

# The Duckiebot Operation Manual

## Contents

### Duckiebot Assembly

- [Duckiebot Configurations](#)
- [Getting the Duckiebot hardware](#)
- [Assembly - Duckiebot \*\*DB21J\*\*](#)
- [Assembly - Duckiebot \*\*DB21M\*\*](#)
- [Duckiebot FAQ Guide](#)

### Software Setup

- [Setup - Laptop](#)
- [Setup - Accounts](#)
- [Setup - Duckiebot SD Card](#)
- [Setup - Booting the Duckiebot](#)
- [Setup - Duckiebot Dashboard](#)

### Duckiebot Operations

- [Handling - Duckiebot \*\*DB21\*\*](#)
- [Operation - Use the Dashboard](#)
- [Operation - Make it Move](#)
- [Operation - Make it See](#)
- [Operation - Make it Shine](#)
- [Operation - Software Tools](#)
- [Calibration - Camera](#)
- [Calibration - Wheels](#)
- [Operation - Taking and verifying a log](#)

### Demos

- [Introduction to Demos](#)
- [Duckiebot Supported Demos](#)
- [Duckiebot Legacy Demos](#)

### Learning Experiences

- [Introduction to Learning Experiences](#)
- [Supported Learning Experiences](#)
- [Experimental Learning Experiences](#)

### Troubleshooting Guides

- [Operation - Networking](#)
- [Debug - Re-flash Microcontroller](#)
- [Debug - Duckiebot Update](#)
- [Requesting a Support Connection](#)
- [Reset Dashboard](#)

### Reference Reading

- [Version Control with Git](#)
- [Docker Basics](#)

- [Secure shell \(SSH\)](#)
- [Handling circuits and batteries](#)

Welcome to the operation manual for your Duckiebot!

In this manual, you will learn how to assemble and operate your Duckiebot, as well as how to set up your working environment.

## Duckiebot Configurations

|                    |   |
|--------------------|---|
| What you will need | <ul style="list-style-type: none"> <li>• Nothing</li> </ul>   |
| What you will get  | <ul style="list-style-type: none"> <li>• Knowledge of Duckiebot configuration naming conventions and their respective functionalities.</li> </ul> |

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

Duckiebots **DB18** onwards can be obtained from the [Duckietown project store](#).

### Overview

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

Legend:

- "JN": NVIDIA Jetson Nano
- "RPI": Raspberry Pi
- "ToF": Time of flight
- "IMU": Inertial Measurement Unit (Accelerometer, Gyroscope)

### 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** and then **DB21J**, improve the **DB21M** by:

- expanding the onboard memory from 32 GB to 64 GB;

- 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 **DB21J** Duckiebot, follow [these](#) instructions.

**Note**

You can obtain a **DB21J** Duckiebot from the [Duckietown project shop](#).

## Duckiebot MOOC Founder's edition, or **DB21M**

The **DB21M** is the first Duckiebot equipped with an 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.



*Fig. 1* The Duckiebot version **DB21M**.

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](#)).

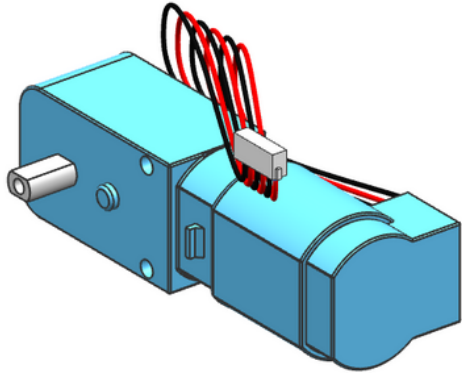
To assemble a **DB21M** Duckiebot, follow [these](#) instructions.

**Note**

**DB21M** Duckiebots are no longer manufactured. If you want to obtain one you might try to find inventory leftovers from Duckietown official distributors, or reach out to the [Duckietown hardware team](#).

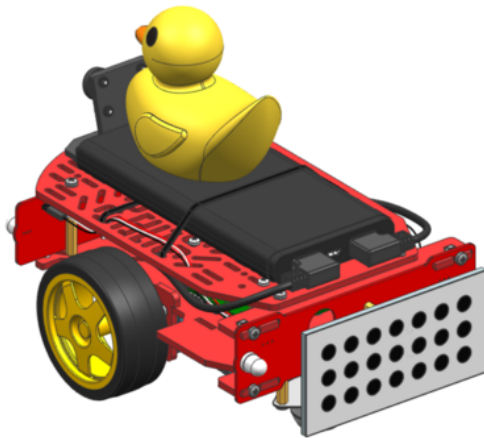
## Duckiebot version 2019, or DB19

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 .



*Fig. 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:



*Fig. 3* The complete Duckiebot DB19.

To assemble a DB19 Duckiebot, follow [these](#) instructions.

### **Note**

DB19 Duckiebots have been extremely successful but are no longer manufactured, and to the best of our knowledge only a few still exist unboxed.

## Duckiebot version 2018, or DB18

You have a DB18 Duckiebot if, e.g., you have pledged to the Kickstarter.

There are two configuration of the DB18.

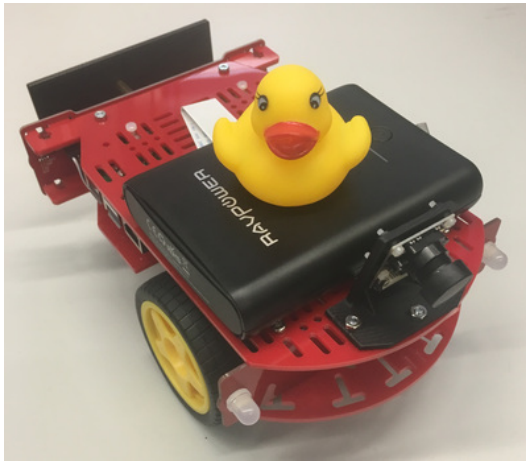
### The DB18 configuration

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

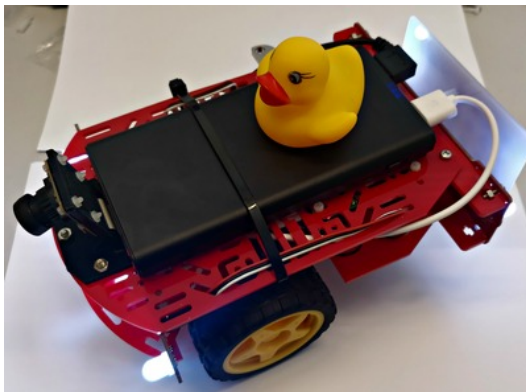
Kickstarter.

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.



*Fig. 4* A **DB18** Duckiebot assembly.



*Fig. 5* Another **DB18** Duckiebot assembly, with different battery.

To assemble a **DB18** Duckiebot, follow [these](#) instructions.

**Note**

**DB18** Duckiebots are no longer manufactured. There are very few unopened **DB18** boxes in the world. For additional information, reach out to contact the [Duckietown hardware team](#).

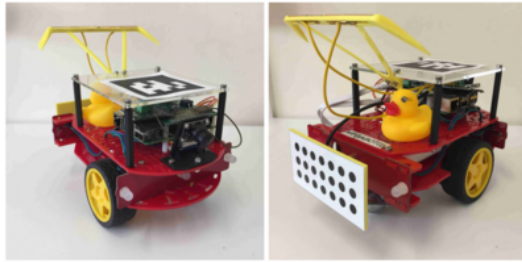
### The **DB18-Robotarium** configuration

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.

Autolabs 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 Autolab, contact the [Duckietown team](#).



**Fig. 6** A Duckiebot in **DB18-Robotarium** configuration.

## Duckiebot versions 2017, or **DB17**

In the **DB17** version, we had several 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  $\mathbb{O} = \{w, j, d, p, l, c\}$ .

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

The elements of  $\mathbb{O}$  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 joystick 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;

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:

- **DB17-11**: *adds* a PWM hat to **DB17**, in addition to a short USB angled power cable and a M-M power wire;
- **DB17-12**: *adds* a bumpers set to **DB17**, in addition to the mechanical bits to assemble it;
- **DB17-13**: *adds* a LED hat and 5 RGB LEDs to **DB17-1112**, in addition to the F-F wires to connect the LEDs to the LED board.

Note: introducing the PWM hat in **DB17-11** induces a *replacement* of the [spliced cable](#) powering solution for the DC motor hat. Details can be found in .

- **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: (**DB17-11**) avoid using a sliced power cable to power the DC motor hat in **DB17**, and (**DB17-12**) for purely aesthetic reasons.

# Getting the Duckiebot hardware

What you will need

- Knowledge of [Duckiebot hardware configurations](#)

What you will get

- Parts to assemble a Duckiebot.

## Foreword

You can acquire Duckiebots in two ways:

- **“One click” solution** (DB18 and above): you can source complete hardware kits on [the Duckietown project online store](#). These kits are tested to work with Duckietown software, are guaranteed to work, and come with a 12 months warranty. This is the officially supported and recommended path to getting started with Duckietown.
- **Do it yourself** (DB17): the DB17 Duckiebot configuration is made of components that you can source independently. This configuration is no longer actively supported, but you can find the bill of materials for this DIY approach below.

## Acquiring the parts for a DB17

Here, we provide a link to all bits and pieces that are needed to build a DB17-jwd 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.

**DB17** Bill of materials

| Item  | Cost (USD)                     |
|---|--------------------------------|
| <a href="#">Chassis</a>   | 20                             |
| <a href="#">Camera with 160-FOV Fisheye Lens</a>                | 39                             |
| <a href="#">Camera Mount</a>                                    | 4                              |
| <a href="#">300mm Camera Cable</a>                              | 2                              |
| <a href="#">Raspberry Pi 3 - Model B+</a>                       | 39                             |
| <a href="#">Heat Sinks</a>                                      | 3                              |
| <a href="#">Power supply for Raspberry Pi</a>                   | 7.50                           |
| <a href="#">16 GB Class 10 MicroSD Card</a>                     | 10                             |
| <a href="#">Micro SD card reader</a>                            | 6                              |
| <a href="#">DC Motor HAT</a>                                    | 22.50                          |
| <a href="#">2 Stacking Headers</a>                              | 2.50/piece                     |
| <a href="#">Battery Pack</a>                                    | 25                             |
| <a href="#">16 Nylon Standoffs (M2.5 12mm F 6mm M)</a>          | 0.06/piece                     |
| <a href="#">4 Nylon Hex Nuts (M2.5)</a>                         | 0.02/piece                     |
| <a href="#">4 Nylon Screws (M2.5x10)</a>                        | 0.05/piece                     |
| <a href="#">2 Zip Ties (300x5mm)</a>                            | 9                              |
| <a href="#">Wireless Adapter (5 GHz) (DB17-w)</a>               | 25                             |
| <a href="#">Joypad (DB17-j)</a>                                 | 10.50                          |
| <a href="#">Tiny 32GB USB Flash Drive (DB17-d)</a>              | 10                             |
| <a href="#">PWM/Servo HAT (DB17-11)</a>                         | 17.50                          |
| <a href="#">Power Cable (DB17-11)</a>                           | 7.80                           |
| <a href="#">Male-Male Jumper Wire (DB17-11)</a>                 | 1.95                           |
| <a href="#">8 M3x10 pan head screws (DB17-12)</a>               | 7                              |
| <a href="#">8 M3 nuts (DB17-12)</a>                             | 7                              |
| Bumpers set (DB17-12)   | 7 (custom made)                |
| Bumper bracers set (DB17-12)                                    | 7 (custom made)                |
| <a href="#">LEDs (DB17-13)</a>                                  | 10                             |
| <a href="#">LED HAT (DB17-13)</a>                               | 9/piece (but 3 pieces minimum) |
| <a href="#">20 Female-Female Jumper Wires (300mm) (DB17-13)</a> | 8                              |
| <a href="#">4 4 pin female header (DB17-13)</a>                 | 0.60/piece                     |
| <a href="#">12 pin male header (DB17-13)</a>                    | 0.48/piece                     |
| <a href="#">2 16 pin male header (DB17-13)</a>                  | 0.61/piece                     |
| <a href="#">3 pin male header (DB17-13)</a>                     | 0.10/piece                     |
| <a href="#">2 pin female shunt jumper (DB17-13)</a>             | 2/piece                        |
| <a href="#">40 pin female header (DB17-13)</a>                  | 1.50                           |
| <a href="#">5 200 Ohm resistors (DB17-13)</a>                   | 0.10/piece                     |
| <a href="#">10 130 Ohm resistors (DB17-13)</a>                  | 0.10/piece                     |
| <a href="#">Soldering tools</a>                                 | 20                             |
| Total for DB17 configuration                                    | 193                            |
| Total for DB17-w configuration                                  | 218                            |
| Total for DB17-j configuration                                  | 203                            |
| Total for DB17-d configuration                                  | 203                            |
| <b>Total for DB17-wjd configuration</b>                         | <b>238</b>                     |
| <b>Total for DB17-l configuration</b>                           | <b>367</b>                     |

Table 1 DB17 Bill of materials

## Chassis

We selected the Magician Chassis as the basic chassis for the robot ([The "Magician" chassis was very popular. It is used in DB17 through DB19 Duckiebot models.](#)).



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).



*Fig. 7* The "Magician" chassis was very popular. It is used in [DB17](#) through [DB19](#) Duckiebot models.

## Raspberry Pi 3 - Model B

Note: It is recommended 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 ([Fig. 8](#)), a small but powerful computer.



*Fig. 8* The Raspberry Pi 3 Model B is a 1.2GHz, 64-bit quad-core ARMv8 CPU, 1GB RAM little computer.

## Power Supply

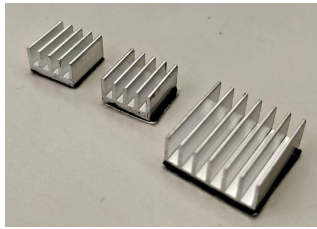
We want a hard-wired power source (5VDC, 2.4A, Micro USB) to supply the Raspberry Pi ([The power supply.](#)) while not driving. This charger can double down as battery charger as well.



*Fig. 9* The power supply.

## Heat Sinks

The Raspberry Pi will heat up significantly during use. It is recommended adding heat sinks, as in [The heat sinks.](#) 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.



*Fig. 10* The heat sinks.

### Class 10 MicroSD Card

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



*Fig. 11* The MicroSD card.

### Mirco SD card reader

A microSD card reader ([The Mirco SD card reader.](#)) 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.



*Fig. 12* The Mirco SD card reader.

### 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 ([The Raspberry Pi camera with fisheye lens.](#)).



*Fig. 13* The Raspberry Pi camera with fisheye lens.

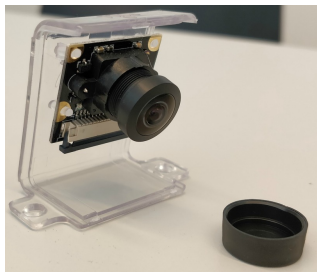
### Camera Mount

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



*Fig. 14* The camera mount.

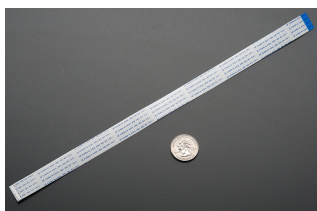
The assembled camera (without camera cable), is shown in ([The camera on its mount.](#)).



*Fig. 15* The camera on its mount.

### 300mm Camera Cable

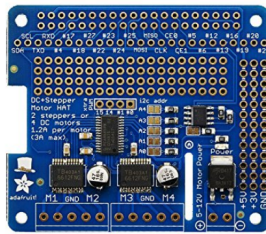
A longer (300 mm) camera cable [A 300 mm camera cable for the Raspberry Pi.](#) makes assembling the Duckiebot easier, allowing for more freedom in the relative positioning of camera and computational stack.



*Fig. 16* A 300 mm camera cable for the Raspberry Pi.

## DC Motor HAT

We use the DC Stepper motor HAT ([The stepper motor HAT.](#)) to control the DC motors that drive the wheels. This item will require [soldering](#) to be functional. This HAT has dedicated PWM and H-bridge for driving the motors.



*Fig. 17* The stepper motor HAT.

## Stacking Headers

We use a long 20x2 GPIO stacking header ([The stacking headers.](#)) to connect the Raspberry Pi with the DC Motor HAT. This item will require [soldering](#) to be functional.



*Fig. 18* The stacking headers.

## Battery

The battery ([The battery.](#)) 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.



*Fig. 19* The battery.

## 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.



*Fig. 20* Standoffs, Nuts and Screws.

## Zip Ties

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



*Fig. 21* The zip ties.

## Configuration DB17 - w

### Wireless Adapter (5 GHz)

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



*Fig. 22* The Edimax AC1200 EW-7822ULC WiFi adapter.

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

## Configuration DB17 - j

### 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 ([A Wireless joypad.](#)) does not include batteries.



**Fig. 23** A Wireless joystick.

Requires: 2 AA 1.5V batteries ([Batteries.](#)) not included in the [bill of materials.](#)



**Fig. 24** Batteries.

## Configuration **DB17-d**

### Tiny 32 GB USB Flash Drive

In configuration **DB17-d**, the Duckiebot is equipped with an “external” hard drive ([The tiny 32 GB USB flash drive.](#)). 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.



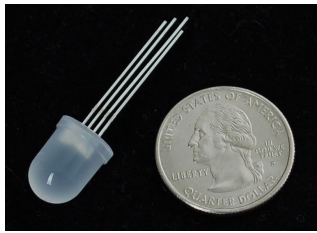
**Fig. 25** The tiny 32 GB USB flash drive.

## Configuration **DB17-1**

### LEDs

This Duckiebot is equipped with 5 RGB LEDs ([The DB17 RGB LEDs.](#)). 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.

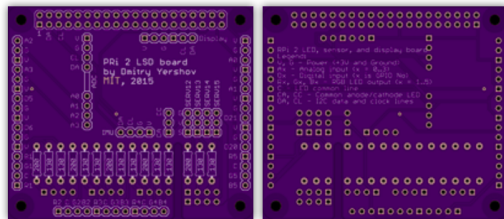


*Fig. 26* The DB17 RGB LEDs.

## LED HAT

The LED HAT ([The LED HAT](#)) 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](#).

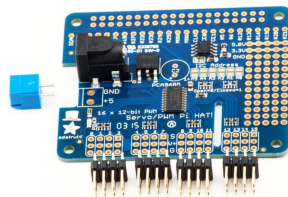
This board is custom designed and can only be ordered in minimum runs of 3 pieces. The price scales down quickly with quantity, and lead times may be significant, so it is better to buy these boards in bulk.



*Fig. 27* The LED HAT.

## PWM/Servo HAT

The PWM/Servo HAT ([The PWM-Servo HAT](#)) 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.



*Fig. 28* The PWM-Servo HAT.

## Power Cable

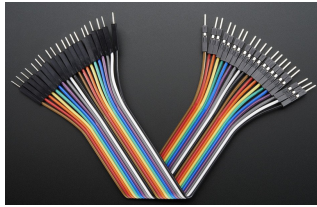
To power the PWM/Servo HAT from the battery, we use a short (30cm) angled male USB-A to 5.5/2.1mm DC power jack cable ([The 30cm angled USB to 5.5/2.1mm power jack cable](#)).



*Fig. 29* The 30cm angled USB to 5.5/2.1mm power jack cable.

### Male-Male Jumper Wires

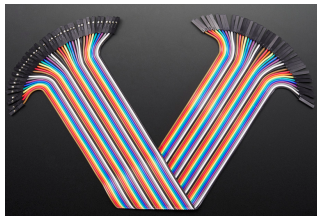
The Duckiebot needs one male-male jumper wire ([Male-male jumper wires.](#)) to power the DC Stepper Motor HAT from the PWM/Servo HAT.



*Fig. 30* Male-male jumper wires.

### Female-Female Jumper Wires

20 Female-Female Jumper Wires ([Female-female jumper wires.](#)) are necessary to connect 5 LEDs to the LED HAT.



*Fig. 31* Female-female jumper wires.

### 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.

### 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.

### Caster (**DB17-c**)

The caster ([The caster wheel.](#)) 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.

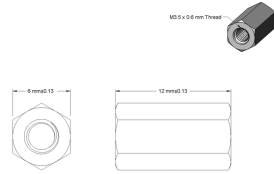




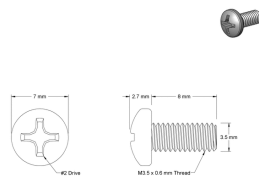
**Fig. 32** The caster wheel.

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

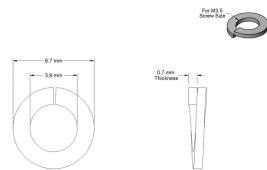
- 4 standoffs (M3 12mm F-F) ([Standoffs for caster wheel.](#)),
- 8 screws (M3x8mm) ([Screws for caster wheel.](#)), and
- 8 split lock washers ([Split lock washers for caster wheel.](#)).



**Fig. 33** Standoffs for caster wheel.



**Fig. 34** Screws for caster wheel.



**Fig. 35** Split lock washers for caster wheel.

**Note**

The caster wheel use is to be considered experimental and has been dropped in official configurations starting from **DB18**.

## Assembly - Duckiebot **DB21J**

What you will need

- Duckiebot **DB21** parts ([get a DB21-Jx](#)). If you are unsure what version of Duckiebot you have, check the overview of existing [Duckiebot configurations](#).
- A micro SD card with the Duckiebot image on it. The procedure to flash the SD card is explained [here](#).
- 3-4 hours of assembly time.

What you will get

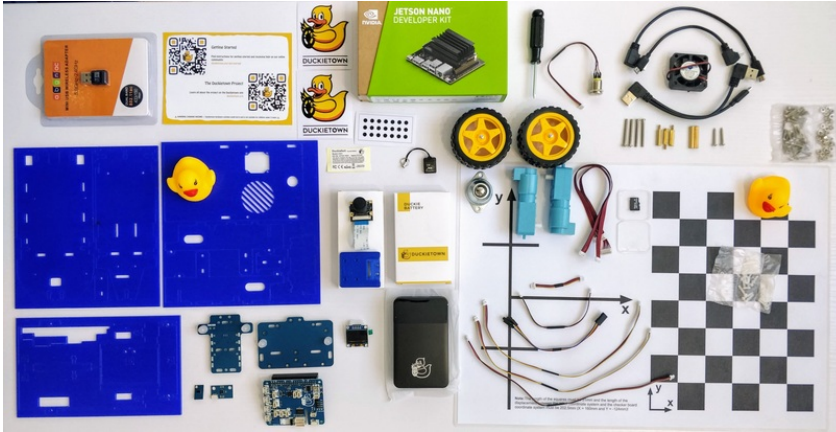
- An assembled Duckiebot in configuration **DB21J**.

