiebot version 2018, or DB18
You have a DB18 Duckiebot if, e.g., you have pledged to the Kickstarter.

There are two configurations of the DB18.

1) The DB18 configuration

The main configuration is labeled plainly as DB18 and is designed to operate on any Duckietown. You have the DB18 if, e.g., you are a student attending the 2019 graduate level classes in ETH or the University of Montreal, or you have pledged to Summer 2018 Kickstarter.

The DB18 supports different power bank models depending on the geographical region, but all these solutions are functionally equivalent, although their form factor is different.

You can recognize a DB18 from previous versions for having only one board in addition to the Raspberry Pi, a backplate, and the computational stack mounted in the bottom deck.

Figure 2.4. A Duckiebot DB18 assembly.
To assemble a DB18 Duckiebot, follow these instructions.

You can obtain a DB18 Duckiebot from the Duckietown project shop.

2) The DB18-Robotarium configuration

The DB18-Robotarium configuration adds to the DB18 the hardware necessary to operate in Robotariums (a.k.a. Duckietown Autolabs): continuously operating Duckietowns. They are otherwise identical to the DB18.

The additional hardware consists of a top localization April Tag infrastructure and an “auto-charging” mod, which allows Duckiebots to dock to charging stations and estimate the residual battery charge.

Robotariums are experimental Duckietown features, currently under development. You will find DB18-Robotarium models in university research labs.

If you are interested in obtaining DB18-Robotarium Duckiebots, or in building your Duckietown Robotarium, contact the Duckietown team.
2.6. **Duckiebot versions 2017, or DB17**

In the **DB17** version, we had several different configurations.

The configurations are defined with a root: **DB17-**, indicating the “bare bones” Duckiebot used in the Fall 2017 synchronized course, and an appendix **y** which can be the union (in any order) of any or all of the elements of the optional hardware set \( \mathcal{O} = \{ w, j, d, p, l, c \} \).

A **DB17** Duckiebot can navigate autonomously in a Duckietown, but cannot communicate with other Duckiebots.

The elements of \( \mathcal{O} \) are labels identifying optional hardware that aids in the development phase and enables the Duckiebot to talk to other Duckiebots. The labels stand for:

- **w**: 5 GHz wireless adapter to facilitate streaming of images;
- **j**: wireless joypad that facilitates manual remote control;
- **d**: USB drive for additional storage space;
- **c**: a different castor wheel to replace the preexisting omni-directional wheel;
- **p**: PWM hat for convenient powering of the DC motor hat;
- **l**: includes LEDs, LED hat, bumpers and the necessary mechanical bits to set the bumpers in place. Note that the installation of the bumpers induces the replacement of a few **DB17** components;

**Note:** During the Fall 2017 course, three Duckietown Engineering Co. branches (Zurich, Montreal, Chicago) are using these configuration naming conventions.
Moreover, all institutions release hardware to their Engineers in training in two phases.

It may be convenient at times to refer to hybrid configurations including any of the DB17-jwcd in conjunction with a subset of the DB17-l components. In order to disambiguate, we define partial upgrades as:

- **DB17-l1**: adds a PWM hat to DB17, in addition to a short USB angled power cable and a M-M power wire;
- **DB17-l2**: adds a bumpers set to DB17, in addition to the mechanical bits to assemble it;
- **DB17-l3**: adds a LED hat and 5 RGB LEDs to DB17-l12, in addition to the F-F wires to connect the LEDs to the LED board.

**Note:** introducing the PWM hat in DB17-l1 induces a replacement of the spliced cable powering solution for the DC motor hat. Details can be found in Unit C-15 - Assembly - Duckiebot DB17.

**Functions**: DB17-l is the necessary configuration to enable communication between Duckiebots, hence fleet behaviors (e.g., negotiating the crossing of an intersection). Subset configurations are sometimes used in a standalone way for: (DB17-l1) avoid using a sliced power cable to power the DC motor hat in DB17, and (DB17-l2) for purely aesthetic reasons.
UNIT B-3

Getting the Duckiebot hardware

Knowledge and activity graph

| Requires: | Knowledge of Duckiebot hardware configurations |
| Results: | Parts to assemble a Duckiebot. |

3.1. Foreword

You can acquire Duckiebots in two ways.

- **Do it yourself** (DB17): the DB17 Duckiebot configuration is made of components that you can source independently.
- **“One click” solution** (DB18 and above): you can source complete hardware kits on the Duckietown project online store.

3.2. Acquiring the parts for a DB18

The DB18 configuration has the same functionalities of a fully equipped DB17 Duckiebot.

3.3. Acquiring the parts for a DB17

Here, we provide a link to all bits and pieces that are needed to build a DB17 Duckiebot, along with the 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.
- International deliveries are subject to additional custom clearances and import fees.
- Substitutions are OK for the mechanical components, and not OK for all the electronics, unless you are OK in writing some software. Limited technical support will be offered for hardware customizations.
- Buying the parts for more than one Duckiebot makes each one cheaper than buying
only one.
• For some components, the links we provide contain more bits than actually needed.

1) **DB17** Bill of materials
## Table 3.1. Bill of Materials

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chassis</td>
<td>USD 20</td>
</tr>
<tr>
<td>Camera with 160-FOV Fisheye Lens</td>
<td>USD 39</td>
</tr>
<tr>
<td>Camera Mount</td>
<td>USD 4</td>
</tr>
<tr>
<td>300mm Camera Cable</td>
<td>USD 2</td>
</tr>
<tr>
<td>Raspberry Pi 3 - Model B+</td>
<td>USD 39</td>
</tr>
<tr>
<td>Heat Sinks</td>
<td>USD 3</td>
</tr>
<tr>
<td>Power supply for Raspberry Pi</td>
<td>USD 7.50</td>
</tr>
<tr>
<td>16 GB Class 10 MicroSD Card</td>
<td>USD 10</td>
</tr>
<tr>
<td>Micro SD card reader</td>
<td>USD 6</td>
</tr>
<tr>
<td>DC Motor HAT</td>
<td>USD 22.5</td>
</tr>
<tr>
<td>2 Stacking Headers</td>
<td>USD 2.50/piece</td>
</tr>
<tr>
<td>Battery Pack</td>
<td>USD 25</td>
</tr>
<tr>
<td>16 Nylon Standoffs (M2.5 12mm F 6mm M)</td>
<td>USD 0.06/piece</td>
</tr>
<tr>
<td>4 Nylon Hex Nuts (M2.5)</td>
<td>USD 0.02/piece</td>
</tr>
<tr>
<td>4 Nylon Screws (M2.5x10)</td>
<td>USD 0.05/piece</td>
</tr>
<tr>
<td>2 Zip Ties (300x5mm)</td>
<td>USD 9</td>
</tr>
<tr>
<td>Wireless Adapter (5 GHz) (DB17-w)</td>
<td>USD 25</td>
</tr>
<tr>
<td>Joypad (DB17 - j)</td>
<td>USD 10.50</td>
</tr>
<tr>
<td>Tiny 32GB USB Flash Drive (DB17 - d)</td>
<td>USD 10</td>
</tr>
<tr>
<td>PWM/Servo HAT (DB17 - l1)</td>
<td>USD 17.50</td>
</tr>
<tr>
<td>Power Cable (DB17 - l1)</td>
<td>USD 7.80</td>
</tr>
<tr>
<td>Male-Male Jumper Wire (DB17 - l1)</td>
<td>USD 1.95</td>
</tr>
<tr>
<td>8 M3x10 pan head screws (DB17 - l2)</td>
<td>USD 7</td>
</tr>
<tr>
<td>8 M3 nuts (DB17 - l2)</td>
<td>USD 7</td>
</tr>
<tr>
<td>Bumpers set (DB17 - l2)</td>
<td>USD 7 (custom made)</td>
</tr>
<tr>
<td>Bumper bracers set (DB17 - l2)</td>
<td>USD 7 (custom made)</td>
</tr>
<tr>
<td>LEDs (DB17 - l3)</td>
<td>USD 10</td>
</tr>
<tr>
<td>LED HAT (DB17 - l3)</td>
<td>USD 9/piece (but 3 pieces minimum)</td>
</tr>
<tr>
<td>20 Female-Female Jumper Wires (300mm) (DB17 - l3)</td>
<td>USD 8</td>
</tr>
<tr>
<td>4 4 pin female header (DB17 - l3)</td>
<td>USD 0.60/piece</td>
</tr>
<tr>
<td>12 pin male header (DB17 - l3)</td>
<td>USD 0.48/piece</td>
</tr>
<tr>
<td>2 16 pin male header (DB17 - l3)</td>
<td>USD 0.61/piece</td>
</tr>
<tr>
<td>3 pin male header (DB17 - l3)</td>
<td>USD 0.10/piece</td>
</tr>
<tr>
<td>2 pin female shunt jumper (DB17 - l3)</td>
<td>USD 2/piece</td>
</tr>
<tr>
<td>40 pin female header (DB17 - l3)</td>
<td>USD 1.50</td>
</tr>
<tr>
<td>5 200 Ohm resistors (DB17 - l3)</td>
<td>USD 0.10/piece</td>
</tr>
<tr>
<td>10 130 Ohm resistors (DB17 - l3)</td>
<td>USD 0.10/piece</td>
</tr>
<tr>
<td>Soldering tools</td>
<td>USD 20</td>
</tr>
<tr>
<td>Total for DB17 configuration</td>
<td>USD 193</td>
</tr>
<tr>
<td>Total for DB17 - w configuration</td>
<td>USD 218</td>
</tr>
<tr>
<td>Total for DB17 - j configuration</td>
<td>USD 203</td>
</tr>
<tr>
<td>Total for DB17 - d configuration</td>
<td>USD 203</td>
</tr>
<tr>
<td>Total for DB17 - wjd configuration</td>
<td>USD 238</td>
</tr>
<tr>
<td>Total for DB17 - l configuration</td>
<td>USD 367</td>
</tr>
</tbody>
</table>
3.4. Chassis

We selected the Magician Chassis as the basic chassis for the robot (Figure 3.1). 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 2 DC motors and wheels as well as the structural part, in addition to a screwdriver and several necessary mechanical bits (standoffs, screws and nuts).

Figure 3.1. The Magician Chassis

3.5. Raspberry Pi 3 - Model B

Note: It is recommend to upgrade to Raspberry Pi 3 model B+. In this case the 5 GHz wireless adapter is no longer necessary.

The Raspberry Pi is the central computer of the Duckiebot. Duckiebots use Model B (Figure 3.2) (A 1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM), a small but powerful computer.
1) Power Supply

We want a hard-wired power source (5VDC, 2.4A, Micro USB) to supply the Raspberry Pi (Figure 3.3) while not driving. This charger can double down as battery charger as well.

2) Heat Sinks

The Raspberry Pi will heat up significantly during use. It is warmly recommended to add heat sinks, as in Figure 3.4. Since we will be stacking HATs on top of the Raspberry Pi with 15 mm standoffs, the maximum height of the heat sinks should be well below 15 mm. The chip dimensions are 15x15mm and 10x10mm.
3) Class 10 MicroSD Card

The MicroSD card (Figure 3.5) is the hard disk of the Raspberry Pi. 16 GB of capacity are sufficient for the system image.

4) Micro SD card reader

A microSD card reader (Figure 3.6) is useful to copy the system image to a Duckiebot from a computer to the Raspberry Pi microSD card, when the computer does not have a native SD card slot.
3.6. Camera

The Camera is the main sensor of the Duckiebot. All versions equip a 5 Mega Pixels 1080p camera with wide field of view (160°) fisheye lens (Figure 3.7).

1) Camera Mount

The camera mount (Figure 3.8) serves to keep the camera looking forward at the right angle to the road (looking slightly down). The front cover is not essential.
Figure 3.8. The Camera Mount

The assembled camera (without camera cable), is shown in (Figure 3.9).

Figure 3.9. The Camera on its mount

2) 300mm Camera Cable

A longer (300 mm) camera cable Figure 3.10 makes assembling the Duckiebot easier, allowing for more freedom in the relative positioning of camera and computational stack.
3.7. DC Motor HAT

We use the DC Stepper motor HAT (Figure 3.11) to control the DC motors that drive the wheels. This item will require soldering to be functional. This HAT has dedicate PWM and H-bridge for driving the motors.

1) Stacking Headers

We use a long 20x2 GPIO stacking header (Figure 3.12) to connect the Raspberry Pi with the DC Motor HAT. This item will require soldering to be functional.
3.8. Battery

The battery (Figure 3.13) 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. The battery linked in the table above comes with two USB to microUSB cables.

3.9. 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 Raspberry Pi to the chassis and the HATs stacked on top of the Raspberry Pi.

The Duckiebot requires 8 standoffs, 4 nuts and 4 screws.
3.10. Zip Tie

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

3.11. Configuration DB17-w

1) Wireless Adapter (5 GHz)

The Edimax AC1200 EW-7822ULC 5 GHz wireless adapter (Figure 3.16) boosts the connectivity of the Duckiebot, especially useful in busy Duckietowns (e.g., classroom). This additional network allows easy streaming of images.
3.16. The Edimax AC1200 EW-7822ULC wifi adapter

This component is not necessary if upgrading to Raspberry Pi 3 Model B +.

3.12. Configuration DB17 - j

1) 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 linked in the table (Figure 3.17) does not include batteries.

---

Figure 3.17. A Wireless Joypad

**Requires:** 2 AA 1.5V batteries (Figure 3.18) not included in the bill of materials.
3.13. Configuration **DB17-d**

1) **Tiny 32GB USB Flash Drive**

In configuration **DB17-d**, the Duckiebot is equipped with an “external” hard drive (Figure 3.19). This add-on 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.

3.14. Configuration **DB17-l**

3.15. **LEDs**
This Duckiebot is equipped with 5 RGB LEDs (Figure 3.20). LEDs can be used to signal to other Duckiebots, or just make fancy patterns.

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

![RGB LEDs](image)

Figure 3.20. The RGB LEDs

1) LED HAT

The LED HAT (Figure 3.21) 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.

This board is custom designed 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.

![LED HAT](image)

Figure 3.21. The LED HAT

2) PWM/Servo HAT

The PWM/Servo HAT (Figure 3.22) mates to the LED HAT and provides the signals to control the LEDs, without taking computational resources away from the Raspberry Pi itself. This item will require soldering.
Figure 3.22. The PWM-Servo HAT

3) Power Cable

To power the PWM/Servo HAT from the battery, we use a short (30cm) angled male USB-A to 5.5/2.1mm DC power jack cable (Figure 3.23).

Figure 3.23. The 30cm angled USB to 5.5/2.1mm power jack cable.

4) Male-Male Jumper Wires

The Duckiebot needs one male-male jumper wire (Figure 3.24) to power the DC Stepper Motor HAT from the PWM/Servo HAT.

Figure 3.24. Premier Male-Male Jumper Wires

5) Female-Female Jumper Wires
20 Female-Female Jumper Wires (Figure 3.25) are necessary to connect 5 LEDs to the LED HAT.

Figure 3.25. Premier Female-Female Jumper Wires

3.16. Bumpers
These bumpers are designed to keep the LEDs in place and are therefore used only in configuration DB17-1. They are custom designed parts, so they must be produced and cannot be bought. We used laser cutting facilities.

3.17. Headers, resistors and jumper
Upgrading DB17 to DB17-1 requires several electrical bits: 5 of 4 pin female header, 2 of 16 pin male headers, 1 of 12 pin male header, 1 of 3 pin male header, 1 of 2 pin female shunt jumper, 5 of 200 Ohm resistors and finally 10 of 130 Ohm resistors.
These items require soldering.

3.18. Caster (DB17-c)
The caster (Figure 3.26) is an DB17-c component that substitutes the steel omni-directional wheel that comes in the Magician Chassis package. Although the caster is not essential, it provides smoother operations and overall enhanced Duckiebot performance.
Figure 3.26. The caster wheel

To assemble the caster at the right height we will need to purchase:

- 4 standoffs (M3 12mm F-F) ([Figure 3.27a - Standoffs for caster wheel.](#)),
- 8 screws (M3x8mm) ([Figure 3.27b - Screws for caster wheel.](#)), and
- 8 split lock washers ([Figure 3.27c - Split lock washers for caster wheel.](#)).
(b) Screws for caster wheel.

(c) Split lock washers for caster wheel.

Figure 3.27. Mechanical bits to assemble the caster wheel.

**Note:** The caster wheel use is to be considered experimental and has been dropped in official configurations starting from DB18.
PART C
Duckiebot assembly and setup

In this section you will find the necessary information on how to get started with your Duckiebot quest.

How to get help

There are several ways to get help:

1. Join the Duckietown Stack Overflow space. This is a private space and requires invitation. Sign up on the Duckietown website to receive an invitation.

2. Join the Duckietown Slack, #how-to-get-help channel (or a more appropriate one depending on the specificity of your question).

3. If you are familiar with Git, you can create an issue for this repository.
UNIT C-1

Setup - Laptop

KNOWLEDGE AND ACTIVITY GRAPH

- **Requires:** An Ubuntu 20.04 environment with sufficient free disk space (recommended 100 GB).
- **Requires:** Internet connection.
- **Requires:** About 10 minutes.
- **Results:** A laptop ready to be used for Duckietown.

1.1. Minimal Laptop Requirements

These installation steps make sure that you have a minimal “sane” environment, which includes:

1. Git;
2. Docker;
3. The Duckietown Shell.

1.2. Native installation vs virtual machines

Having Ubuntu installed natively on your laptop is recommended but not strictly required.

If you are running Ubuntu in a VM make sure that you are using a Bridged network adapter (for example VirtualBox uses NAT by default). This allows you to be on the same subnetwork as your Duckiebot.

Sometimes when running a VMware machine on a Mac OS host, it is neccessary to have two network adapters: *Share with my Mac* for connecting to the internet and *Bridged Networking* for connecting to the Duckiebot.

For more information about networking with VMWare, see [here](#).

1.3. Setup for Ubuntu 20.04
1) System installation

Install Ubuntu 20.04.

**Note:** if you have a robot, you must install Ubuntu natively.

- For instructions, see for example this online tutorial.

2) Basic dependencies

Installs pip3, git, git-lfs, curl, wget:

```bash
$ sudo apt update
$ sudo apt install -y python3-pip git git-lfs curl wget
```

3) Docker

Install Docker by following the instructions here.

Adds user to “docker” group:

```bash
$ sudo adduser `whoami` docker
```

**Note:** you need to log out and back in for the group change to take effect.

**Warning:** If you missed this step, you will later run into docker permission issues.

Make sure you have docker-compose installed:

```bash
$ sudo apt-get install docker-compose
```

4) Duckietown Shell

**Note:** If you are not using Ubuntu environment and you would like to install Duckietown Shell, you should refer to this page.

Install the Duckietown Shell using the following command,

```bash
$ pip3 install --no-cache-dir --user --upgrade duckietown-shell
```
The first thing you need to do with the Duckietown Shell, is set the Duckietown software distribution you want to work with, for this version of the book, we use daffy. Set the shell to use the daffy distribution by running the following command,

```
$ dts --set-version daffy
```

5) For virtual machines

For VMWare, install the package open-vm-tools:

```
$ sudo apt install open-vm-tools
```

### 1.4. Setting up Mac OS X

Warning: this configuration is not officially supported. There might be problems with implementation. We strongly urge you to use Ubuntu 20.04 environment.

1) Basic dependencies

You will need to find the instructions for git, git-lfs.

2) Quartz

You will also need the latest version of XQuartz.

You can install using brew as follows:

```
$ brew install xquartz
```

Or, download from [here](#) and follow the instructions.

After installing XQuartz, run it in the command line with:

```
$ open -a XQuartz
```

Go to “Preferences” and in the “Security” tab make sure that the checkbox next to “Allow connections from network clients” is set. Now close XQuartz.

You may want to add the following lines to your `.bashrc` file:
export IP=$(ifconfig en0 | grep inet | awk '$1=="inet" {print $2}+')
xhost +$IP

**Note:** You may also need to add `/usr/X11/bin` to your system path in order for it to find `xhost`.

These will find your IP and then allow incoming connections to it in order to be able to popup windows from within docker containers.

Alternatively, you can run them each time before you want to use X11 forwarding. This is done for you when you run things through the duckietown shell.

3) Docker

- (For Linux): Follow [these instructions](#);
- (For Mac): Follow [these instructions](#);
- (Important): Follow [post-install instructions](#);

4) Duckietown Shell

Follow [these instructions](#).
This section describes how to register your Duckietown account and set up the Duckietown authentication token, as well as other accounts you will need.

### Knowledge and Activity Graph

| Requires: | Internet connection. |
| Requires: | About 10 minutes. |
| Requires: | Laptop with Duckietown Shell command installed and correctly setup. |
| Results: | Duckietown token correctly set up. |

#### 2.1. Sign up on the Duckietown website

To register on the Duckietown website, go to:

https://www.duckietown.org/site/register

and click through to pick a community.

**Note:** If you are a student, please use your official student email address, so that you can be sorted in the right group.

#### 2.2. Find the Duckietown token

The Duckietown Token allows to authenticate your devices to the Duckietown network.

The token is a string of letters and numbers that looks something like this:

```
dt1-7vEuJsaxeXXXXX-43dzqWFnWd8KBa1yev1g3UKnzVxZkkTbfSJnxzuJjWaANEf4y6XSXBWTp7y
```

To find your token, first [login to duckietown.org](https://www.duckietown.org/site/your-token), then open the page:
Note: It may take up to 5 minutes after first creating your account for a token to be generated.

2.3. Tell the Duckietown Shell your token

Use the command

```
$ dts tok set
```

and follow the prompt.

Verify your token was successfully set by running the shell command,

```
$ dts tok status
```

You should see a message like the following,

```
dts: Correctly identified as uid = ***
```

2.4. Troubleshooting

1) “DTS Shell object has no attribute sprint” when using `dts tok set`

You have to completely reinstall `dts` and its commands. Do that by:
1. Delete the `~/.dt-shell` folder
2. Uninstall `dts` by running `pip uninstall duckietown-shell`
3. Reinstall `dts` by following the procedure in Laptop Setup

2.5. Other Accounts

1) Github

You will find it useful to have an account on Github if you don’t have one already. You can get one [here](#).
UNIT C-3

Setup Duckiebot SD Card

This page is for the DB18 configuration and above (including Jetson Nano configurations).

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** An SD card of size at least 32 GB
- **Requires:** A computer with Ubuntu
- **Requires:** At least 20 GB of free space on the computer
- **Requires:** An internet connection
- **Requires:** SD card reader
- **Requires:** Duckietown Shell, as configured in Subsection 1.3.4 - Duckietown Shell.
- **Requires:** Docker, as configured in Subsection 1.3.3 - Docker.
- **Requires:** Duckietown Token, as configured in Unit C-2 - Setup - Account.
- **Requires:** 2.5 hours on average (depends on internet connection and sd-card adapter used)
- **Results:** A correctly configured Duckiebot SD card, ready to be used to give life to your Duckiebot.

3.1. **Choose a name for your robot**

Pick a hostname for your robot. This will be the name of your robot and has to be unique within a fleet of robots connected to the same network. A valid hostname satisfies all the following requirements:

- it is lowercase
- it starts with a letter
- it contains only letters, numbers, and underscores

3.2. **Burn the SD card**
1) Video Tutorial

Figure 3.1. How to burn a Duckiebot SD card.

2) Step-by-Step Instructions

**Note:** Though the suggested operating system for this operation is Ubuntu 20.04, this should work on any Unix-like operating system. If you are using dts through WSL or experience any issues while performing this procedure, when prompted to enter the device name, simply provide a path to a file, for example `/home/user/duckiebot_sd_card.img`. The program will proceed by creating a disk image on that file that you can later transfer to an SD card using any standard flashing tool, e.g., etcher, dd.

**Note:** If you are using a microSD to SD card adapter, make sure the adapter does not have the write protection enabled. Check [this link](#) to learn more.

Plug the SD card in your computer using an SD card reader. If your computer does not have one, you will find a USB to microSD card adapter in your Duckiebot kit.

Initialize the SD card by running the following command,

```
$ dts init_sd_card --hostname hostname --type type --configuration configuration --wifi wifi
```

where,
--hostname          Name of the robot to flash the SD card for.
--type              The type of your device. Types are `duckiebot` (default),
                    `watchtower`, `traffic_light`.
--configuration     The model of your robot. This is associated with
                    `--type` option. E.g. `DB21M`, `DB19`, or `DB18`.

Other options are:

--wifi              A comma-separated list of WiFi networks, each network is passed in the format `wifi_name: wifi_password`
                    default: duckietown:quackquack
--country           Country code.
                    default: US

**Note:** the default username and password for all Duckiebots are “duckie” and “quackquack”, respectively.

**Warning:** for the “Self-Driving Cars with Duckietown” online course on edX, the robot configuration to choose is `DB21M`.