## Foreword

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

## 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](#)
- [Troubleshooting section](#)

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

## Preliminary Steps

### 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 and pliers might ease some passages.

#### **Note**

Both NVIDIA Jetson Nano 2 GB and 4 GB are supported, but the SD cards must be initialized differently, as described in [Setup - Duckiebot SD Card](#).

### Plastic cover

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

### 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).

| Picture | Dimension      | Number | Index |
|---------|----------------|--------|-------|
|         | Nylon nut M2   | 4      | N2    |
|         | Nylon nut M2.5 | 12     | N2.5  |
|         | Metal nut M3   | 12     | M3    |

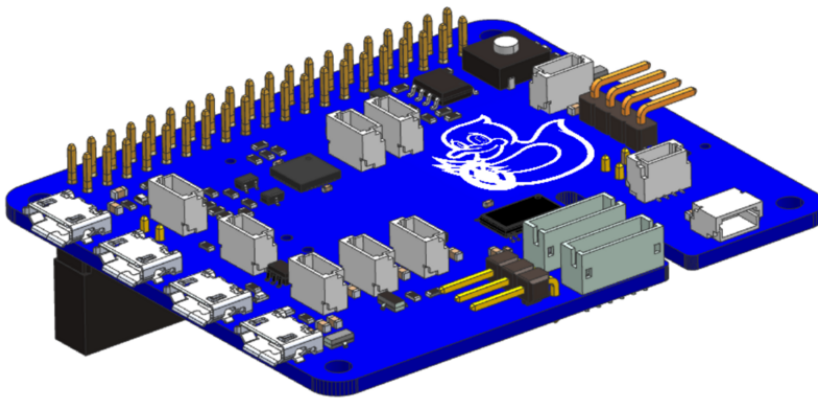
| Picture | Dimension             | Number | Index   |
|---------|-----------------------|--------|---------|
|         | Nylon screw M2x8mm    | 4      | N2x8    |
|         | Nylon screw M2.5x10mm | 12     | N2.5x10 |
|         | Metal screw M3x8mm    | 4      | M3x8    |
|         | Metal screw M3x12mm   | 26     | M3x12   |
|         | Metal screw M3x30mm   | 4      | M3x30   |

| Picture | Dimension                     | Number | Index    |
|---------|-------------------------------|--------|----------|
|         | Metal F-F stand-off M3x25mm   | 2      | FF3x25   |
|         | Metal F-F stand-off M2.5x18mm | 3      | MF2.5x18 |
|         | Spacer M6x12x1.5mm            | 1      | Spacer   |

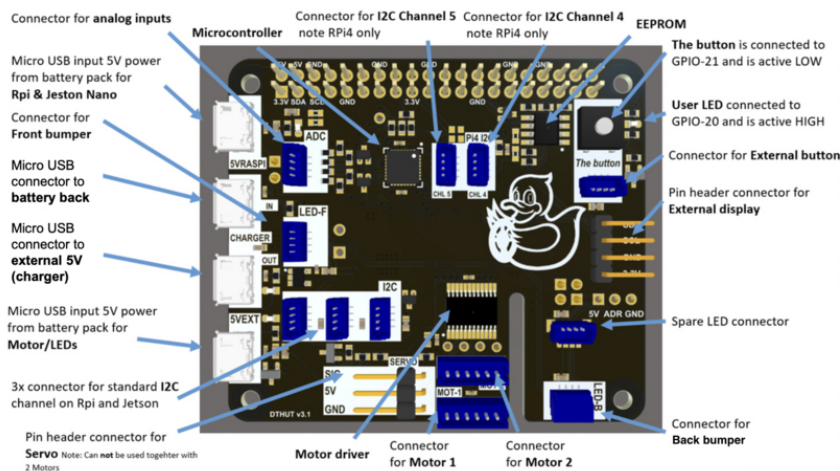
In addition, there are also shorter screws, which can be used in some cases.

## HUT

Hut is the electronics board:



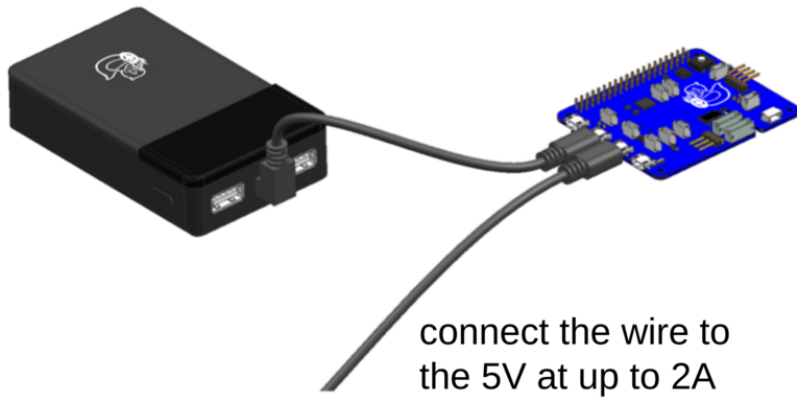
It contains the connectors for various sensors, the fan, motors, external button, etc. Also it is mounted to the NVIDIA Jetson Nano. A more detailed scheme is shown in the figure:



## Charge Duckiebattery via the HUT

This preliminary step allows us to start charging the battery while confirming the functionality of the HUT.

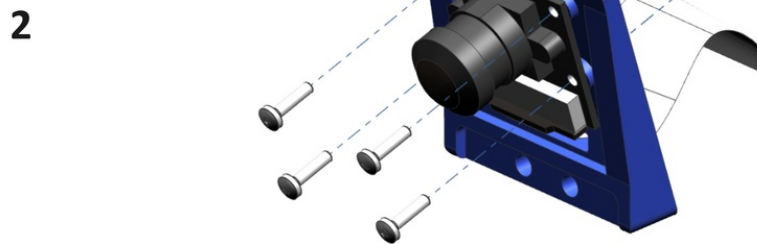
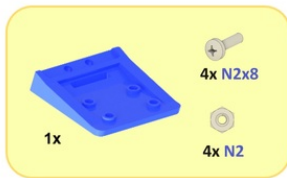
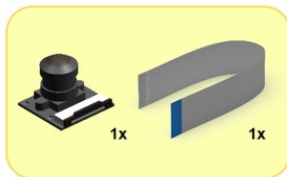
- Connect the battery and the HUT board as shown and make sure a green LED on the HUT is lit.
- Wait 30 minutes and then push the button on the battery.
- Check that the state of charge LEDs on the battery start blinking.
- Leave this setup until the battery is charged. This may take up to 5 hours.



You can familiarize with how the Duckiebattery works by reading its [handling instructions](#).

## Camera Assembly

This section (steps 1 to 2) guides you through the assembly of the Camera:



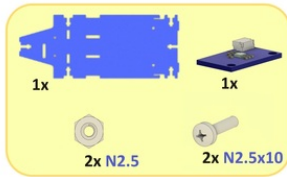
## Base-plate

In the following steps (3 to 18) we will build the *base-plate* assembly of the Duckiebot.

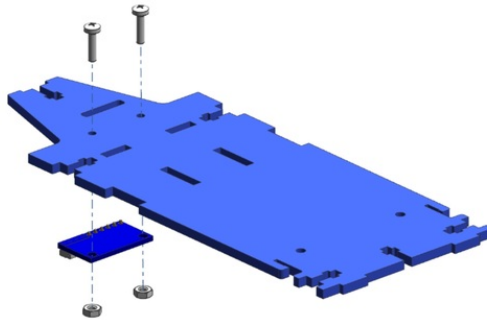


**Note**

You could try to use shorter screws in case the screws do not fully insert into the standoffs.

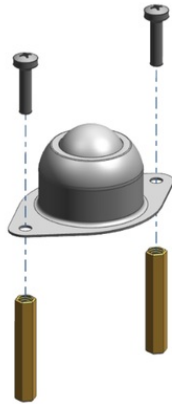


**3**





4

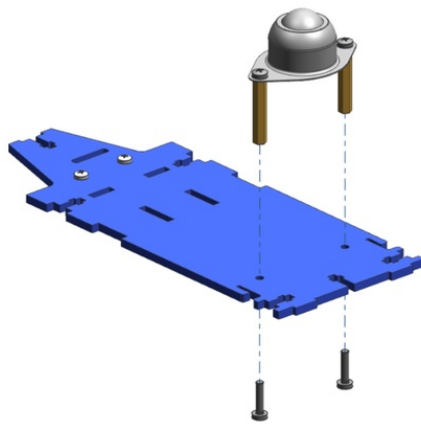


**Note**

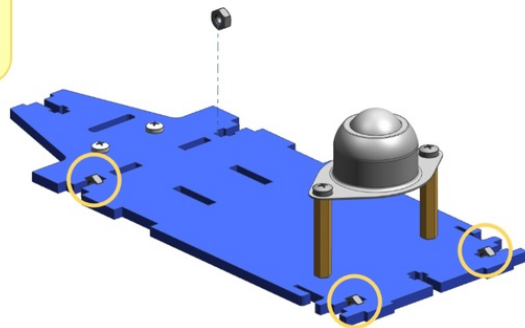
Occasionally manufacturing tolerances (on the nut and the chassis) might prevent a flush fit. Trying a different nut or changing its orientation might solve the problem. Sometimes it may be convenient to use pliers.

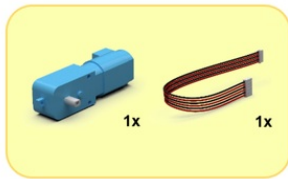


5

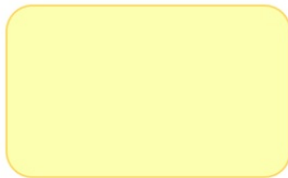
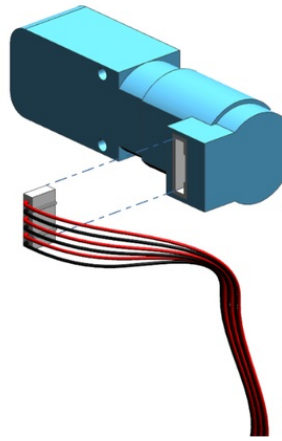


6

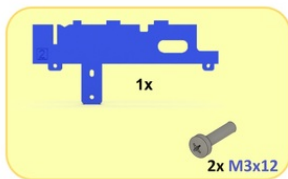
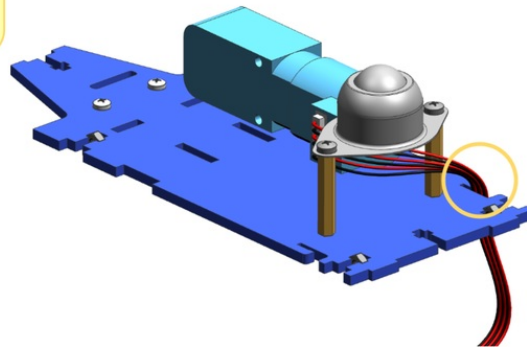




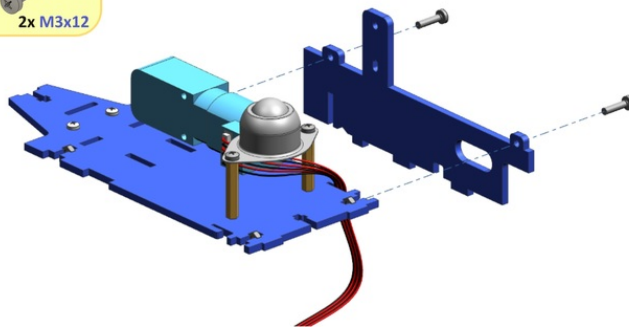
7

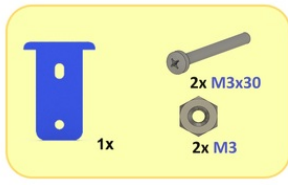


8

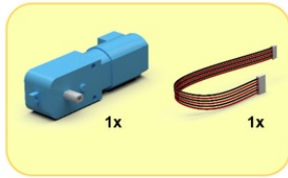
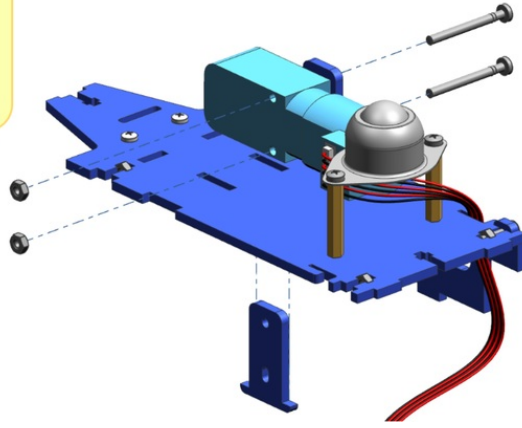


9

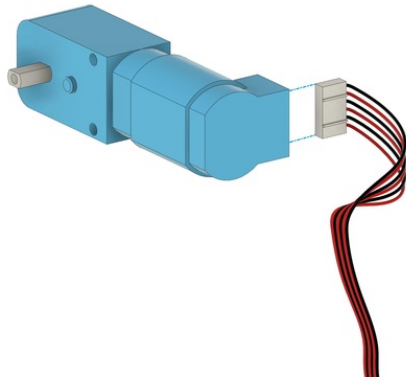




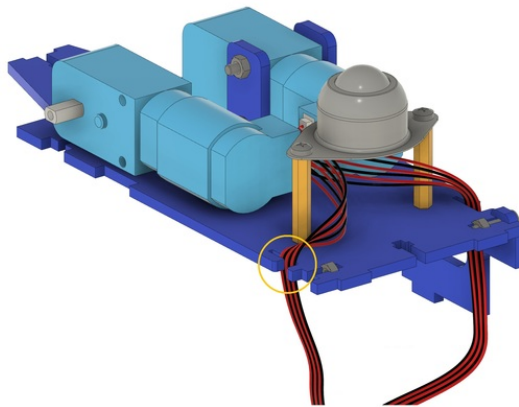
10



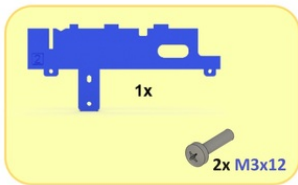
11



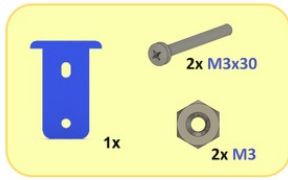
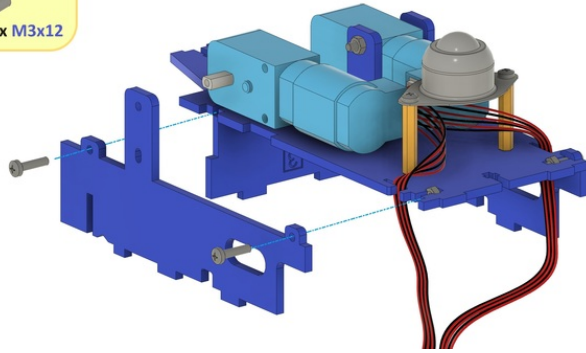
12



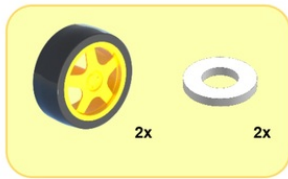
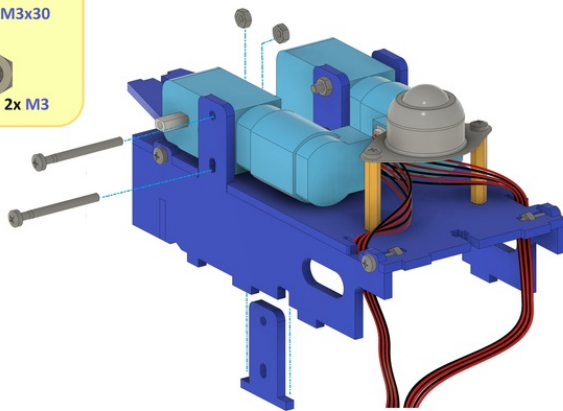




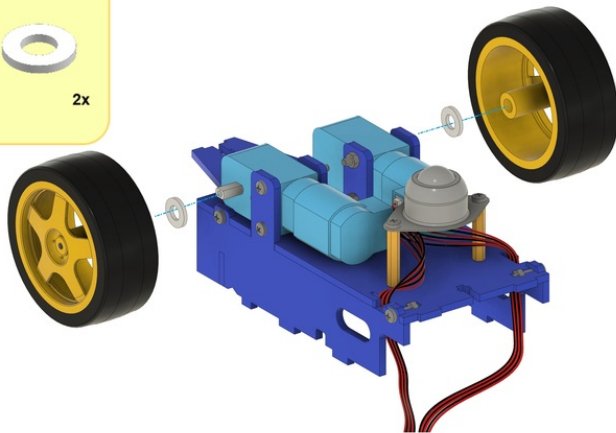
13



14

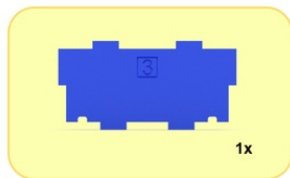
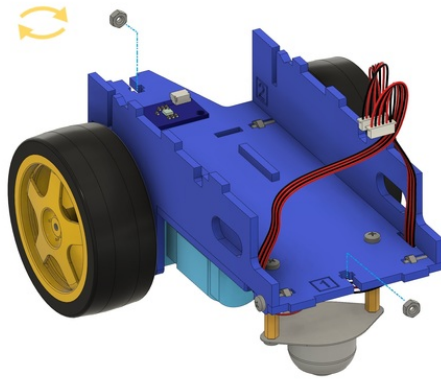


15

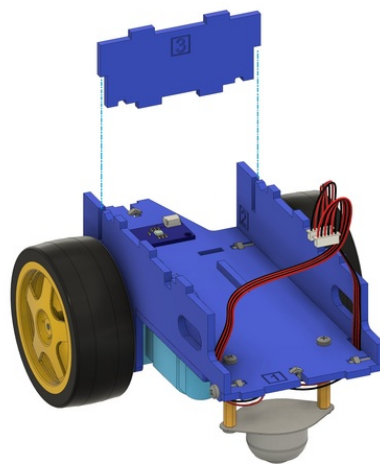




16



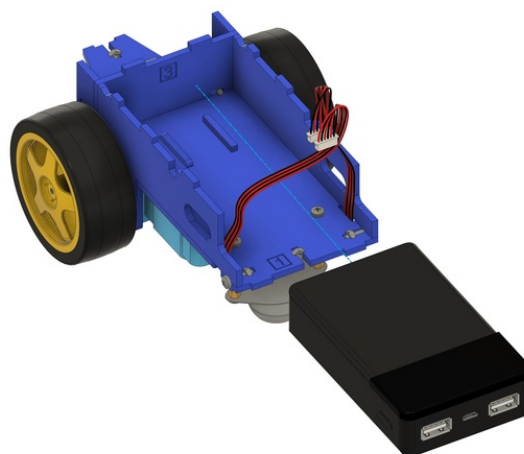
17



Remove the USB cable from the Duckiebattery, connected in the [battery related preliminary step](#).



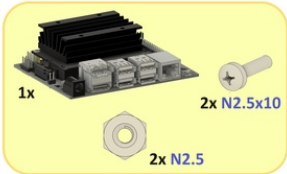
18



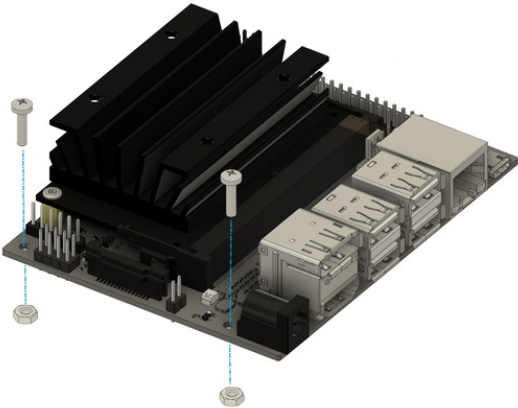
Before proceeding, verify that no component is wiggling. The only things moving should be the cables and the sphere in the omni-wheel (and, yes, the motor axles). Proceed to gently tighten the screws of the offending parts, if necessary.

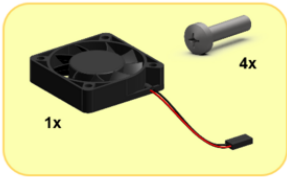
### Computation Unit

The following steps (19 to 27) guide through the assembly of the *Computation* unit:

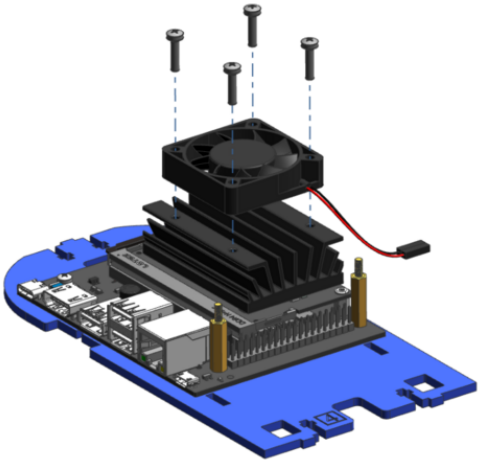


19

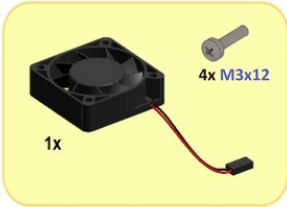
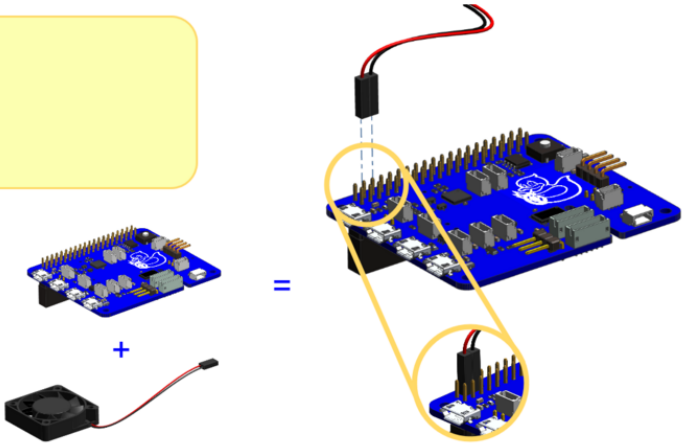




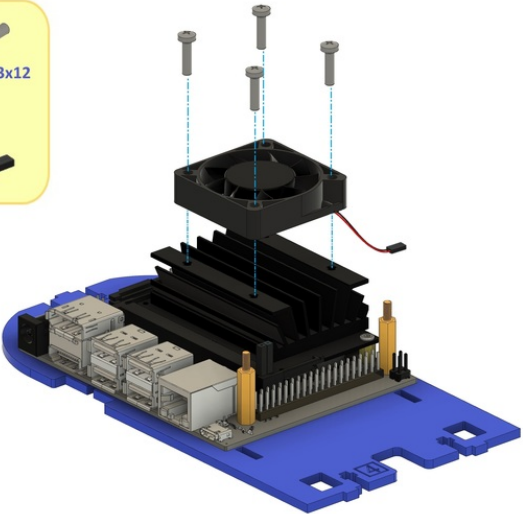
20

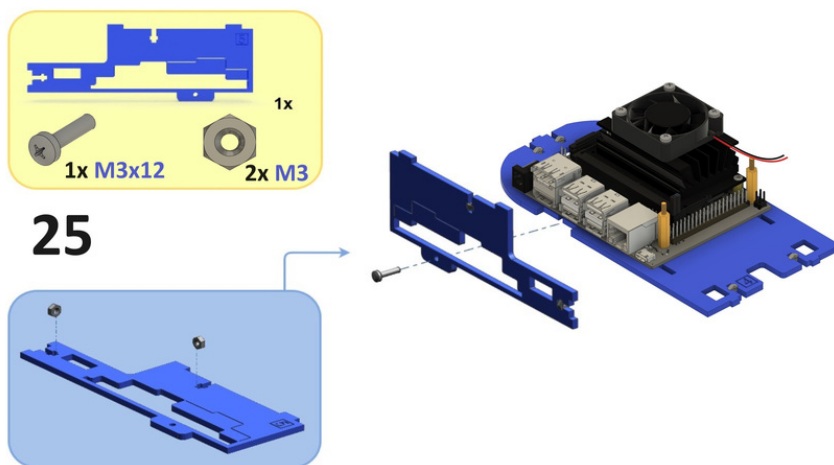
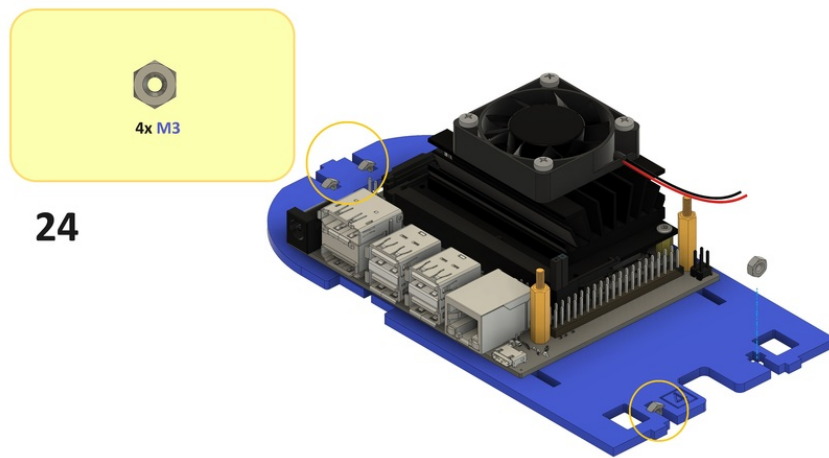
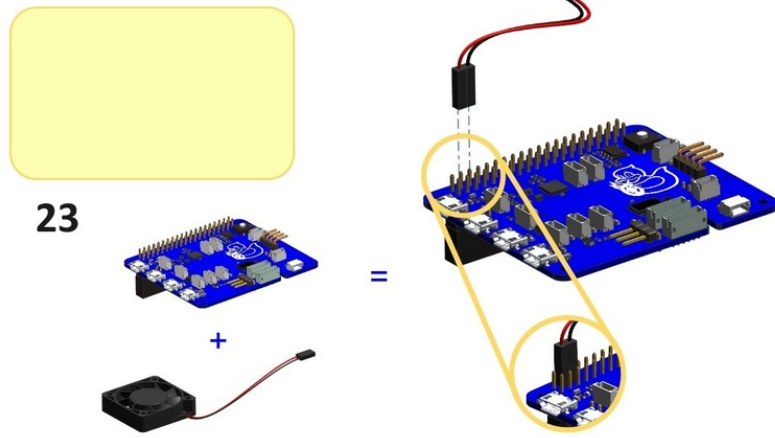


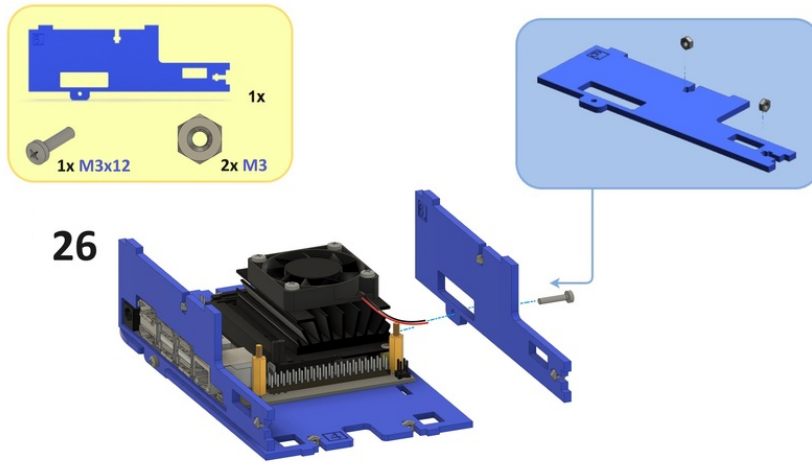
21



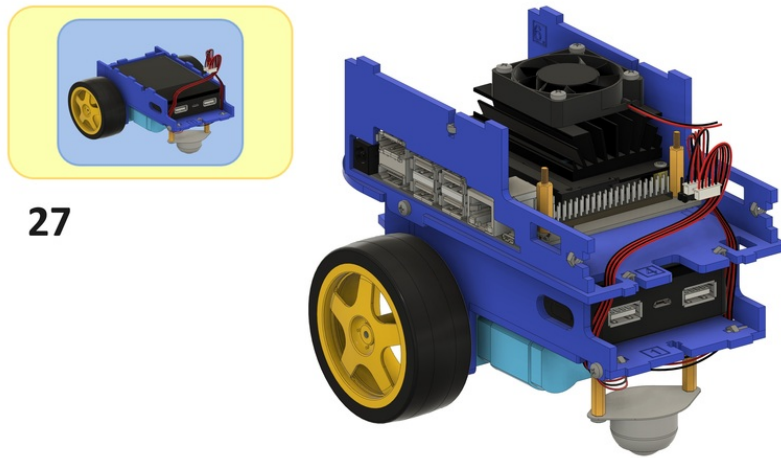
22





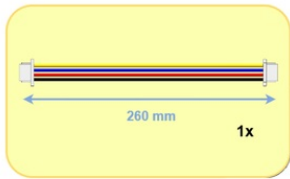
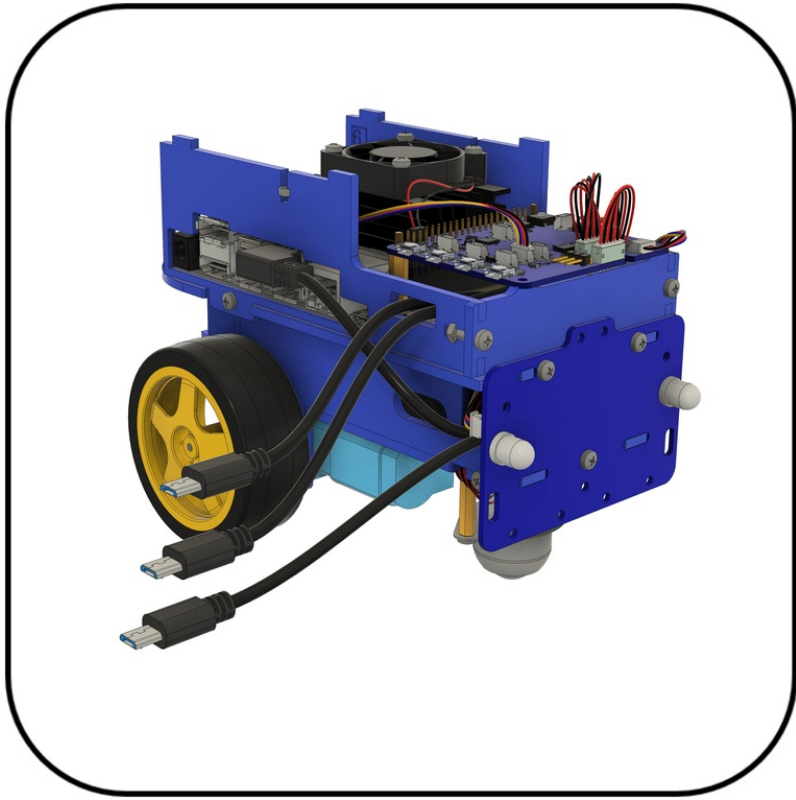


Now connect it to the base-plate (i.e, the rest of the chassis assembled in steps 3 to 18).  
Verify the chassis components are locked correctly.

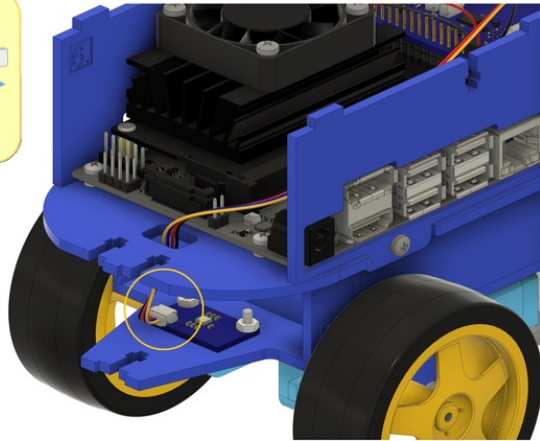


## Rear Assembly

The following steps (28 to 40) guide through the assembly of the rear part of the Duckiebot:

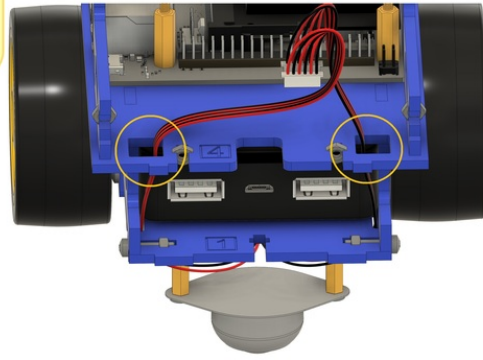


28

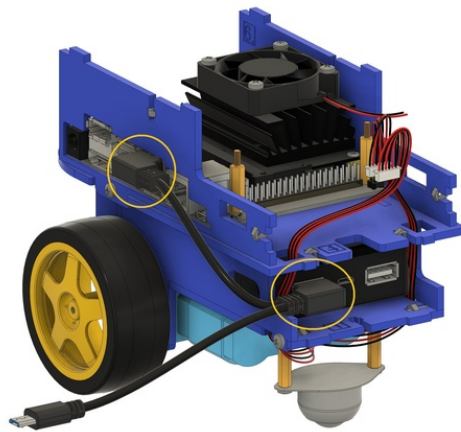




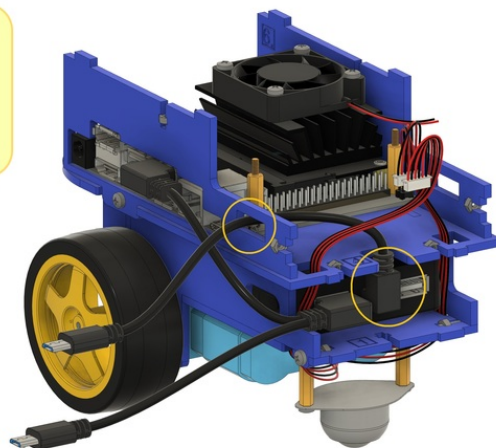
29



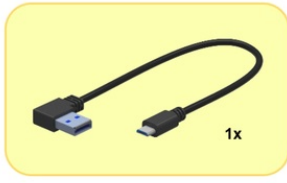
30



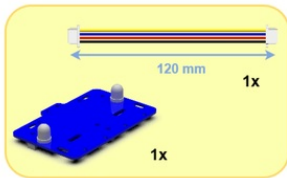
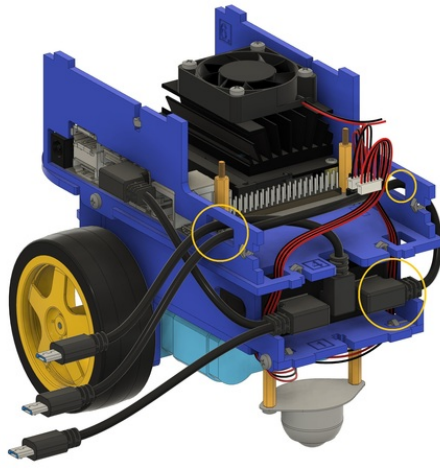
31



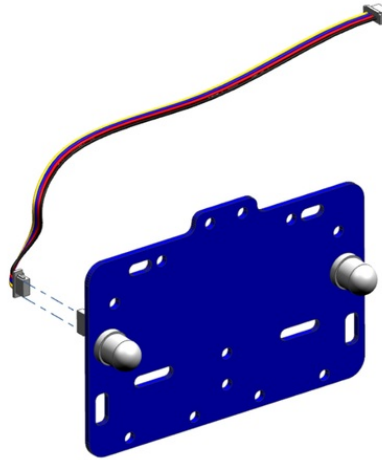




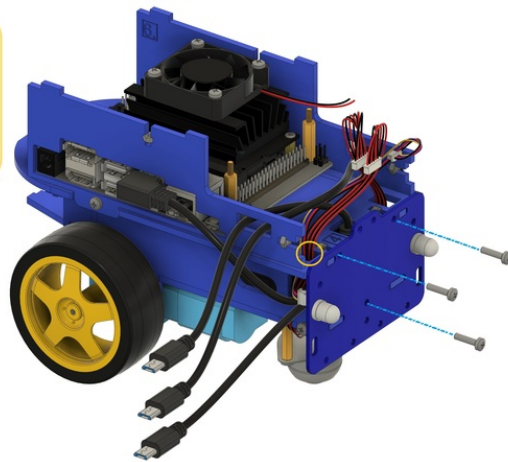
32

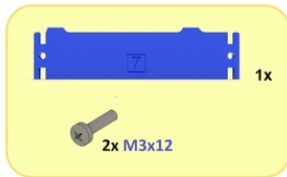


33

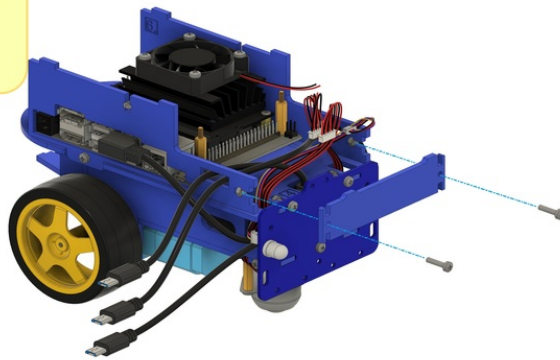


34



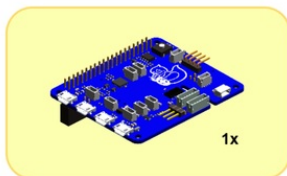


35

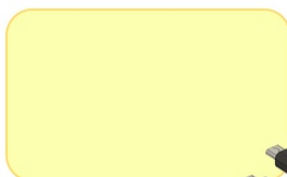
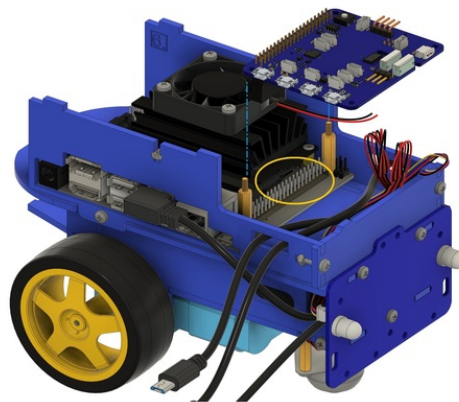


**Note**

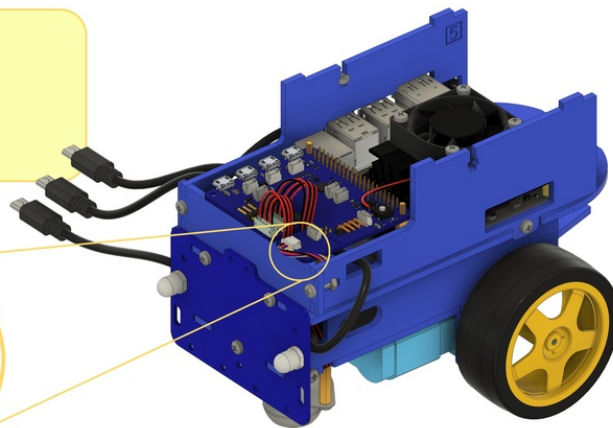
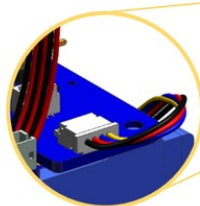
Make sure that both rows of pins on the 40-pin expansion header on the Jetson Nano are connected to the corresponding contacts of HUT.



36

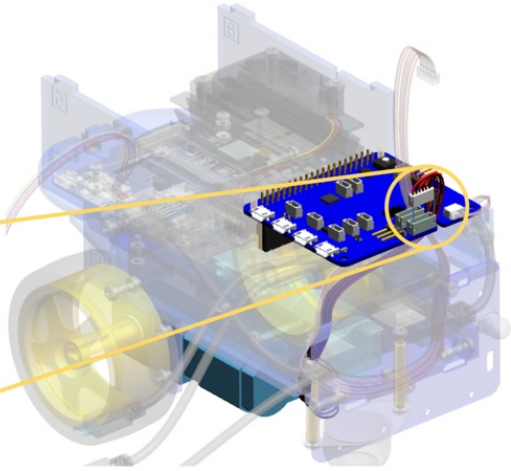


37

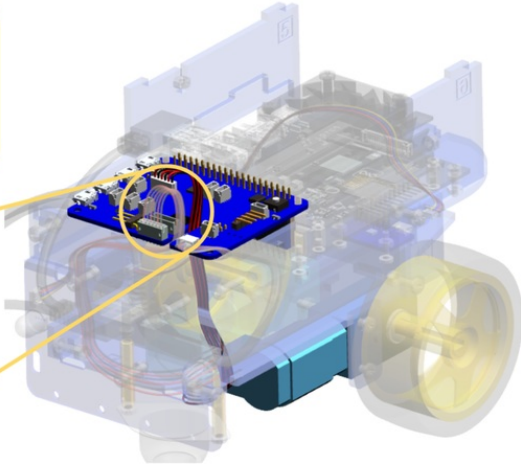
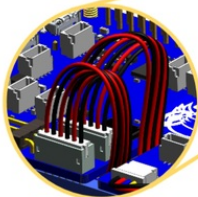




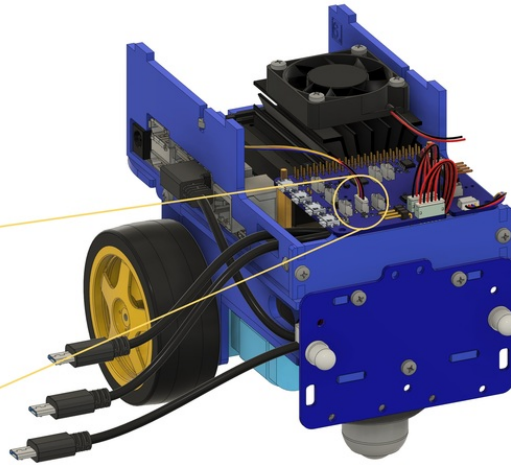
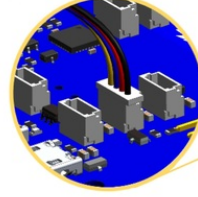
38

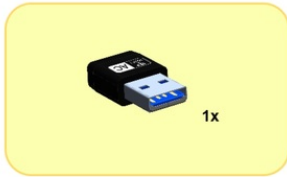


39

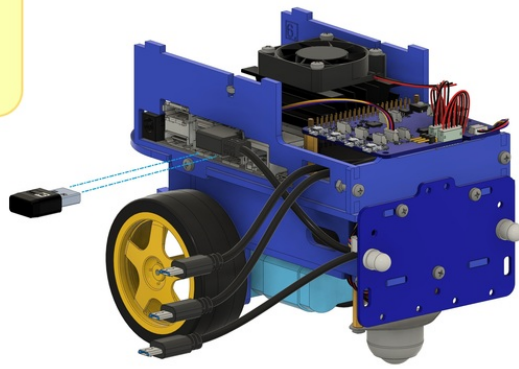


40



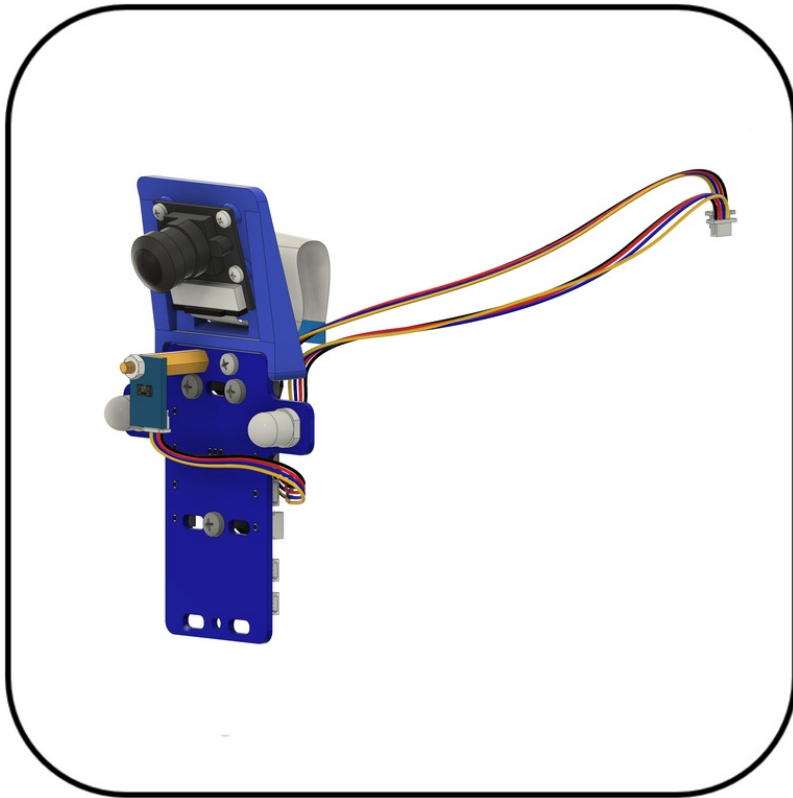


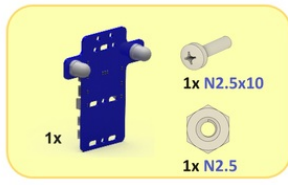
41



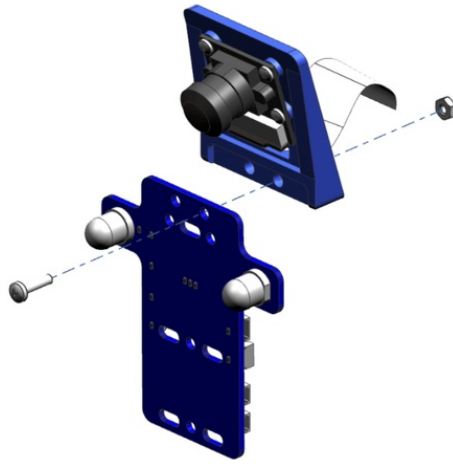
## Front Assembly

Steps 42 to 51 guide you through the assembly of the front bumper:

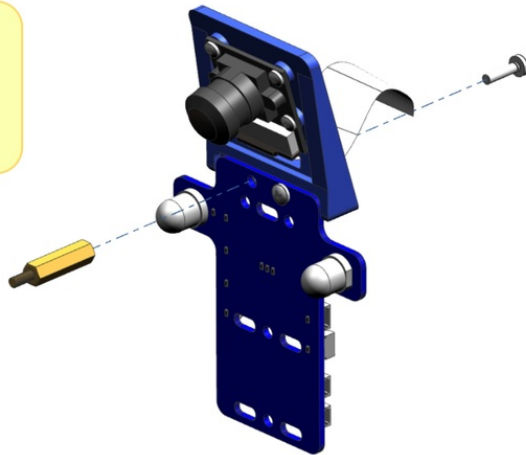




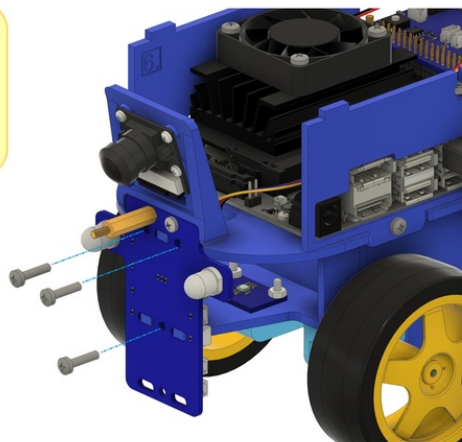
42



43

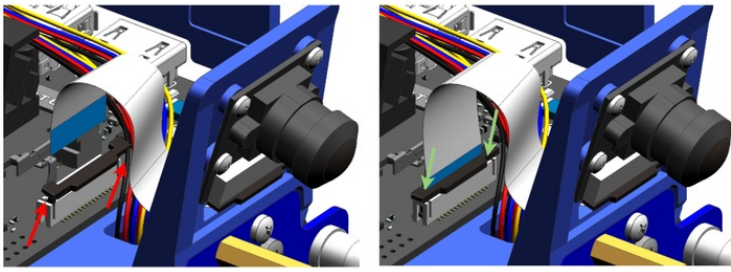


44

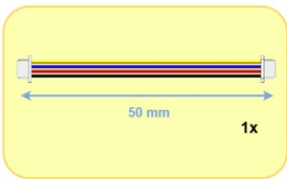
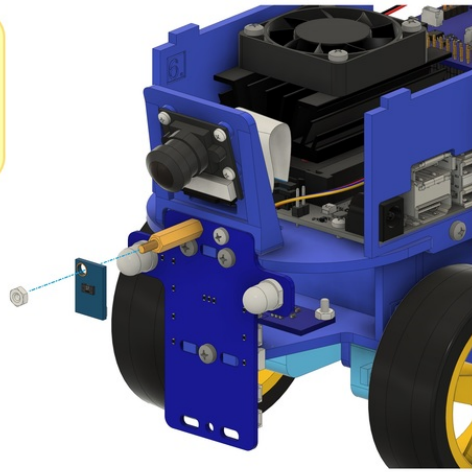




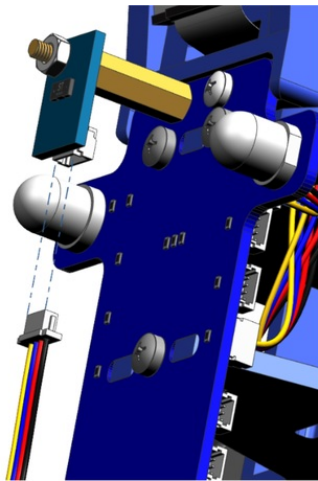
45



46

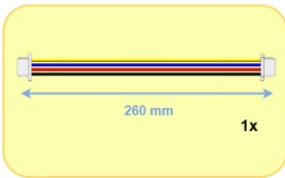
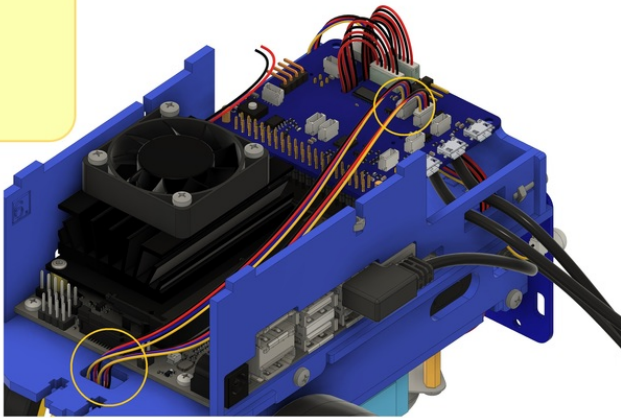


47

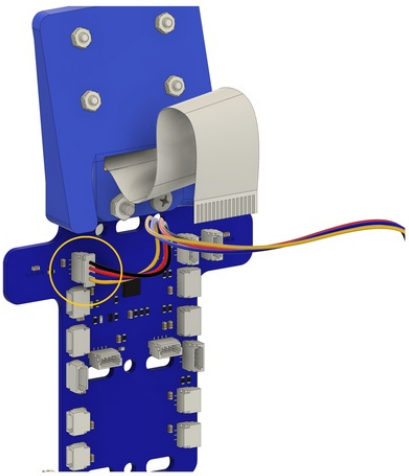




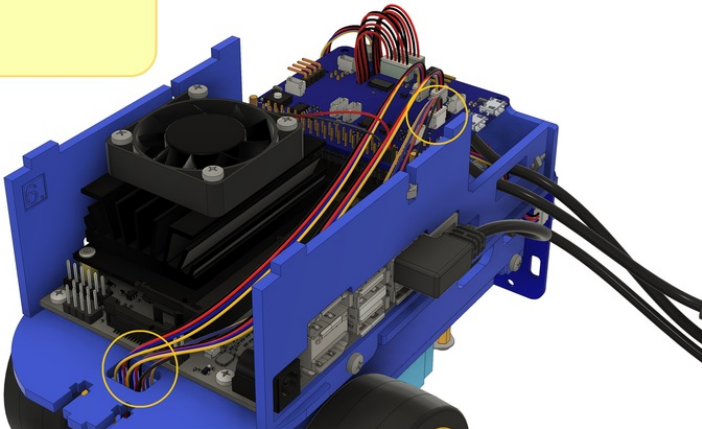
48



49

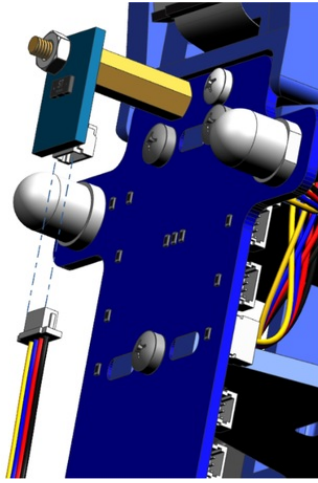


50





51

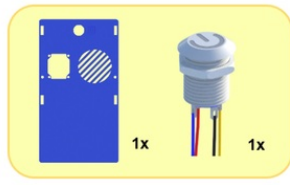


## Top Deck Assembly

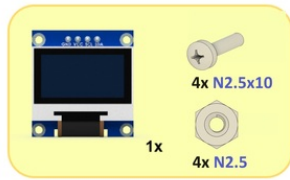
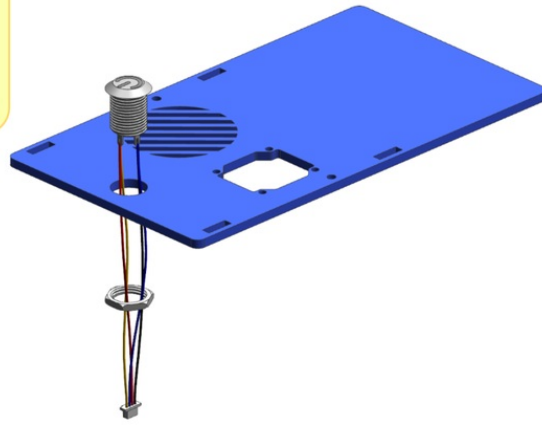
This last section (steps 52 to 62) guide through the assembly of the top deck:



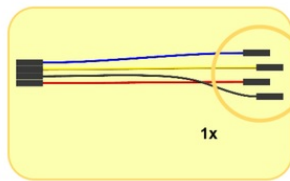
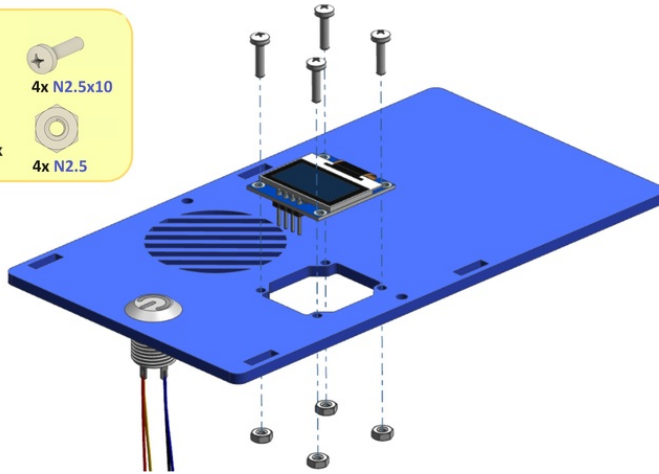




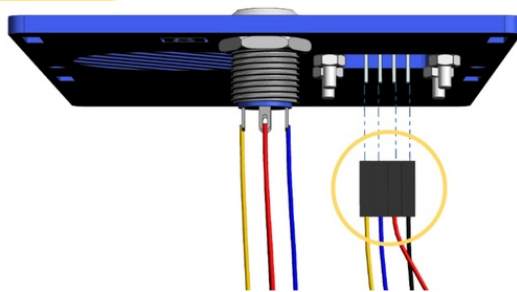
52

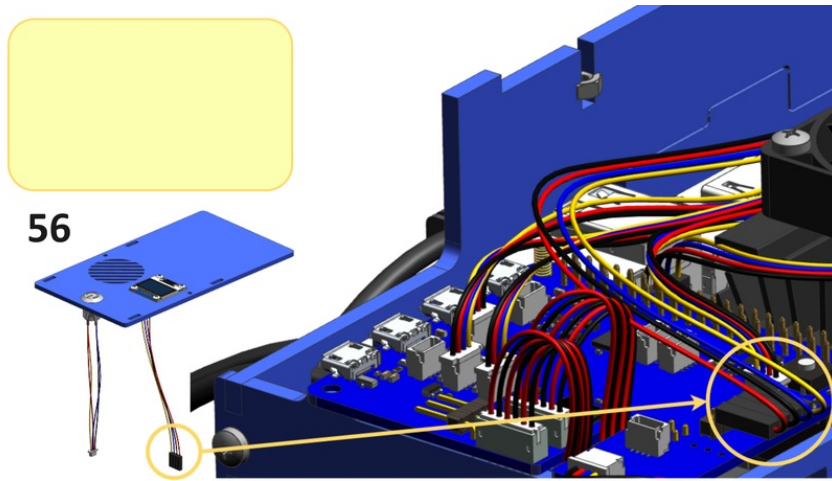
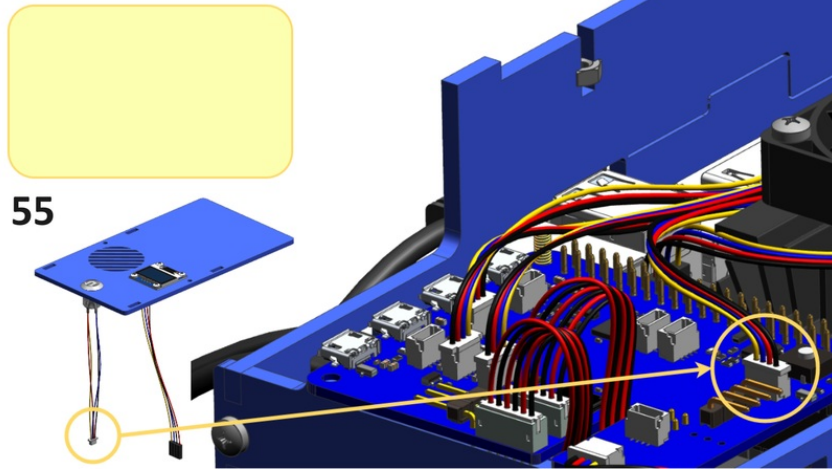


53



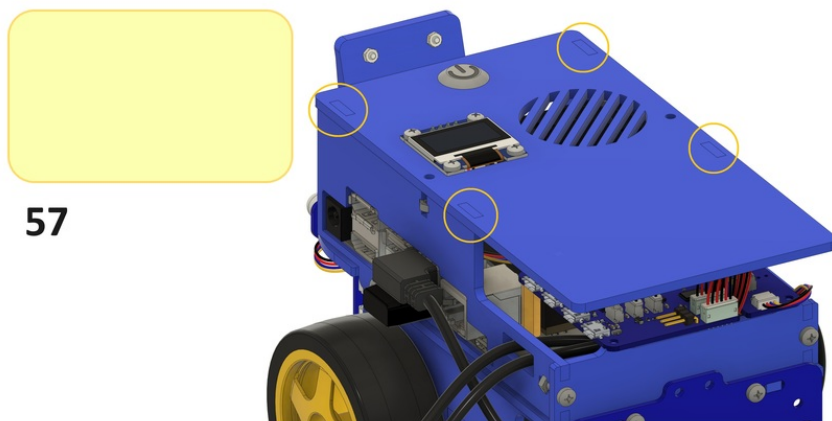
54





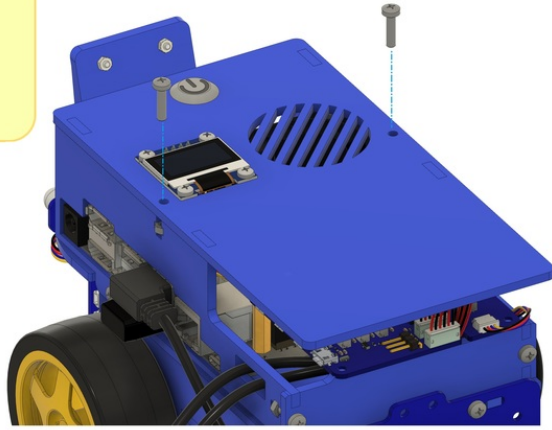
**Note**

On the display board you will see 4 pins: SDA, SCL, VCC, GND. Make sure that the correct color wire is connected to the corresponding pin on the HUT.

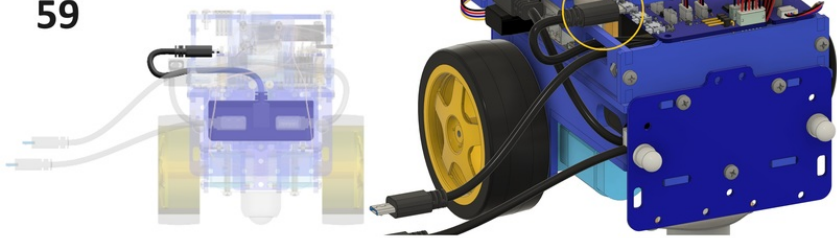




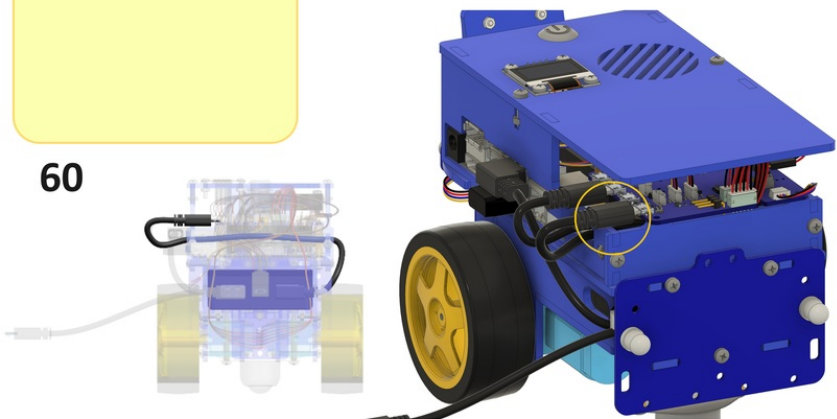
58



59

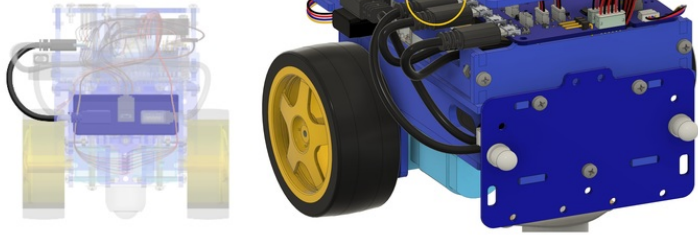


60



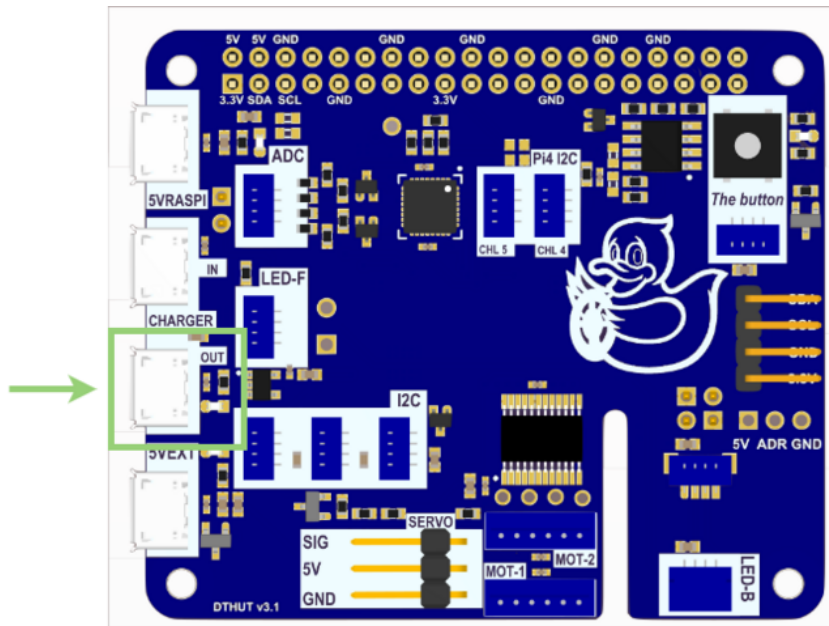


61



## 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.



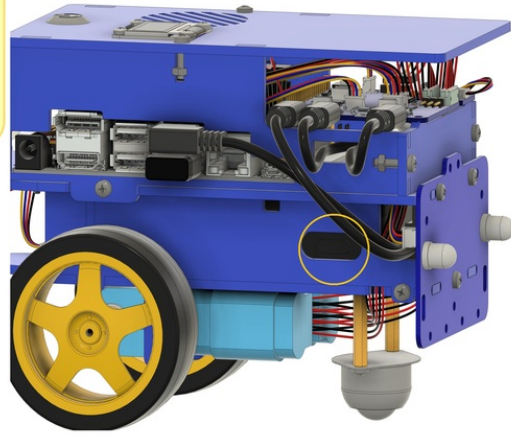
### **Warning**

Always plug and unplug USB cables from the HUT with care!

Make sure the SD card was flashed successfully without any error during the process and it is plugged in Jetson Nano under the main board. ONLY then you can continue to follow the instructions. Once your Duckiebot is fully charged, you can press the button of the battery on the side to power it up. 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.



62



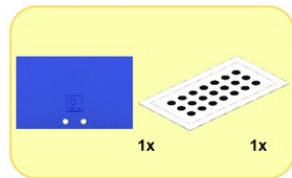
Congratulations, your Duckiebot **bb21j** is now completely assembled.

### Additional Parts

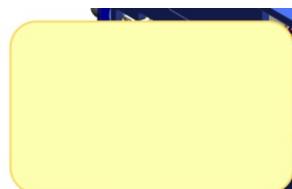
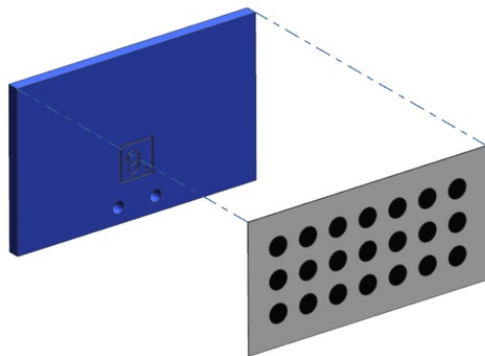
These additional parts are not always necessary.

#### Back Pattern

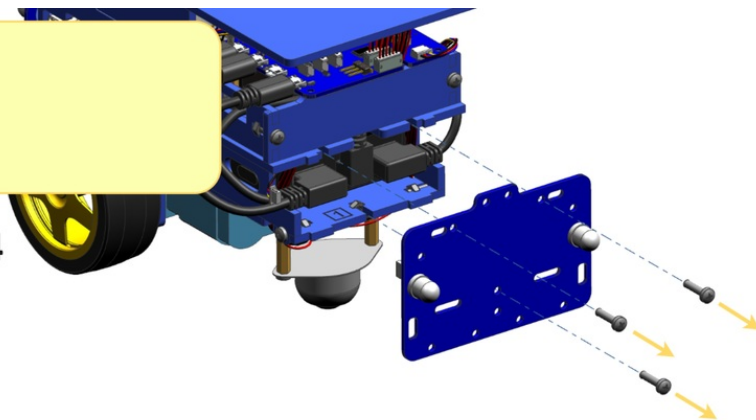
The back pattern enables the traffic management behavior used in challenges with vehicles (e.g., **LFV**, **LFIV**, etc.).