If you plan on connecting with the Duckiebot over different networks (e.g., at home and in class), you can list them like this:

```sh
$ dts init_sd_card ... --wifi duckietown:quackquack,myhomenetwork:myhomepassword,myuninetwork:myunipassword
```

**Note:** There should be no space after the commas.

Watchtowers and traffic lights by default have Wi-Fi not configured, as we recommend hard wiring them with Ethernet cables. Default Wi-Fi settings for other robot types is “duckietown:quackquack”.

Each network defined in the list can support the following arguments:

- Open networks (no password) network: "ssid"
- PSK (Pre-shared key) protected networks (no password) network: "ssid:psk"
- EAP (Extensible Authentication Protocol) protected networks network: "ssid:username:password"
Make sure to set your country correctly with the \texttt{--country} option (e.g., CA for Canada, CH for Switzerland, US for the United States of America). Neglecting this sometimes will result in specific Wi-Fi hot-spots not being seen by the Duckiebot.

Additional options for \texttt{init_sd_card} exist. For a full list of the options, run:

\begin{verbatim}
$ dts init_sd_card --help
\end{verbatim}

After you run the \texttt{dts init_sd_card} command, follow the instructions that appear on screen.

Part of this procedure includes accepting the Duckietown Software License, Terms and Conditions and Privacy Policy, as well as robot configuration-specific licenses due to the presence of third party software in the SD card.

The next step is that of choosing among all the devices connected to your computer, which one represents the SD card that you want to flash for your Duckiebot. Given the danger of choosing a wrong device (from data loss to OS files corruption), the program will guide you through this step by asking the size of the SD card. Devices that do not match the given size will not be shown.

Type in or copy-paste the device name from the list and press \texttt{Enter}.

At this point, the SD card is being flashed. A verification step follows to make sure that the flashing went well.

On successful end of the procedure, the drives will be automatically ejected and you can just remove the SD card from your laptop.

\subsection*{3.3. Booting the Duckiebot}

Now insert the SD card as shown in the video below into your robot and push the button on the battery to power up the Duckiebot.
Warning: Unless you are using a Duckiebattery (available in the DB21M Duckiebot), don’t charge the battery while you are doing the initialization (or in general when the Duckiebot is turned on). The external power supply might not be able to provide sufficient current and the Raspberry Pi will reboot. Should that happen during the initialization procedure, you will likely have to burn the SD card again.

3.4. Monitoring the First Boot

You know that your Raspberry Pi, or NVIDIA Jetson Nano, has successfully booted when you see it using the `dts fleet discover` utility. Open a terminal and run the command:

```
$ dts fleet discover
```

The command above (`fleet discover`) will show a list of all the Duckiebots reachable on your network. Leave this tool open, it will refresh automatically every second, so there is no need to manually restart it.

You should see your Duckiebot in a few minutes after you inserted your SD card and power on the robot.

The list will look like the following.

![Output of 'dts fleet discover'](image)

Figure 3.2. Output of ‘dts fleet discover’

During the first boot, the robot will automatically reboot several times. Wait for the “Status” column to read “Ready” and turn solid green.

**Note:** If the command above returns an error about the library `zeroconf` being missing, run `pip3 install zeroconf` and retry.

When the column “Dashboard” reads “Up” and turns solid green, you can proceed to the robot’s first setup that can be performed through any Web Browser.
Figure 3.3. Output of `dts fleet discover` (Dashboard Up)

When the Dashboard is Up, open your browser and visit the URL `http://hostname.local/`. You will see the following page,

![Dashboard Page](image)

Figure 3.4

This is the dashboard of your Duckiebot. The Dashboard is built using a framework called `\compose\`. You will see how to configure it in **Unit C-7 - Setup - Dashboard**.

### 3.5. Troubleshooting

1) SD card flashing

**Symptom:** The SD card doesn’t seem to be written.
**Symptom:** The SD card process seems extremely fast, and there is no data on my SD card.

**Resolution:** Check if your SD card has a write protection switch or it is in read-only mode.

**Resolution:** Make sure you inputted the correct drive name during the initialization procedure.

**Symptom:** The flashing procedure fails with a **Bad archive** error.

**Resolution:** This happens when the downloaded compressed disk image file appears corrupted. You can force the re-download by adding the option `--no-cache` to the `init_sd_card` command.

**Symptom:** The verification process fails with error **Please set up a token using "dts tok set".**

**Resolution:** Make sure you completed the Duckietown token setup procedure [Unit C-2 - Setup - Account](#).

---

### 2) First Boot

**Symptom:** The red LED on the Raspberry Pi is OFF.

**Resolution:** Press the button on the side of the battery ([Figure 3.5](#)).

![Figure 3.5. The power button on the RAVPower Battery.](#)

**Symptom:** The Raspberry Pi has power but it does not boot.

**Resolution:** Initialize the SD card if not done already. If problem persists, try again.

**Symptom:** I cannot ping the Duckiebot.

**Resolution:** Check the networking section of the book to see if your network is set up correctly.

**Symptom:** I am not sure whether the Duckiebot is properly initialized.

**Resolution:** As long as the fleet discover tool shows ready, your Duckiebot should be ready. You can also visit [http://hostname:9000](http://hostname:9000) to see all the container status. Generally, as long as you see the Duckiebot web UI is up, your Duckiebot should be correctly initialized.
**Symptom:** (only for DB18 and DB19) The LEDs light up in a variety of colors when the battery is plugged in for the first time.

**Resolution:** The LEDs of the (DB18 and DB19) Duckiebot should light up in white as soon as you power the Duckiebot. If the LEDs turn on and shine in any different color than white, probably the code on the microcontroller is corrupted. You can reflash it using the procedure in Unit C-20 - Debug - Re-flash Microcontroller.

**Symptom:** On first boot, the lights of the Duckiebot do not turn white (might be blue).

**Resolution:** Run the following commands:

```
$ dts duckiebot update <hostname>
```

### 3.6. SSH to the Duckiebot

Next, try to log in using SSH, using

```
$ ssh duckie@hostname.local
```

This should succeed without password. The default password is `quackquack`.

### 3.7. Rebooting the Duckiebot

**Warning:** Do not test these commands before the Duckiebot has completed its first boot. If the Duckiebot gets rebooted/shutdown while the first boot has not finished, the Duckiebot might become unreachable and you will have to reflash the SD card.

To reboot your Duckiebot, use the command,

```
$ dts duckiebot reboot hostname
```

### 3.8. Turn off the Duckiebot

To turn off your Duckiebot, use the command,

```
$ dts duckiebot shutdown hostname
```

Then wait 20 seconds.
Warning: If you disconnect the power before shutting down properly using `shutdown`, the system might get corrupted.

If you have a Duckiebot that is powered by the official Duckiebattery, e.g., DB21M, this procedure will shutdown the battery as well. This means that you do not need to manually disconnect any component from the battery. Learn more about handling the Duckiebattery in the DB21 handling instructions.

If you DO NOT have a Duckiebot that is powered by the official Duckiebattery, disconnect the USB cable from the battery.

Warning: If you disconnect frequently the cable at the computational unit’s end, you might damage the port.

Warning: (DB18 and DB19) Pressing the battery button does not shut down the power to the Duckiebot, it only activates the battery. If not in use anymore, disconnect the cables. The battery will automatically shut down if no load is detected over a period of 10 mins.
UNIT C-4
Assembly - Duckiebot DB21

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** Duckiebot DB21 parts ([get a DB21](#)). If you are unsure what version of Duckiebot you have, check the overview of existing Duckiebot configurations.
- **Requires:** A micro SD card with the Duckiebot image on it. The procedure to flash the SD card is explained [here](#).
- **Requires:** 3 hours of assembly time.
- **Results:** An assembled Duckiebot in configuration DB21.

4.1. **Foreword**

These instructions are your friend. Follow them carefully, especially if it’s the first time you assemble a Duckiebot. Small variations might cause big effects (e.g., don’t flip your cables!).

4.2. **Overview**

A Duckiebox contains the following components:
The assembly process is divided in 6 parts. They must be completed in the following order:

- **Part 1: Preliminary Steps**
- **Part 2: Drive Train**
- **Part 3: Computation Unit**
- **Part 4: Rear Assembly**
- **Part 5: Front Assembly**
- **Part 6: Top Deck Assembly**

**FAQ section**

The FAQ section at the bottom of this page provides resolutions to common problems.

### 4.3. Preliminary Steps

1) **Unboxing**
Unbox all of your components and lay them out on a flat surface. Ensure that you have well lit, uncluttered space to work on.

**Note:** “The Duckiebox hides but does not steal”. Your Duckiebot chassis might be under the white protection foam inside the box. To reach it, pull out the white foam from the box after removing everything. Mind that the upper part of the inside foam has several side pockets in addition to a main compartment where components are located.

Although not necessary, a small (M2.5) wrench might ease some passages.

**Note:** both NVIDIA Jetson Nano 2 GB and 4 GB are supported, but the sd-cards must be initialized differently: [Unit C-3 - Setup Duckiebot SD Card](#).

2) Plastic cover

Peel the plastic cover from all the chassis parts on both sides.

3) Screws, Nuts and Stand-offs

Verify each connecting part before using them. This will prevent undesirable effects (e.g., nylon screws prevent electrical shorts; bigger screws might damage the chassis).

<table>
<thead>
<tr>
<th>Picture</th>
<th>Dimension</th>
<th>Picture</th>
<th>Dimension</th>
<th>Picture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nylon screw M2x8mm</td>
<td></td>
<td>Nylon nut M2</td>
<td></td>
<td>Metal stand-off M2.5x18mm</td>
</tr>
<tr>
<td></td>
<td>Nylon screw M2.5x10mm</td>
<td></td>
<td>Nylon nut M2.5</td>
<td></td>
<td>Metal stand-off M3x25mm</td>
</tr>
<tr>
<td></td>
<td>Metal screw M3x12mm</td>
<td></td>
<td>Metal nut M3</td>
<td></td>
<td>Spacer M6x12x1.5mm</td>
</tr>
<tr>
<td></td>
<td>Metal screw M3x30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4) Charge Duckiebattery via the HUT

This preliminary step allows us to start charging the battery while confirming the functionality of the HUT.
• Connect the battery and the HUT board as shown and make sure a green LED on the HUT is lit.
• Wait 30 minutes and then push the button on the battery.
• Check that the state of charge LEDs on the battery start blinking.
• Leave this setup until the battery is charged. This may take up to 5 hours.

Figure 4.3
You can get familiar with how the Duckiebattery works by reading its handling instructions.

4.4. Base-plate
In the following steps (1 to 16) we will build the base-plate assembly of the Duckiebot.
Figure 4.4

Figure 4.5
Note: Occasionally manufacturing tolerances (on the nut and the chassis) might prevent a flush fit. Trying a different nut or changing its orientation might solve the problem.
Figure 4.19
Remove the USB cable from the Duckiebattery, connected in the battery related preliminary step.

Figure 4.20
Before proceeding, verify that no component is wiggling. The only things moving should be the cables and the sphere in the omni-wheel (and, yes, the motor axles). Proceed to gently tighten the screws of the offending parts, if necessary.

4.5. Computation Unit

The following steps (17 to 25) guide through the assembly of the *Computation* unit:

Figure 4.21
Now connect it to the base-plate (i.e, the rest of the chassis assembled in steps 1 to 16). Verify the chassis components are locked correctly.
4.6. Rear Assembly

The following steps (26 to 39) guide through the assembly of the rear part of the Duckiebot:
Figure 4.31

Figure 4.32
Figure 4.33

Figure 4.34
Figure 4.41

Figure 4.42
Figure 4.43

Figure 4.44
4.7. **Front Assembly**

Steps 40 to 52 guide through the assembly of the front bumper:
4.8. Top Deck Assembly
This last section (steps 53 to 63) guide through the assembly of the top deck:
4.9. Power your Duckiebot

One of the USB ports on the HUT will remain free. You can use this port to charge the Duckiebot. To avoid putting additional stress on the connector, you can leave this cable plugged in and store it under the blue top lid.
**Warning:** Always plug and unplug USB cables from the HUT with care!

Once your Duckiebot is fully charged, you can press the button of the battery on the side to power it up (do this ONLY if a flashed SD card has been inserted). It is important to make sure the battery is charged to prevent undesired shutdown during the first boot, which will compromise the initialization sequence and require the sd card to be re-flashed.

![Diagram of Duckiebot]

Figure 4.72

Congratulations, your Duckiebot DB21M is now completely assembled.

### 4.10. Additional Parts

These additional parts are not always necessary.

1) **Back Pattern**

The back pattern enables the traffic management behavior of the indefinite navigation demo, and it might be useful in challenges with vehicles (e.g., LFV, LFIV, etc.).
2) April Tag

This top facing April Tag enables localization in Duckietown Autolabs.
4.11. Check the outcome

- Look at the Overview of interlocking parts and make sure you have used each type at least once.
- Check all cable connectors and make sure they are plugged in completely. Do not use force on the Duckiebot, it is (almost) never useful, and it might lead to undesirable outcomes.
- Make sure you have flashed your SD card with the latest version of the Duckiebot DB21 image.

**Note:** Version 1.2.2 is the minimum requirement for enabling battery code updates. Make sure you have at least this version (>22 March 2021).
- Make sure the SD card is inserted in Jetson Nano in the dedicated SD card slot under the main board. Do not plug it in the adapter and in a USB port. If you have already inserted a flashed SD card, you are allowed to push the magic button on the battery.

4.12. Troubleshooting
**Symptom:** I can’t find the blue chassis.

**Resolution:** It’s *under* the white foam in the Duckiebox. Remove the inner packaging to access it.

**Symptom:** Camera cable needs to be twisted to make the pins on the cable matching those in the connector. Is this normal?

**Resolution:** Yes this is normal. It might look a little nicer if you wire the camera cable around the metal stand-off next to the plug.

**Symptom:** The Duckiebattery does not fit flush in the compartment.

**Resolution:** Position it as it fits (at an angle). It will make the assembly a little trickier but everything will work out in the end.

**Symptom:** I don’t have enough screws of a specific type.

**Resolution:** Each package has enough screws of each type, plus spares of some. It might happen to inadvertently use one type instead of the correct one, which will result in shortages towards the final stages. Following the instructions carefully will prevent this from happening.

**Symptom:** I can’t screw the omni-directional wheel right; the screws don’t fit all the way in the standoffs.

**Resolution:** Sometimes manufacturing inefficiencies make the thread inside the stand-off shorter than it should. This happens only occasionally, and it is not the norm. The solution is to orient, in case of need, the shorter threaded stand-off side towards above, on the side of the chassis.

**Symptom:** A piece broke while I was trying to assemble it!

**Resolution:** Mistakes happen. Some damages will not influence the functionality of the robot, others will be fixable at home with some tools, others could be showstoppers. Please take a picture of the damage and email hardware@duckietown.com.

**Symptom:** The wheels tend to fall off the motors.

**Resolution:** This is due to manufacturing tolerances. You may remove the distance disks used in the assembly between motors and wheels, but make sure that the wheels are still not touching the screws of the motor mounts. In worst cases, glue can be used to keep the wheels in place (not recommended).

**Symptom:** My Duckiebot is driving backwards when pressing the key for straight forward.

**Resolution:** You have swapped the motor cables. Double-check the motor cables are connected to their respective ports as indicated in the assembly instructions above.

**Symptom:** I don’t understand what’s going on with the connections
**Resolution:** This simplified block diagram of data and electrical connections of the DB21M might help:

![Block diagram](image)

Figure 4.79. Block diagram of electrical and data connections for the DB21 and DB21M.

**Symptom:** I followed the instruction to the letter, but there is something off I can’t quite put my finger on.

**Resolution:** You forgot to put a duckie on top of your Duckiebot!
5.1. Foreword

These instructions are your friend. Follow them carefully, especially if it’s the first time you assemble a DB21M. Small variations might cause big effects (e.g., don’t flip your cables!).

5.2. Video tutorial

Figure 5.1. DB21M: What’s in the box and assembly.

5.3. Overview

A Duckiebox contains the following components:
Figure 5.2. Overview of all parts in your Duckiebox

The assembly process is divided in 9 parts. They must be completed in the following order:

- **Part 1: Preliminary Steps**
- **Part 2: Drive Train**
- **Part 3: Battery Pack Installation**
- **Part 4: Computational Unit and Rear Assembly**
- **Part 5: Cable Management**
- **Part 6: Front Assembly**
- **Part 7: Top Plate Assembly**
- **Part 8: Power your Duckiebot**
- **Part 9: Additional Parts**
- **FAQ**

The FAQ section at the bottom of this page provides resolutions to common symptoms.
5.4. Preliminary Steps

1) Unboxing

Unbox all of your components and lay them out on a flat surface. Ensure that you have well lit, uncluttered space to work on.

**Note:** “The Duckiebox hides but does not steal”. Your Duckiebot chassis is under the white protection foam. To get to the chassis, pull out the white foam from the box after removing everything. Mind that the upper part of the inside foam has several side pockets in addition to a main compartment where components are located. Although not necessary, a small (M2.5) wrench might ease some of the next passages.

**Note:** we currently support the NVIDIA Jetson Nano 2GB developer kit, not 4GB.

2) Plastic cover

Peel the plastic cover from all the chassis parts on both sides.

3) Screws, Nuts and Stand-offs

Each type of screw, nut and stand-off is labeled with an index. You will find corresponding labels on the pictures at each step. Using the right parts at each step will prevent undesirable effects (e.g., nylon screws prevent electrical shorts).
### Figure 5.3. Overview of interlocking parts

<table>
<thead>
<tr>
<th>Index</th>
<th>Picture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td></td>
<td>Nylon screw M2x8mm</td>
</tr>
<tr>
<td>B2</td>
<td></td>
<td>Nylon screw M2.5x10mm</td>
</tr>
<tr>
<td>B3</td>
<td></td>
<td>Metal screw M3x8mm</td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td>Metal screw M3x12mm</td>
</tr>
<tr>
<td>B5</td>
<td></td>
<td>Metal screw M3x30</td>
</tr>
<tr>
<td>N1</td>
<td></td>
<td>Nylon nut M2</td>
</tr>
<tr>
<td>N2</td>
<td></td>
<td>Nylon nut M2.5</td>
</tr>
<tr>
<td>N3</td>
<td></td>
<td>Metal nut M3</td>
</tr>
<tr>
<td>N4</td>
<td></td>
<td>Nylon nut M3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Picture</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td></td>
<td>Nylon stand-off M3x5mm</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td>Metal stand-off M2.5x18mm</td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td>Metal stand-off M3x25mm</td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td>Spacer M6x12x1.5mm</td>
</tr>
<tr>
<td>S5</td>
<td></td>
<td>Spacer M2.5x5x1</td>
</tr>
</tbody>
</table>

5.5. Drive Train

1) Step 0 - Charge battery via HUT

- Connect the battery and the HUT board as shown and make sure a green LED on the HUT is lit.
- Wait 30 minutes and then push the button on the battery.
- Check that the state of charge LEDs on the battery start blinking.
- Leave this setup until the battery is fully charged. This may take up to 5 hours.
In the following steps (1 to 12) you will build the base plate assembly.
2) Step 1

Take the base plate (number 02 on the bottom side) and insert two metal M3 nuts (N3) as shown.

It might appear tricky at first to make the nuts fit. Once on the inset, gently rotate each nut until it falls in place (note the final orientation of the screw in the picture).
**Note:** Occasionally manufacturing tolerances (on the nut and the laser cut chassis) might prevent a flush fit. Trying a different nut or changing the insert direction might solve the problem.

3) **Step 2**

Check the hole pattern in the middle of the plate to make sure you are holding it the right orientation, then insert two of the motor plates in the base plate.
4) Step 3

Connect one of the 6-pin motor cable to the blue motor.
Figure 5.8

**Note:** It might be easier if you place the base plate on a flat surface so hold the motor plates. Use one hand to hold the nut, and the other to drive the screwdriver.

5) **Step 4**

Place the motor between the two plates and tighten it with two M3x30 screws (B5) and two metal M3 nuts (N3). Tighten the screws enough so that the final assembly does not wobble. Don’t use excessive force to avoid getting hurt (or braking something).
**Note:** Pre-bend the cable before carefully passing it through the cable management inset on the chassis. Don’t use force to avoid braking the inset.

6) **Step 5**

Connect the second 6-pin motor cable to the second motor.
7) Step 6

Insert the other two motor plates into the base plate, similarly to step 2.
8) Step 7

Tighten the second motor with two M3x30 metal screws (B5) and two metal M3 nuts (N3) and place the cable, similarly to steps 3 and 4.

Figure 5.12

9) Step 8

Connect one of the longest 4-pin cables to the white connector of the IMU (Inertial Measurement Unit) board.
10) Step 9

Attach the IMU board to the base plate using two nylon M2.5x10 screws (B2) and two nylon M2.5 nuts (N2).
11) Step 10

Flip the assembly and verify that the three cables are routed through their corresponding holes or slits (cable of the *left* motor through the *left* slit, cable of the *right* motor through the *right* slit).
12) Step 11
Flip the assembly again and mount the two M3x25 stand-offs (S3) to the bottom plate. Use two metal M3x8 screws (B3).
13) **Step 12**

Mount the omni-wheel to the stand-offs using two of the metal M3x8 screws (B3).
**Note:** If you note the screws don’t go all the way, try flipping the stand-off.

14) Verify the assembly

Before proceeding, verify that no component is wiggling. The only things moving should be the cables and the sphere in the omni-wheel (and, yes, the motor axles). Proceed to gently tightening the screws of the offending parts, if necessary.

Congratulations, you just built the base plate Duckiebot assembly!

### 5.6. Battery Pack Installation

The following steps (13 to 18) guide through the *Duckibattery* assembly:
1) Step 13

Take the upper plate (01) and 8 metal M3 nuts (N3). Compare the hole in the green circles with the hole position on your plate and make sure they agree (if the number 01 is visible on top, you are good to go).
2) Step 14

Insert 4 nylon M2.5x10 screws (B2) from the top and tighten them with 4 nylon M2.5 nuts (N2).
3) Step 15

Take the two plates with numbers 04L and 04R and insert three metal M3 nuts (N3) into each of them.
4) Step 16

Connect these two plates to the upper plate with a metal M3x8 screw (B3) each. Note the slightly different holes in the side plates to mount them in the right way!
5) Step 17

Insert the battery between the two side plates. Make sure the Mack the duck is on the bottom. The USB ports of the battery are not the same. Flipping the battery at this stage will cause the Duckiebot not to power on.
6) Step 18

Take the two small plates labeled 07 and lock the battery in place.

![Image of Duckiebot with labeled parts]

Figure 5.25

5.7. Computation Unit and Rear Assembly

At this point, we are starting to see the final shape of the Duckiebot! The following steps (19 to 27) will help assemble the lower frame and mount the NVIDIA Jetson Nano board:
Figure 5.26

1) Step 19

Take the lower half of the Duckiebot from steps 1 to 12 again and mount it directly to the assembly you have just created using 4 metal M3x8 (B3) screws. Make sure all the plates are locked in place correctly.
**Note:** This is a tricky step and might require a few tries to get it right. Remember that the Roboticist is patient and welcomes challenges with joy and determination!

2) **Step 20**

Check the routing path of the two motor cables again from step 10 and wire them through the holes in the upper plate.
3) Step 21

Do a similar procedure for the cable of the IMU unit.
4) Step 22

Take the two yellow driving wheels and push them to the motors using one distance disk (S4) between each of them.
5) Step 23

1. Insert the SD card to the slot on the NVIDIA Jetson Nano board. Make sure the metal pins of the SD card are facing the heat sink.
2. Place the Jetson Nano on the 4 screws from step 14 but DO NOT tighten the Jetson with any nuts or stand-offs yet!
6) Step 24

Take the side cover carrying the number 05L. Insert a nylon M2.5 nut (N2) and a metal M3 nut (N3) into the plate (note the engravings: N2 and N3 on the plate itself).

Then secure the plate to the frame using two metal M3x8 screws (B3).

You may need to lift the Jetson Nano a little to fit the side cover plate under the NVIDIA Jetson Nano board.
7) Step 25

Place 4 of the (S5) spacers on the nylon screws holding the NVIDIA Jetson Nano board.
8) Step 26

Tighten the NVIDIA Jetson Nano board with the 6 brass stand-offs (S2). Put two stand-offs on each of the front screws but only one each on the back.
9) **Step 27**

Take the side cover carrying the number 05R. Insert a nylon M2.5 nut (N2) and a metal M3 nut (N3) into the plate, similarly to step 24 (note the engraving: N2 and N3 on the plate itself). Then screw the plate to the frame using two metal M3x8 screws (B3).
5.8. Cable Management

1) Step 28

Take the USB cable that has three connectors. Connect the Duckiebattery and the NVIDIA Jetson Nano board with the USB-A ports as shown in the picture below.
Note: The micro USB connector must not be connected at that stage!

2) Step 29

Take the angled micro USB to micro USB cable and plug the angled connector to the middle port on the Duckiebattery. Wire the cable through the chassis as shown in the picture below. The micro USB end must be unplugged at that stage!