63

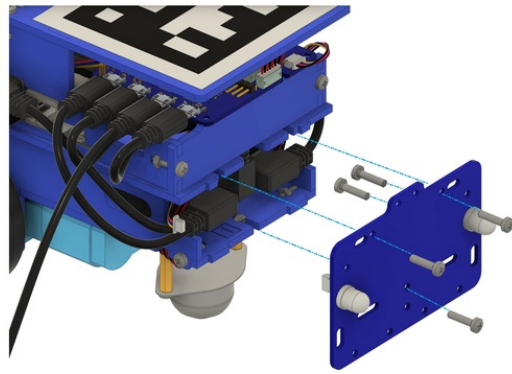


64

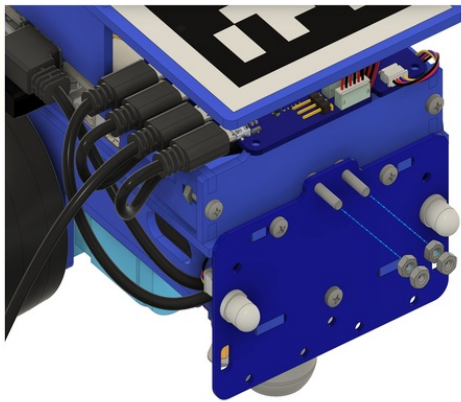




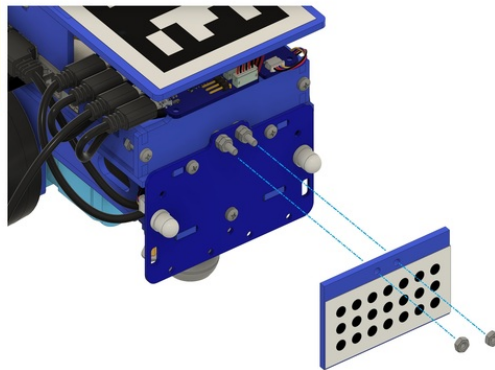
65



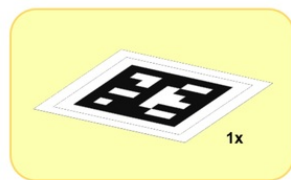
66



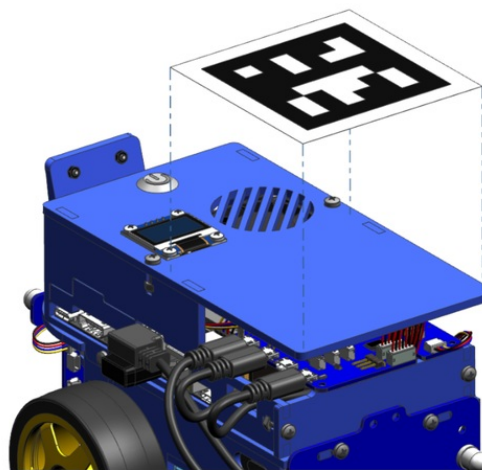
67



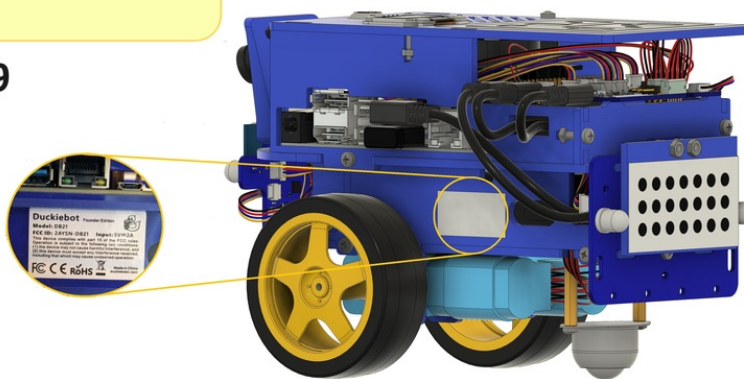
This top facing April Tag enables localization in [Duckietown Autolabs](#).



68



69



**Note**

Please place the sticker somewhere on the Duckiebot.

## 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 image (configuration **DB21M** if using a Jetson Nano 2 GB, **DB21J** if using a Jetson Nano 4 GB).

**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.
- Which LEDs you should see:
  - Blue LEDs when the robot is off and you plug in the charger (after it's off);
  - White LEDs when the robot is on;
  - Random color LEDs when the robot is powering on.

## Troubleshooting

| Troubleshooting |   |
|-----------------|---|
| SYMPTOM         | I can't find the blue chassis.  |
| RESOLUTION      | It's <i>under</i> the white foam in the Duckiebox. Remove the inner packaging to access it. |

| 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. |

| Troubleshooting |  |
|-----------------|--|
| 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. |

| Troubleshooting |   |
|-----------------|---|
| 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. |

| Troubleshooting |   |
|-----------------|---|
| SYMPTOM         | I can't screw the omni-directional wheel right; the screws don't fit all the way in the standoffs.  |
| RESOLUTION      | Occasionally the standoffs are not fully threaded due to manufacturing inefficiencies. The solution is to orient, in case of need, the shorter threaded stand-off side towards above, on the side of the chassis. Alternatively, shorter screws (provided in the package) can be used. If everything else fails, a "dirty" but effective solution is to use two spare nuts to mitigate tolerances, as shown in the picture below: |





### Troubleshooting

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](mailto:hardware@duckietown.com) for assistance.

### Troubleshooting

SYMPTOM

The wheels wiggle and/or 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 the wheels are not touching the screws of the motor mounts. Alternatively, screws are provided to fix the wheels to the motor axles. Make sure not to tighten the screws too hard, or they will add resistance to the spinning of the wheels (you can find the sweet spot by turning the wheel by hand and feeling the resistance torque).



**Fig. 36** Screws will keep the wheels in place. Do not tighten too hard!

### Troubleshooting

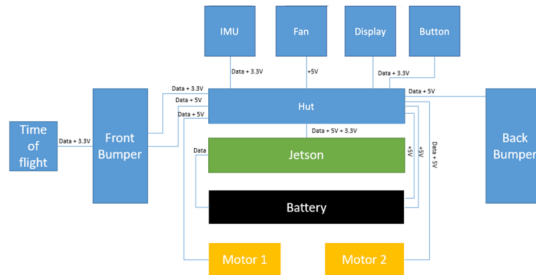
**SYMPTOM** My Duckiebot is driving backwards when pressing the key for straight forward.

**RESOLUTION** Try swapping the motor cables on the HUT connectors. Double-check the motor cables are connected to their respective ports as indicated above.

### Troubleshooting

**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:



**Fig. 37** Block diagram of electrical and data connections for the **DB21J** and **DB21M**.

### Troubleshooting

**SYMPTOM** ToF sensor not detected

**RESOLUTION** If you have Jetson Nano with 2 camera ports, try to plug ToF into CH4 instead of CH6 on the front LED bumper board.

### Troubleshooting

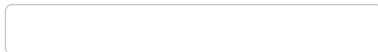
**SYMPTOM** I have a non-functional sticker with weird symbols left over. What to do with it?

**RESOLUTION** Duckiebots are FCC and CE certified, which means they comply with (and surpass) material quality (e.g., RoHS 2.0) and electrical interference standards (FCC, CE). You should place the sticker somewhere on the Duckiebot. We suggest a position out of sight (of other Duckiebots) to prevent detection issues in more advanced applications. For example, under the wheels:





**Fig. 38** Place the sticker somewhere a human can read it, but another Duckiebot cannot.



### Troubleshooting

|            |  |
|------------|--|
| 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!   |

## Assembly - Duckiebot DB21M

### Attention

These instructions are not of the latest Duckiebot model. Unless you are handling a legacy Duckiebot (version **DB21M**), you should follow these instructions instead: [DB21J assembly instruction](#).

### See also

Not sure what is your Duckiebot? Read more about [Duckiebot configurations](#).

#### What you will need

- Duckiebot **DB21M** parts ([get a DB21M](#)). If you are unsure what version of Duckiebot you have, check this overview of all existing [Duckiebot configurations](#).
- A micro SD card with the Duckiebot image on it. The procedure to flash the SD card is explained [here](#).
- 3 hours of assembly time.

#### What you will get

- An assembled Duckiebot in configuration **DB21M**.

## 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!).

## Video tutorial

## **i** What's in the box and assembly video

[Duckiebot DB21M unpacking and assembly](#) from [Duckietown](#) on [Vimeo](#).

## Overview

A Duckiebox contains the following components:



*Fig. 39* 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](#)
- [Troubleshooting](#)

The Troubleshooting section at the bottom of this page provides resolutions to common symptoms.

## Preliminary Steps

### 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.

## Tip

“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.

## Plastic cover

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

## 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).

| Index | Picture   | Dimension             |
|-------|---|-----------------------|
| B1    |    | Nylon screw M2x8mm    |
| B2    |    | Nylon screw M2.5x10mm |
| B3    |    | Metal screw M3x8mm    |
| B4    |    | Metal screw M3x12mm   |
| B5    |    | Metal screw M3x30     |
| N1    |   | Nylon nut M2          |
| N2    |  | Nylon nut M2.5        |
| N3    |  | Metal nut M3          |
| N4    |  | Nylon nut M3          |






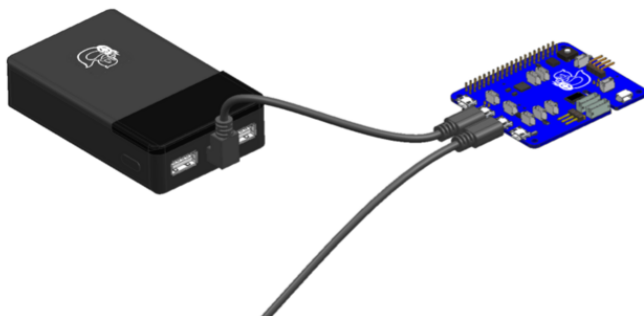
| Index | Picture   | Dimension                 |
|-------|---|---------------------------|
| S1    |  | Nylon stand-off M3x5mm    |
| S2    |  | Metal stand-off M2.5x18mm |
| S3    |  | Metal stand-off M3x25mm   |
| S4    |  | Spacer M6x12x1.5mm        |
| S5    |  | Spacer M2.5x5x1           |

Fig. 40 Overview of interlocking parts

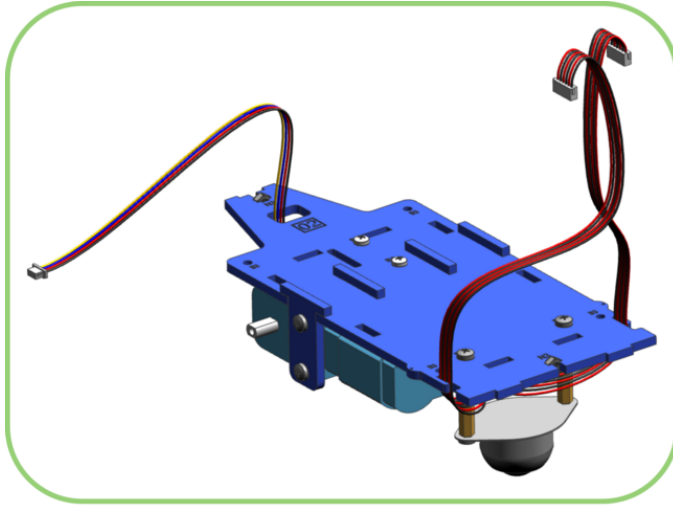
## Drive Train

### 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.



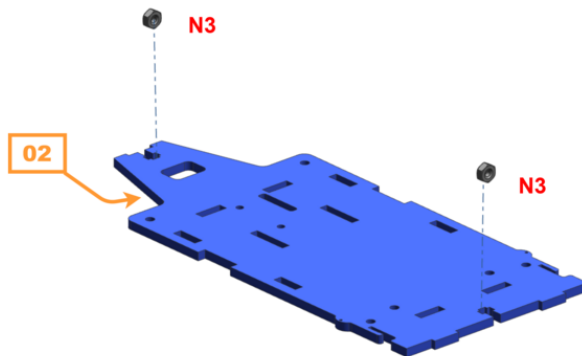
In the following steps (1 to 12) you will build the base plate assembly.



### Step 1

Take the baseplate (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).

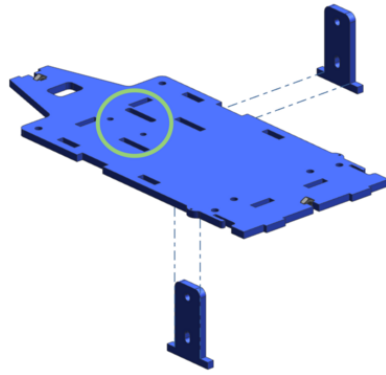


#### **Note**

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.

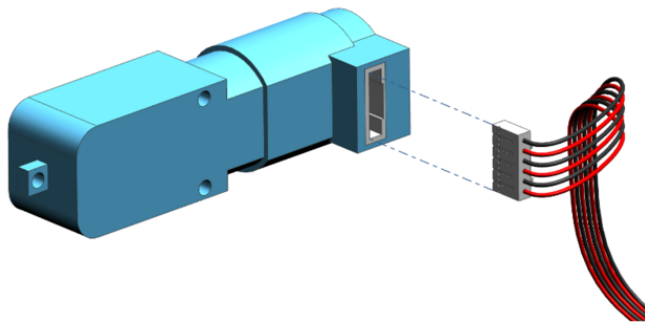
### Step 2

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.



### Step 3

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

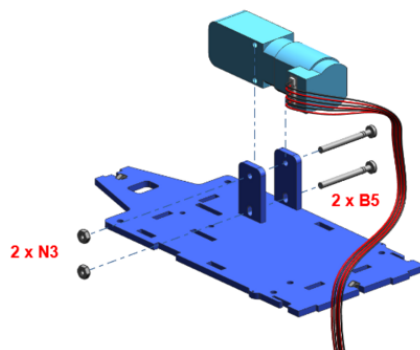


#### Tip

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.

### 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).

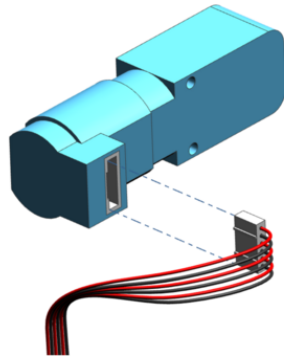


#### Warning

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.

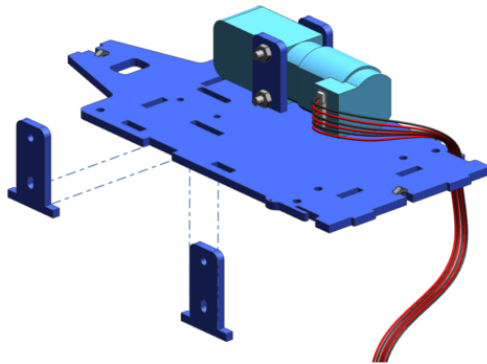
### Step 5

Connect the second 6-pin motor cable to the second motor.



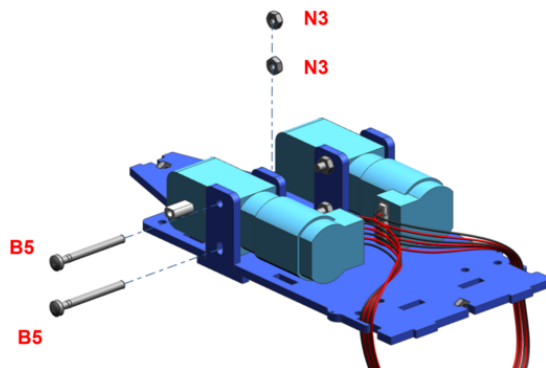
### Step 6

Insert the other two motor plates into the base plate, similarly to step 2.



### Step 7

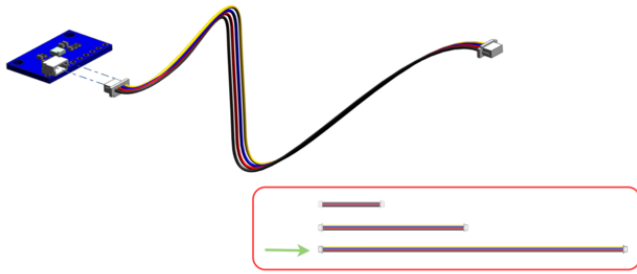
Tighten the second motor with two M3x30 metal screws (B5) and two metal M3 nuts (N3) and place the cable, similarly to steps 3 and 4.



### Step 8

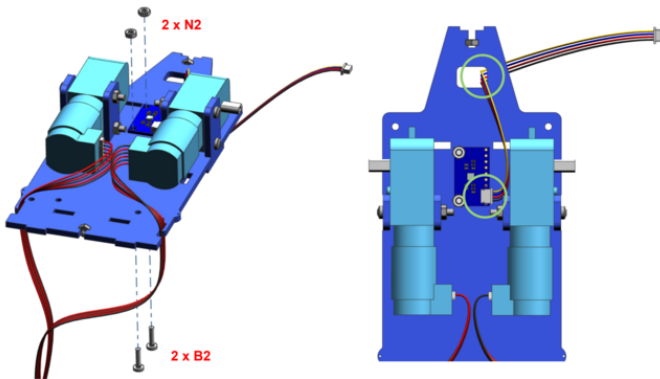
Connect one of the longest 4-pin cables to the white connector of the IMU (Inertial Measurement Unit) board.





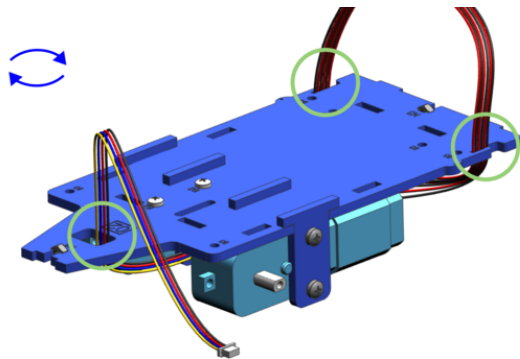
### Step 9

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



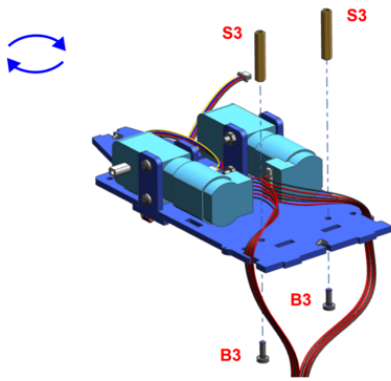
### Step 10

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).



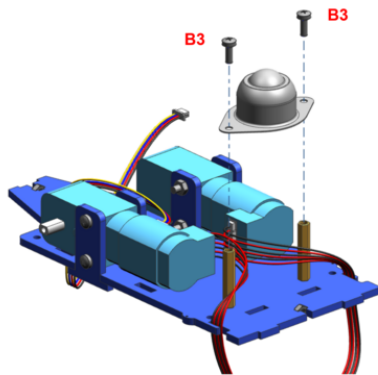
### Step 11

Flip the assembly again and mount the two M3x25 stand-offs (**S3**) to the bottom plate. Use two metal M3x8 screws (**B3**).



## Step 12

Mount the omni-wheel to the stand-offs using two of the metal M3x8 screws (B3).



### Tip

If you note the screws don't go all the way, try flipping the stand-off.

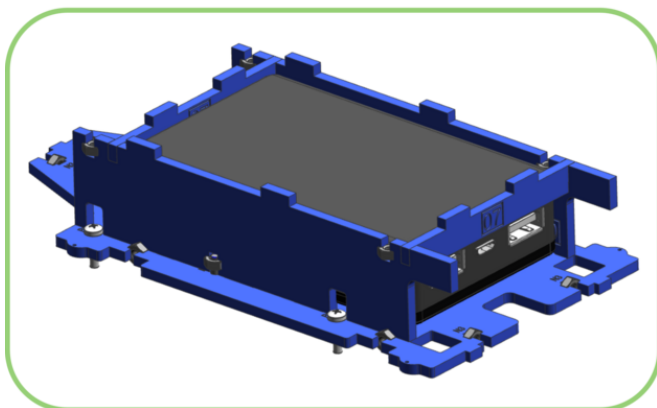
## Verify the assembly

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!

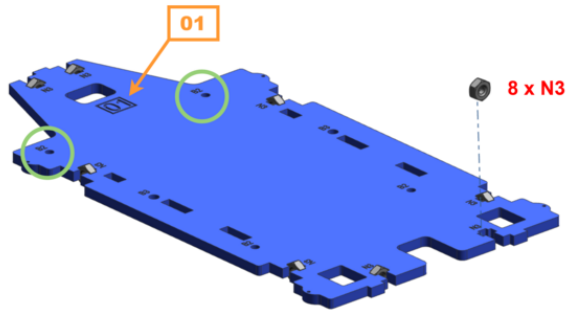
## Battery Pack Installation

The following steps (13 to 18) guide through the *Duckiebattery* assembly:



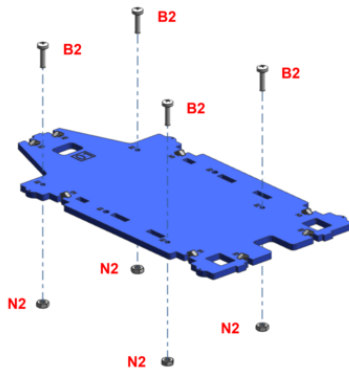
## Step 13

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).



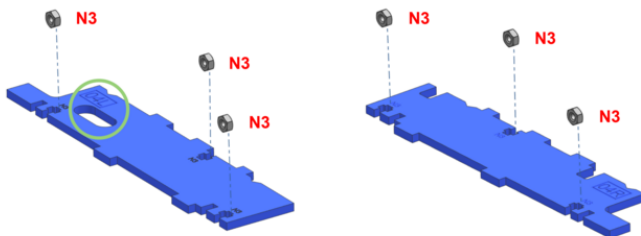
#### Step 14

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



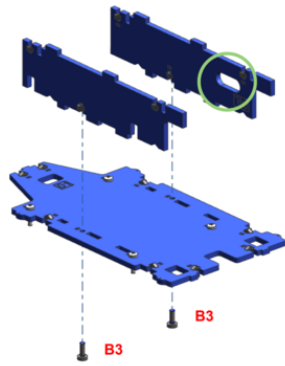
#### Step 15

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



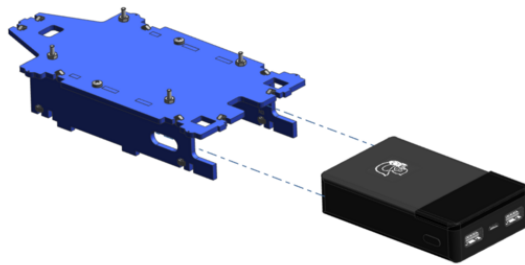
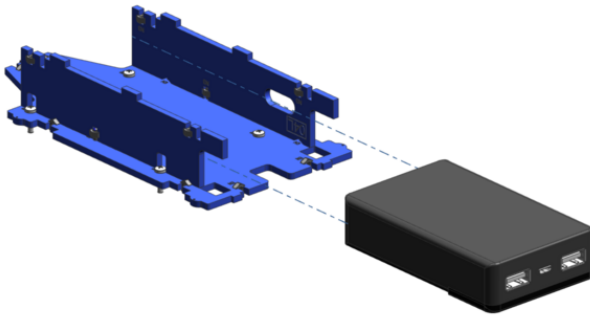
#### Step 16

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!