Note: An experienced Roboticist at this point would mark the free end of this cable (e.g., with a sticker or some tape), to make it distinguishable from the other free roaming cable with a micro USB connector, so to make it very improbable to mix the two up and prevent future headaches!
3) Step 30

Now take the last USB cable, with a micro USB plug on one side and an angled USB-A plug on the other side. Connect the USB-A end to the last (right) port on the Duckiebattery. Again, wire the cable through the chassis as shown and leave the micro USB connector free on the other side.
4) Step 31

Connect the Wi-Fi dongle to the upper USB-A port on the NVIDIA Jetson Nano board.
5) Step 32

Take the back bumper board and connect the 4-pin cable of medium length to the white plug on the board.
6) Step 33

Wire the cable attached to the back bumper through the same hole in the upper plate as the motor cable of the left driving motor.
7) Step 34

When attaching the back bumper board to the chassis, make sure the pins of the lower and upper plate all fit well. Tighten the board with three metal screws (B3).
8) Step 35

Take the plate carrying the number 06 and press two nylon M2.5 nuts (N2) into the corresponding slits.
9) Step 36

Mount the plate number 06 to the back of the chassis using two metal M3x8 screws (B3).

**Note:** the 06 is not symmetric and the orientation matters. If the number 06 is pointing towards the Duckiebot, we are good to go!
10) Step 37

Take the fan and mount it on top of the heat sink of the NVIDIA Jetson Nano board using 4 metal M3x12 screws (B4). Make sure the cable of the fan is pointing to the back right side (it might be necessary to use a little force on these screws, as the thread has to cut its way through the heat sink the first time).

**Note:** You don’t need to tighten the screws completely but the fan must sit tight.
11) Step 38

Take the PCB with the Duckietown logo on it (we’ll call it the HUT from now on. A DB21M is equipped with a HUT v3.1).
Plug in the fan cable to the two pins as shown (note the orientation of the black and red cables!).
12) Step 39

Gently press the pin connector of the HUT on the pins on the NVIDIA Jetson Nano board. Make sure both motor cables are routed through the slit in the HUT board.
13) Step 40

Connect the HUT to the plate in the back using two nylon M2.5x10 screws (B2). Make sure the end of the 4-pin cable connected to the back bumper board is pointing to the right hand side.
14) Step 41

Connect the 4-pin cable of the back bumper board to the white connector in the back right corner of the HUT.
15) Step 42

Connect the first 6-pin motor cable of the left motor to the connector placed on the edge of the HUT.
16) Step 43

Connect the second 6-pin motor cable of the right motor to the other connector.
Note: If you swap the motor cables, you Duckiebot will probably drive backwards when it is supposed to drive forwards :)

17) Step 44

Connect the 4-pin cable from the IMU to the corresponding plug shown in the picture.
5.9. Front Assembly
The following steps 45 to 52 will guide you through the assembly of the camera unit as well as some more electronics and cables.
1) Step 45

If not already done, open the plug of the camera and push one side of the camera cable in. The orientation of the cable should be such that the copper pins on the camera cable face the camera plate. Then close the plug completely. Make sure to take off the plastic cap from the lens.
2) Step 46

Wire the camera cable through the 3D printed camera mount starting from the front. Then use 4 nylon M2x8 screws (B1) and 4 nylon M2 nuts (N1) on the other side to tighten the camera to the mount.
3) Step 47

Mount the camera part to the front bumper board using only one metal M3x8 screw (B3) and one metal M3 nut (N3).
4) Step 48

Mount the single nylon M3x5 stand-off (S1) with a metal M3x8 screw (B3) from the other side.
5) Step 49

Take one of the longest 4-pin cables and connect it to the front bumper board as shown.
6) **Step 50**

Wire the cable that you have just connected (step 49) through the upper plate and the connect it to the connector on the HUT, as shown below.
7) Step 51

Take the last of the long 4-pin cables and connect it to the front bumper board, similarly to step 49.
8) Step 52

Again, wire this cable through the upper plate and connect it to the HUT, similarly to step 50.
9) **Step 53**

Mount the front bumper board to the upper and lower plate using three metal M3x8 screws (B3). Make sure they are locked in place correctly.
10) Step 54

Open the camera slit on the NVIDIA Jetson Nano board by raising it on the sides (with care), and put in the other end of the camera cable.

**Note:** The orientation of the cable should be such that the blue part of the cable faces the camera (i.e., facing towards the front end of the Duckiebot).
11) Step 55

Attach the small blue distance sensor to the stand-off on the front bumper and tighten it with a nylon M3 nut (N4).
Make sure to take off the small transparent cover from the sensor.

12) Step 56
Take the shortest 4-pin cable and connect the bottom of the time of flight sensor to the
front bumper, as shown below.

5.10. Top Plate Assembly
The following steps 57 to 64 show the assembly of the top plate of the DB21M, containing a button and a screen.
1) Step 57

Remove the nut from the button, if necessary, and wire the button cables through the hole on the top plate (marked as 03).

**Note:** Mind the orientation; if the number is pointing downwards, we are good to go! Once the button is pushed in completely, tighten it again with the flat nut.
Figure 5.69

2) Step 58

Mount the screen to the plate in a way the pins of the screen are pointing towards the button. Use 4 nylon M2.5 screws (B2) and 4 nylon M2.5 nuts (N2) for this.
3) Step 59

Have a look at the pin descriptions on the screen. Take the 4-pin cable with the long black connectors and connect the 4 loose ends to the screen.

Follow this pattern: GND-black, VCC-red, SCK-yellow, SDA-blue.
4) Step 60

Connect the end of the cable from the button to the connector on the HUT, as below.
5) Step 61

Connect the end of the cable from the screen to the 4 male pins on the HUT as shown. Check the colors of the cables so that the same goes to the same, i.e.: GND-black, 3.3V-red, SCL-yellow, SDA-blue.
Gently place the cover plate on the chassis. Make sure the screws of the fan and the pins of the side plates are locked in place properly.
7) Step 63

Tighten the cover part using two nylon M2.5x10 screws (B2).
8) Step 64

Then, tighten the cover using two nylon M2.5 nuts (N2).
5.11. Power your Duckiebot

In this step we will plug the various power cables to the HUT. One port will remain free. You can use this port to charge the Duckiebot.
Warning: *Always plug and unplug USB cables from the HUT with care!*

1) Step 65

Take the black USB cable that you have connected in step 29 and connect it to the micro USB connector on the HUT as shown.
2) Step 66

Similarly, connect the other USB cable (routed through the same hole) to the HUT.
3) Step 67

Finally, connect the last cable to the **HUT**.
4) Step 68

At that point, your Duckiebot is fully assembled! For charging, connect the charging cable to the last free micro USB connector on the HUT. To avoid putting additional stress on the connector, you can leave this cable plugged in and store it somewhere under the blue top lid.
Figure 5.81

Once your Duckiebot is fully charged, you can press the button of the battery on the side to power it up (do this ONLY if a flashed SD card has been inserted).
5.12. **Additional Parts**

1) **Step 69**

If you have an April tag take some glue and put it in between the two nylon screws on the top of you Duckiebot.
2) Step 70

If you have a circle pattern put it on the back plate of your Duckiebot.
5.13. Check the outcome

- Look at the Overview of interlocking parts and make sure you have used each type at least once.
- Check all cable connectors and make sure they are plugged in completely. Do not use force on the Duckiebot, it is (almost) never useful and it might lead to undesirable outcomes.
- Make sure all USB cables to the Jetson Nano and the HUT are plugged in completely, and in the correct order. Several configurations exist for which the Duckiebot will do something, but only one, described above, is the correct one.
• Make sure you have flashed your SD card with the latest version of the Duckiebot DB21M image.

**Note:** Version 1.2.2 is the minimum requirement for enabling battery code updates. Make sure you have at least this version (>22 March 2021).

• Make sure the SD card is inserted in Jetson Nano in the dedicated SD card slot under the main board. Do not plug it in the adapter and in a USB port. If you have already inserted a flashed SD card, you are allowed to push the magic button on the battery.

5.14. **Troubleshooting**

**Symptom:** Camera cable needs to be twisted to make the pins on the cable matching those in the connector. Is this normal?

**Resolution:** Yes this is normal. It might look a little nicer if you wire the camera cable around the metal stand-off next to the plug.

**Symptom:** The Duckiebattery does not fit flush in the compartment.

**Resolution:** Position it as it fits (at an angle). It will make the assembly a little trickier but everything will work out in the end.

**Symptom:** I don’t have enough screws of a specific type.

**Resolution:** Each package has enough screws of each type, plus spares of some. It might happen to inadvertently use one type instead of the correct one, which will result in shortages towards the final stages. Following the instructions carefully will prevent this from happening.

**Symptom:** I can’t screw the omni-directional wheel right; the screws don’t fit all the way in the standoffs.

**Resolution:** Sometimes manufacturing inefficiencies make the thread inside the stand-off shorter than it should. This happens only occasionally and it is not the norm. The solution is to orient, in case of need, the shorter threaded stand-off side towards above, on the side of the chassis.

**Symptom:** A piece broke while I was trying to assemble it!

**Resolution:** Mistakes happen. Some damages will not influence the functionality of the robot, others will be fixable at home with some tools, others could be showstoppers. Please take a picture of the damage and send an email to hardware@duckietown.com.

**Symptom:** The wheels tend to fall off the motors.
Resolution: You may remove the distance disks you put in step 22. But make sure that the wheels are still not touching the screws of the motor mounts.

Symptom: My Duckiebot is driving backwards when pressing the key for straight forward.

Resolution: You have swapped the motor cables. Please check the steps 42 and 43 again and make sure you connected the cables the right way.

Symptom: One of the black USB cables is too short to connect it to the HUT.

Resolution: The customized cables may undergo some manufacturing tolerances. If it does not fit, there is a second way to connect the cables. However, some minor functionalities might differ in that configuration (e.g. the fan might continue working when shutting down the NVIDIA Jetson Nano).

Figure 5.85

Symptom: I don’t understand what’s going on with the connections

Resolution: This simplified block diagram of data and electrical connections of the DB21M might help:
Figure 5.86. *DB21M* block diagram of electrical and data connections.

**Symptom:** I followed the instruction to the letter, but there is something off I can’t quite put my finger on.

**Resolution:** You forgot to put a duckie on top of your Duckiebot!
UNIT C-6
Handling - Duckiebot DB21, DB21M

KNOWLEDGE AND ACTIVITY GRAPH

**Requires:** An assembled DB21 or DB21M. Find the assembly instructions [here](#).

**Requires:** An initialized DB21 or DB21M with image version at least 1.2.2. Find the initialization instructions [here](#). Check your current firmware version before proceeding.

**Results:** Knowledge on standard protocols to turn on, turn off, charge, and update the Duckiebattery software version on a DB21 or DB21M.

**Note:** the above box contains important information on the requirements. Make sure to read and follow them before proceeding.

6.1. Duckiebot DB21M handling tutorial video

![Figure 6.1. Duckiebattery power on, shutdown and charging protocols.](image)

6.2. How to charge a DB21M

To charge your Duckiebot, follow these steps:

- Plug in the charging cable to the free microUSB port on the HUT.

**Note:** to minimize mechanical stress on the HUT we recommend plugging in the
charging cable once, and leaving the USB port end free to plug and unplug from charging instead. You can arrange the cable under the DB21M top plate during operations for cable management.

- Plug in the charger to a 5V 2A source.

**Note:** the battery can draw up to 2A. Feeding a higher amperage will not be a problem, but wrong voltage will send the battery in protection mode.

- If the Duckiebot is turned on when charging, a battery charge indicator will appear on the top right of the screen. If the Duckiebot is turned off, the LEDs will turn on. In both cases, a small LED on the HUT near the charger port will turn green, indicating incoming power.

### 6.3. How to power off a DB21M

**Warning:** The proper shutdown protocol for a DB21M requires having the Duckiebattery software version 2.0.0 or later. To check the version of your battery, follow the instruction to “Verify current battery version” on How to update a Duckiebattery.

Make sure the Duckiebot has completed the booting process. You can verify this by checking the “Status” after running dts fleet discover on your laptop: a green Ready message will indicate that the Duckiebot has completed the booting process.

**Note:** There are three methods to power off a DB21M (recommended method: “With the top button”):

1. With the **top** button:
   
   a. Press the **top** button (not the battery button) for 5 seconds and release.
   b. What to expect:
      
      a. The user should see the top button blinks for 3 seconds
      b. the Duckiebot front and back LEDs should be turned off
      c. Then in about 10 seconds, the Jetson and fan should stop.
   
   c. Troubleshooting: If the display just switched to the next page and the top button did not blink, try again and push harder on the top button during the 5 seconds.

2. With **duckietown-shell**:
   
   a. dts duckiebot shutdown `hostname`
   b. What to expect:
a. The Jetson and fan should stop in about 10 seconds.

b. If the charging cable is not attached, the front and back LEDs should also be turned off.

3. With the Duckiebot dashboard:
   a. Open a browser
   b. Navigate to http:// hostname.local
   c. In the Top-Right corner, click on the Power options, and choose “Shutdown”. Then confirm the action.
   d. What to expect: the same as the “With duckietown-shell” method.

**Warning:** The following “hard” power shutdown should be only be used if the three methods above failed to shut down the Duckiebot, because it might lead to software and hardware issues.

As a last resort, one could still perform a “hard” power shutdown of the DB21M: ssh duckie@ hostname.local sudo poweroff - unplugging the micro USB cable from the port marked as 5Vraspi on the HUT.

### 6.4. How to power on a DB21M

To power on a Founder Edition Duckiebot, press the button on the battery once.

The Duckiebot LEDs, as well as the Jetson Nano board booting LED will turn on.

After a few seconds, the Wi-Fi dongle will start blinking. The Duckiebot LEDs will then turn to a steady white color, followed by the button and screen on the top plate powering on, as shown in the tutorial video.

### 6.5. How to update a Duckiebattery

To update the software running on the micro-controller in the Duckiebattery, or just checking the current version of it, follow this procedure.

When reporting issues on Stack Overflow, please include the step number, e.g. Step 4.i.b, the actions performed, and a description of the unexpected/unknown outcome.

**Important:**

1. Before the battery upgrade, please make sure the battery has at least 15% of charge.
2. Run all following commands on the desktop/laptop

Make sure the Duckiebot is powered on and connected to the network. You can verify
the latter by launching, e.g., `dts fleet discover` and finding that your Duckiebot is on the list.

All following `hostname` refers to the name of the Duckiebot to which the battery is plugged in.

1. Please update the `duckietown-shell` utility:
   a. `pip3 install --user --upgrade --no-cache-dir duckietown-shell`
   b. `dts update`
   c. `dts desktop update`

2. Update the Duckiebot:
   a. `dts duckiebot update hostname`

3. Reboot the Duckiebot:
   a. `ssh duckie@hostname.local sudo reboot`
   b. Wait until the Duckiebot reboots and the display shows information (especially about the battery).
   c. You could verify the battery related software is up and running by checking whether the display reacts correctly to charging states when a charging cable is plugged in and unplugged.

4. Upgrade the battery firmware:
   a. `dts duckiebot battery upgrade hostname`
      a. Note: When prompted to “double-click” on the battery button, make sure to quickly click twice the battery button.
      b. Note: Do not worry if you are unsure if you actually pressed the button twice or not, as the battery upgrade process will verify this.
      c. Follow the instructions in the terminal.
   b. If the command finished with the error: `SAM-BA operation failed INFO:UpgradeHelper:An error occurred while flashing the battery. ERROR:dts:The battery reported the status 'GENER-IC_ERROR'`, please try flashing again with: `dts duckiebot battery upgrade --force hostname`
   c. If the command finished with any other error: single press the battery button, and start from step 3 again one more time. If there are still errors, please report on StackOverflow.
5. Prepare for post-upgrade checks
   a. If the battery indicates the charging states correctly, and shows the percentage number normally, proceed to step 6
   b. If the display shows “NoBT” (No battery detected), then single press the battery button, and run:
      a. `ssh duckie@hostname .local sudo reboot`
      b. Wait for the reboot (as described in step 3)
      c. Then proceed to step 6

6. Verify current battery version:
   a. Method 1:
      a. `dts duckiebot battery check_firmware hostname`
      b. Verify the battery version should be "2.0.2"
   b. Method 2:
      a. Open a browser window
      b. Navigate to `http://hostname .local/health/battery/info`
      c. Verify the battery version should be "2.0.2"

6.6. How to update a HUT
Instructions on how to flash a Duckietown HUT board can be found here.

Note: (re)flashing a HUT is typically not needed. A notable exception is for HUT version 3.15 which comes with DB21s. The HUT version can be read on the board itself.
UNIT C-7
Setup - Dashboard

KNOWLEDGE AND ACTIVITY GRAPH

| Requires: Laptop configured, according to Unit C-1 - Setup - Laptop. |
| Requires: You have configured the Duckiebot as documented in Unit C-3 - Setup Duckiebot SD Card. |

This section shows how to install the Duckietown Dashboard on the Duckiebot.

7.1. The \compose\ platform
\compose\ is a CMS (Content Management System) platform that provides functionalities for fast-developing web applications. Custom applications are developed as external packages that can be installed using the built-in Package Store.

The Duckiebot Dashboard is a package that you can install on your instance of \compose\ running on your Duckiebot. To make it easier for you to get started, we provide a Docker image with \compose\ and all the packages you need. Follow the instructions in the next step to get started.

Visit the official documentation page for further information about \compose\.

7.2. Setting up dashboard

1) Video Tutorial

Figure 7.1. Dashboard setup tutorial.
2) Step-by-Step Instructions

You can find your duckietown dashboard at:

http://YOUR_DUCKIEBOT_NAME.local/

If the above address does not work, remove the .local part and just do http://YOUR_DUCKIEBOT_NAME/

**Note:** if .local does not work, that means your router’s default domain name is set to something else. It will be helpful if you figure out what that is. And keep in mind that any instruction later that includes .local should be just ignored.

Steps 1, 2:

By default, \compose\ uses Google Sign-In to authenticate the users. In Duckietown, we use authentication based on personal tokens. You should be able to retrieve yours by visiting the page:

https://www.duckietown.org/site/your-token

You should notice that the first two steps already appear to be completed. Do not worry about creating an administrator account (Step 2) for now, a new administrator account will be automatically created the first time we login using a Duckietown token.

Step 3:

**Step 3** is about configuring the dashboard.
You can complete this step as you please. Feel free to update all the fields, and remember, you can always update your choices through the page **Settings** after you authenticate using your personal token.

When you are happy with your choices, click on **Next**.

**Step 4:**

The **Step 4: Complete** tab should now be open, as shown below.
You can go ahead and press **Finish**.

_**First Login:**_

If everything went as planned, the dashboard is now configured and ready to go! You should be able to see the login page, as shown below.
**Note:** Since your dashboard does not have an administrator account yet, the first user to login will be automatically assigned the role of administrator. If you have multiple tokens, make sure to keep note of which one you used for the first login.

If you have not retrieved your personal Duckietown Token as described in Unit C-2 - Setup - Account yet, it is now time to do so. Once you have your personal Duckietown token, go ahead and click on the button **Sign in with Duckietown**. You should now see a dialog like the one shown below,
Copy/paste or type in your personal token and press Login. Wait for the token to be validated, and if your token is correct, you will be redirected to your profile page, similar to the one shown below.
As you might have noticed, the side bar to the left now shows many more pages. Some pages are accessible by all users (e.g., Robot), others only by administrators (e.g., Settings, Package Store).

Take your time to visit all the pages and get comfortable with the platform. We will discuss the functionalities offered by each page in the next sections.
UNIT C-8
Operation - Use the Dashboard

KNOWLEDGE AND ACTIVITY GRAPH

Requires: Laptop configured, according to Unit C-1 - Setup - Laptop.
Requires: You have configured the Duckiebot as documented in Unit C-3 - Setup Duckiebot SD Card.
Requires: You have completed the Dashboard setup as documented in Unit C-7 - Setup - Dashboard.

This section shows how to use the Duckietown Dashboard on the Duckiebot.

8.1. What is in the Dashboard?
The following video provides a brief tour of the most important features of the Duckietown Dashboard on your Duckiebot.

Figure 8.1. Dashboard operation tutorial.

To see all the available components within the dashboard, you will need to first login to the dashboard. Inside the dashboard, you will see a navigation panel on your left hand side. There are 7 subpages of dashboard. They are:
8.2. **Portainer**

Portainer is a provided tool for managing all the docker containers that are running on the Duckiebot. Using portainer tools, you can quickly see the status of the containers on your Duckiebot.
You can select containers to see all the containers on the Duckiebot.

For more information about portainer, you can find them in this page.

8.3. Robot Page

In this page you will find several tabs that help you see and understand the Duckiebot status. The default tab is Info.

1) Info

In this tab, you can find information for your robot, including your robot name, type, configuration, and critical information such as CPU usage, temperature, and other crucial robot vitals.
Note: from this page you can read the Duckiebot’s firmware version, i.e., the version of the base image used during initialization. This might inform if a re-initialization is needed, e.g., to perform the DB21 battery update.

2) Mission Control

This is the Mission Control tab.
In this tab you can see what the Duckiebot sees and you can see lateral and angular speed of your robot, and a plot of left and right motor speed. This is the tab that lets you monitor and control your Duckiebot. The top of the page should be similar to the following image,

**Note:** If you do not see the camera view, make sure you are accessing the dashboard using `https://ROBOT_HOSTNAME.local/` instead of directly accessing the dashboard using robot IP address.

**Did you know?** The page contains 4 blocks by default. Feel free to drag them around and rearrange them as you please. You can also use the menu button of each block to resize them.

3) **Health**

This is the Health Page. It will show you a plot of the robot’s health status such as temperature, frequency, and CPU usage. It is a good debug tool to watch your code’s
This is the Architecture Page. It will allow you to visualize all the published ROS topics and see their details. It is a useful tool to see what is running and what is not. You can also use this tool as a replacement of `rqt-graph`. For more instructions on rqt-graph, you can see it [here](#).
Figure 8.6
UNIT C-9

Operation - Make it move

KNOWLEDGE AND ACTIVITY GRAPH

**Requires:** A Duckiebot in DB18 or later configurations.

**Requires:** Laptop configured according to [Unit C-1 - Setup - Laptop](#).

**Requires:** You have configured the Duckiebot as documented in [Unit C-3 - Setup Duckiebot SD Card](#).

**Results:** You can make your Duckiebot move.

This section describes how to make your Duckiebot move.

### 9.1. Keyboard control

The easiest way to move a Duckiebot is by keyboard control. This video shows how to drive a Duckiebot using the keyboard, through the Duckietown Shell.

![Duckiebot keyboard control](image)

**Figure 9.1.** Duckiebot keyboard control.

1) **Duckietown Shell**

To move your Duckiebot using your computer's keyboard open a terminal and run:

```
$ dts duckiebot keyboard_control ROBOT_NAME
```

which, after startup, will open an arrows interface window:
0x0

Figure 9.2. The keyboard control graphical user interface

**Note:** input Duckiebot ![hostname], do not include .local part.

The following keys control the Duckiebot:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARROW_KEYS</td>
<td>Steer your Duckiebot</td>
</tr>
<tr>
<td></td>
<td>Quit</td>
</tr>
<tr>
<td>q</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>Turn on Lane Following</td>
</tr>
<tr>
<td>s</td>
<td>Stop Lane Following</td>
</tr>
<tr>
<td>i</td>
<td>Toggle Anti-instagram</td>
</tr>
</tbody>
</table>

The a, s, i function require the lane following demo to be running.

**Warning:** This does not currently work on Mac OSX.

2) The no-window way with Duckietown shell (For Mac Users)

For some reason messages published on Mac inside the container don’t make it all the way to the robot.

When the popup arrow window is not responsive, running the stack directly on the Duckiebot might help:

```bash
$ dts duckiebot keyboard_control [hostname] --cli
```

### 9.2. Troubleshooting

**Symptom:** Duckiebot goes backwards, even though I command it to go forward.

**Resolution:** If you have a DB17 or DB18, revert the polarities (plus and minus cables) of the cables that go to the motor driver (HUT) for both motors.
**Symptom:** The robot doesn’t move

**Resolution:** Check that the `duckiebot-interface` container is running

Open the Portainer interface and check the running containers. You should see one called `dt18_03_roscore_duckiebot-interface_1`. You can also determine this by running:

```
$ docker -H ROBOT_NAME local ps
```

and look at the output to find the Duckiebot interface container and verify that it is running.

**Resolution:** One of the base images is out of date Update your Duckiebot with the command

```
$ dts duckiebot update ROBOT_NAME
```

**Symptom:** Everything seems fine, I can see the commands being sent to the Duckiebot (e.g., through the Dashboard > Mission Control), but the Duckiebot does not move. My Dashboard > Robot > Components page show a red alert for the HUT.

**Note:** if you have a HUT v3.15 you will stumble on this problem the first time you try to move your Duckiebot.

**Resolution:** Re-flash your HUT following the procedure described in Unit C-20 - Debug - Re-flash Microcontroller.
UNIT C-10
Operation - Make it see

KNOWLEDGE AND ACTIVITY GRAPH

- **Requires:** A Duckiebot in DB18 or above configuration.
- **Requires:** Laptop configured, according to Unit C-1 - Setup - Laptop.
- **Requires:** You have configured the Duckiebot as documented in Unit C-3 - Setup Duckiebot SD Card.
- **Results:** You can see the output of the camera.

This section describes how to see what your Duckiebot sees.

10.1. Check the camera hardware
If you have a DB18 or DB19, it might be useful to do a quick camera hardware check as documented in Section 16.6 - Camera.

10.2. Viewing an Image Stream on Your Laptop
The imagery is streaming from your Duckiebot by default on startup. To see it, open a terminal on your laptop and run:

```bash
$ dts start_gui_tools DUCKIEBOT_NAME
```

**Warning:** Note that in here you input Duckiebot ![hostname], do not include .local part.

This will start a container with access to the ROS messages of the Duckiebot. At the prompt, run:

```bash
$ rqt_image_view
```

The command will open a window where you can view the image.

You have to select the right topic from the drop-down menu:
1) Troubleshooting

**Symptom:** I see a black image like this:

![Black Image](image1.png)

**Resolution:** Remove the cap of the camera.

**Symptom:** When I try to do `rqt_image_view`, I don’t see the window on my machine.

**Resolution:** Sometimes the window does not successfully spawn on the first try. You can `Ctrl + C` to terminate the process before trying again.

**Symptom:** `libGL error` when running `dts start_gui_tools`

**Resolution:** If you have an error like that when running `dts start_gui_tools` or another command with a GUI on the Duckiebot, then you are likely having issues with an NVIDIA graphics card:

```
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast nvidia docker
```
This could occur on a computer that has two graphics cards, e.g., a NVIDIA GPU and an integrated Intel card. Switch to the Intel card by following the official guidelines for your OS and graphics card.

**Symptom:** I don’t see any image.

**Resolution:** use `rostopic hz /hostname/camera_node/image/compressed` and see if the image is being published. Images should be published at roughly 30 Hz.

⋆ For more information about `rostopic`, see Unit C-11 - Operation - Using `start_gui_tools, ROS tools and no-vnc`.

⋆ you can see the images as your robot sees them with `rostopic echo /hostname/camera_node/image/compressed`. `Ctrl+c` on the terminal once you’ve seen enough.

**Symptom:** My image topic is not being published.

**Resolution:** Check that the `duckiebot-interface` is running

Open the Portainer interface and check the running containers. You should see one called `duckiebot-interface`, using image `duckietown/dt-duckiebot-interface:daffy-arm32v7`.

You can also determine this by running:

```
$ docker -H DUCKIEBOT_NAME.local ps
```

and look at the output to find the Duckiebot interface container and verify that it is running.

If that image is not running, you should manually start that or check to see if you `init_sd_card` procedure was correct.

To manually start the `duckiebot-interface`, do:

```
$ docker -H DUCKIEBOT_NAME.local run --name duckiebot-interface -v /data:/data --privileged --network=host -dit --restart unless-stopped duckietown/dt-duckiebot-interface:daffy-arm32v7
```

**Symptom:** The camera is not detected from Duckiebot

**Resolution:** (DB18, DB19 only) remove the battery pack and check the camera cable for damage.

**Symptom:** The images are out of focus.
The camera focus can be *manually* adjusted by rotating the lens of the image sensor. As always with dealing with hardware, exercise care and do not use force.

**Symptom:** I cannot rotate the lens and change the camera focus.

**Resolution:** You need to break the glue. Very occasionally cameras come with the lens glued in place. Apply a bit more force the first time you adjust the lens to break the glue’s adhesion.

### 10.3. Viewing the image in no-vnc

For instructions using the no-vnc tool, see [here](#).

### 10.4. Viewing the image stream on the Dashboard

If you followed the instructions in [Unit C-7 - Setup - Dashboard](#), you should have access to the Duckiebot dashboard.

Open the browser and visit the page `http://hostname.local/`. Login using your duckietown token, and select robot panel on the left hand side navigation bar. Once selected you should see mission control page there. If you are unfamiliar with the dashboard, you can find more information here: [Section 8.1 - What is in the Dashboard?](#)

The bottom of the page shows the camera block. You should be able to see the camera feed in the camera block, as shown in the image below.

![Camera Feed](image.png)

**Figure 10.3**

By default, the camera stream is throttled down to 8 frames per second. This is to min-
imize the resources used by your browser while streaming images from the robot. Feel free to increase the data stream frequency in the Properties tab of the camera block.

**Note:** If you see a black image in the camera block, make sure that you removed the protective cap that covers the camera lens of your Duckiebot.
11.1. A note on start_gui_tools

If you are very familiar with how ROS works and like command line interface, you can run:

```bash
$ dts start_gui_tools DUCKIEBOT_NAME
```

to obtain a terminal (container actually) that is connected to the Duckiebot ROS network. In this terminal you can perform all ROS commands.

**Note:** You can only start one instance of `start_gui_tools` container. If you want multiple terminal instances, it is recommended to use `no-vnc`.

**Note:** You can add the AND Symbol at the end of your command to run it in the background.

11.2. Starting no-vnc images

To start a image that runs `no-vnc`, type:

```bash
$ dts start_gui_tools --vnc DUCKIEBOT_NAME
```

**Warning:** Note that in here you input the Duckiebot hostname; do not include `localhost` part.

To use `no-vnc`, use your browser and navigate to:

```
http://localhost:8087/
```

You can treat this environment as a typical Ubuntu machine with ROS installed, and configured to talk with your Duckiebot.
11.3. **Verifying the output by using the ROS utilities and command line**

Open up a terminal and use the commands below to check the data streams in ROS.

1) **List topics**

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

```bash
$ rostopic list
```

* For more information about `rostopic`, see Section 1.5 - rostopic.

You should see at least the following topics:

```
/hostname/camera_node/camera_info
/hostname/camera_node/image/compressed
/rosout
/rosout_agg
```

There might be other topics if you started other demos.

2) **Show topics frequency**

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

```bash
$ rostopic hz /hostname/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
```

Use `Ctrl-C` to stop `rostopic`.

3) **Show topics data**

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

```bash
$ rostopic echo /hostname/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.

11.4. rqt_image_view tool
To see what your Duckiebot sees, you can click on the RQT Image View application icon on the desktop. You will see the rqt_image_view starts up

![The rqt image view window with dropdown menu](image)

Figure 11.1. The rqt image view window with dropdown menu

11.5. rqt_graph tool
If you want to explore the relationship between all the nodes, topics and tf, you can open up a terminal and run:

```
$ rqt_graph
```

This will open up a window that contains all the ROS topics being published, all the ROS nodes running, and it is a very handy tool to understand the relationship between nodes.

11.6. ROS Troubleshooting

**Symptom:** My ros commands are not working. I cannot use tab to auto complete ROS commands.

**Resolution:** You can fix that by sourcing devel/setup.bash.

```
$ source /code/catkin_ws/devel/setup.bash
```

**Symptom:** I cannot connect to ROS master.
Resolution: Go to protainer of the robot to make sure the following containers are running without errors:

- ROS
- car-interface
- duckiebot-interface

If they are not running, refer to `docker troubleshooting` to make sure those containers are running.
UNIT C-12

Calibration - Camera

This section describes the intrinsics and extrinsics calibration procedures.

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** You can see the camera image on the laptop. The procedure is documented in [Unit C-10 - Operation - Make it see](#).
- **Results:** That your camera intrinsics and extrinsics are calibrated and stored on the Duckiebot.

12.1. Materials

If you do not have one already, download and print a PDF of the calibration checkerboard:

- **A3-format.**

![Checkerboard Image](image)

Figure 12.1

**Note:** the squares must have side equal to 0.031 m = 3.1 cm. Please measure this, as having the wrong size will make your Duckiebot crash.

**Note:** In case your squares are not of the right size, make sure your printer settings are on A3 format, no automatic scaling, 100% size.
Fix the checkerboard to a rigid planar surface that you can move around.

Warning: If the pattern is not rigid the calibration will be useless. You can print on thick paper or adhere to something rigid to achieve this.

1) Optional material

You will also need a “lane” during the extrinsics calibration procedure. This is not necessary.

12.2. Intrinsic Calibration

Every camera is a little bit different so we need to do a camera calibration procedure to account for the small manufacturing discrepancies. This process will involve displaying a predetermined pattern to the camera and using it to solve for the camera parameters. For more information on the details see the slides. The procedure is basically a wrapper around the ROS camera calibration tool.

1) Launch the intrinsic calibration application

Next you can launch the intrinsic calibration program with:

```
$ dts duckiebot calibrate_intrinsics hostname
```

You should see a display screen open on the laptop (Figure 12.2).
Note: If you only see a window with black screen during the startup, try to resize the window manually using cursor, and you should see the window content correctly.

2) Calibration dance

Position the checkerboard in front of the camera until you see colored lines overlaying the checkerboard. You will only see the colored lines if the entire checkerboard is within the field of view of the camera.

You should also see colored bars in the sidebar of the display window. These bars indicate the current range of the checkerboard in the camera’s field of view:

- X bar: the observed horizontal range (left - right)
- Y bar: the observed vertical range (top - bottom)
- Size bar: the observed range in the checkerboard size (forward - backward from the camera direction)
- Skew bar: the relative tilt between the checkerboard and the camera direction

Also, make sure to focus the image by rotating the mechanical focus ring on the lens.
of the camera.

**Warning:** Do not touch the focus anymore, ever, as it will invalidate calibration.

Now move the checkerboard right/left, up/down, and tilt the checkerboard through various angles of relative to the image plane. After each movement, make sure to pause long enough for the checkerboard to become highlighted. Once you have collected enough data, all four indicator bars will turn green. Press the “CALIBRATE” button in the sidebar.

Calibration may take a few moments. Note that the screen may dim. Don’t worry, the calibration is working.

![Image of calibration process](image.png)

Figure 12.3

3) **Save the calibration results**

If you are satisfied with the calibration, you can save the results by pressing the “COMMIT” button in the side bar. (You never need to click the “SAVE” button.)
This will automatically save the calibration results on your Duckiebot:

```
/data/config/calibrations/camera_intrinsic/hostname.yaml
```

You can view or download the calibration file using the Dashboard running at http://hostname.local under File Manager in the sidebar on the left, navigating to config/calibrations/camera_intrinsic/hostname.yaml.

4) Keeping your calibration valid

Warning: Do not change the focus during or after the calibration, otherwise your calibration is no longer valid.

Warning: Do not use the lens cover anymore; removing the lens cover may change the focus.

12.3. Extrinsic Camera Calibration
1) Setup

Arrange the Duckiebot and checkerboard according to Figure 12.5. Note that the axis of the wheels should be aligned with the y-axis.

![Figure 12.5](image)

Figure 12.5

Figure 12.6 shows a view of the calibration checkerboard from the Duckiebot. To ensure proper calibration there should be no clutter in the background.
2) **Launch the extrinsic calibration pipeline**

**Run:**

```
$ dts duckiebot calibrate_extrinsics hostname
```

First the output will instruct you place your robot on the calibration box and press Enter. If all goes well the program will complete.

This will automatically save the calibration results on your Duckiebot:

```
/data/config/calibrations/camera_extrinsic/ hostname.yaml
```

You can view or download the calibration file using the Dashboard running at http://hostname.local under File Manager in the sidebar on the left, navigating to config/calibrations/camera_extrinsic/ hostname.yaml.

**Troubleshooting:**

**Symptom:** You see a long complicated error message that ends with something about
findChessBoardCorners failed.

Resolution: Your camera is not viewing the full checkerboard pattern. Most likely part of the chess board pattern is occluded. Possibly you didn’t assemble your Duckiebot correctly or you did not put it on the calibration pattern properly.
UNIT C-13
Calibration - Wheels

KNOWLEDGE AND ACTIVITY GRAPH

Requires: You can make your robot move as described in Unit C-9 - Operation - Make it move.

Results: Calibrate the wheels of the Duckiebot such that it goes in a straight line when you command it to. Set the maximum speed of the Duckiebot.

For the theoretical treatment of the odometry calibration see Unit E-2 - Odometry Calibration.

13.1. Step 1: Make your robot move
Follow instructions in Unit C-9 - Operation - Make it move to make your robot movable either with a joystick or the keyboard.

13.2. Step 2: Get a base container with a command line
Now you need another container to run so that you can edit the calibrations and see the results. To get a base container with a command line you can run:

```
$ dts start_gui_tools hostname
```

13.3. Step 3: Perform the calibration

1) Calibrating the trim parameter
The trim parameter is set to 0.00 by default, under the assumption that both motors and wheels are perfectly identical. You can change the value of the trim parameter by running the command:

```
$ rosparam set /hostname/kinematics_node/trim trim value
```

Use some tape to create a straight line on the floor (Figure 13.1).
Place your Duckiebot on one end of the tape. Make sure that the Duckiebot is perfectly centered with respect to the line.

Command your Duckiebot to go straight for about 2 meters. Observe the Duckiebot from the point where it started moving and annotate on which side of the tape the Duckiebot drifted (Figure 13.2).

Measure the distance between the center of the tape and the center of the axle of the
Duckiebot after it traveled for about 2 meters (Figure 13.3). Make sure that the ruler is orthogonal to the tape.

Figure 13.3. Measure the amount of drift after 2 meters run

If the Duckiebot drifted by less than 10 centimeters you can stop calibrating the trim parameter. A drift of 10 centimeters in a 2 meters run is good enough for Duckietown. If the Duckiebot drifted by more than 10 centimeters, continue with the next step.

If the Duckiebot drifted to the left side of the tape, decrease the value of $r$, by running, for example:

```
$ rospack set /hostname/kinematics_node/trim -0.1
```

If the Duckiebot drifted to the right side of the tape, increase the value of $r$, by running, for example:

```
$ rospack set /hostname/kinematics_node/trim 0.1
```

Repeat this process until the robot drives straight
2) Calibrating the **gain** parameter

The gain parameter is set to 1.00 by default. You can change its value by running the command:

```
$ rosparam set /hostname/kinematics_node/gain gain value
```

3) Store the calibration

When you are all done, save the parameters by running:

```
$ rosservice call /hostname/kinematics_node/save_calibration
```

The first time you save the parameters, this command will create the file

4) Final check to make sure it’s stored

The calibration result is saved on your Duckiebot:

```
/data/config/calibrations/kinematics/ hostname.yaml
```

You can view or download the calibration file using the Dashboard running at `http://hostname.local` under File Manager in the sidebar on the left, navigating to `config/calibrations/kinematics/ hostname.yaml`. 
UNIT C-14

Operation - Taking and verifying a log

KNOWLEDGE AND ACTIVITY GRAPH

**Requires:** Unit C-9 - Operation - Make it move

**Requires:** Unit C-10 - Operation - Make it see

**Results:** A verified log.

14.1. Preparation

**Note:** You may not have the folder. SSH into your robot and execute:

```
$ sudo mkdir /data/logs
```

**Note:** It is recommended but not required that you log to your USB and not to your SD card.

→ To mount your USB see [here](#).

14.2. Record the log

1) **Option: Minimal Logging on the Duckiebot**

```
$ dts duckiebot demo --demo_name make_log_docker --duckiebot_name DUCKIEBOT_NAME --package_name duckietown_demos
```

This will only log the imagery, camera_info, the control commands and a few other essential things.

2) **Option: Full Logging on the Duckiebot**

To log everything that is being published, run the base container on the Duckiebot:
14.3. Stop logging

You can stop the recording process by stopping the container:

```
$ docker -H DUCKIEBOT_NAME.local stop demo_make_log_docker
```

or demo_make_log_full_docker as the case may be. You can also do this through the portainer interface.

14.4. Getting the log

1) Download through browser (Recommended)

Using the dashboard file tab, you can access files on your Duckiebot.

Go to /logs and you should see all your logs there. Simply click on the log you want to transfer to your computer and it will download through your browser.

If for some reason you are having trouble accessing dashboard, you can directly specify the webpage at http://hostname.local:8082/logs/.

2) Using a USB drive

If you mounted a USB drive, you can unmount it and then remove the USB drive containing the logs (recommended).

- For unmounting instructions see here

3) Using SCP

Otherwise you can copy the logs from your robot onto your laptop. Assuming they are on the same network execute:

```
$ scp linux_username@hostname.local:/data/logs/* path-to-local-folder
```

You can also download a specific log instead of all by replacing * with the filename.

```
14.5. Verify a log

**Note:** This procedure requires ROSBAG to be installed. If you have not installed that already, you can do so via:

```
$ sudo apt-get install python3-rosbag
```

Either copy the log to your laptop or from within your container do

```
$ rosbag info FULL_PATH_TO_BAG --freq
```

Then:

- verify that the “duration” of the log seems “reasonable” - it’s about as long as you ran the log command for
- verify that the “size” of the log seems “reasonable” - the log size should grow at about 220MB/min
- verify in the output that your camera was publishing very close to **30.0Hz** and verify that your virtual joysick was publishing at a rate of around **26Hz**.

An example of the output looks like this:

```bash
path: avlduck2_2020-08-05-01-54-18.bag
version: 2.0
duration: 12.7s
start: Aug 04 2020 21:54:18.73 (1596592458.73)
end: Aug 04 2020 21:54:31.42 (1596592471.42)
size: 69.9 MB
messages: 756
compression: none [77/77 chunks]
types: sensor_msgs/CameraInfo
c9a58c1b0b154e0e6da7578cb991d214
  sensor_msgs/CompressedImage
8f7a12909da2c9d3332d540a0977563f
  topics: /avlduck2/camera_node/camera_info 374 msgs @ 29.9 Hz : sensor_msgs/CameraInfo
     /avlduck2/camera_node/image/compressed 382 msgs @ 29.8 Hz : sensor_msgs/CompressedImage
```
UNIT C-15

Assembly - Duckiebot DB17

This page is for the DB17 configuration used in classes in 2017. The docker based software stack of 2018 is currently not guaranteed to work out of the box with the DB17 hardware configurations.

**Knowledge and activity graph**

- **Requires:** Duckiebot DB17 parts. The acquisition process is explained in Section 3.3.1 - DB17 Bill of materials.
- **Requires:** A microSD card with the Duckiebot image already on it. This procedure is explained [here](#).
- **Requires:** Time: about 1-1.5 hours (45 minutes for an experienced Duckiebot builder).
- **Results:** An assembled Duckiebot in configuration DB17.

There are several steps in this procedure. Many steps rely on previous ones, so make sure you perform them in order!

- **Part 1:** Assemble the DB17-jwd
- **Part 2:** Solder the boards
- **Part 3:** Prepare the power cable for DB17-jwd
- **Part 4:** Assemble the bumpers for DB17-l
- **Part 5:** Assemble the DB17-l

### 15.1. Assembly instructions (DB17-jwd)

Once you have received the parts and soldered the necessary components, it is time to assemble them in a Duckiebot. Here, we provide the assembly instructions for the configuration DB17-wjd.

**Knowledge and activity graph**

- **Requires:** Duckiebot DB17-wjd parts. The acquisition process is explained in Section 3.3 - Acquiring the parts for a DB17.
Requires: Having soldered the DB17-wjd parts. The soldering process is explained in Section 15.2 - Assembly instructions (DB17): soldering.

Requires: Having prepared the power cable. The power cable preparation is explained in Subsection 15.2.5 - Assembly instructions (DB17-jwd): power cable. Note: Not necessary if you intend to build a DB17-l configuration.

Results: Time: about 30 minutes.

Results: An assembled Duckiebot in configuration DB17-wjd.

Note: The FAQ section at the bottom of this page may already answer some of your comments, questions or doubts.

This section is comprised of 14 parts. Each part builds upon some of the previous parts, so make sure to follow them in the following order.

- **Part I: Motors**
- **Part II: Wheels**
- **Part III: Omni-directional wheel**
- **Part IV: Chassis standoffs**
- **Part V: Camera kit**
- **Part VI: Heat sinks**
- **Part VII: Raspberry Pi 3**
- **Part VIII: Top plate**
- **Part IX: USB Power cable**
- **Part X: DC Stepper Motor HAT**
- **Part XI: Battery**
- **Part XII: Upgrade to DB17-w**
- **Part XIII: Upgrade to DB17-j**
- **Part XIV: Upgrade to DB17-d**

1) **Motors**

Open the Magician Chassis package and take out the following components:

- Chassis-bottom (1x)
- DC Motors (2x)
- Motor holders (4x)
- M3x30 screw (4x)
- M3 nuts (4x)

Figure 15.1 shows the components needed to complete this part of the tutorial.

Figure 15.1. Components needed to mount the motors.

Video tutorial:
The following video shows how to attach the motors to the bottom plate of the chassis.

Figure 15.2
2) Step-by-step guide

**Step 1:**
Pass the motor holders through the openings in the bottom plate of the chassis as shown in Figure 15.3.

![Figure 15.3. The sketch of how to mount the motor holders.](image)

**Step 2:**
Put the motors between the holders as shown in Figure 15.4.
Figure 15.4. The sketch of how to mount the motors.

**Note:** Orient the motors so that their wires are inwards (i.e., towards the center of the plate).

**Step 3:**
Use 4 M3x30 screws and 4 M3 nuts to secure the motors to the motor holders. Tighten the screws to secure the holders to the bottom plate of the chassis as shown in Figure 15.5.
Figure 15.5. The sketch of how to secure the motors to the bottom plate.

Check the outcome:

Figure 15.6 shows how the motors should be attached to the bottom plate of the chassis.
Figure 15.6. The motors are attached to the bottom plate of the chassis.

3) Wheels

From the Magician Chassis package take the following components:

- Wheels (2x)

Figure 15.7 shows the components needed to complete this part of the tutorial.

Figure 15.7. The wheels.

*Video tutorial:*

The following video shows how to attach the wheels to the motors.
Check the outcome:

Figure 15.9 shows how the wheels should be attached to the motors.

Figure 15.9. The wheels are attached to the motors.

4) Omni-directional wheel

The Duckiebot is driven by controlling the wheels attached to the DC motors. Still, it requires a passive support on the back. In this configuration an omni-directional wheel is attached to the bottom plate of the chassis to provide such support.

From the Magician Chassis package take the following components:
• Steel omni-directional wheel (1x)
• Long metal spacers (2x)
• M3x6 screws (4x)

Figure 15.10 shows the components needed to complete this part of the tutorial.

![Components](image)

Figure 15.10. The omni-directional wheel with *2* long spacers and *4* M3x6 screws.

**Video tutorial:**

The following video shows how to attach the omni-directional wheel to the bottom plate of the chassis.

![Video](image)

Figure 15.11

5) Step-by-step guide
Step 1:
Secure the long spacers to the plate using 2 M3x6 screws and the omni-directional wheel to the spacers using also 2 M3x6 screws as shown in Figure 15.12.

Check the outcome:
Figure 15.13 shows how the omni-directional wheel should be attached to the plate.
Figure 15.13. The omni-directional wheel is attached to the plate.

6) Chassis standoffs

From the Magician Chassis package take the following components:

- Long metal spacers/standoffs (4x)
- M3x6 screws (4x)

From the Duckiebot kit take the following components:

- M3x5 nylon spacers/standoffs (4x)

Figure 15.14 shows the components needed to complete this part of the tutorial.
Figure 15.14. The standoffs to mount on the bottom plate.

Video tutorial:
The following video shows how to attach the standoffs to the bottom plate of the chassis.

Figure 15.15

7) Step-by-step guide

Step 1:
Secure the long metal spacers to the bottom plate using 4 M3x6 screws as shown in Figure 15.16.
Figure 15.16. The sketch of how to mount the standoffs on the plate.

**Step 2:**
Attach the 4 nylon standoffs on top of the metal ones.

**Check the outcome:**
Figure 15.17 shows how the standoffs should be attached to the plate.
8) Camera kit

From the Magician Chassis package take the following components:

- M3x10 flathead screws (2x)
- M3 nuts (2x)

From the Duckiebot kit take the following components:

- Camera Module (1x)
- (Optional) 300mm Camera cable (1x)
- Camera mount (1x)

Note: If you have camera cables of different lengths available, keep in mind that both are going to work. We suggest to use the longer one, and wrap the extra length under the Raspberry Pi stack.

Figure 15.18 shows the components needed to complete this part of the tutorial.
Figure 15.18. The parts needed to fix the camera on the top plate.

*Video tutorial:*
The following video shows how to secure the camera to the top plate of the chassis.

Figure 15.19

9) **Step-by-step guide**

*Step 1 (Optional):*
If you do not have the 300mm Camera cable you can jump to *Step 3.*
If you do have the long camera cable, the first thing to do is removing the shorter cable
that comes attached to the camera module. Make sure to slide up the black connectors of the camera port on the camera module in order to unblock the cable.

**Step 2:**
Connect the camera cable to the camera module as shown in [Figure 15.20](#).

![Figure 15.20](#).

**Step 3:**
Attach the camera module to the camera mount as shown in [Figure 15.21](#).

![Figure 15.21](#).