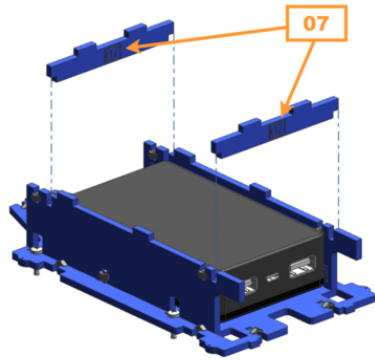
### Step 17

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.



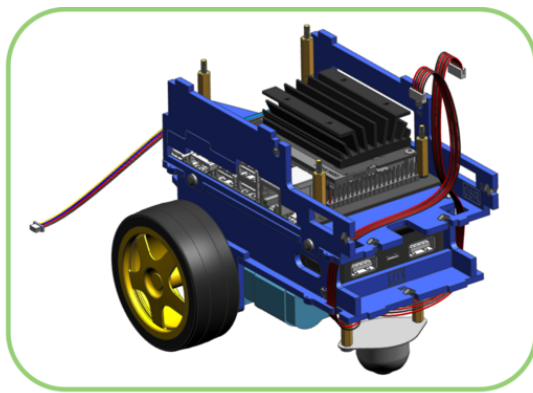
### Step 18

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



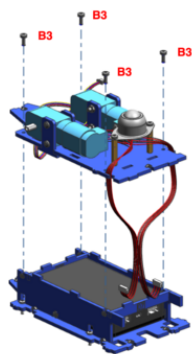
## Computation Unit and Rear Assembly

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:



### 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.

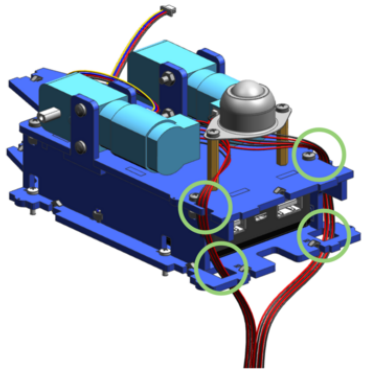


#### **Attention**

This is a tricky step and might require a few tries to get it right. Remember that the Robotcist is patient and welcomes challenges with joy and determination!

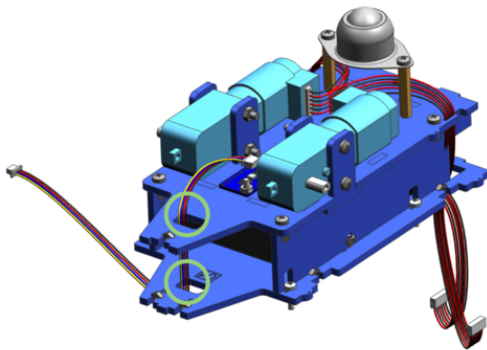
### Step 20

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



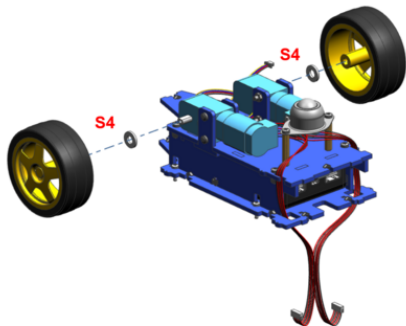
### Step 21

Do a similar procedure for the cable of the IMU unit.



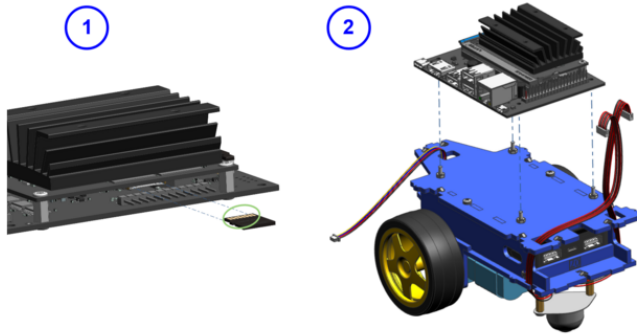
### Step 22

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



### 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!

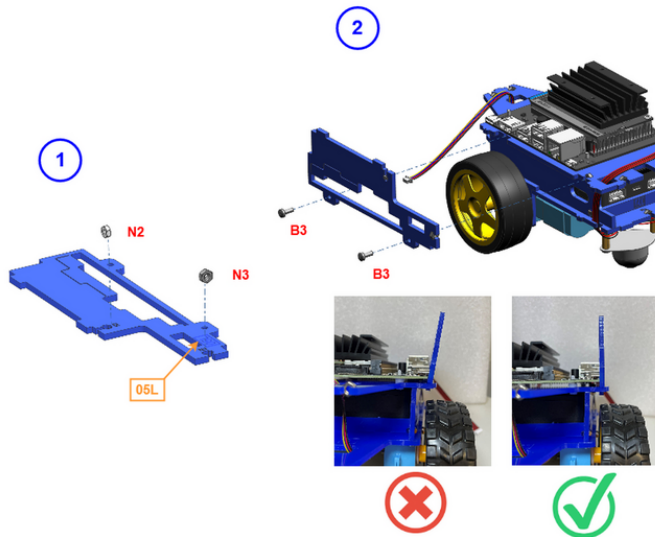


### Step 24

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).

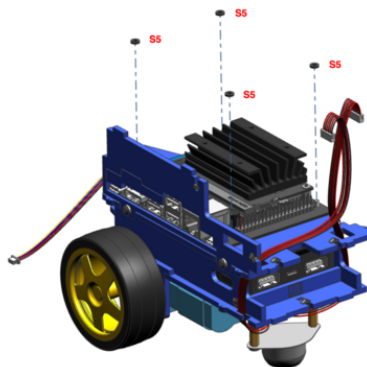
Then secure the plate to the frame using two metal M3x8 screws (**B3**).

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



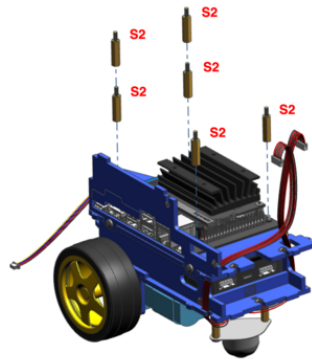
### Step 25

Place 4 of the (**s5**) spacers on the nylon screws holding the NVIDIA Jetson Nano board.



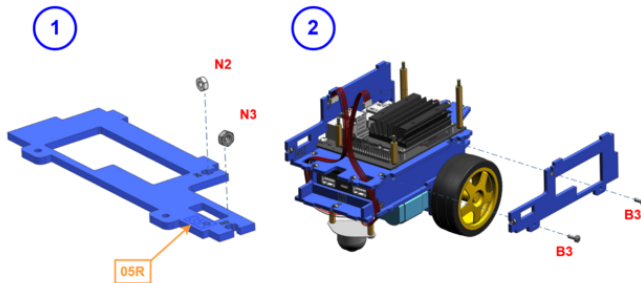
### Step 26

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.



### Step 27

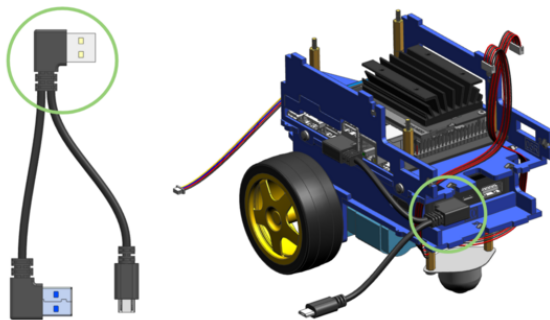
Take the side cover carrying the number **05R**. Insert a nylon M2.5 nut (**N2**) and a metal M3 nut (**N3**) into the plate, similarly to step 24 (note the engraving: **N2** and **N3** on the plate itself). Then screw the plate to the frame using two metal M3x8 screws (**B3**).



## Cable Management

### 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.



#### **Attention**

The micro USB connector must **not** be connected at that stage!

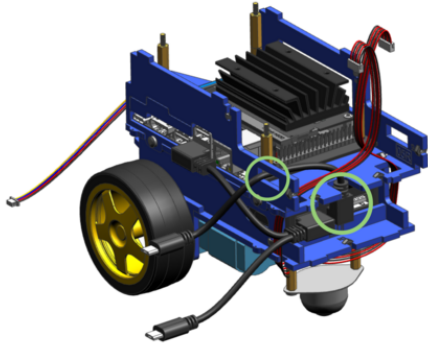
### Step 29

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!



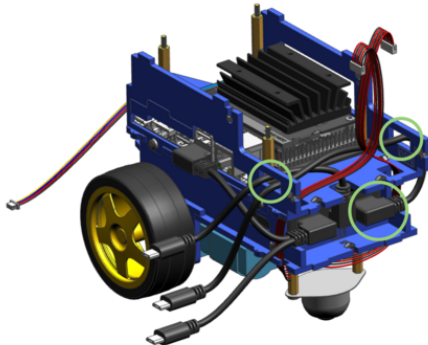
 **Tip**

An experienced Robotcist 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!



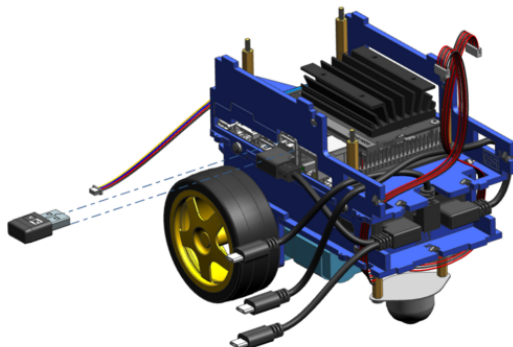
**Step 30**

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.



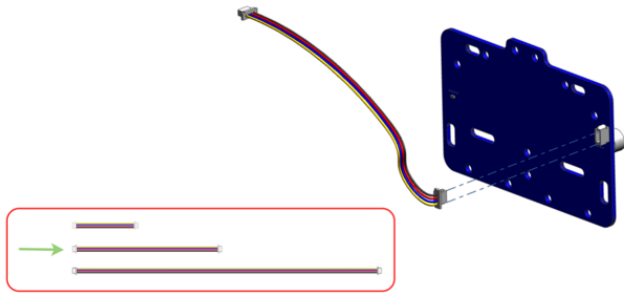
**Step 31**

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



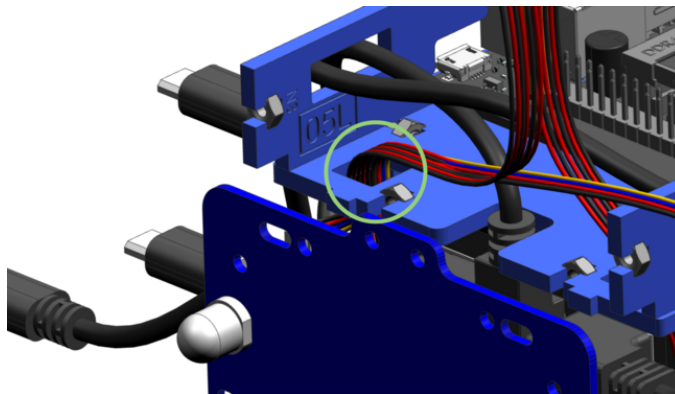
**Step 32**

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



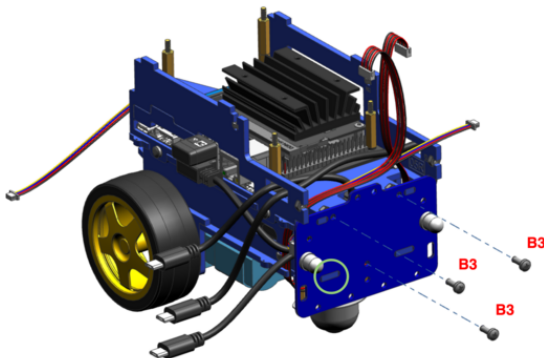
### Step 33

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.



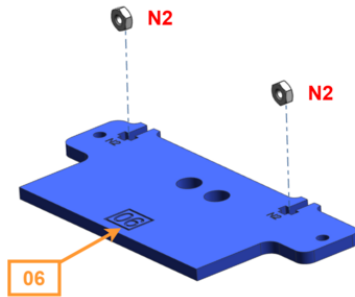
### Step 34

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).



### Step 35

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

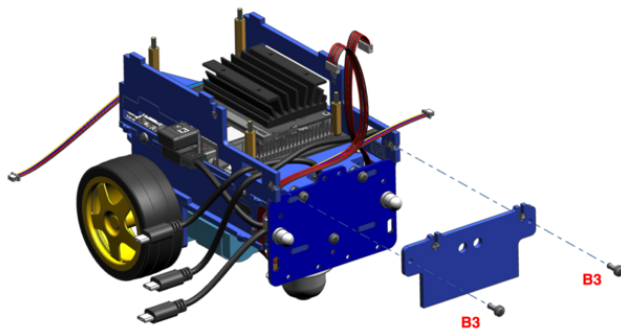


### Step 36

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

#### **i Attention**

The 06 is not symmetric and the orientation matters. If the number 06 is pointing towards the Duckiebot, we are good to go!

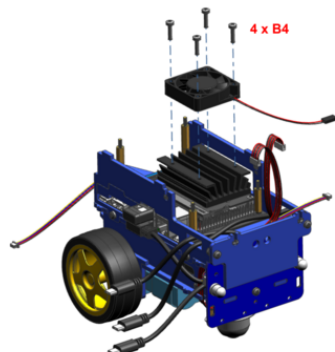


### Step 37

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 it's way through the heat sink the first time).

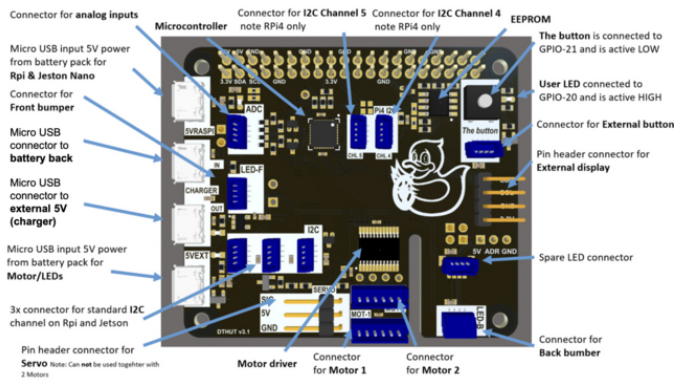
#### **i Note**

You don't need to tighten the screws completely but the fan must sit tight.

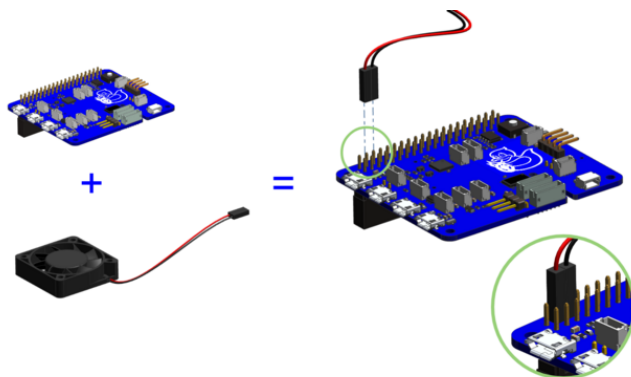


### Step 38

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**).

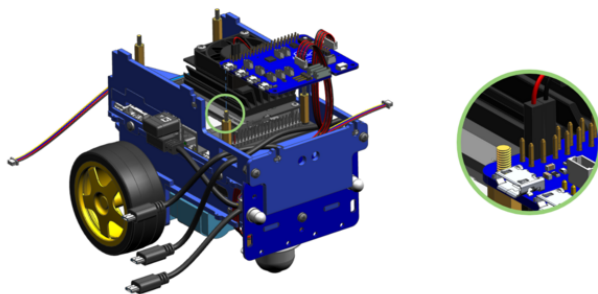


Plug in the fan cable to the two pins as shown (note the orientation of the black and red cables!).



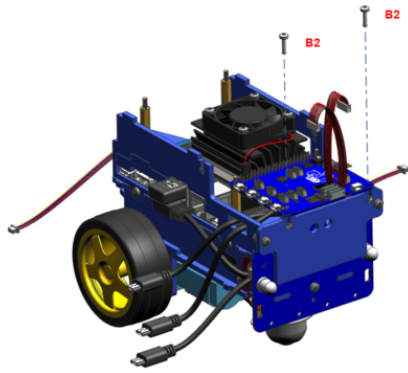
### Step 39

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.



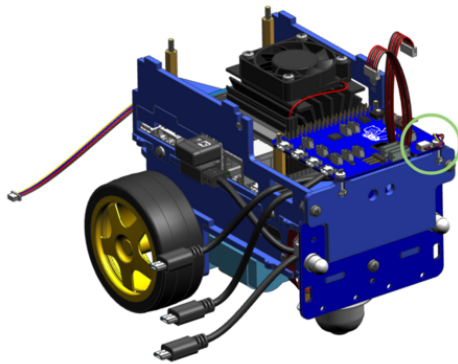
### Step 40

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.



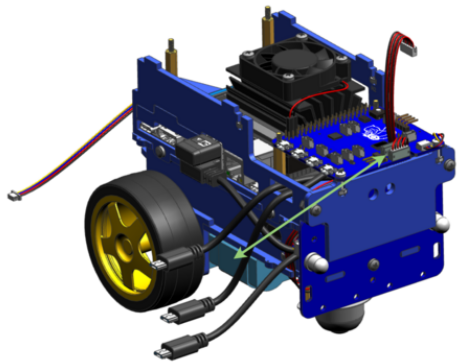
#### Step 41

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



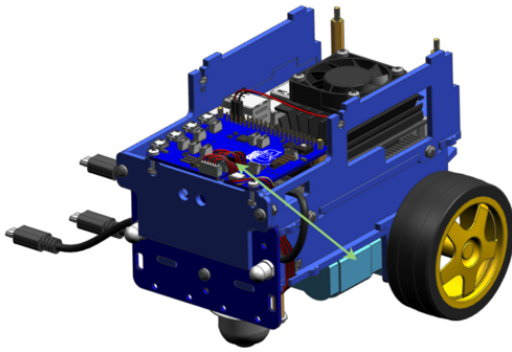
#### Step 42

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



#### Step 43

Connect the second 6-pin motor cable of the right motor to the other connector.

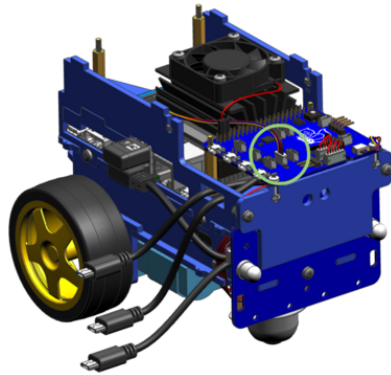


**Note**

If you swap the motor cables, your Duckiebot will probably drive backwards when it is supposed to drive forwards :)

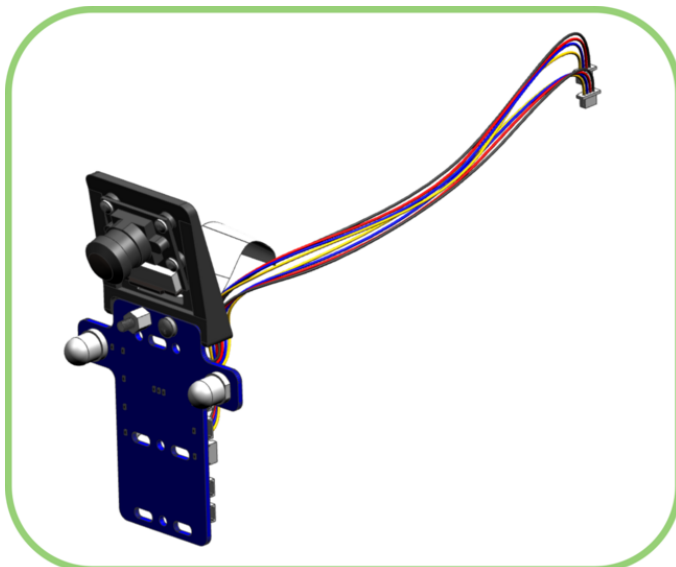
**Step 44**

Connect the 4-pin cable from the IMU to the corresponding plug shown in the picture.



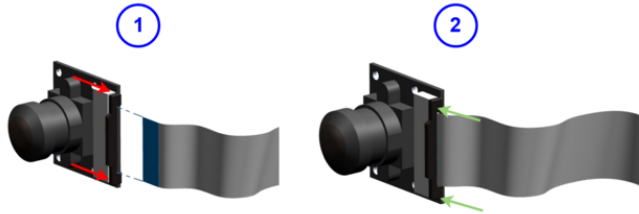
**Front Assembly**

The following steps 45 to 52 will guide you through the assembly of the camera unit as well as some more electronics and cables.



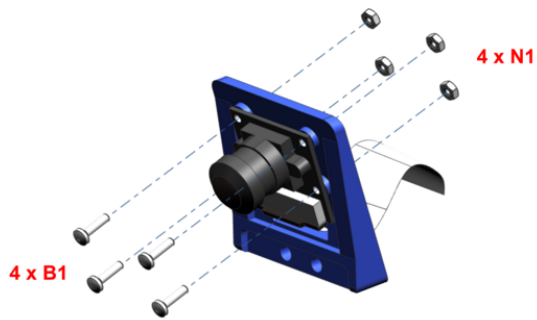
**Step 45**

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.



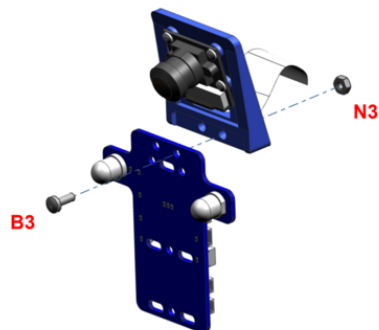
#### Step 46

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.