**Note:** The camera is just press-fitted to the camera mount, no screws/nuts are needed.

**Step 4:**
Secure the camera mount to the top plate by using the 2 M3x10 flathead screws and the nuts as shown in [Figure 15.22](#).
Figure 15.22. How to attach the camera mount to the top plate.

*Check the outcome:*

**Figure 15.23** shows how the camera should be attached to the plate.

Figure 15.23. The camera attached to the plate.

10) Heat sinks

From the Duckiebot kit take the following components:

- Raspberry Pi 3 (1x)
• Heat sinks (2x)
• Camera mount (1x)

Figure 15.24 shows the components needed to complete this part of the tutorial.

Video tutorial:
The following video shows how to install the heat sinks on the Raspberry Pi 3.

11) Step-by-step guide

Step 1:
Remove the protection layer from the heat sinks.
Step 2:
Install the big heat sink on the big “Broadcom”-labeled integrated circuit (IC).

Step 3:
Install the small heat sink on the small “SMSC”-labeled integrated circuit (IC).

Check the outcome:
Figure 15.26 shows how the heat sinks should be installed on the Raspberry Pi 3.

Figure 15.26. The heat sinks installed on the Raspberry Pi 3.

12) Raspberry Pi 3

From the Magician Chassis package take the following components:
- Top plate (with camera attached) (1x)

From the Duckiebot kit take the following components:
- Raspberry Pi 3 (with heat sinks) (1x)
- M2.5x12 nylon spacers/standoffs (8x)
- M2.5 nylon hex nuts (4x)

Figure 15.27 shows the components needed to complete this part of the tutorial.
Figure 15.27. The parts needed to mount the Raspberry Pi 3 on the top plate.

**Video tutorial:**
The following video shows how to mount the Raspberry Pi 3 on the top plate of the chassis.

![Video Tutorial Image]

13) **Step-by-step guide**

**Step 1:**
Mount 8 M2.5x12 nylon standoffs on the Raspberry Pi 3 as shown in Figure 15.29.
Step 2:
Use the M2.5 nylon hex nuts to secure the Raspberry Pi 3 to the top plate as shown in Figure 15.30.

Check the outcome:
Figure 15.31 shows how the Raspberry Pi 3 should be mounted on the top plate of the chassis.
14) Top plate

From the Magician Chassis package take the following components:

- Bottom plate (with motors, wheels and standoffs attached) (1x)
- Top plate (with camera and Raspberry Pi 3 attached) (1x)
- M3x6 screws (4x)

Figure 15.32 shows the components needed to complete this part of the tutorial.
The following video shows how to secure the top plate on top of the bottom plate.

Figure 15.33

15) Step-by-step guide

Step 1:
Pass the motor wires through the openings in the top plate.

Step 2:
Use 4 M3x6 screws to secure the top plate to the nylon standoffs (mounted on the bottom plate in Subsection 15.1.6 - Chassis standoffs) as shown in Figure 15.34.
Figure 15.34. How to secure the top plate to the bottom plate.

Check the outcome:
Figure 15.35 shows how the top plate should be mounted on the bottom plate.
16) USB Power cable

The power cable preparation is explained in Subsection 15.2.5 - Assembly instructions (DB17-jwd): power cable.

17) DC Stepper Motor HAT

From the Duckiebot kit take the following components:

- USB power cable (prepared in Subsection 15.2.5 - Assembly instructions (DB17-jwd): power cable) (1x)
- DC Stepper Motor HAT (1x)
- M2.5x10 Nylon screws (or M2.5x12 nylon standoffs) (4x)

Figure 15.36 shows the components needed to complete this part of the tutorial.
Figure 15.36. The parts needed to add the DC Stepper Motor HAT to the Duckiebot.

Video tutorial:
The following video shows how to connect the DC Stepper Motor HAT to the Raspberry Pi 3.

Figure 15.37

18) Step-by-step guide

Step 1:
Connect the wires of the USB power cable to the terminal block on the DC Stepper Motor HAT labeled as “5-12V Motor Power” as shown in Figure 15.38. The black wire goes to the negative terminal block (labeled with a minus: -) and the red wire goes to the positive terminal block (labeled with a plus: +).
Step 2:
Pass the free end of the camera cable through the opening in the DC Stepper Motor HAT as shown in Figure 15.39.

Step 3:
Connect the free end of the camera cable to the CAMERA port on the Raspberry Pi 3 as shown in Figure 15.40.
To do so, you will need to gently pull up on the black connector (it will slide up) to allow the cable to insert the port. Slide the connector back down to lock the cable in place, making sure it “clicks”.

**Note:** Make sure the camera cable is inserted in the right direction! The metal pins of the cable must be in contact with the metal terminals in the camera port of the PI. Please be aware that different camera cables have the text on different sides and with different orientation, **do not** use it as a landmark.

**Step 4:**
Attach the DC Stepper Motor HAT to the GPIO header on the Raspberry Pi 3. Make sure that the GPIO stacking header of the Motor HAT is carefully aligned with the underlying GPIO pins before applying pressure.

**Note:** In case you are using a short camera cable, ensure that the camera cable does not stand between the GPIO pins and the the GPIO header socket before applying pressure.

**Step 5:**
Secure the DC Stepper Motor HAT using 4 M2.5x10 nylon screws.

**Note:** If you are planning on upgrading your Duckiebot to the configuration `DB17-l`, you can use 4 M2.5x12 nylon standoffs instead.
Step 6:
Connect the motor wires to the terminal block on the DC Stepper Motor HAT as shown in Figure 15.41.

Figure 15.41. How to connect the motor wires to the terminal block on the DC Stepper Motor HAT. While looking at the Duckiebot from the back, identify the wires for left and right motor. Connect the left motor wires to the terminals labeled as M1 and the right motor wires to the terminals labeled as M2. This will ensure that the pre-existing software that we will later install on the Duckiebot will send the commands to the correct motors.

Check the outcome:
Figure 15.42 shows how the DC Stepper Motor HAT should be connected to the Raspberry Pi 3.
19) Battery

From the Duckiebot kit take the following components:

- Battery (1x)
- Zip tie (1x)
- Short micro USB cable (1x)

**Figure 15.43** shows the components needed to complete this part of the tutorial.

**Video tutorial:**
The following video shows how to add the battery to the Duckiebot and turn it on.
20) Step-by-step guide

*Step 1:*
Pass the zip tie through the opening in the top plate.

*Step 2:*
Slide the battery between the two plates. Make sure it is above the zip tie.

*Step 3:*
Push the free end of the zip tie through the opening in the top plate.

*Step 4:*
Tighten the zip tie to secure the battery.

*Step 5:*
Connect the short micro USB cable to the Raspberry Pi 3.

*Step 6:*
Connect the short micro USB cable to the battery.

*Step 7:*
Connect the USB power cable to the battery.

*Step 8:*
Make sure that the LEDs on the Raspberry Pi 3 and the DC Stepper Motor HAT are on.

*Check the outcome:*

*Figure 15.45* shows how the battery should be installed on the Duckiebot.
21) Upgrade to **DB17-w**

This upgrade equips the Duckiebot with a secondary, faster, Wi-Fi connection, ideal for image streaming. The new configuration is called **DB17-w**.

**Figure 15.46** shows the components needed to complete this upgrade.
Figure 15.46. The parts needed to upgrade the Duckiebot to the configuration DB17-w.

Instructions:
- Insert the USB WiFi dongle into one of the USB ports of the Raspberry Pi.
22) Upgrade to DB17-j

This upgrade equips the Duckiebot with manual remote control capabilities. It is particularly useful for getting the Duckiebot out of tight spots or letting younger ones have a drive, in addition to providing handy shortcuts to different functions in development phase. The new configuration is called DB17-j.

Figure 15.48 shows the components needed to complete this upgrade.
Figure 15.48. The parts needed to upgrade the Duckiebot to the configuration DB17-j.

**Note:** The joystick comes with a USB receiver (as shown in Figure 15.48).

*Instructions:*
- Insert the USB receiver into one of the USB ports of the Raspberry Pi.
- Insert 2 AA batteries on the back side of the joystick.
- Turn on the joystick by pressing the **HOME** button. Make sure that the LED above the **SELECT** button is steady.
23) Upgrade to **DB17-d**

This upgrade equips the Duckiebot with an external hard drive that is convenient for storing videos (logs) as it provides both extra capacity and faster data transfer rates than the microSD card in the Raspberry Pi 3. Moreover, it is easy to unplug it from the Duckiebot at the end of the day and bring it over to a computer for downloading and analyzing stored data. The new configuration is called **DB17-d**.  

Figure 15.50 shows the components needed to complete this upgrade.
Instructions:

- Insert the USB drive into one of the USB ports of the Raspberry Pi.
Figure 15.51. Upgrade to DB17-d completed.

- Mount your USB drive.

**TODO: re-add instructions from 2017 Duckiebook version.**

The following was marked as "todo".

**TODO: re-add instructions from 2017 Duckiebook version.**

Location not known more precisely.

Created by function n/a in module n/a.

24) FAQ

**Q:** If we have the bumpers, at what point should we add them?

**Answer:** You shouldn’t have the bumpers at this point. The function of the bumpers is to keep the LEDs in place, i.e., they belong to DB17-l configuration. These instructions cover the DB17-wjd configurations. You will find the bumper assembly instruc-
tions in Section 15.3 - Bumper Assembly.

**Q:** Yeah but I still have the bumpers and am reading this page. So?

**Answer:** The bumpers can be added after the Duckiebot assembly is complete.

**Q:** I found it hard to mount the camera (the holes weren’t lining up).

**Answer:** Sometimes in life you have to push a little to make things happen. (But don’t push too much or things will break!)

**Q:** The long camera cable is a bit annoying - I folded it and shoved it in between two hats.

**Answer:** The shorter cable is even more annoying. We suggest wrapping the long camera cable between the chassis and the Raspberry Pi. With some strategic planning, you can use the zip ties that keep the battery in place to hold the camera cable in place as well.

**Q:** I need something to cut the end of the zip tie with.

**Answer:** Scissors typically work out for these kind of jobs (and no, they’re not provided in a Fall 2017 Duckiebox).

### 15.2. Assembly instructions (DB17): soldering

**Note:** It is better to be safe than sorry. Soldering is a potentially hazardous activity. There is a fire hazard as well as the risk of inhaling toxic fumes. Stop a second and make sure you are addressing the safety standards for soldering when following these instructions. If you have never soldered before, seek advice.

In this instruction set we will assume you have soldered something before and are acquainted with the soldering fundamentals. If not, before proceeding, read this great tutorial on soldering:

→ [Alternative instructions: how to solder on Headers and Terminal Block](#)

**Note:** Very general tips in soldering

- solder the components according to their height - from lowest to highest.
- soldering is potentially dangerous - prepare a clean, well lit and aerated working place before starting.
- It’s ok to ask for help! Especially is you have never soldered before.
1) Assembly instructions (DB17-jwd): DC Motor HAT

**Requirements:** Parts: Duckiebot DB17 parts. The acquisition process is explained in Subsection 3.3.1 - DB17 Bill of materials. The configurations are described in Unit B-2 - Duckiebot Configurations. In particular you need:

- GPIO Stacking Header
- DC and Stepper Motor HAT for Raspberry Pi
- Soldering tools

**Requirements:** Experience: novice-level experience with soldering.

**Requirements:** Time: 20 minutes

**Results:** Soldered DC Motor HAT

**Preparing the components:**

Take the GPIO stacking header Figure 15.52 out of Duckiebox and sort the following components from DC motor HAT package:

- Adafruit DC/Stepper Motor HAT for Raspberry Pi
- 2-pin terminal block (2x), 3-pin terminal block (1x)

Figure 15.52. GPIO_Stacking_Header
Soldering instructions:

1) Make a 5 pin terminal block by sliding the included 2 pin and 3 pin terminal blocks into each other Figure 15.54.

2) Slide this 5 pin block through the holes just under “M1 GND M2” on the board. Solder it on (we only use two motors and do not need connect anything at the “M3 GND M4” location) (Figure 15.57);

3) Slide a 2 pin terminal block into the corner for power. Solder it on. (Figure 15.56);

4) Slide in the GPIO Stacking Header onto the 2x20 grid of holes on the edge opposite the terminal blocks and with vice versa direction (Figure 15.55). Solder it on.

**Note:** stick the GPIO Stacking Header from bottom to top, different orientation than terminal blocks (from top to bottom).
2) Assembly instructions (DB17-1): soldering the PWM / Servo HAT and LED boards

**Knowledge and activity graph**

**Requires:** Duckiebot DB17-1 parts. The acquisition process is explained in Subsection 3.3.1 - DB17 Bill of materials. The configurations are described in Unit B-2 - Duckiebot Configurations. In particular you need:

- **GPIO Stacking Header**
- **Adafruit PWM / Servo HAT**
- **LED HAT**

**Requires:** Time: 30 minutes

**Results:** A soldered PWM / Servo HAT for DB17-1 configuration.

16-channel PWM/Servo HAT:
(Alternative instructions: how to solder on the PWM/Servo HAT)

**Prepare the components:**
Put the following components on the table according the Figure

- **GPIO Stacking Header** for A+/B+/Pi 2
- **Adafruit HAT** Mini Kit of 16-Channel PWM / Servo HAT for Raspberry Pi
3x4 headers (4x)
- 2-pin terminal block
- 16-Channel PWM / Servo HAT for Raspberry Pi (1x)

- **LED HAT**

Figure 15.58.

3) **PWM / Servo HAT soldering instructions**

1. Solder the 2 pin terminal block next to the power cable jack
2. Solder the four 3x4 headers onto the edge of the HAT, below the words “Servo/PWM Pi HAT!”
3. Solder the GPIO Stacking Header at the top of the board, where the 2x20 grid of holes is located.

4) **LSD board soldering instructions**

The LSD board you received is unpopulated and should look like this:
Prepare the components:
Put the following components according to the figure on the table:

- 1 x 40 pin female header
- 5 x 4 pin female header
- 2 x 16 pin male header
- 1 x 12 pin male header
- 1 x 3 pin male header
- 1 x 2 pin female shunt jumper
- 5 x 200 Ohm resistors
- 10 x 130 Ohm resistors
- 3 x 4 pin male header for servos
Soldering instructions:
1. Put the resistors on the top of the board according to silkscreen markings, solder it on from the bottom side.

Tips:
1. Solder all female headers to the bottom of the board. Alignment becomes easy if the female headers are plugged into the PWM heat, and the LSD board rests on top.
2. Solder all male headers to the top of the board. Male header positions are outlined on the silkscreen.

LED connection:

Parts list:
• 4 x 6” female-female jumper cable

Instructions:
1. Connect LED accordingly to silkscreen indication on PRi 2 LSD board
2. Silkscreen legend: Rx, Gx, Bx are red, green, and blue channels, accordingly, where x is the LED number; C is a common line (either common anode or common cathode)
3. For adafruit LEDs are common anode type. The longest pin is common anode. Single pin on the side of common is red channel. The two other pins are Green and Blue channels, with the blue furthest from the common pin.
4. Both types of LEDs are supported. Use shunt jumper to select either common anode (CA) or common cathode (CC) on 3-pin male header. Note, however, that all LEDs on the board must be of the same type.

Putting things together:
1. Stack the boards
   - Screw the first eight standoffs into the Pi - provide hints on the location of standoffs and the suggested orientation of the boards w/r to the chassis
   - Connect the camera to the Pi
   - Stack the DC/Stepper Motor HAT onto the Pi, aligning both sets of GPIO pins over each other and screw the standoffs to secure it. Try to not bend the camera connector too much during this step
   - Stack the 16-channel PWM/Servo HAT onto the Pi, both sets of GPIO pins over each other and screw the standoffs to secure it

2. Slide the battery between the two chassis plates

3. Power the PWM/Servo HAT and Pi connecting them to the battery with the cables included in the duckiebox

4. Power the DC/Stepper motor from the PWM/Servo HAT using the male-to-male cable in the duckiebox, connect the positive

5. Connect the Pi to the board

6. Done!

5) Assembly instructions (DB17-jwd): power cable

In configuration DB17 we will need a cable to power the DC motor HAT from the battery. The keen observer might have noticed that such a cable was not included in the DB17 Duckiebot parts chapter. Here, we create this cable by splitting open any USB-A cable, identifying and stripping the power wires, and using them to power the DC motor HAT. If you are unsure about the definitions of the different Duckiebot configurations, read Unit B-2 - Duckiebot Configurations.

It is important to note that these instructions are relevant only for assembling a DB17-wjdc configuration Duckiebot (or any subset of it). If you intend to build a DB17-li configuration Duckiebot, you can skip these instructions.

**Knowledge and activity graph**

- **Requires:** One male USB-A to anything cable.
- **Requires:** A pair of scissors.
- **Requires:** A multimeter (only if you are not purchasing the suggested components)
- **Requires:** Time: 5 minutes
Results: One male USB-A to wires power cable

Video tutorial:
The following video shows how to prepare the USB power cable for the configuration DB17.

![Image](image_url)

Figure 15.60

Step 1: Find a cable:
To begin with, find a male USB-A to anything cable.

If you have purchased the suggested components listed in Unit B-3 - Getting the Duckiebot hardware, you can use the longer USB cable contained inside the battery package (Figure 15.61), which will be used as an example in these instructions.

![Image](image_url)

Figure 15.61. The two USB cables in the suggested battery pack.

Put the shorter cable back in the box, and open the longer cable (Figure 15.62)
Figure 15.62. Take the longer cable, and put the shorter one back in the box.

*Step 2: Cut the cable:*

**Check before you continue**
Make sure the USB cable is *unplugged* from any power source before proceeding.

Take the scissors and cut it ([Figure 15.63](#)) at the desired length from the USB-A port.

Figure 15.63. Cut the USB cable using the scissors.

The cut will look like in [Figure 15.64](#).
**Step 3: Strip the cable:**

Paying attention not to get hurt, strip the external white plastic. A way to do so without damaging the wires is shown in Figure 15.65.

---

**Figure 15.65.** Stripping the external layer of the USB cable.

After removing the external plastic, you will see four wires: black, green, white and red (Figure 15.66).

---

**Figure 15.66.** Under the hood of a USB-A cable.

Once the bottom part of the external cable is removed, you will have isolated the four
wires (Figure 15.67).

![Image of USB-A cable with wires exposed]

Figure 15.67. The four wires inside a USB-A cable.

**Step 4: Strip the wires:**

**Check before you continue**

Make sure the USB cable is *unplugged* from any power source before proceeding.

Once you have isolated the wires, strip them, and use the scissors to cut off the data wires (green and white, central positions) (Figure 15.68).

![Image of USB-A cable with power wires stripped and data wires removed]

Figure 15.68. Strip the power wires and cut the data wires.

If you are not using the suggested cable, or want to verify which are the data and power wires, continue reading.

**Step 5: Find the power wires:**

If you are using the USB-A cable from the suggested battery pack, black and red are the power wires and green and white are instead for data.

If you are using a different USB cable, or are curious to verify that black and red actu-
ally are the power cables, take a multimeter and continue reading.
Plug the USB port inside a power source, e.g., the Duckiebot’s battery. You can use some scotch tape to keep the cable from moving while probing the different pairs of wires with a multimeter. The voltage across the pair of power cables will be roughly twice the voltage between a power and data cable. The pair of data cables will have no voltage differential across them. If you are using the suggested Duckiebot battery as power source, you will measure around 5V across the power cables (Figure 15.69).

Figure 15.69. Finding which two wires are for power.

**Step 6: Test correct operation:**
You are now ready to secure the power wires to the DC motor HAT power pins. To do so though, you need to have soldered the boards first. If you have not done so yet, read Section 15.2 - Assembly instructions (DB17): soldering.
If you have soldered the boards already, you may test correct functionality of the newly crafted cable. Connect the battery with the DC motor HAT by making sure you plug the black wire in the pin labeled with a minus: - and the red wire to the plus: + (Figure 15.70).

Figure 15.70. Connect the power wires to the DC motor HAT
15.3. Bumper Assembly

Knowledge and Activity Graph

- **Requires**: Duckiebot \( \text{DB17-lc} \) parts.
- **Requires**: Having the Duckiebot with configuration \( \text{DB17-wjd} \) assembled. The assembly process is explained in Section 15.1 - Assembly instructions (DB17-jwd).
- **Requires**: Time: about 15 minutes.
- **Results**: A Duckiebot with Bumpers (configuration DB17-l2)

1) Locate all required parts

The following should be included in your parts envelope (See image below for these components):

- 1x front bumper (Camera side)
- 1x rear bumper (the side of Caster/Omnidirectional wheel)
- 2x rear bumper brace (the side of Caster/Omnidirectional wheel)
- 8x M2.5x10 nylon or metal screws
- 8x M2.5 nuts

The following is not included in your parts envelope but will be needed for assembly:

- Small screwdriver

Figure 15.71. Components in Duckiebot package.
2) Reminder: Use care when assembling!

Use care when assembling, make sure to be gentle! Tighten screws just enough so that the parts will remain stationary. When inserting LEDs, gently work them into their holders. While the acrylic is relatively tough, it can be fractured with modest force. We don’t have many replacements (at this moment) so we may not be able to replace a broken part.

*Remove protective paper:*

Peel protective layer off of all parts on all sides.
3) Assembly Rear Spacers

_Disassembly the spacers between the both chassis:

The backside of duckiebot before assembling the bumpers looks as Figure 15.74:

Now remove the spacers and the (short)metal screws from the standoffs (configuration ‘DB17-wjd’) and replace it with 4 M3x10 nylon screws for connecting the chassis and the bumper spacers.
Figure 15.75. The spacers configuration for the bumpers, Left: old configuration, Right: new configuration

M3x10 screws attaching bottom rear brace:

Figure 15.76. Attach the spacers with M3*10 nylone screws

Back View, fully assembled:
Note: For the ETH students, the M3*10 nylon screws for attaching the rear spacers with chassis were already distributed during the duckiebot ceremony and not included in second distribution. Please reuse them!

Mount Rear Bumper:
Carefully guide rear bumper on to rear bumper brace tabs. Ensure that the hole for charging aligns with the charging port on your battery.
Locate 4 M2.5 nylon/metall nuts and 4 M2.5*10 nylon screws. Place a nut in the wide part of the t-slot and thread a screw into the nut as shown in the following pictures. Note: Use care when assembling! If you are having trouble with the nuts falling out, take a small piece of transparent tape and place it over both sides of the t-slot with the nut inside. It won’t look as nice but it will be much easier to assemble.

✔ Test the screws and the nuts once by screwing them together before you use it for the bumpers. It make the following assembly process much easier.

**Note:** For ETH 2017, the screws and nuts using for this step are M2.5*10 nylon screws (white) and M2.5 nylon or metall nuts from the envelope.
Figure 15.79. Hold the nuts with the fingers
Figure 15.80. Screw the screws into nuts while holding the nuts with the fingers

The completed rear bumper should look like this:
Figure 15.81. Completed rear bumpers

Congrats! your rear bumper assembly is complete!

Mount Front Bumper:

The front side of Duckiebot in DB17-wjd should look like in Figure 15.82.

Take 2x M2.5*10 nylon screws and 2x M2.5 nylon nuts and install them as shown in the following pictures. The first picture shows the correct holes to mount these screws (The correct position is the widest pair of 3mm holes beside the camera). The nuts should tightened on by a few threads (these are the two nuts that are not yet tightened at the top of the second picture):
Figure 15.82. Front side of configuration DB17-wjd, without bumpers

✔ Before tighten the front bumper, you should organize the wires of LEDs going through the right holes of chassis. The center LED should be bent at a right angle in the direction that the wire is fed through the body.

Take the front bumper and carefully press the LEDs into the bumper holders. Take care that the wires are routed behind the front bumper. Also note that the front center LED wire should not be crushed between the bumper and the right spacer (you will likely fracture the bumper if you try to force it). The center LED should be bent at a right angle in the direction that the wire is fed through the body (see Figure 15.83).
Figure 15.83. Insert the LEDs before tighten the front bumper.

Position the bumper so that the nuts align with the t-slots. You may need to loosen or tighten the screws to align the nuts. You may also need to angle the front bumper when inserting to get it past the camera screws.
Gently tighten the nuts. The front bumper should now stay in position.

15.4. Assembly instructions (DB17-1)

Figure 15.84. Completed front bumpers

**Knowledge and activity graph**

- **Requires:** Duckiebot DB17-lc parts. The acquisition process is explained in Section 3.3.1 - DB17 Bill of materials.
- **Requires:** Soldering DB17-lc parts. The soldering process is explained in Section 15.2 - Assembly instructions (DB17): soldering.
- **Requires:** Having assembled the Duckiebot in configuration DB17 (or any DB17-wjd). The assembly process is explained in Section 15.1 - Assembly instructions (DB17-jwd).
- **Requires:** Time: about 30 minutes.
- **Results:** An assembled Duckiebot in configuration DB17-wjdlc.

1) Assembly the Servo/PWM hat (DB17-l1)
Recommend: If you have bumpers, it is recommend to have them assembled before the PWM hat. The assembly process is explained in Section 15.3 - Bumper Assembly.

Locate the components for Servo/PWM hat:

- Soldered PWM hat (1x)
- Nylon Standoffs (M3.5 12mm F-F) (4x)
- Power cable: short angled male USB-A to 5.5/2.1mm DC power jack cable
- Male-Male Jumper Wires (1x)
- Screwdriver

Remove the hand-made USB power cable from DC Motor HAT:

From now on, the DC Motor Hat will be powered by the PWM HAT via male-male jumper wire. Before that, the previous hand-made USB power cable needed to be removed. Insert the male-male jumper wire into + power terminal on the DC motor HAT (DC-end).
Figure 15.86. Insert the male-male wire into + terminal block on the DC motor HAT

Stack the PWM HAT above the DC motor HAT:
Put a soldered Servo/PWM HAT board (in your Duckiebox) with 4 standoffs on the top of Stepper Motor HAT.
Insert the other end of male-male jumper wire into “+5”V power terminal on the PWM HAT (PWM-end). It leads the power to DC motor HAT.

Figure 15.87. Insert the PWM-end into +5V terminal on PWM HAT

Power Supply for PWM HAT:
To power the PWM/Servo HAT from the battery, plugin a short (30cm) angled male
USB-A to 5.5/2.1mm DC power jack cable into PWM HAT. The other end of the power cable will plugin to the battery when it is in use.

Figure 15.88. Plugin the short angled male DC power cable

2) Assembling the Bumper Set (DB17-l2)

For instructions on how to assemble your bumpers set, refer to: Section 15.3 - Bumper Assembly.

3) Assembling the LED HAT and LEDs (DB17-l3)

For instructions on how to assemble the LED HAT and related LEDs, refer to: Section 15.5 - Assembling the DB17-lc.

15.5. Assembling the DB17-lc

**Knowledge and activity graph**

**Requires:** Duckiebot DB17-lc parts. The acquisition process is explained in Section 3.3 - Acquiring the parts for a DB17.

**Requires:** Soldering DB17-lc parts. The soldering process is explained in Section 15.2 - Assembly instructions (DB17): soldering.

**Requires:** Having assembled PWM Hat on the Duckiebot with configuration DB17-wjd. The assembly process is explained in Unit C-15 - Assembly - Duckiebot.
**DB17.**

**Requires:** Time: about 15 minutes.

**Results:** A Duckiebot with LEDs attached and functioning (configuration DB17-l3)

1) Locate all required parts

To attach the LEDs on the Duckiebots, the following components are needed:
- Soldered LSD board with Jumper
- LED (5x)
- Female-female wires (20x)
- Nylon standoffs M2.5*12
- some Tape

![Components-List for LED configuration](image)

2) Connecting Wires to LEDs

**LEDs:**

The LEDs are common anode type. The longest pin is called the common. The single pin on the side of common is red channel. The two other pins are Green and Blue channels, with the blue furthest from the common pin.
Use the long wires with two female ends. Attach one to each of the pins on the LED. To figure out the order to connect them to the LSD hat, use the legend on the silkscreen and the information above. i.e. RX - means the red pin, CX - means the common, GX means the green, and BX means the blue. The “X” varies in number from 1-5 depending on which LED is being connected as discussed in the next section.
Figure 15.91. Locate 5 groups of 4 pins with label RX, C, GX, BX) on the LSD board

✔ Use Tape to keep the LEDs stick with the wires.

3) Connecting LEDs to LSD Hat

Silkscreen legend: Rx, Gx, Bx are red, green, and blue channels, accordingly, where x is the LED number; C is a common line (either common anode or common cathode).

Define the following names for the lights:

- “top” = top light - the “top” light is now at the bottom
- fl = front left
- fr = front right
- br = back right
- bl = back left

The LEDs are wired according to Figure 15.92.
Mappings from the numbers on the LED hats to the positions shown (TOP is now the one in the middle at the front)

- FR -> 5
- BR -> 4
- TOP -> 3
- BL -> 2
- FL -> 1

4) Running the Wires Through the Chassis

It is advised that the LED cables are routed through the positions noted in the images below before installing the bumpers:

Front Left, Front Middle, and Front Right LED Wiring suggestion:
5) Final tweaks

Adjust the LED terminals (particularly in the front) so that they do not interfere with the wheels. This can be accomplished by bending them up, away from the treads.
UNIT C-16

Assembly - Duckiebot DB18

This page is for the DB18 configuration used in classes in 2018 and 2019. For the instructions from 2017 see the DB17 Duckiebot operation manual.

**Knowledge and activity graph**

- **Requires:** Duckiebot DB18 parts (get them [here](#)). You may want to check out the different configurations [here](#).
- **Requires:** A microSD card with the Duckiebot image already on it. This procedure is explained [here](#).
- **Requires:** 1.5 hours of assembly time.
- **Results:** An assembled Duckiebot in configuration DB18.

**Note:** Make sure you visit the FAQ section at the bottom of this page. This may already answer your comments, questions or doubts.

The complete assembly process is divided into 9 subparts. Make sure to complete them in the following order:

- **Part 1:** What is in the box?
- **Part 2:** Motors
- **Part 3:** Bottom chassis
- **Part 4:** Computation unit
- **Part 5:** Camera
- **Part 6:** Top chassis
- **Part 7:** Back bumper
- **Part 8:** Front bumper
- **Part 9:** Battery and Duckie

### 16.1. What is in the box

All the pieces in your Duckiebox are shown in Figure 16.1. Note that the battery and camera calibration pattern are not shown in the picture.
Moreover, you might have slightly different components than those shown: for example, different USB power cables, slightly different sets of screws or a camera mount and backplate of different colors. Do not worry, these instructions can be followed anyway.

Figure 16.1. Components in Duckiebot package (Duckiebox).

Some of the components in your Duckiebox will not be used at this stage, e.g., the traffic signs and stands. Keep these aside, they will come in handy in other parts of the book.

Finally, you should have several spares, especially for the mechanical bits (nuts and screws). These are included just in case you drop a few and can’t find them anymore (especially the tiny nylon ones like to hide in the cracks!).

16.2. Preliminary Steps

1) Unboxing
Unbox all of your components and lay them out on a flat surface. Ensure that you have well lit, uncluttered space to work on.

Although not necessary, you might find useful getting a small (M2.5) wrench to ease some of the passages.

2) Do the bumper braces fit?

Take the rear bumper bracers and the back bumper. The back bumper will be mounted in the last steps as a press fit to the rear bumper bracers. Try to fit the bracers into the holes of the back bumper.

Some bumper bracers have a plastic protective film which is a residue of the manufacturing process. If you struggle in the press fitting, peel off the plastic cover from one side of the bracer. If this does not help, peel off the plastic cover from both sides.

3) Plastic cover

Peel the plastic cover from all the chassis parts (except the bumper bracers) on both sides.

4) Screws

Note that a few among all of your metal screws are special. They are “countersunk” screws. Keep these aside. They will be needed in Part 7: Back bumper.

Figure 16.2. Countersunk screws have a tapered head.

Every time you read M3x8 screw, a M3x10 will do the same trick. You can exchange them at will.

Do not exchange metal and nylon screws though. The latter are not electrically conductive and are passive protections to potential short circuits that can damage your Duckiebot beyond repair. This is especially true for - Part 3: Bottom chassis; make sure
you use the correct nylon screws at that step. Remember that instructions are your friend! At least at this stage, try to follow them precisely.

If regardless of this foreword you still choose to try and figure it out yourself, it’s ok, but for the love of what is precious, do not plug the battery in until you have performed a visual inspection here.

16.3. Motors

From the Duckiebox package take the following components:

- Bottom plate (1x)
- Yellow dc motors (2x)
- Motor holders (4x)
- M3x30 screw (4x)
- M3 nuts (4x)

1) Step 1

Pass two of the motor holders through the openings in the bottom plate of the chassis as shown in Figure 16.3. If you have troubles fitting the holders, it is probably because you have not removed the protective plastic film. Note which holes you are using. You should be using the middle section ones.
Figure 16.3. How to mount the motor holders.

Put one motor between the holders as shown in Figure 16.4.

Figure 16.4. How to mount the first DC-motor.

**Note:** Orient the motors that their wires are pointing to the inside (i.e., towards the center of the plate).
**Note:** Use your screwdriver.

Use 2 M3x30 screws and 2 M3 nuts to secure the motor to the motor supports. Pass the screws through the holes from the outside inwards, then tighten the screws to secure the holders to the bottom plate of the chassis.

**Note:** You might find aligning the holes to be a little bit hard. It is easier to first get the top screw aligned and place the screw in, and then push in the side support piece in to place. If that still doesn't work, get the bottom screw and the side support in first, align it, and secure with nuts. Then try to push the top screw through. You might have to use the screw to expand the hole a bit.

2) Step 2

Do the same for the second motor as well as shown in [Figure 16.5](#) and [Figure 16.6](#).

![Figure 16.5. How to mount the second motor supports](image-url)
Figure 16.6. How to mount the second DC-motor

**Note:** Tighten the motors firmly, as a wobbly motor might cause troubles later on.

3) Step 3

Wire the cables of the two motors through the bottom plate of the chassis and make sure they are well separated (Figure 16.7).

Figure 16.7. Cables through central hole

16.4. **Bottom chassis**

The Duckiebot is driven by controlling the wheels attached to the DC motors. Still,
it requires a *passive* support on the back. In this configuration an omni-directional wheel is attached to the bottom plate of the chassis to provide such support.

From the Duckiebox package take the following components: - Steel omni-directional wheel (1x) - M3x25 metal spacers (2x) - M3x10 metal screws (2x) - M3x10 nylon screws (2x)

1) Step 4

Assemble the omni-directional wheel as shown in figure **Figure 16.8**.

![Figure 16.8. Assembly of the omni-directional wheel](image)

2) Step 5

Then mount it to the bottom plate of the Duckiebot as shown in figure **Figure 16.9**.
3) Step 6

From the Duckiebot kit take the following components:

- M3x10 screws (4x)
- M3x25 metal spacers (2x)
- M3x30 metal spacers (2x)
- 1x Rear bumper bracer

and mount the rear bracer to the bottom plate with the 25cm spacers (Figure 16.10)!
4) Step 7

Now use the M3x30 standoffs (longer ones) for the front of the Duckiebot (Figure 16.11).
5) Step 8

From the Duckiebot kit take the following components:

- Wheels (2x)

Figure 16.12 shows how to mount the big yellow driving wheels to the bottom chassis.

Figure 16.12. Mounting the driving wheels

**Note:** Mind that there is a particular orientation at which the wheels will fit. Don’t force them too much!

Figure 16.13 shows how the assembly should look like after mounting the wheels.
Figure 16.13. The wheels mounted on the bottom plate assembly.

16.5. **Computation unit**

From the Duckiebot kit take the following components:

- Raspberry Pi 3B+ (1x)
- Heat sink (1x) (or 2x if there are two of them)
- Camera cable (1x) (This should be in the same box as the camera)
- Micro SD card (1x)

**Note:** You probably have two heat sinks, make sure you use the bigger one for sure.
The smaller one can be used as shown in the following pictures or apply it on the USB port you will later use for the external 32GB USB dongle.

**Note:** You will notice there is a camera cable already attached to the camera. We won’t use this one since it is a bit short. You will find a longer camera cable inside the same box as the camera.

*Figure 16.14* shows the components needed to complete the following steps.

![The bigger heat sinks and the Raspberry Pi 3 B+](image)

*Figure 16.14. The bigger heat sinks and the Raspberry Pi 3 B+.*

1) **Step 9**

Peel the cover from the bottom of the heat sink and place it on the Raspberry Pi microchip, as shown in *Figure 16.15*. Make sure to put it on the
2) Step 10

Insert the SD card in the slot as shown in figure Figure 16.16.

**Note:** If the card is not flashed yet, do the initialization first: Unit C-3 - Setup Duckiebot SD Card, but only until the section Section 3.2 - Burn the SD card, because you will need the fully assembled Duckiebot for the following steps of the Duckiebot Initialization.
Figure 16.16. Insert SD card

3) Step 11

Make sure that the visible metal connectors of the camera cable match the ones in the Raspberry Pi (Figure 16.17), then plug in the cable and push down the black wings to fasten the connection, making sure it “clicks”.

Note: Please be aware that different camera cables have the text on different sides and with different orientation, do not use it as a landmark.
Figure 16.17. Put the camera cable in the open plug
Then close the camera plug once the cable is pushed in properly. You can check by pulling at the cable once you have closed the plug and you mustn’t be able to tear it out at all.

Figure 16.18. Close camera cable plug

4) Step 12
From the Duckiebot kit take the following components:
• Duckiebot Hut (1x)
• Top plate of the chassis (1x)
• M2.5x12 nylon spacers (4x)
• M2.5x10 Nylon screws (4x)
• M2.5x4 Nylon spacers (4x)
• M2.5 nylon nuts (4x)
• USB to micro USB cables (2x)
• Set of three female to female (F/F) jumper wires (1x)

Figure 16.19 shows the components needed to complete this part of the tutorial.

Note: In the newest version of the Hut, the pins are removed. This is to prevent them from being shorted. You will not need them so you can neglect them in the next few pictures.

Note: It is more convenient to not separate the jumper cables, but leave them in two
sets of three.

Place the spacers on the bottom part of the Duckiebot hut and secure them using the M2.5 nylon nuts, see figure Figure 16.20

![Figure 16.20. Mount spacers to Hut](image)

5) Step 13

Mount the pre-assembled Hut and Raspberry Pi to the top plate with nylon screws as shown in figure Figure 16.21
Figure 16.21. Mount computation unit

**Note:** Make sure that the Raspberry Pi GPIO pins fit into the Hut connector.

6) Step 14

Connect the F/F jumper wire to the pins on the Hut as shown in figure Figure 16.22 and make sure the camera cable is put in the slit of the Hut
7) Step 15

Finally, attach the two USB cables to the power plugs on the Hut and Raspberry Pi, as in Figure 16.23.

16.6. Camera

From the Duckiebox package take the following components:

- Raspberry Pi camera
• Camera mount (1x)
• M3x10 screws (3x)
• M3 nuts (3x)

1) Step 16

Open the camera cable plug on the Raspberry Pi camera as shown in figure Figure 16.24.

Note: Make sure the cable is routed through the camera mount as well!
Close the plug once the cable is pushed in completely as in Figure 16.25
Figure 16.25. Close camera plug

Insert the M3x10 nylon screws from the front to screw the camera to its mount, see Figure 16.26

Figure 16.26. Screw the camera to its mount
Note: Make sure the camera is as parallel to its mount as possible, i.e. facing downwards at this very angle.

2) Step 17

Screw the camera mount to the top plate with three M3x10 screws and metal M3 nuts, see Figure 16.27.

Figure 16.27. Screw the mount to the top plate

16.7. Top chassis

From the Duckiebox package take the following components: - Metal screws M3x10 (4x) - Back bumper bracer (1x)

1) Step 18

Put the top plate on the bottom part of the chassis as shown in figure Figure 16.28 but do not screw it in yet. This is how it should look like in the end.
Figure 16.28. Placing the top plate

2) Step 19

Now open the slots for the cables of the DC motors on the Hut with a small screwdriver and make sure you mount the cables in correctly, see figure Figure 16.29. Probably you have to take the top and bottom part of the chassis apart again.

Figure 16.29. Connect the DC motor cables to the Hut

3) Step 20

Place the second rear bumper bracer on top of the chassis spacers in the back of the...
chassis. Again, the holes should align (Figure 16.30). If you experience a slight misalignment, carefully pass an M3 screw through the top plate and the top bumper bracer first, then move the standoff until you can plug the screw in. At the end, put in the second M3 screw as well.

![Rear bumper bracer positioning](image)

**Figure 16.30. Rear bumper bracer positioning**

4) **Step 21**

Fasten the top plate in the front using the other two M3x10 screws, as in(Figure 16.31).
5) Step 22

You can use a zip tie to help with cable management (Figure 16.32). Tip: you can use the same zip tie to keep the camera cable down and the USB cables below from touching the wheels.
16.8. Back bumper

From the Duckiebot kit take the following components:

- Back bumper (1x)
- M3x10 nylon screws (2x)
- M3x10 metal screws (4x)
- M3x10 countersunk screws (2x)
- M3 metal nut (4x)
- M3x25 spacers (2x)
- Back pattern plate (1x)
- Circle pattern sticker (1x)

**Note:** In the picture, nylon screws are used. However you probably used them to assemble the omni-directional wheel, then use the metal screws. 

*Figure 16.33* shows the components needed to complete this upgrade.
Figure 16.33. The parts needed to mount the circle grid holder.

**Note:** You could have a back plate of a different color with respect to the picture, e.g., black or white. They are all functionally equivalent.

1) Step 23

Mount the spacers using the *metal* screws, as in **Figure 16.34**.

Figure 16.34. How to mount the spacers
**Note:** Pay attention that the spacers should be in the same direction as the LEDs.

2) **Step 24**

Fasten the back plate to the back bumper standoffs using the countersunk screws. The back plate is not symmetric, place it such that it is centered with respect to the back bumper as shown in picture *Figure 16.35*.

![Figure 16.35. Mount the pattern plate](image)

3) **Step 25**

Peel off the sticker with the circle pattern and put it on the pattern plate mounted before, see picture *Figure 16.36*.

![Figure 16.36. Pattern sticker](image)
4) Step 26

Connect a new jumper wire (F/F) to the connector on the back bumper, see picture **Figure 16.37**.

![Figure 16.37. Jumper wire to back bumper](image)

No matter what exact solution you choose concerning the cable management, the objective is to avoid any cables from touching the wheels in the end. One solution could be something like it is shown in figure **Figure 16.38**.
5) Step 27

Screw the back bumper to the rest of the chassis with four metal screws and nuts, see picture Figure 16.39.

Note: Pay attention that the pins of the bumper bracers are properly put in the holes of the back bumper. It might be a little bit difficult to place it properly.
**Note:** Wire the cable mounted in step 26 through the top plate of the chassis as shown in the picture as well.

### 16.9. Front bumper

From the Duckiebot kit take the following components:

- Front bumper (1x)
- M3x10 metal screws (2x)

1) **Step 28**

At that point you should have arrived with two loose cables in the front. Make sure to connect both of them to the front bumper. The one from step 26 on the right hand side and the other on the left hand side, see picture [Figure 16.40](#).

2) **Step 29**

First, click in the front bumper without any screws. Maybe you have to bend the top and bottom plate of the chassis a little bit in order to make it fit, see pictures [Figure 16.41](#) and [Figure 16.42](#).
Figure 16.41
Now use the two M3x10 metal screws to finally fix the top plate of the chassis properly. This will now prevent the front bumper from falling out again. Probably you will have to move the standoffs a little bit in order to make the holes flush, see figure Figure 16.43.
16.10. Battery and Duckie

From the Duckiebot kit take the following components:

- Battery (1x)
- Zip tie (1x)
- Duckie (N+1x)

1) Visual inspection

Before plugging in the battery, make sure the Hut’s GPIO pins are (a) not touching each other, (b) not touching any metal screws (in case you did not follow these instructions exactly) and (c) are free from any external object that might have gotten stuck there during the assembly process.

**Note:** If the GPIOs of the Hut get shorted (there is an electrically conductive connection between them) when you plug in the battery, you might damage the Raspberry Pi beyond repair!
2) Step 31

Place the battery and fix it using a zip tie (Figure 16.44), then connect the USB cables.

**Note:** Do not unplug/replug the bumper wires when the power is on. You could break the bumpers!

Figure 16.44. Fixing the top plate

There are different batteries concerning the outer dimensions, so the result could look like either of the following pictures Figure 16.45 or Figure 16.46:
Now place a Duckie on top of your brand new Duckiebot, but be careful not to hurt the Duckie!

3) Check the outcome
As a final check, verify that no cable is touching the wheels. You can use the provided zip ties to ensure that cables stay out of the way.

16.11. **Hardware verification**

If you have assembled your Duckiebot DB18 but you are not sure if you did it correctly, you can use this tool test that. These steps are not mandatory but do serve as a good troubleshooting tool. It takes about 20 minutes.

---

**Note:** The procedure consists of downloading and flashing a test image on an SD card. If you have only one SD card, you might want to do that before you do the steps in **Unit C-3 - Setup Duckiebot SD Card**.

1. Download this SD card image
2. Extract the .img image from the archive.
3. Flash the image on an SD card. If you use Ubuntu, you can use the USB Image Writer tool that it comes with.

**Note:** Make sure you write it to the right device! You can damage your system if you
select a different device!
4. Now put the SD card in the assembled robot and power it up.
5. Put the robot on the ground.

If everything is successful within 30 to 60 seconds you should your robot’s lights start to change. If your robot is assembled correctly you should observe the following behavior:

1. All LEDs are **green** for 2 seconds.
2. All LEDs are **red** for 2 seconds.
3. The LEDs on the **right** side of the robot will be **red**, the rest are white, and the robot is turning **right** (i.e. *into* the direction of the red LEDs).
4. The LEDs on the **left** side of the robot will be **red**, the rest are white, and the robot is turning **left** (i.e. *into* the direction of the red LEDs).
5. The robot stops.
6. After a few seconds all LEDs are green.
7. End of the test. Everything appears to be assembled correct.

This is a video of what you should observe if your robot is assembled correctly:

![Robot with LEDs illuminated](image)

### 16.12. Troubleshooting

<table>
<thead>
<tr>
<th><strong>Symptom</strong></th>
<th>The omni-directional wheel, the back bumper or the Raspberry Pi are challenging to mount.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resolution</strong></td>
<td>Sometimes in life you have to push a little to make things happen. (But don’t push too much or things will break!)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Symptom</strong></th>
<th>I have a different color of the back patter plate. Do I get the wrong one?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resolution</strong></td>
<td>The color of this part can be of any and all are functionally equivalent.</td>
</tr>
</tbody>
</table>
Symptom: My battery is different from the one shown in the pictures! Did I get the wrong box?

Resolution: If there is a duckie in or on your box, you most probably got the right one. We support different battery models. All supported models are functionally equivalent, although the form factor varies.

Symptom: The robot doesn’t move in the right direction or moves backwards instead of forward.

Resolution: You have attached the motor wires in the wrong sockets. You will have to open the robot, fix that and test again.

Symptom: At the end of the procedure the LEDs don’t turn green and the LED under the camera is red.

Resolution: This means that the test procedure could not obtain an image from the camera. Check that your camera cable is attached correctly at both ends and that your camera is not broken.
This page is for the DB19 Duckiebot configuration, used in classes worldwide in 2018 and 2019.

These instructions are fundamentally the same as the instructions for the DB18. However this version includes DC motors with built-in encoders.

For older instructions from 2017 see the DB17 Duckiebot operation manual.

**Knowledge and activity graph**

- **Requires:** Duckiebot DB19 parts (get them [here](#)). You may want to check out the different configurations [here](#).
- **Requires:** A microSD card with the Duckiebot image already on it. This procedure is explained [here](#).
- **Requires:** 1.5 hours of assembly time.
- **Results:** An assembled Duckiebot in configuration DB19.

**Note:** Make sure you visit the FAQ section at the bottom of this page. This may already answer your comments, questions or doubts.

The complete assembly process is divided into 9 subparts. Make sure to complete them in the following order:

- **Part 1:** What is in the box?
- **Part 2:** Motors
- **Part 3:** Bottom chassis
- **Part 4:** Computation unit
- **Part 5:** Camera
- **Part 6:** Top chassis
- **Part 7:** Back bumper
- **Part 8:** Front bumper
- **Part 9:** Battery and Duckie
17.1. What is in the box

All the pieces in your Duckiebox are shown in Figure 17.1. Note that the battery and camera calibration pattern are not shown in the picture.

Moreover, you might have slightly different components than those shown: for example, different USB power cables, slightly different sets of screws or a camera mount and backplate of different colors. Do not worry, these instructions can be followed anyway.

![Figure 17.1. Components in Duckiebot package (Duckiebox).](image)

Some of the components in your Duckiebox will not be used at this stage, e.g., the traffic signs and stands. Keep these aside, they will come in handy in other parts of the book.

Finally, you should have several spares, especially for the mechanical bits (nuts and screws). These are included just in case you drop a few and can’t find them anymore (especially the tiny nylon ones like to hide in the cracks!).
17.2. Preliminary Steps

1) Unboxing

Unbox all of your components and lay them out on a flat surface. Ensure that you have well lit, uncluttered space to work on.
Although not necessary, you might find useful getting a small (M2.5) wrench to ease some of the passages.

2) Do the bumper bracers fit?

Take the rear bumper bracers and the back bumper. The back bumper will be mounted in the last steps as a press fit to the rear bumper bracers. Try to fit the bracers into the holes of the back bumper.
Some bumper bracers have a plastic protective film which is a residue of the manufacturing process. If you struggle in the press fitting, peel off the plastic cover from one side of the bracer. If this does not help, peel off the plastic cover from both sides.

3) Plastic cover

Peel the plastic cover from all the chassis parts (except the bumper bracers) on both sides.