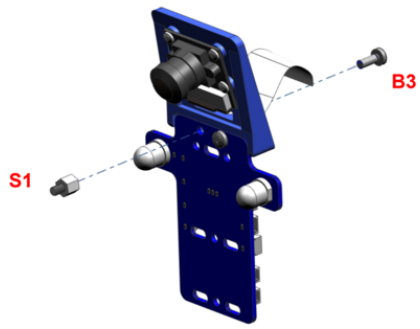
#### Step 47

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



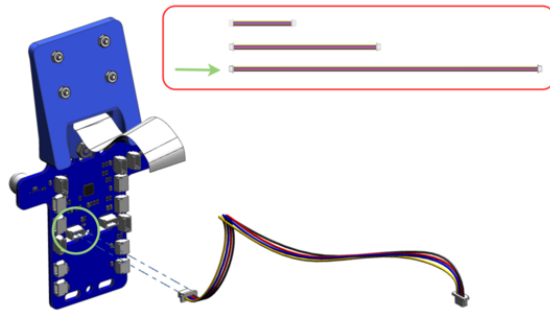
#### Step 48

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



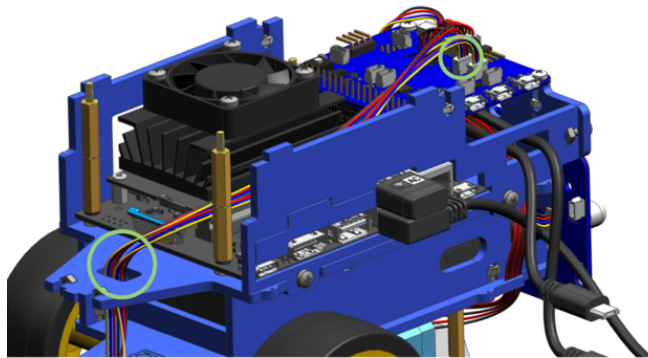
#### Step 49

Take one of the longest 4-pin cables and connect it to the front bumper board as shown.



#### Step 50

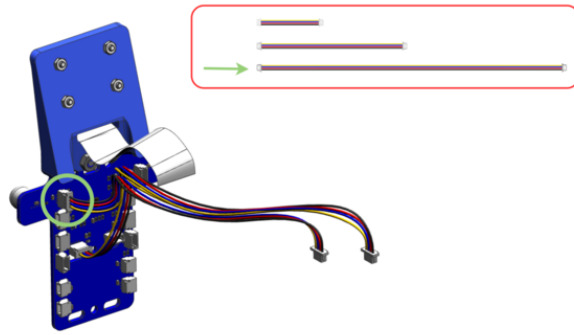
Wire the cable that you have just connected (step 49) through the upper plate and the connect it to the connector on the **HUT**, as shown below.



#### Step 51

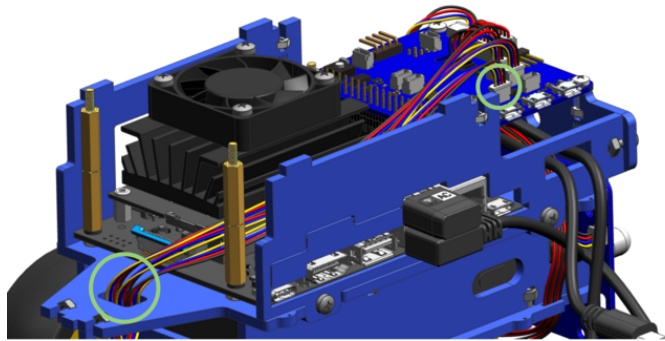
Take the last of the long 4-pin cables and connect it to the front bumper board, similarly to step 49.





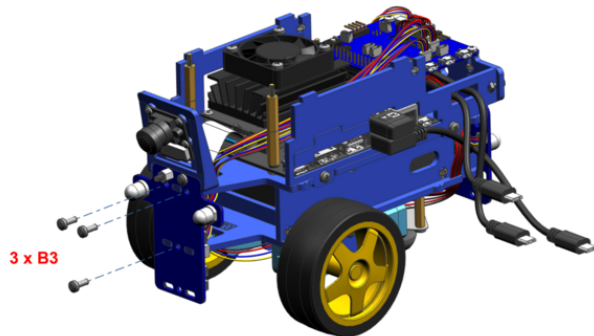
### Step 52

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



### 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.

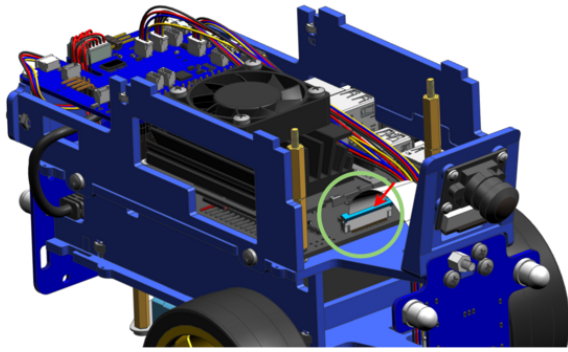


### Step 54

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.

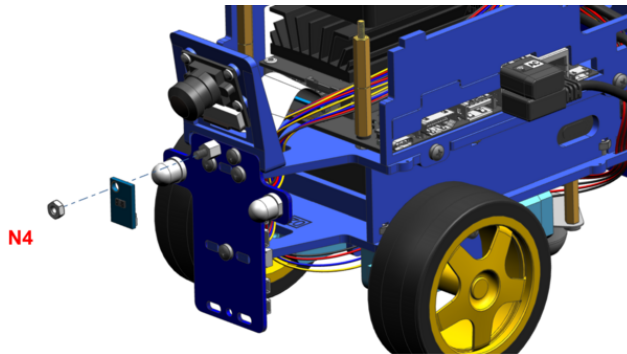
#### **Attention**

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).

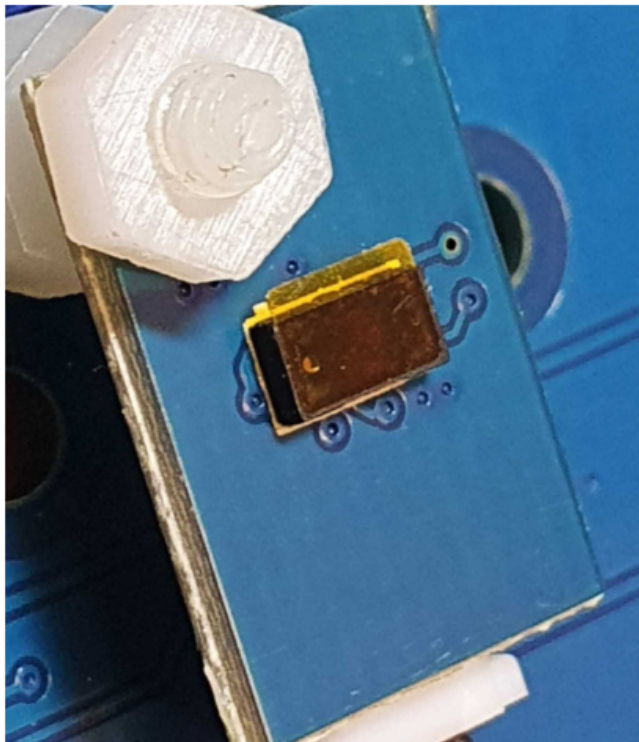


### Step 55

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.



### Attention (04/2023)

Do not proceed to Step 56.

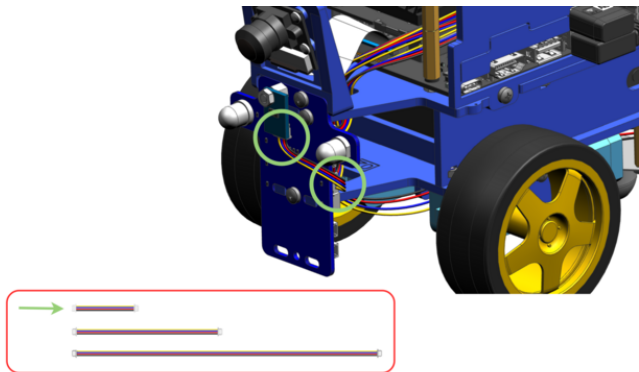
It is now recommended that you connect your ToF Sensor directly to the Duckiebot HUT. To do this:

1. Locate the 260mm cable that you connected to the I2C port on the front bumper in Step 49.
2. Disconnect the cable from the front bumper I2C port. Do not disconnect the other end of the cable from the HUT.
3. Connect the now free end of the cable into the ToF sensor port shown in Step 56.
4. Disregard Step 56.

You can now continue on to the [Top Plate Assembly](#).

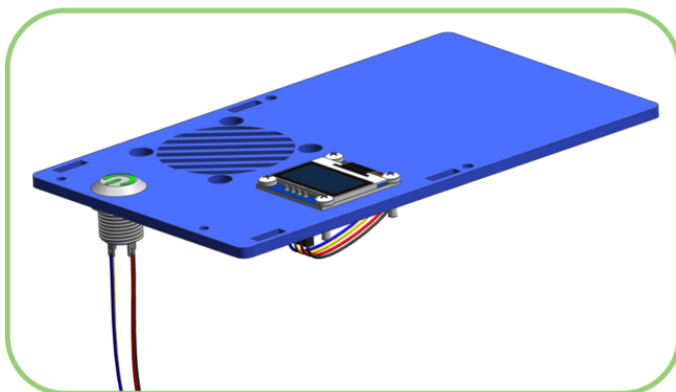
### Step 56

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



### 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.



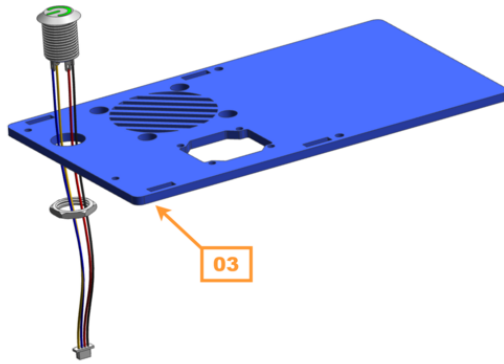
### Step 57

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

**! Attention**

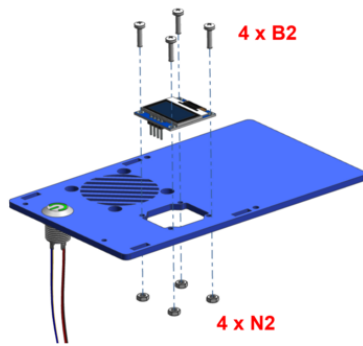
Mind the orientation; if the number is pointing downwards, we are good to go!

Once the button is pushed in completely, tighten it again with the flat nut.



**Step 58**

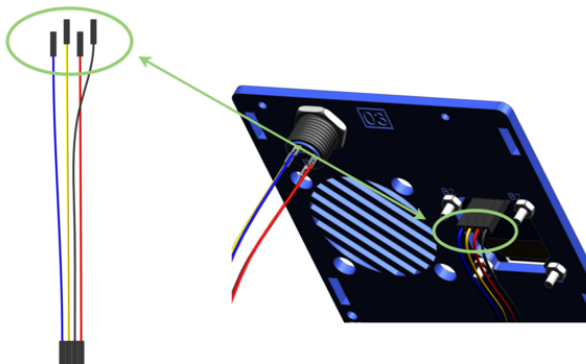
Mount the screen to the plate in a way the pins of the screen are pointing towards the button. Use 4 nylon M2.5 screws (B2) and 4 nylon M2.5 nuts (N2) for this.



**Step 59**

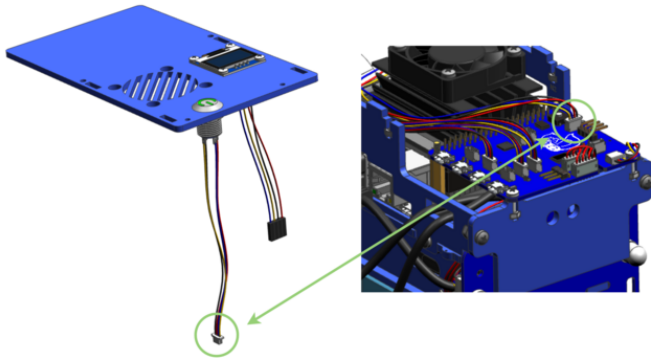
Have a look at the pin descriptions on the screen. Take the 4-pin cable with the long black connectors and connect the 4 loose ends to the screen.

Follow this pattern: GND-black, VCC-red, SCK-yellow, SDA-blue.



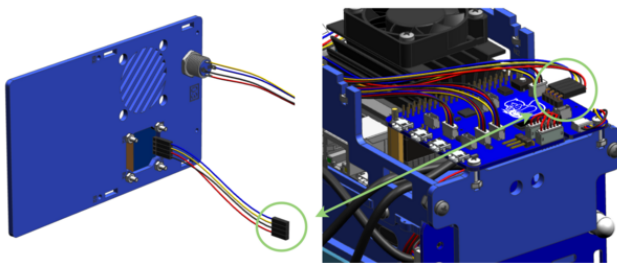
**Step 60**

Connect the end of the cable from the button to the connector on the HUT, as below.



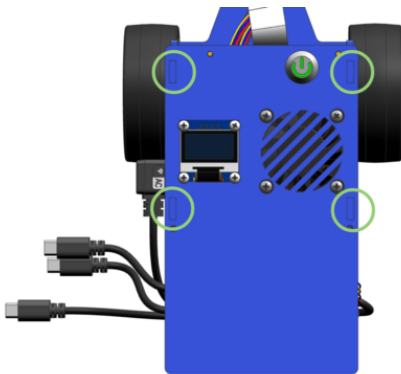
### Step 61

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.



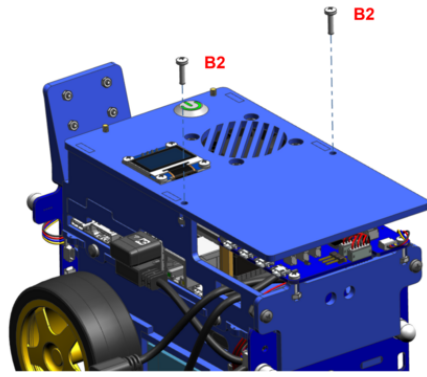
### Step 62

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.



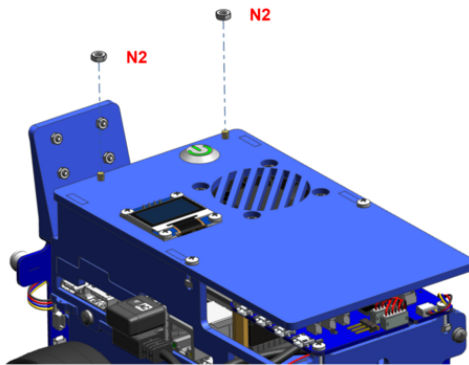
### Step 63

Tighten the cover part using two nylon M2.5x10 screws (**B2**).



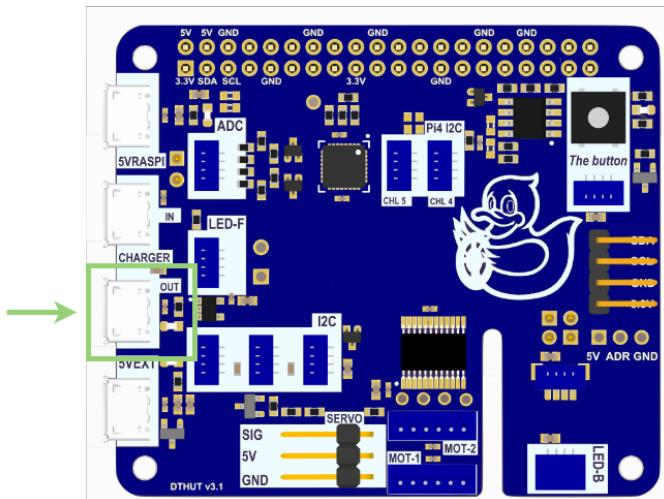
#### Step 64

Then, tighten the cover using two nylon M2.5 nuts (N2).



#### 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.

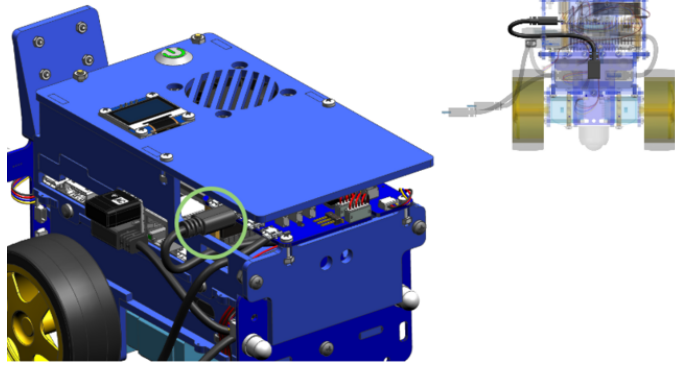


#### **Warning**

Always plug and unplug USB cables from the **HUT** with care!

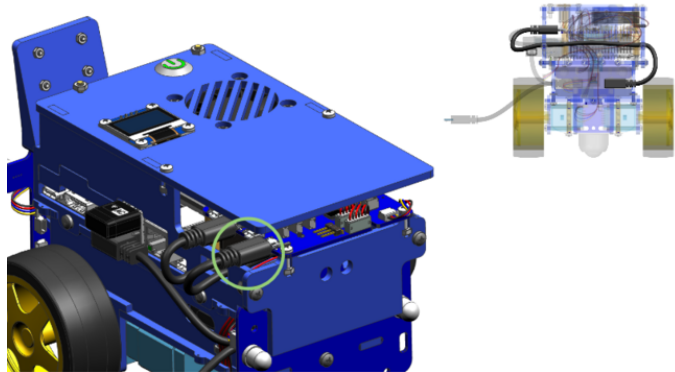
#### Step 65

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.



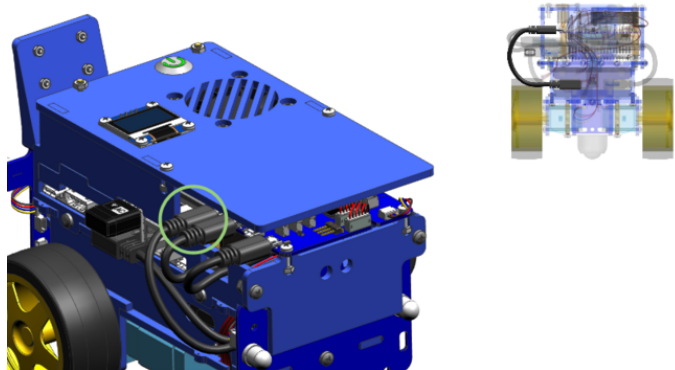
#### Step 66

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



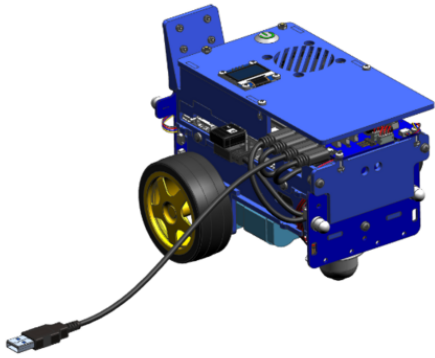
#### Step 67

Finally, connect the last cable to the **HUT**.

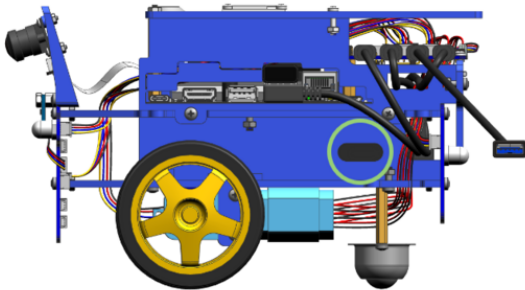


#### Step 68

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.



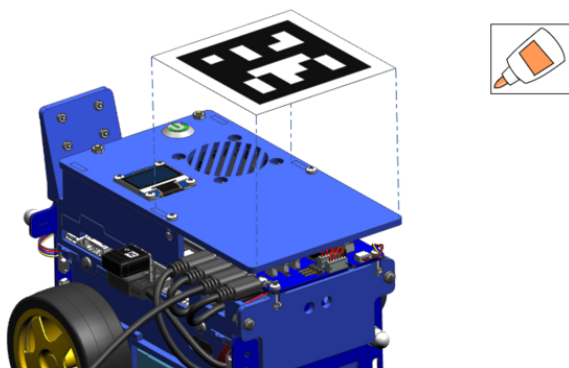
Once your Duckiebot is fully charged, you can press the button of the battery on the side to power it up (do this ONLY if a flashed SD card has been inserted).



## Additional Parts

### Step 69

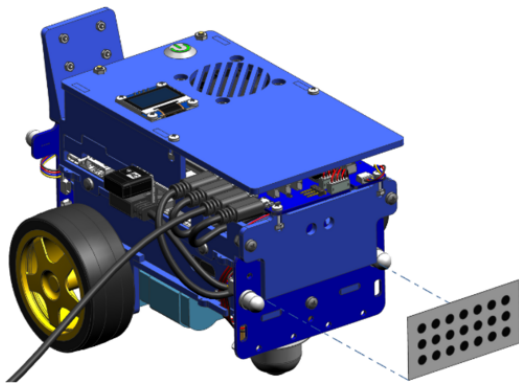
If you have an April tag take some glue and put it in between the two nylon screws on the top of you Duckiebot.





## Step 70

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



## 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.

## Troubleshooting

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | I can't find the blue chassis.  |
| RESOLUTION | It's <i>under</i> the white foam in the Duckiebox. Remove the inner packaging to access it. |

### 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. |

### Troubleshooting

|            |  |
|------------|--|
| 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. |

### Troubleshooting

|            |   |
|------------|---|
| 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. |

### Troubleshooting

|            |  |
|------------|--|
| 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 standoff 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. |

### Troubleshooting

|            |  |
|------------|--|
| 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 <a href="mailto:hardware@duckietown.com">hardware@duckietown.com</a> . |

### Troubleshooting

|            |  |
|------------|--|
| 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. |

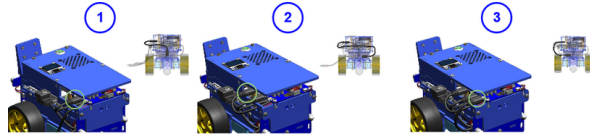
### Troubleshooting

|            |   |
|------------|---|
| 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. |

### Troubleshooting

|            |  |
|------------|--|
| 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).



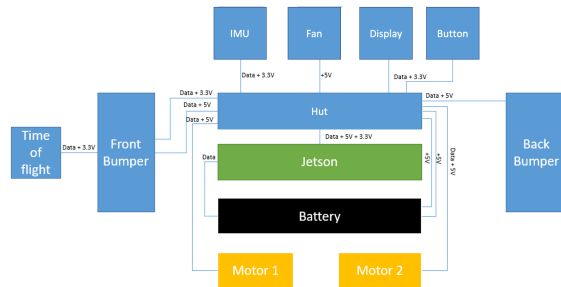
### Troubleshooting

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:



### Troubleshooting

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!