4) Screws

Note that a few among all of your metal screws are special. They are “countersunk” screws. Keep these aside. They will be needed in Part 7: Back bumper.

Figure 17.2. Countersunk screws have a tapered head.
Every time you read M3x8 screw, a M3x10 will do the same trick. You can exchange them at will.
Do not exchange metal and nylon screws though. The latter are not electrically conductive and are passive protections to potential short circuits that can damage your Duckiebot beyond repair. This is especially true for - Part 3: Bottom chassis; make sure you use the correct nylon screws at that step.

Remember that instructions are your friend! At least at this stage, try to follow them precisely.

If regardless of this foreword you still choose to try and figure it out yourself, it’s ok, but for the love of what is precious, do not plug the battery in until you have performed a visual inspection here.

17.3. Motors
From the Duckiebox package take the following components:

- Bottom plate (1x)
- Blue dc motors (2x)
- Motor holders (4x)
- M3x30 screw (4x)
- M3 nuts (4x)

1) Step 1
Pass two of the motor holders through the openings in the bottom plate of the chassis as shown in Figure 17.3. If you have troubles fitting the holders, it is probably because you have not removed the protective plastic film. Note which holes you are using. You should be using the middle section ones.
Figure 17.3. How to mount the motor holders.

Put one motor between the holders as shown in Figure 17.4.

**Note:** Orient the motors so that their wires are pointing to the inside (i.e., towards the center of the plate).
**Note:** Use your screwdriver.

Use 2 M3x30 screws and 2 M3 nuts to secure the motor to the motor supports. Pass the screws through the holes from the outside inwards, then tighten the screws to secure the holders to the bottom plate of the chassis.

**Note:** You might find aligning the holes to be a little bit hard. It is easier to first get the top screw aligned and place the screw in, and then push in the side support piece in to place. If that still doesn’t work, get the bottom screw and the side support in first, align it, and secure with nuts. Then try to push the top screw through. You might have to use the screw to expand the hole a bit.

2) Step 2

Do the same for the second motor as well as shown in **Figure 17.5** and **Figure 17.6**

![Figure 17.5. How to mount the second motor supports](image-url)
Figure 17.6. How to mount the second DC-motor

**Note:** Tighten the motors firmly, as a wobbly motor might cause troubles later on.

3) Step 3

Wire the cables of the two motors through the bottom plate of the chassis and make sure they are well separated (Figure 17.7).

Figure 17.7. Cables through central hole

### 17.4. Bottom chassis

The Duckiebot is driven by controlling the wheels attached to the DC motors. Still, it requires a *passive* support on the back. In this configuration an omni-directional
wheel is attached to the bottom plate of the chassis to provide such support.

From the Duckiebox package take the following components: - Steel omni-directional wheel (1x) - M3x25 metal spacers (2x) - M3x10 metal screws (2x) - M3x10 nylon screws (2x)

1) Step 4

Assemble the omni-directional wheel as shown in figure **Figure 17.8**.

![Figure 17.8. Assembly of the omni-directional wheel](image)

2) Step 5

Then mount it to the bottom plate of the Duckiebot as shown in figure **Figure 17.9**.
Figure 17.9. Mounting the omni-directional wheel

**Note:** Sometimes you might need a bit of force to get the screw and the standoffs fully connected.

3) **Step 6**

From the Duckiebot kit take the following components:

- M3x10 screws (4x)
- M3x25 metal spacers (2x)
- M3x30 metal spacers (2x)
- 1x Rear bumper bracer

and mount the rear bracer to the bottom plate with the 25cm spacers (Figure 17.10)!
4) Step 7

Now use the M3x30 standoffs (longer ones) for the front of the Duckiebot (Figure 17.11).
5) Step 8

From the Duckiebot kit take the following components:

- Wheels (2x)

Figure 17.12 shows how to mount the big yellow driving wheels to the bottom chassis.

![Mounting the driving wheels](image)

**Note:** Mind that there is a particular orientation at which the wheels will fit. Don’t force them too much!

Figure 17.13 shows how the assembly should look like after mounting the wheels.
17.5. **Computation unit**

From the Duckiebot kit take the following components:

- Raspberry Pi 3B+ (1x)
- Heat sink (1x) (or 2x if there are two of them)
- Camera cable (1x) (This should be in the same box as the camera)
- Micro SD card (1x)

**Note:** You probably have two heat sinks, make sure you use the bigger one for sure.
The smaller one can be used as shown in the following pictures or apply it on the USB port you will later use for the external 32GB USB dongle.

**Note:** You will notice there is a camera cable already attached to the camera. We won’t use this one since it is a bit short. You will find a longer camera cable inside the same box as the camera.

*Figure 17.14* shows the components needed to complete the following steps.

![Figure 17.14. The bigger heat sinks and the Raspberry Pi 3 B+.

1) Step 9

Peel the cover from the bottom of the heat sink and place it on the Raspberry Pi microchip, as shown in *Figure 17.15*. Make sure to put it on the
2) Step 10

Insert the SD card in the slot as shown in figure Figure 17.16.

**Note:** If the card is not flashed yet, do the initialization first: Unit C-3 - Setup Duckiebot SD Card, but only until the section Section 3.2 - Burn the SD card, because you will need the fully assembled Duckiebot for the following steps of the Duckiebot Initialization.
3) Step 11

Make sure that the visible metal connectors of the camera cable match the ones in the Raspberry Pi (Figure 17.17), then plug in the cable and push down the black wings to fasten the connection, making sure it “clicks”.

**Note:** Please be aware that different camera cables have the text on different sides and with different orientation, do **not** use it as a landmark.
Then close the camera plug once the cable is pushed in properly. You can check by pulling at the cable once you have closed the plug and you mustn’t be able to tear it out at all.

4) Step 12

From the Duckiebot kit take the following components:
• Duckiebot Hut (1x)
• Top plate of the chassis (1x)
• M2.5x12 nylon spacers (4x)
• M2.5x10 Nylon screws (4x)
• M2.5x4 Nylon spacers (4x)
• M2.5 nylon nuts (4x)
• USB to micro USB cables (2x)
• Set of three female to female (F/F) jumper wires (1x)

Figure 17.19 shows the components needed to complete this part of the tutorial.

Figure 17.19. The parts needed to assemble the Raspberry Pi and the Hut.

Note: In the newest version of the Hut, the pins are removed. This is to prevent them from being shorted. You will not need them so you can neglect them in the next few pictures.

Note: It is more convenient to not separate the jumper cables, but leave them in two
sets of three.
Place the spacers on the bottom part of the Duckiebot hut and secure them using the M2.5 nylon nuts, see figure Figure 17.20

5) Step 13
Mount the pre-assembled Hut and Raspberry Pi to the top plate with nylon screws as shown in figure Figure 17.21
Figure 17.21. Mount computation unit

**Note:** Make sure that the Raspberry Pi GPIO pins fit into the Hut connector.

6) Step 14

Connect the F/F jumper wire to the pins on the Hut as shown in figure Figure 17.22 and make sure the camera cable is put in the slit of the Hut.
Finally, attach the two USB cables to the power plugs on the Hut and Raspberry Pi, as in Figure 17.23.

17.6. Camera
From the Duckiebox package take the following components:
1) Step 16

Open the camera cable plug on the Raspberry Pi camera as shown in figure Figure 17.24.

Figure 17.24. Open camera cable plug

**Note:** Make sure the cable is routed through the camera mount as well!

Close the plug once the cable is pushed in completely as in Figure 17.25
Figure 17.25. Close camera plug

Insert the M3x10 nylong screws from the front to screw the camera to its mount, see Figure 17.26

Figure 17.26. Screw the camera to its mount
**Note:** Make sure the camera is as parallel to its mount as possible, i.e. facing downwards at this very angle.

2) **Step 17**

Screw the camera mount to the top plate with three M3x10 screws and metal M3 nuts, see **Figure 17.27**.

![Figure 17.27. Screw the mount to the top plate](image)

**17.7. Top chassis**

From the Duckiebox package take the following components: - Metal screws M3x10 (4x) - Back bumper bracer (1x)

1) **Step 18**

Put the top plate on the bottom part of the chassis as shown in figure **Figure 17.28** but do not screw it in yet. This is how it should look like in the end.
Now open the slots for the cables of the DC motors on the Hut with a small screwdriver and make sure you mount the cables in correctly, see figure Figure 17.29. Probably you have to take the top and bottom part of the chassis apart again.
3) Step 20

Place the second rear bumper bracer on top of the chassis spacers in the back of the chassis. Again, the holes should align (Figure 17.30). If you experience a slight misalignment, carefully pass an M3 screw through the top plate and the top bumper bracer first, then move the standoff until you can plug the screw in. At the end, put in the second M3 screw as well.

![Figure 17.30. Rear bumper bracer positioning](image)

4) Step 21

Fasten the top plate in the front using the other two M3x10 screws, as in(Figure 17.31).
5) Step 22

You can use a zip tie to help with cable management (Figure 17.32). Tip: you can use the same zip tie to keep the camera cable down and the USB cables below from touching the wheels.
17.8. Back bumper

From the Duckiebot kit take the following components:

- Back bumper (1x)
- M3x10 nylon screws (2x)
- M3x10 metal screws (4x)
- M3x10 countersunk screws (2x)
- M3 metal nut (4x)
- M3x25 spacers (2x)
- Back pattern plate (1x)
- Circle pattern sticker (1x)

**Note:** In the picture, nylon screws are used. However you probably used them to assemble the omni-directional wheel, then use the metal screws.

Figure 17.33 shows the components needed to complete this upgrade.
Figure 17.33. The parts needed to mount the circle grid holder.

**Note:** You could have a back plate of a different color with respect to the picture, e.g., black or white. They are all functionally equivalent.

1) Step 23

Mount the spacers using the *metal* screws, as in Figure 17.34.

Figure 17.34. How to mount the spacers
Note: Pay attention that the spacers should be in the same direction as the LEDs.

2) Step 24

Fasten the back plate to the back bumper standoffs using the countersunk screws. The back plate is not symmetric, place it such that it is centered with respect to the back bumper as shown in picture Figure 17.35.

![Figure 17.35. Mount the pattern plate](image)

3) Step 25

Peel off the sticker with the circle pattern and put it on the pattern plate mounted before, see picture Figure 17.36.

![Figure 17.36. Pattern sticker](image)
4) Step 26

Connect a new jumper wire (F/F) to the connector on the back bumper, see picture Figure 17.37.

Figure 17.37. Jumper wire to back bumper

No matter what exact solution you choose concerning the cable management, the objective is to avoid any cables from touching the wheels in the end. One solution could be something like it is shown in figure Figure 17.38.
5) Step 27

Screw the back bumper to the rest of the chassis with four metal screws and nuts, see picture Figure 17.39.

*Note:* Pay attention that the pins of the bumper bracers are properly put in the holes of the back bumper. It might be a little bit difficult to place it properly.
**Note:** Wire the cable mounted in step 26 through the top plate of the chassis as shown in the picture as well.

### 17.9. Front bumper

From the Duckiebot kit take the following components:

- Front bumper (1x)
- M3x10 metal screws (2x)

#### 1) Step 28

At that point you should have arrived with two loose cables in the front. Make sure to connect both of them to the front bumper. The one from step 26 on the right hand side and the other on the left hand side, see picture Figure 17.40.

![Figure 17.40. Wiring the front bumper](image)

#### 2) Step 29

First, click in the front bumper without any screws. Maybe you have to bend the top and bottom plate of the chassis a little bit in order to make it fit, see pictures Figure 17.41 and Figure 17.42.
Figure 17.41
3) Step 30

Now use the two M3x10 metal screws to finally fix the top plate of the chassis properly. This will now prevent the front bumper from falling out again. Probably you will have to move the standoffs a little bit in order to make the holes flush, see figure Figure 17.43.
17.10. Battery and Duckie

From the Duckiebot kit take the following components:

- Battery (1x)
- Zip tie (1x)
- Duckie (N+1x)

1) Visual inspection

Before plugging in the battery, make sure the Hut’s GPIO pins are (a) not touching each other, (b) not touching any metal screws (in case you did not follow these instructions exactly) and (c) are free from any external object that might have gotten stuck there during the assembly process.

**Note:** If the GPIOs of the Hut get shorted (there is an electrically conductive connection between them) when you plug in the battery, you might damage the Raspberry Pi beyond repair!
2) Step 31

Place the battery and fix it using a zip tie (Figure 17.44), then connect the USB cables.

**Note:** Do not unplug/replug the bumper wires when the power is on. You could break the bumpers!

Figure 17.44. Fixing the top plate

There are different batteries concerning the outer dimensions, so the result could look like either of the following pictures [Figure 17.45](#) or [Figure 17.46](#):
Now place a Duckie on top of your brand new Duckiebot, but be careful not to hurt the Duckie!

3) Check the outcome
Note: As a final check, verify that no cable is touching the wheels. You can use the provided zip ties to ensure that cables stay out of the way.

17.11. Troubleshooting

Symptom: The omni-directional wheel, the back bumper or the Raspberry Pi are challenging to mount.
Resolution: Sometimes in life you have to push a little to make things happen. (But don’t push too much or things will break!)

Symptom: I have a different color of the back patter plate. Do I get the wrong one?
Resolution: The color of this part can be of any and all are functionally equivalent.

Symptom: My battery is different from the one shown in the pictures! Did I get the wrong box?
Resolution: If there is a duckie in or on your box, you most probably got the right one. We support different battery models. All supported models are functionally equivalent, although the form factor varies.

Symptom: The robot doesn’t move in the right direction or moves backwards instead
of forward.

**Resolution:** You have attached the motor wires in the wrong sockets. You will have to open the robot, fix that and test again.
UNIT C-18

Debug - Duckiebot Update

KNOWLEDGE AND ACTIVITY GRAPH

**Requires:** A Duckiebot that has been initialized

**Requires:** A computer with an Ubuntu OS,

**Requires:** Duckietown Shell, Docker, etc., as configured in Unit C-1 - Setup - Laptop.

**Requires:** Duckietown Token set up as in Unit C-2 - Setup - Account.

**Requires:** An internet connection to the Duckiebot, configured as in Unit C-19 - Operation - Networking.

**Results:** An up to date Duckiebot!

18.1. **Understand what is the difference between OTA update and Release Update**

The update method described in this page will allow you to receive an **On The Air** (OTA) update within the distribution you so choose (e.g., *daffy*), and it can improve your Duckiebot performance. Note that this is different from using the *init_sd_card* tool: Using the *init_sd_card* tool will only provide the latest release version, not the latest Duckiebot software version.

18.2. **Update Duckiebot container using dts command**

If your Duckiebot has not been used for a while and a new image has been released, you don’t necessarily need to re-flash the Duckiebot image as described in the initialization procedure. Instead, you can use `dts duckiebot update` command to update your Duckiebot.

```bash
$ dts duckiebot update DUCKIEBOT_NAME
```

You will see a prompt similar to this:
Type in `y` for yes to continue updating. If you would like to abort, you can use `Ctrl-C` to stop the update.

**Note:** This process is expected to take a while to complete.
UNIT C-19
Operation - Networking

This page is for the DB18 configuration used in classes in 2018. For last year’s instructions see here.

KNOWLEDGE AND ACTIVITY GRAPH

<table>
<thead>
<tr>
<th>Requires:</th>
<th>A Duckiebot that is initialized according to Unit C-3 - Setup Duckiebot SD Card.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires:</td>
<td>Patience (channel your inner Yoda).</td>
</tr>
<tr>
<td>Results:</td>
<td>A Duckiebot that you can connect to and that is connected to the internet.</td>
</tr>
</tbody>
</table>

The instructions here are ordered in terms of preference, the first being the most preferable and best.

By default on boot your robot will look for a network with a “duckietown” SSID, unless you changed it in the SD card flashing procedure. You can connect to your robot wirelessly by connecting to that network.

This page describes how to get your robot connected to the wide-area network (internet).

19.1. Add WiFi Networks without reinitializing the SD card

To add networks at a later stage or modify existing settings, edit the file wpa_supplicant.conf in the main partition of the SD card.

For robots based on Raspberry Pi, (e.g., DB17, DB18, DB19), this file is located at /etc/wpa_supplicant/wpa_supplicant.conf in the root partition; For robots based on Nvidia Jetson Nano, (e.g., DB21M), this file is located at /etc/wpa_supplicant.conf in the APP partition;

New networks can be created by adding a new network= paragraph, and then entering the network information. An example network configuration is shown below:
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=CH

network={
    id_str="network_1"
    ssid="comnet23243"
    psk="MSNDJWKE32"
    key_mgmt=WPA-PSK
}
network={
    id_str="network_2"
    ssid="TPlink23432"
    psk="ksnbn4wn3"
    key_mgmt=WPA-PSK
}

19.2. Testing if your Duckiebot is Connected to the Internet

Some networks block pings from passing through, so a better way is to execute on your duckiebot:

⚠️ $ sudo curl google.com

which will try to download the Google homepage. If it is successful, you should see an output like:

```html
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

19.3. Option 1: Connect your Duckiebot to the internet through a WiFi
router that you control

If you are working from your home, for example, you simply need to make the Duckiebot connect to your home network. You may have input the proper SSID and password when you initialized the SD card, in which case, your Duckiebot should be connected to the internet already.

If you didn’t enter the right SSID and password for your network or you want to change you need to connect to your robot somehow (e.g. with Ethernet) and then edit the file `/etc/wpa_supplicant/wpa_supplicant.conf` as explained in the Duckiebot initialization procedure.

This is the best option.

19.4. Option 2: Bridge the internet connection through your laptop with Ethernet

This method assumes that you can connect your laptop to a network but it is one that you don’t control or is not open. For example, on campus many networks are more protected, e.g., with PEAP. In that case, it can be difficult to get your configurations right on the Duckiebot. An alternative is bridge the connection between your laptop and your Duckiebot whenever you need internet access on the robot.

1) Ubuntu

1. Connect your laptop to a wireless network.
2. Connect the Duckiebot to your laptop via an Ethernet cable.
3. Make a new Ethernet connection:
4. Network Settings... (or run the command `nm-connection-editor`)
5. Click “Add”
6. Type -> Ethernet
7. Connection Name: “Shared to Duckiebot”
8. Select “IPV4” tab
9. Select Method
10. Select “Shared to other computers”
11. Click apply.

Now, you should be able to SSH to your Duckiebot:
Check whether you can access the internet from your Duckiebot:

```
ssh hostname
```

Now, try to pull a Docker image:

```
sudo docker pull duckietown/rpi-simple-server # This should complete successfully
```

If the previous command does not work, you may need to change the system date. To do so, run the following command:

```
sudo date -s "2018-09-18 15:00:00" # Where this is the current date in YYYY-MM-DD HH-mm-ss
```

Untested instructions [here](#)

**19.5. Option 3: Push Docker Images from Laptop**

Since we are primarily using the internet to pull Docker images, we can simply connect the laptop and the Duckiebot then push Docker images from the laptop over SSH like so:

```
docker save duckietown/image-name | ssh -C hostname docker load
```

Then the image will be available on your Duckiebot.

If you can connect to your laptop (e.g. through a router) but do not have internet access then you can proceed for now, but everytime you see a command starting with:

```
docker run ...
```


note that you will need to pull onto your laptop and push to your Duckiebot in order to load the latest version of the image.

19.6. Troubleshooting

1) I cannot ping the Duckiebot

| Symptom: | ping `robot_name` does not work. |
| Resolution: | Check if your laptop and Duckiebot are connected to the same network. |

Additional debugging steps:

- Step 1: Check that your Raspberry Pi is responsive by observing the blinking LED on the Raspberry Pi.
- Step 2: Connect your Duckiebot with the laptop using the ethernet cable. Check if you are able to ping the Duckiebot. This will provide you an hint if there is an issue with the robot or network.
- Step 3: Check that this file: `/etc/wpa_supplicant/wpa_supplicant.conf` contains all the wifi networks in the correct syntax that you want to connect.
- Step 4: If it’s your private access point, then you can access your router, typically connecting to `192.168.0.1`, where you can see all the devices connected. Make sure that both your Duckiebot and your laptop are in the list.
- Step 5: Check the file `~/.ssh/config` has the correct name hostname with `hostname.local` defined.

19.7. I cannot access my Duckiebot via SSH

| Symptom: | When I run `ssh `robot_name`.local` I get the error `ssh: Could not re-solve hostname `robot_name`.local.` |
| Resolution: | Make sure that your Duckiebot is ON. Connect it to a monitor, a USB mouse and a keyboard. Run the command: |

```
$ sudo service avahi-daemon status
```

You should get something like the following:
Avahi is the module that in Ubuntu implements the mDNS responder. The mDNS responder is responsible for advertising the hostname of the Duckiebot on the network so that everybody else within the same network can run the command `ping robot_name.local` and reach your Duckiebot. Focus on the line containing the hostname published by the `avahi-daemon` on the network (i.e., the line that contains `robot_name_in_avahi.local`). If `robot_name_in_avahi` matches the `robot_name`, go to the next Resolution point. If `robot_name_in_avahi` has the form `robot_name-XX`, where `XX` can be any number, modify the file `/etc/avahi/avahi-daemon.conf` as shown below.

Identify the line

```
use-ipv6=yes
```

and change it to

```
use-ipv6=no
```

Identify the line

```
#publish-aaaa-on-ipv4=yes
```

and change it to

```
publish-aaaa-on-ipv4=no
```
Restart Avahi by running the command

```
$ sudo service avahi-daemon restart
```

19.8. I can SSH to the Duckiebot but not without a password

Check the file `.ssh/config` and make sure you add your `ssh` key there, in case it doesn’t exists.

The `init_sd_card` procedure should generate a paragraph in the above file in the following format:

```
# --- init_sd_card generated ---
Host duckiebot
  User duckie
  Hostname duckiebot.local
  IdentityFile /home/user/.ssh/DT18_key_00
  StrictHostKeyChecking no
# ------------------------------
```

Do:

```
$ ssh-keygen -f "/home/user/.ssh/known_hosts" -R hostname.local
```

It will generate a key for you, if it doesn’t exists.

19.9. Unable to communicate with Docker

**Symptom:** Error message appears saying `I cannot communicate with docker`. Also a warning `"DOCKER_HOST" is set to hostname.local` is present.

**Resolution:** Unset the `DOCKER_HOST`, running:

```
$ unset DOCKER_HOST
```

19.10. Can ping but the robot doesn’t move with the virtual joystick

**Symptom:** You can ping the robot, `ssh` into it, start the demos, but the commands from the virtual joystick do not seem to reach the robot.
A possible cause is that your computer’s firewall is blocking the incoming traffic from the robot.

**Resolution:** Check the settings for the firewall on your computer and make sure that any incoming traffic from the IP address of the robot is allowed on all ports. Keep in mind that if your robot’s IP address changes, you might need to update the rule.
20.1. When and why should I run this procedure?

This procedure flashes the microcontroller on the Duckietown HUT. This microcontroller is responsible for translating the duty cycle commands from the onboard computer to actual \textit{PWM} signals that control the motors and the LEDs (because they are “addressable” LEDs) of the Duckiebots.

A typical example of when is necessary to flash the microcontroller is when commands are sent to the motors, e.g., through keyboard control, the motors signals on the dashboard/mission control show that signals are correctly being sent, but the Duckiebot does not move.

This procedure will not be useful to fix problems such as one motor working and not the other, or LEDs showing unexpected colors when the motors work.

20.2. How to flash the microcontroller

Ssh into your Duckiebot by running:

```
$ ssh duckie@DUCKIEBOT NAME.local
```

Install the packages needed to compile the microcontroller firmware:
# Debug - Re-flash Microcontroller

Clone the firmware for the microcontroller using the following command:

```bash
$ git clone https://github.com/duckietown/fw-device-hut.git
```

Navigate inside the repository you cloned:

```bash
$ cd fw-device-hut
```

Copy the `avrdude.conf` file in the `/etc` folder of the robot. If you are running a Duckiebot with an NVIDIA Jetson Nano board run:

```bash
$ sudo cp _avrdudeconfig_jetson_nano/avrdude.conf /etc/avrdude.conf
```

else, if you have a Raspberry Pi based Duckiebot, use:

```bash
$ sudo cp _avrdudeconfig_raspberry_pi/avrdude.conf /etc/avrdude.conf
```

Then test the `avrdude` and set the low-level configuration with:

```bash
$ make fuses
```

A successful outcome looks like:

```
avrdude: verifying ...
avrdude: 1 bytes of efuse verified

avrdude: safemode: Fuses OK (E:FF, H:DF, L:E2)

avrdude done. Thank you.
```
If you see the message `make: warning: Clock skew detected. Your build may be incomplete. or the process is not stopping, stop the process pressing `ctrl-c and run:

```
$ find -exec touch \{{}\} \;
```

And then retry running the `make fuses` command.

Remove all temporary files by running:

```
$ make clean
```

Compile the firmware and upload it to the microcontroller:

```
$ make
```

The resulting output should be:
......

```
sudo avrdude -p attiny861 -c linuxgpio -P -q -U flash:w:main.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e930d (probably t861)

avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed

    To disable this feature, specify the -D option.

avrdude: erasing chip

avrdude: reading input file "main.hex"

avrdude: input file main.hex auto detected as Intel Hex

avrdude: writing flash (2220 bytes):

Writing | ################################################## | 100% 0.75s

avrdude: 2220 bytes of flash written

avrdude: verifying flash memory against main.hex:

avrdude: load data flash data from input file main.hex:

avrdude: input file main.hex auto detected as Intel Hex

avrdude: input file main.hex contains 2220 bytes

avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.58s

avrdude: verifying ...

avrdude: 2220 bytes of flash verified

avrdude: safemode: Fuses OK (E:FF, H:DF, L:E2)

avrdude done. Thank you.
```

Remove the cloned repository to free up space:
and finally reboot the Duckiebot:

```bash
$ sudo reboot
```

After reboot your Duckiebot should move normally and LEDs respond nominally. The Dashboard / components page will show a green status for the HUT, too.
This part describes a set of demos and exercises that demonstrate the functionality of the robots. Some of these demos require different hardware on the robot, or in Duckietown.
UNIT D-1
General Demo Running Procedure

This page describes the basic procedure for running demos. Some demos have specific requirements that must be adhered to, but the general process of running them through the Duckietown shell is standardized.

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** A Duckiebot in DB18 configuration that is initialized
- **Requires:** Laptop configured, according to Unit C-1 - Setup - Laptop.
- **Requires:** Other requirements are demo specific, see the specific pages

1.1. Start demos

In the Duckietown software repo, there are two main types of launch files: node-specific (nuclear) ones, and demo launch files. A node launch file handles only a single node and its parameters. A demo launch file combines multiple of node launch files and adds the necessary connections in order to stage demos that use dozens of nodes. The launch procedure for both types is very similar. The generic command is:

```
$ dts duckiebot demo --duckiebot_name $DUCKIEBOT_NAME --demo_name $DEMO_NAME --package_name $PACKAGE_NAME --image duckietown/$IMAGE:daffy-arm32v7
```

This command will start the $DEMO_NAME.launch launch file in the $PACKAGE_NAME package from the duckietown/$IMAGE:daffy-arm32v7 Docker image on the $DUCKIEBOT_NAME Duckiebot.

**Warning:** If you want the image to run on your computer, you should use $:daffy-amd64$ tag, if you want to run it on duckiebot, you should use $:daffy-arm32v7$ as the images are architect dependent.

**Note:** Currently daffy is the development branch and the dts commands work by default with the master19 version. That is why you should always specify the image.
with the daffy tag!
You can find the specific command for each demo in the corresponding part of the book.

1.2. Debug options
You can open a terminal in the container running the demo you want by appending the option `--debug` to the command. An example is:

```
$ dts duckiebot demo --duckiebot_name DUCKIEBOT_NAME --demo_name DEMO_NAME --package_name PACKAGE_NAME --image duckietown/IMAGE:daffy --debug
```

This enables you to access to the ROS debug informations of the nodes that are launched. This is the same output that you can see in the logs window of the particular container on Portainer.
UNIT D-2

General Exercise Running Procedure

This page describes the exercises infrastructure. This infrastructure affords a seamless method to build on existing baselines, test them in simulation, test them on robot hardware either remotely or locally, and then evaluate and submit them as challenges with the AIDO challenges infrastructure.

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** A Duckiebot that is initialized
- **Requires:** Laptop configured, according to Unit C-1 - Setup - Laptop.
- **Requires:** That you are able to to submit a challenge according to Unit C-3 - Make your first submission.

2.1. Video Tutorial

![Video Tutorial](image)

Figure 2.1. Learn how to use the dt-exercises infrastructure.

2.2. Getting Started

Fork the dt-exercises repository and clone it onto your computer.

Set up an upstream remote. From inside the directory you just cloned:

```
$ git remote add upstream git@github.com:duckietown/dt-exercises.git
```
Now to pull anything new from the original repository you can do:

```bash
$ git pull upstream daffy
```

Enter the `dt-exercises` folder that you just cloned do:

```bash
$ cd dt-exercises
```

In here you will see a number of folders. Each folder corresponds to an exercise.

### 2.3. The Anatomy of an Exercise

Exercises should contain all of the following:

1) **`config.yaml`**

This contains information about the exercise. Example:

```yaml
# Exercise configuration file

agent_base: "duckietown_baseline" # the agent base image to use
ws_dir: "exercise_ws" # directory that contains the code
agent_run_cmd: "run_all.sh" # the script in "launchers" to run the agent with
notebooks:
  - notebook:
      package_name: "encoder_pose"
      name: "odometry_activity"
  - notebook:
      package_name: "lane_controller"
      name: "control_activity"
```

The `agent_base` indicates which image to use as a baseline to build from. The mappings are listed [here](#). Many of the existing exercises are build on the `duckietown_baseline` image which contains all of the code in the `dt-core repository`.

**Note:** In the case that you are using the `duckietown_baseline`, any package/node
that you create in the `exercise_ws` directory will be run instead of the one in the `dt-core repository` if the package name and node name match. This is achieved through `workspace overlaying`.

The `ws_dir` indicates the name of the subdirectory that contains the code that should be mounted into the image.

The `agent_run_cmd` indicates the command that should be run when the container is run to start things.

The `notebooks` contains the list of the notebooks that have to be converted to python scripts. For each notebook must be specified the name of the `notebook name`, and the name of the package where the generated script has to be copied `package_name`. Multiple notebooks can be listed.

2) `exercise_ws`  3) `assets`  4) `launchers`  5) `notebooks`

2) `exercise_ws`

As indicated above, the `exercise_ws` directory is where the code should go. For the case of ROS packages, they should go inside a `src` directory inside `exercise_ws`.

3) `assets`

The `assets` folder contains two subfolders, `setup` and `calibrations`.

- The `setup` subfolder contains all the configuration information (mostly in terms of environment variables) that are needed to run the various docker images that we are running (which depends on the configuration).
- The `calibrations` folder contains robot calibrations with a similar directory structure as is on the Duckiebot.

4) `launchers`

The `launchers` folder contains scripts that can be run by the agent. Specifically, the one that is indicated in the `config.yaml` file will be run by default when the agent container is run or when your exercise is submitted through the challenges infrastructure.

**Note:** You can specify different launchers to run depending on whether you are testing/developing with the `exercises` infrastructure or submitting through the `challenges` infrastructure.

5) `notebooks`

The `notebooks` folder contains pedagogical notebooks that can be run. Some parts of
the notebooks indeed are dedicated to tests, to check if the code is working properly before starting the simulation or testing on the Duckiebot.

Before running the test cells make sure you run also the cells with the code. There are different ways to run a cell:

1. click on the `play` button on the top left of the cell.
2. pressing `crtl+enter`.

The code in the notebooks can also be compiled and become accessible inside the code in the `exercise_ws` directory.

In order to do so, from inside the exercise folder run:

```
$ dts exercises build
```

This command convert the notebook into a python script and place it inside the package in `exercise_ws` directory specified in the `config.yaml` file.

The same is when running the `run` command, with the difference that in this case the ROS workspace is not built:

```
$ dts exercises run options
```

6) **requirements.txt**

The `requirements.txt` file contains any specific python requirements that you need for your submission. Note that these are requirements need over and above the `base image`.

7) **Dockerfile**

The `Dockerfile` contains the recipe for making your submission. In the normal case, this is relatively straightforward. We install the requirements, copy in the code and run a `launcher`.

2.4. The Exer**Exercises API**

In the following we will describe the current commands that are supported within `dts exercises` and how they are used.

1) **Building your code**
You can start by building your code with:

```
$ dts exercises build
```

If you go inside the `exercises_ws` folder you will notice that there are more folders that weren’t there before. These are build artifacts that persist from the building procedure because of mounting.

**Note:** every time you run a `dts exercises` command you have to be inside an exercise folder or you will see an error.

2) Testing your code

With `dts exercises test` you can test your agent:

1. in the simulated environment,
2. on your robot but with the agent code running on your laptop,
3. with all of the code running on your robot.

**Running in Simulation:**

You can run your current solution in the gym simulator with:

```
$ dts exercises test --sim
```

Then you can look at what’s happening by looking through the “novnc” browser at http://localhost:8087.

If you are running an exercise with a ROS-based baseline, you can use all of the existing ROS tools from this browser desktop. For example, open up the `rqt_image_view`, resize it, and choose `/agent/camera_node/image/compressed` in the dropdown. You should see the image from the robot in the simulator.

You might want to launch a virtual joystick by opening a terminal and doing:

```
$ dt-launcher-joystick
```

If you are running the `duckietownBaseline`, by default the duckiebot is in joystick control mode, so you can freely drive it around. You can also set it to `LANE FOLLOWING` mode by pushing the `a` button when you have the virtual joystick active. If you do so you will see the robot move forward slowly and never turn.
You might also explore the other outputs that you can look at in `rqt_image_view`. Also useful are some debugging outputs that are published and visualized in `RViz`. You can open `RViz` through the terminal in the `novnc` desktop by typing:

```
$ rviz
```

In the window that opens click “Add” the switch to the topic tab, then find the `segment_markers`, and you should see the projected segments appear. Do the same for the `pose_markers`.

Another tool that may be useful is `rqt_plot` which also can be opened through the terminal in novnc. This opens a window where you can add “Topics” in the text box at the top left and then you will see the data get plotted live.

All of this data can be viewed as data through the command line also. Take a look at all of the `rostopic` command line utilities.

TODO: add pictures.

**Testing Your agent on the Robot:**

If you are using a Linux laptop, you have two options, local (i.e., on your laptop) and remote (i.e., on the Duckiebot). If you are Mac user stick to the remote option. To run “locally”

```
$ dts exercises test --duckiebot_name ROBOT_NAME --local
```

To run on the Duckiebot:

```
$ dts exercises test --duckiebot_name ROBOT_NAME
```

The following was marked as "todo".

TODO: add pictures.

Location not known more precisely.

Created by function n/a in module n/a.
In both cases you should still be able to look at things through novnc by pointing your browser to http://localhost:8087. If you are running on Linux, you can load up the virtual joystick and start lane following as above.

*Interactive Mode:*

You may find it annoying to completely shut down all of the running images and restart them to make a simple change to your code. To make things faster, you can use the `--interactive` flag with `dts exercises test`.

In this case, when all of the containers other than the agent have started, you will be given a command line inside the agent container (overriding the command specified in `config.yaml`). From here you can run your launcher from the command line manually.

**Note:** If you are running an exercise based on the `duckeitown_baseline` image, the first time you will have to start the “interface” part of the agent. To do this run

```
laptop container $ launchers/run_interface.sh
```

You will see some output of some ros nodes starting. At the end, if you push ENTER you will get your command line back. Then you can run the lane_following demo using your lane_controller by running

```
laptop container launchers/run_agent.sh
```

You can do the normal thing of going to novnc and putting it into lane following mode or driving around with the joystick or whatever.

If you would like to change your code and re-run, just edit your code on your laptop, and then go to that terminal and do CTRL-C. You will see everything start to shut down. Then you can simply rerun the agent and it will have the new code that you just modified since it’s mounted into the agent container. So just do `launchers/run_agent.sh` again and it will start up again.

**Note:** You will see an output from the anti-instagram node saying it’s waiting for the first image. Don’t worry, if you go to novnc and put the agent in lane following mode or drive with the joystick, it will start to receive images and that output will go away.

**Note:** There is a timeout on the simulator, so if you do CTRL-C and then spend a while editing your code, it’s likely that the simulator will have shut down. So either
leave it running while you edit your code or just restart everything. You can get out of your terminal by typing

```
  laptop container $ exit
```
UNIT D-3
Lane following

This is the description of lane following demo.

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** Wheels calibration completed.
- **Requires:** Camera calibration completed.
- **Requires:** Joystick demo has been successfully launched.

3.1. Video of expected results

![Outcome of the lane following demo](image)

Figure 3.1. Outcome of the lane following demo.

3.2. Duckietown setup notes

Assumption about Duckietown:
- A Duckietown with white and yellow lanes. No obstacles on the lane.
- Layout conform to Duckietown Appearance Specifications
- Required tiles types: straight tile, turn tile
- Configurated wireless network for communicating with Duckiebot.
- Good consistent lighting, avoid natural lighting.
3.3. Duckiebot setup notes

- Make sure the camera is heading ahead.
- Duckiebot in configuration **DB18**

3.4. Pre-flight checklist

- Turn on joystick (if applicable).
- Turn on battery of the duckiebot.
- Place duckiebot in lane so that enough of the lane lines are visible to the camera.
- Verify you can ping your duckiebot over the network.
- **IMPORTANT** Make sure no containers are running on the duckiebot which use either the camera or joystick. We will run these ROS nodes together in a new container.

3.5. Demo instructions

1) Start the demo containers

Running this demo requires almost all of the main Duckietown ROS nodes to be up and running. As these span 3 Docker images (**dt-duckiebot-interface**, **dt-car-interface**, and **dt-core**). The **dt-duckiebot-interface** and **dt-car-interface** container typically starts with robot startup. You will need to start **dt-core** manually.

First, check to make sure that **dt-duckiebot-interface** and **dt-car-interface** are running on your duckiebot via portainer, if not, do:

```
$ dts duckiebot demo --demo_name duckiebot-interface --duckiebot_name DUCKIEBOT_NAME --package_name duckiebot_interface --image duckietown/dt-duckiebot-interface:daffy-arm32v7

$ dts duckiebot demo --demo_name car-interface --duckiebot_name DUCKIEBOT_NAME --package_name car_interface --image duckietown/ dt-car-interface:daffy-arm32v7
```

Then, we are ready to start the high-level pipeline for lane following:

```
$ dts duckiebot demo --demo_name lane_following --duckiebot_name DUCKIEBOT_NAME --package_name duckietown_demos
```
You have to wait a while for everything to start working. While you wait, you can check in Portainer if all the containers started successfully and in their logs for any possible issues.

2) Make your Duckiebot drive autonomously!

If you have a joystick you can skip this next command, otherwise we need to run the keyboard controller:

```
$ dts duckiebot keyboard_control $DUCKIEBOT_NAME
```

Start the lane following. The Duckiebot should drive autonomously in the lane. Intersections and red lines are neglected and the Duckiebot will drive across them like it is a normal lane. You can regain control of the bot at any moment by stopping the lane following and using the (virtual) joystick. Resuming the demo is as easy as pressing the corresponding start button.

Et voilà! We are ready to drive around autonomously.

3) Visualize the detected line segments (optional)

This step is not necessary but provides a nice visualization of the line segments that the Duckiebot detects.

For that, we need to make `lane_filter_node` publish all the image topics. To do that, you can use `start_gui_tools` to get a shell that is connected to the ROS system.

```
$ dts start_gui_tools $DUCKIEBOT_NAME
```

Then, set the ROS parameter `verbose` to `true`:

```
$ rosparam set /$DUCKIEBOT_NAME/line_detector_node/verbose true
```

so that `line_detector_node` will publish the image_with_lines.

**Note:** For this part, you can also use no-vnc, or directly use any container that can communicate with the ROS system.
Now you can run `rqt_image_view` and select the `/DUCKIEBOT_NAME/line_detector_node/image_with_lines` and you should see something like this:

![Figure 3.2. Outcome of the line detector node.](image)

4) Extras

Here are some additional things you can try:

- Get a [remote stream](#) of your Duckiebot.
- Try to change some of the ROS parameters to see how your Duckiebot’s behavior will change.

3.6. Troubleshooting

1) The duckiebot does not move

- Check if you can manually drive the duckiebot
- Try re launching `dts duckiebot keyboard_control hostname`
- Check if ROS messages are received on the robot on the `hostname/joy` topic

2) The Duckiebot does not stay in a straight lane

- Check `rqt_image_view` and look at `image_with_lines`.
- Check if you see enough segments. If not enough segments are visible, reset the Anti-Instagram filter.
- Check if you see more segments and the color of the segments are according to the color of the lines in Duckietown
- Check your camera [calibrations](#) are good.

3) The Duckiebot does not drive nicely through intersections
This feature is not implemented for this demo. The duckiebot assumes only normal lanes during this demo.

4) The Duckiebot cuts white line while driving on inner curves (advanced)

Solution (advanced):
Set alternative controller gains. While running the demo on the Duckiebot use the following to set the gains to the alternative values:

```
$ rosparam set /DUCKIEBOT_NAME/lane_controller_node/k_d -45
$ rosparam set /DUCKIEBOT_NAME/lane_controller_node/k_theta -11
```

Those changes are only active while running the demo and need to be repeated at every start of the demo if needed. If this improved the performance of your Duckiebot, you should think about permanently change the default values in your catkin_ws.
UNIT D-4
Lane following with vehicle avoidance (LFV)

This is the description of lane following with vehicle avoidance demo.

**Knowledge and activity graph**

- **Requires:** Wheels calibration completed.
- **Requires:** Camera calibration completed.
- **Requires:** Joystick demo has been successfully launched.
- **Requires:** Lane Following demo gas veeb successfully completed

TODO: Add LFV result video

The following was marked as "todo".

Todo: Add LFV result video

Location not known more precisely.
Created by function n/a in module n/a.

4.1. Duckietown setup notes

Assumption about Duckietown:
- A Duckietown with white and yellow lanes. No obstacles on the lane.
- Layout conform to Duckietown Appearance Specifications
- Required tiles types: straight tile, turn tile
- Configurated wireless network for communicating with Duckiebot.
- Good consistent lighting, avoid natural lighting.

4.2. Duckiebot setup notes
• Make sure the camera is heading ahead.
• Duckiebot in configuration DB18 or DB19
• Make sure the vehicle obstacle has the circle pattern installed.

4.3. Pre-flight checklist
• Turn on joystick (if applicable).
• Turn on battery of the duckiebot.
• Place duckiebot in lane so that enough of the lane lines are visible to the camera.
• Verify you can ping your duckiebot over the network.
• **IMPORTANT** Make sure no containers are running on the duckiebot which use either the camera or joystick. We will run these ROS nodes together in a new container.
• Make sure your duckiebot can perform lane following.

4.4. Demo instructions

1) Start the demo containers

Running this demo requires almost all of the main Duckietown ROS nodes to be up and running. As these span 3 Docker images (dt-duckiebot-interface, dt-car-interface, and dt-core). The dt-duckiebot-interface and dt-car-interface container typically starts with robot startup. You will need to start dt-core manually.

First, check to make sure that dt-duckiebot-interface and dt-car-interface are running on your duckiebot via portainer, if not, do:

```bash
$ dts duckiebot demo --demo_name duckiebot-interface --duckiebot_name DUCKIEBOT_NAME --package_name duckiebot_interface --image duckietown/dt-duckiebot-interface:daffy-arm32v7

$ dts duckiebot demo --demo_name car-interface --duckiebot_name DUCKIEBOT_NAME --package_name car_interface --image duckietown/dt-car-interface:daffy-arm32v7
```

Then, we are ready to start the high-level pipeline for lane following:
You have to wait a while for everything to start working. While you wait, you can check in Portainer if all the containers started successfully and in their logs for any possible issues.

2) Make your Duckiebot drive autonomously!

If you have a joystick you can skip this next command, otherwise we need to run the keyboard controller:

```bash
$ dts duckiebot demo --demo_name multi_lane_following --duckiebot_name DUCKIEBOT_NAME --package_name duckietown_demos
```

Start the lane following. The Duckiebot should drive autonomously in the lane. Intersections and red lines are neglected and the Duckiebot will drive across them like it is a normal lane. You can regain control of the bot at any moment by stopping the lane following and using the (virtual) joystick. Resuming the demo is as easy as pressing the corresponding start button.

Et voilà! We are ready to drive around autonomously.

3) Expected outcome

The expected result for the lane following with vehicle avoidance is that the duckiebot will blink its tail lights yellow (greenish yellow) when it gets close to the vehicle in the way, while slowing down. At a certain distance, the duckiebot will stop completely and blink its tail light red to signal other duckiebots behind that there is a road anomaly.

4) Visualize the detected vehicle (optional)

This step is not neccessary but provides a nice visualization of the line segments that the Duckiebot detects.

In order to visualize the duckiebot’s vehicle detection you can use the `rqt_image_view` image tool and select the vehicle detection output. For more information about `rqt_image_view` tools, you can find them [here](#).
5) Extras

Here are some additional things you can try:

• Try to change some of the ROS parameters to see how your Duckiebot’s stopping behaviour will change.
• Try to implement a new way of detecting the duckiebot in the way.

4.5. Troubleshooting

1) Generic Lane Following Problems

Generic lane following problems can happen, such as:

• The duckiebot does not move
• The Duckiebot does not stay in a straight lane
• The Duckiebot does not drive nicely through intersections
• The Duckiebot cuts white line while driving on inner curves

For these problems, please refer to the duckiebot lane following tutorial [here](#) for troubleshooting instructions

2) The Duckiebot does not stop behind the vehicle obstacles

Sometimes due to processing latency, the duckiebot will not be able to see the vehicle in front in time. This can also sometimes happen in the real world autonomous vehicles. It is an active research area and feel free to propose better solutions! Currently we use CV tools and the circular pattern to identify and calculate the distance between duckiebots.

To make sure the problem is not caused by the pipeline, you can also use the `rqt_image_view` tool described above to make sure the duckiebot is not failing the circular pattern detection. You will also observe the latency of the image detection through there.
This is the description of the indefinite navigation demo.

**KNOWLEDGE AND ACTIVITY GRAPH**

- **Requires:** Wheel calibration completed.
- **Requires:** Camera calibration completed.
- **Requires:** Fully set up Duckietown.
- **Results:** One or more Duckiebot safely navigating in Duckietown.

### 5.1. Video of expected results

![Image of Duckietown setup](image.png)

Figure 5.1. Demo: indefinite navigation

### 5.2. Duckietown setup notes

To run this demo, you can setup a quite complex Duckietown. The demo supports normal road tiles, intersections and traffic lights. That makes it a level more difficult than the lane following demo. Make sure that your Duckietown complies with the appearance specifications presented in the Duckietown specs. In particular correct street signaling is key to success of intersections handling.

### 5.3. Duckiebot setup notes
One (or possibly more) Duckiebot in configuration DB19/DB21M.

5.4. Pre-flight checklist
Check that every Duckiebot has sufficient battery charge and that they are all properly calibrated.

5.5. Demo instructions

1) Start the demo containers
Running this demo requires almost all of the main Duckietown ROS nodes to be up and running. Make sure that ros, car-interface and duckiebot-interface are running.

Then, we are ready to start the high-level pipeline for indefinite navigation:

```
$ dts duckiebot demo --demo_name indefinite_navigation --duckiebot_name DUCKIEBOT_NAME
```

You have to wait a while for everything to start working. While you wait, you can check in Portainer if all the containers started successfully and in their logs for any possible issues.

2) Make your Duckiebot drive autonomously!
If you have a joystick you can skip this next command, otherwise we need to run the keyboard controller:

```
$ dts duckiebot keyboard_control DUCKIEBOT_NAME
```

Start Ind Navigation   R1   a
Stop Ind Navigation    L1   s

Start indefinite navigation by pressing the above controls. The Duckiebot should drive autonomously in the lane. Intersections and red lines are taken into consideration and the Duckiebot will stop at the red lines. You can regain control of the bot at any moment by stopping the indefinite navigation and using the (virtual) joystick. Resuming the demo is as easy as pressing the corresponding start button.
Et voilà! We are ready to drive around autonomously.

3) Extras

Here are some additional things you can try:

- Get a remote stream of your Duckiebot.
- You can visualize the detected line segments the same way as for the lane following demo.
- Try to change some of the ROS parameters to see how your Duckiebot’s behavior will change.

5.6. Troubleshooting the intersection handling

If your Duckiebot does not yield satisfactory results in intersection navigation, you might want to tune the related parameters. Basically the unicorn_intersection_node (long story for the name) is a mixture of open loop commands and a re-use of the lane filter. During the intersection, namely when the Duckiebot is in the FSM state INTERSECTION_CONTROL, the color perception of lines is changed. As a simple example if the goal is to go straight, the red lines will be perceived as white, so that it will be possible to follow the right white line. On top of this there are a few open loop commands that are used to help the Duckiebot face the correct direction. These parameters are stored in

```
Software/catkin_ws/src/00-infrastructure/duckietown/config/baseline/unicorn_intersection/unicorn_intersection_node/default.yaml
```

You can change them online (while the demo is running) by using:

```
duckiebot-container $ rosparam set your_parameter your_value
```

You can see all the parameters by running:

```
$ rosparam list
```

And check the value of a specific one using:
The ones you might want to modify are the feed-forward parts, stored in `ff_left`, `ff_right` and `ff_straight`. These parameters modify the output $\omega$ (angular velocity, positive in counterclockwise direction) for the time given in `time_left_turn`, `time_straight_turn` and `time_right_turn`, which you might want to change as well.

Maintainer: Contact Gianmarco Bernasconi (ETHZ), Frank (Chude) Qian (UofT) via Slack for further assistance.