## Duckiebot FAQ Guide

This FAQ page collects common roadblocks you might run into when setting up your Duckietown environment and operating your Duckiebot.

Each symptom and resolution are also available on the pages they relate to throughout the manual, so be sure to watch for troubleshooting sections and carefully complete checkpoints as you progress.

If you don't find the solutions you need in this book, be sure to first search the Duckietown Stack Overflow and Slack communities for previous answers, then post your own question following the support guidelines on Slack.

### FAQs: Booting your Duckiebot

#### Troubleshooting

SYMPTOM

I pressed the power button on top to boot my Duckiebot but nothing happened.

RESOLUTION

Power on your Duckiebot using the button on the side of the Duckiebattery. The top button is only for powering off. You can also learn more about how to handle your Duckiebot in [Handling - Duckiebot DB21](#).

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | My Duckiebot does not appear to boot after pressing the power button on the battery. I don't see a green light on the HUT or the Jetson Nano.  |
| RESOLUTION | Refer back to <a href="#">Assembly - Duckiebot DB21J</a> , and check each of your cable connections. Confirm the start and end port of each power cable from the battery. The battery must be charged fully as shown in the first assembly step. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot is getting power but does not appear to be booting. The Wifi dongle is not blinking.           |
| RESOLUTION | Make sure you flashed the SD card following the instructions in <a href="#">Setup - Duckiebot SD Card</a> . |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | My Duckiebot is getting power but does not appear to be booting. The Wifi dongle is not blinking.  |
| RESOLUTION | Make sure that you correctly specified the model of your Duckiebot when initializing the SD card.<br><br>If you have a Duckiebot with a 2GB Jetson Nano - the model is DB21M<br><br>If you have a Duckiebot with a 4GB Jetson Nano - the model is DB21J<br><br>If you are not using a Jetson Nano, the model is the model of your Duckiebot (ex. DB19 or DBR4) |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The Duckiebot screen does not turn on even though it shows up in <code>dtc fleet discover</code> , and the Dashboard is accessible. The ToF sensor and the front bumper are not detected on the Dashboard Components page.                                   |
| RESOLUTION | Disconnect the ToF sensor from the front bumper and use the long cable that originally connected the front bumper to the HUT to connect the ToF sensor directly to that same HUT port. Then reboot. This bypasses a known multiplexer issue on some bumpers. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot appears to be booted and the screen is on, but I can't see it using <code>dtc fleet discover</code> .  |
| RESOLUTION | Your Duckiebot must be connected to the same network as the computer you are using to run the <code>dtc</code> commands. Check the <a href="#">networking section</a> of the book to see if your network is set up correctly. |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I am not sure whether my Duckiebot is properly initialized.  |
| RESOLUTION | As long as the <code>fleet discover</code> tool shows ready, your Duckiebot should be ready. You can also visit the dashboard to confirm that the Duckiebot is serving its status. Generally as long as you see the Duckiebot dashboard is up, your Duckiebot should be correctly initialized. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | I see a permissions error when trying to access the Duckiebot dashboard: <code>Directory '/data/config/permissions' cannot be written.</code>   |
| RESOLUTION | <ul style="list-style-type: none"> <li>Take the sd card from your robot (press in once to release the spring, then remove)</li> <li>Put it in your laptop using the adapter that came with the card</li> <li>Navigate to the root of the card in your terminal. Most OS have an option to right click on the drive when it appears on your desktop or in the sidebar and select a "open in terminal", "new terminal at folder", or similar</li> <li>Run <code>chmod -R 777 ./data/config/permissions</code></li> <li>Eject the drive, place back in the Duckiebot, and power back up</li> </ul> |

## FAQs: Operating your Duckiebot

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The power button on top does not shut off the Duckiebot.   |
| RESOLUTION | The power button needs to be held for three seconds and then released. If this still does not work, run <code>dts duckiebot update &lt;your_robot&gt;</code> and then use <code>dts duckiebot reboot &lt;your_robot&gt;</code> . You may also need to re-flash your HUT following the procedure described in <a href="#">Debug - Re-flash Microcontroller</a> if you have not already. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot has a very low battery charge and is stuck in a boot cycle.  |
| RESOLUTION | Unplug all cables from the HUT except port # that is used to charge the battery. Allow the battery to charge for at least 5 hours before plugging all cables back in their nominal positions. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | When using the keyboard control GUI, I can see the commands being sent to the Duckiebot (e.g., through the <b>Dashboard &gt; Mission Control</b> ), but the Duckiebot does not move. My <b>Dashboard &gt; Robot &gt; Components</b> page shows a red alert for the HUT. |
| RESOLUTION | If you have a HUT v3.1 you will stumble on this problem the first time you try to move your Duckiebot. Re-flash your HUT following the procedure described in <a href="#">Debug - Re-flash Microcontroller</a> .  |

## Troubleshooting

### SYMPTOM

I have reflashed the HUT but the joystick commands still do not work or the Duckiebot operates in a jerky manner. Additionally, the ToF sensor and front bumper are not detected on the dashboard Components page. I may also be having issues with the screen.

### RESOLUTION

Disconnect the ToF sensor from the front bumper and use the long cable that originally connected the front bumper to the HUT to connect the ToF sensor directly to that same HUT port. Then reboot. This bypasses a known multiplexer issue on some bumpers that can cause other HUT misbehaviors.

## Troubleshooting

### SYMPTOM

I'm still having a software issue that the Duckietown team has pushed a new fix for according to StackOverflow.

### RESOLUTION

You can pull the latest images to your Duckiebot by running `dts duckiebot update <duckiebot_name>`. This is always the correct way to reset your Duckiebot's containers. You will never need to reflash the SD card to get updates.

## Troubleshooting

### SYMPTOM

A lot of the hardware components on the [Robot Dashboard / Components](#) tab are not found on DB21-series Duckiebots. And their connector buses are all I2C (see the bottom line of each component card).

### RESOLUTION

There could be 3 reasons typically:

- Please make sure that both rows of GPIO pins on the Jetson are connected accordingly to the connection slots on the HUT.
- A broken component along the I2C bus could lead to this problem. You could perform the following: turn off the Duckiebot, unplug one element from the HUT at a time and boot, and repeat this for other components. If most missing components appear connected at one test, it is likely the unplugged component has a hardware failure. Please contact the Duckietown team for replacement. Please also try to record videos of these tests.
- If all individual component unplugging tests were performed, and the components along the I2C chain are still missing all the time. There might be a Jetson I2C issue. Please try `sudo i2cdetect -r -y 1` after SSH'ed to you robot, to see if that returns a table of I2C addresses identified. Please report to Duckietown Team with the test videos of the previous step and the outcome of running the `i2cdetect` command.

## Setup - Laptop

### What you will need

- A laptop or machine running Ubuntu 20.04 or 22.04
- Alternatively: A laptop or machine running MacOS

### What you will get

- Your laptop configured with Duckietown software and

ready to sign into your accounts

In this section, we will install all the software that you need to run the Duckietown development environment on your laptop. This includes Git, Docker, and the Duckietown Shell.

#### **Note**

If you are unfamiliar with Git or Docker, we strongly recommend reading the following reference pages to gain a working understanding of these tools.

1. [The Duckietown Intro to Git](#)
2. [The Duckietown Intro to Docker](#)

Check out the laptop requirements below to make sure you are ready to proceed, then continue on to the next page to begin setting up your development environment.

If you need help installing Ubuntu, see [Step 0: System Installation](#) below.

## Minimal Laptop Requirements

Duckietown officially supports Ubuntu 20.04 and Ubuntu 22.04.

Please be aware that using an operating system that is not officially supported reduces the ability of the Duckietown team and the whole community to provide you with technical support when needed.

### Officially Supported

|              |  |
|--------------|--|
| Ubuntu 22.04 | This is the recommended operating system for using Duckietown. We strongly suggest using it.         |
| Ubuntu 20.04 | This older version of the Ubuntu operating system will currently work with the Duckietown ecosystem. |

### Unofficially Supported

|                           |   |
|---------------------------|---|
| Other Linux distributions | Having Ubuntu is not an enforced requirement, and everything should in theory work on any Linux operating system.   |
| MacOSX                    | We provide instructions for setting up on MacOSX, but you may run into bugs as you venture further into development.  |
| Ubuntu installed via VM   | This is possible, though network configuration is required. See the section below for more details.   |
| Windows (WSL)             | An experimental setup is in development to run the <code>dts</code> through the Windows Subsystem for Linux (WSL). Select the <code>Windows (Beta)</code> tab to test this. |

### Not Supported

|                  |   |
|------------------|---|
| Windows (Native) | Currently, it is not possible to run the Duckietown development tools on Windows. See below for available VM options to run Ubuntu on your Windows machine. |
|------------------|---|

## 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 necessary 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](#).

## Step 0: System Installation

[Ubuntu](#)   [MacOSX](#)   [Windows \(Beta\)](#)

Install the Ubuntu operating system. Ubuntu 22.04 is strongly suggested.

Follow the [official documentation](#) for instructions.

## Step 1: Dependency Installation

Select the tab for your operating system below, and follow the instructions to begin installing the Duckietown software dependencies.

[Ubuntu](#)   [MacOSX](#)   [Windows \(Beta\)](#)

### 1) Install dependencies

The basic development tools that you will need are `pip3`, `git`, `git-lfs`, `curl`, and `wget`. Install these by running the following commands in the shell:

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

If you are running Ubuntu on a virtual machine, install the package `open-vm-tools` in addition to the normal Ubuntu dependencies:

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

### Checkpoint

Before continuing, run the following test command

Test

```
pip3 --version
```

Expected Result

This command should output a version number for the `pip3` package.

### Tip

Never skip a checkpoint!

If you continue past a test that did not work, you will have further software issues down the line, and they will be more complex to fix. Instead, if you do not get the expected outcome at any checkpoint:

- First check for any troubleshooting sections on the page that might match the problem.



- If you run into any issues that can't be solved using the troubleshooting sections, join the Duckietown community on StackOverflow and Slack following the instructions below and search for previous solutions.

You can join the [Duckietown community on Slack at this link](#). There you can request an invitation to the Duckietown Stack Overflow team.

## Step 2: Docker Installation

Duckietown uses [DockerHub](#) to distribute the containerized version of its software modules, and most Duckietown procedures entail some `docker` operations behind the scenes.

If you are unfamiliar with Docker, we strongly recommend reading the following reference page to gain a working understanding of this tool: [The Duckietown Intro to Docker](#)

[Ubuntu](#)   [MacOSX](#)   [Windows \(Beta\)](#)

### 1) Install Docker

Install Docker by first ensuring that you don't have older versions of Docker on your system

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

Then set up the apt repository containing Docker

```
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg
```

Add the official GPG key

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
```

And set up the repository with

```
echo \
"deb [arch="$(dpkg --print-architecture)" signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Finally, update again and install Docker Engine and Docker Compose

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
sudo apt-get install docker-compose
```

### 2) Set up Docker

Start by adding the user "docker" to your user group, then log out and back in

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

#### **i** Attention

You need to *log out and back in* for this group change to take effect.

---

Checkpoint

Now make sure that Docker was correctly installed by running the following tests

|                 |  |
|-----------------|--|
| Test            | <pre>docker --version docker buildx version</pre>  |
| Expected Result | Make sure the Docker version is <b>v1.4.0+</b> and <b>buildx</b> version <b>v.0.8.0+</b> |

|                 |   |
|-----------------|---|
| Test            | Start the <b>hello-world</b> image with                 |
|                 | <pre>docker run hello-world</pre>                       |
| Expected Result | You should see a message like <b>Hello from Docker!</b> |

## Step 3: Duckietown Shell Installation

[Ubuntu](#)   [MacOSX](#)

The Duckietown Shell is a [command-line interface \(CLI\) program](#) that provides all of the necessary Duckietown operations, such as

- Updating a Duckiebot
- Driving a Duckiebot with a virtual keyboard
- Viewing the camera stream of a Duckiebot from a graphical app
- Using our learning experiences
- (and more!)

### 1) Install the Duckietown Shell (**dts**)

Install the Duckietown Shell using the following command,

```
pip3 install --no-cache-dir --user --upgrade duckietown-shell
```

### 2) Source **dts**

Make sure your system can find local binaries by adding the following to your **.bashrc** file.

#### **Attention**

If you are using **zsh**, replace the **.bashrc** in the commands below with **.zshrc** instead.

```
export PATH=~/local/bin:${PATH}
```

Then source the updates to your current shell or restart your shell.

```
source ~/.bashrc
```

### Checkpoint

To confirm that **dts** was installed successfully, run the following test

|      |                      |
|------|----------------------|
| Test | <pre>which dts</pre> |
|------|----------------------|

Expected Result

This should output a path ending in `dt.s`.

## Windows (Beta)

---

# Setup - Accounts

What you will need

- You have completed all of the steps in the previous [Setup - Laptop](#) section.

What you will get

- Your laptop configured with the Duckietown development environment and ready to run Duckietown robots and learning experiences.

Now that you have the required software installed on your operating system, we will set up the developer accounts you need to use each of the tools. Once you are logged in and have tested your environment, you will be ready to develop with Duckietown!

## Step 0: GitHub Account Setup

Duckietown uses [GitHub](#) to distribute its open-source code and engage with collaborators and end-users. If you are unfamiliar with Git, we strongly recommend reading the following reference page to gain a working understanding of this tool: [The Duckietown Intro to Git](#).

If you do not already have a GitHub account, you can sign up for one at [this link](#).

In addition to the Github account you'll need to create an SSH key, to seamlessly access Github through terminal without having to enter the password each time. Follow the official Github documentation to create a SSH Key [Link](#).

Once you have a GitHub username and SSK key associated with your account, you can continue on to the next page to set up your Duckietown account.

## Step 1: Duckietown Account Setup

### 1) Configure the Duckietown Shell

The first thing you need to do with `dt.s` is to set the Duckietown software distribution you want to work with. For this version of the book, we use `daffy`. Set the shell to use a profile on the `daffy` distribution by running the command

```
dt.s profile switch daffy
```

### 2) Get a Duckietown Token

Now your Duckietown Shell needs a Duckietown token. The Duckietown Token allows you to authenticate yourself and your robots against the Duckietown network.

You can make a Duckietown account for free from the Duckietown Hub. [Make an account here](#).

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

```
dt1-7vEuJsaxeXXXX-  
43dzqWFnWd8KBa1yev1g3UKnzVxZkTbfSJnxzuJjWaANeMf4y6XSXBWtpJ7vWXXXX
```

To find your token, first [log in](#) to your account, then open [the profile page](#) in your browser:

Copy your token to use in the next step.

### 3) Set your token in the Duckietown Shell

You can tell the Duckietown Shell who you are by running the command below and following the instructions presented in the shell

```
dts tok set
```

#### Checkpoint

Run the following tests to check your setup:

|                 |   |
|-----------------|---|
| Test            | If the Duckietown Shell was installed correctly, then you can run a command like this |
|                 | <pre>dts version</pre>  |
| Expected Result | Verify that the version of the commands is set to <b>daffy</b> .                      |

|                 |  |
|-----------------|--|
| Test            | Check if your token was successfully by running    |
|                 | <pre>dts tok status</pre>                          |
| Expected Result | This should output a message like the following,   |
|                 | <pre>dts : Correctly identified as uid = ***</pre> |

If you have encountered issues or something is not behaving as expected, please stop here, it is a good time to ask for help on Stack Overflow.

You can join the [Duckietown community on Slack at this link](#).

There you can request an invitation to the Duckietown Stack Overflow, by following [these instructions](#).

## Step 2: Docker Account Setup

Duckietown leverages containerization to ensure software portability and reproducibility. Most procedures entail the use of **docker** operations behind the scene. That is why we need to set up the logins for docker within **dts**.

We use [DockerHub](#) to distribute the containerized version of its software modules, and most Duckietown procedures entail some **docker** operations behind the scenes.

If you are unfamiliar with Docker, we strongly recommend reading the following reference page to gain a working understanding of this tool: [The Duckietown Intro to Docker](#)

### 1) Create a DockerHub account

If you do not have one already, you can sign up for a DockerHub account at [this link](#).

#### Make an access token

Follow the instructions on [this page](#) to create a new personal access token on DockerHub.

### 2) Log in to Docker

Once you have an account on DockerHub and an access token, you can test them by logging in using the command,

```
docker login -u DOCKER_USERNAME
```

where `DOCKER_USERNAME` is the username you chose when signing up on DockerHub. You will then be prompted for your password, paste the access token we created earlier and press `Enter`.

### 3) Configure shell

We are now going to provide the same username and access token to the shell in order to automate most of the back-end operations involving docker.

#### Attention

- These credentials are **only stored locally**;
- **Never use your account password** instead of an access token;

You can pass your DockerHub credentials to the Duckietown Shell by running the following command,

```
dts config docker credentials set \  
--username DOCKERHUB_USERNAME \  
--password DOCKERHUB_ACCESS_TOKEN
```

#### For developers

With an extra **positional** argument, one could specify a custom docker registry server other than `docker.io`. Check `dts config docker set --help` for more details.

### Checkpoint

Before we move on, let us make sure you have set our credentials correctly.

#### Tip

Never skip a checkpoint!  
If you have trouble with any of these commands, see the FAQs section below.

Test

If your Docker login was successful, you should be able to run

```
dts config docker credentials info
```

Expected Result

You should see an output similar to the following,

```
Docker credentials:  
. registry:  docker.io  
. username:  DOCKERHUB_USERNAME  
. secret:   DOCKERHUB_ACCESS_TOKEN
```

FAQs

If you continue past a test that did not work, you will have further software issues down the line, and they will be more complex to fix. Instead, if you do not get the expected outcome at any checkpoint:

- First check the troubleshooting guide below.
- If you run into any issues that can't be solved using the troubleshooting section, join the Duckietown community on StackOverflow and Slack following the instructions below and search for previous solutions.

You can join the [Duckietown community on Slack at this link](#).

There you can request an invitation to the [Duckietown Stack Overflow](#), in particular, following [these instructions](#).

| Troubleshooting |  |
|-----------------|--|
| SYMPTOM         | I mistakenly set a wrong/unwanted username or password. How can I update the credentials?                                |
| RESOLUTION      | Just run the command again with the correct credentials. Only the latest inputs are stored for the same docker registry. |

| Troubleshooting |  |
|-----------------|--|
| SYMPTOM         | I would like to remove my stored docker credentials. How could I achieve that?   |
| RESOLUTION      | Simply use a text editor to remove the section <code>docker-credentials</code> in <code>~/.dt-shell/config.yaml</code> file. |

## Setup - Duckiebot SD Card

Here we will learn how to initialize an SD card with the Operating System that will run on the Duckiebot on-board computer. This procedure is called **flashing**, or **burning** of the SD card.

|                    |   |
|--------------------|---|
| What you will need | <ul style="list-style-type: none"><li>• An SD card of size at least 32 GB</li><li>• At least 20 GB of free space on the computer</li><li>• An internet connection</li><li>• SD card reader</li><li>• Duckietown Shell, as configured in <a href="#">Step 3: Duckietown Shell Installation</a>.</li><li>• Docker, as configured in <a href="#">Step 2: Docker Installation</a>.</li><li>• Duckietown Token, as configured in <a href="#">Setup - Accounts</a>.</li><li>• 30 minutes on average (depends on internet connection and SD card adapter used)</li></ul> |
| What you will get  | A flashed Duckiebot SD card, ready to be used to give life to your Duckiebot.   |

### **i** For non-Unix-like systems

Though the suggested operating system for this operation is Ubuntu, 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 in that file instead. You can later transfer it 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.

## Step 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:

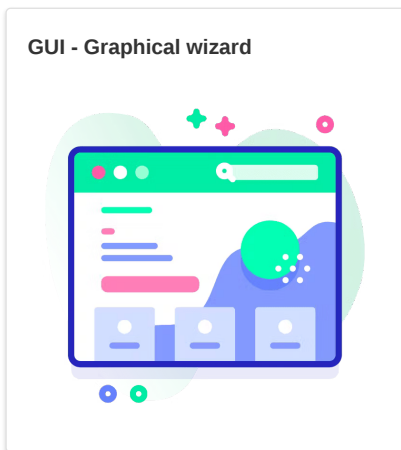
- it is lowercase
- it starts with a letter
- it contains only letters, numbers, and underscores

## Step 2) Burn the SD card

There are two interfaces to the process of burning a Duckiebot SD card:

- GUI: A graphical wizard-like interface;
- CLI: A terminal-based command line interface (for those who crave the terminal experience);

Choose the interface that best fits your preferences and skip the other.



## Burn the SD card - GUI

### Step-by-Step Instructions

#### **Attention**

Please note: You will need to specify the model of your Duckiebot when initializing the SD card.

- If you have a Duckiebot with a **2GB** Jetson Nano - the model is **DB21M**
- If you have a Duckiebot with a **4GB** Jetson Nano - the model is **DB21J**
- If you are not using a Jetson Nano, the model is the model of your Duckiebot (ex. DB19 or DBR4)

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 --gui
```

A new window similar to the one shown below will pop up,

Initialize a new SD card  
Let's initialize a new Duckietown robot!

---

**Name \***  
A (host)name for your new robot

**Type**

**Configuration**  
What model of a Duckiebot are we initializing today?

**SD card size**  
 GB

**Wireless Networks**  
List all WiFi networks that you want your robot to connect to

**Duckietown Terms and Conditions (read) \***

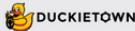
**Duckietown Software License (read) \***

**Duckietown Privacy Policy (read) \***

**License For Customer Use of NVIDIA Software (read) \***

▶ **Advanced options**

\* Required field



Fill in the fields according to the Duckiebot you are initialization, the SD card you are using, and the WiFi network details.

**Note**

If you need to configure **EAP (Extensible Authentication Protocol) protected networks**, use the CLI version of the SD card flashing procedure described in [Burn the SD card - CLI](#).

**Warning**

Given the danger of choosing a wrong device to flash (which can result in data loss and corruption of the Operating System), it is necessary to specify the size of the SD card you are using, so that the program will only show you options that are compatible with your input.

Once you completed the form, click on the **Confirm** button to return to the terminal and confirm the device to be flashed. When prompted, type in or copy-paste the device name from the list and press `Enter`.

At this point, the SD card is being flashed. On successful end of the procedure, the drives will be automatically ejected and you will be instructed to remove the SD card from the SD card reader and insert it into the Duckiebot's SD card slot.

If you experience any issues while flashing the SD card, make sure you check the [Troubleshoot - SD card flashing](#) section before asking for help on Stack Overflow.

**Troubleshoot - SD card flashing**

| <b>Troubleshooting</b> |   |
|------------------------|---|
| SYMPTOM                | <ul style="list-style-type: none"> <li>• The SD card doesn't seem to be written.</li> </ul> |



RESOLUTION

- The flashing process seemed too fast, there is no data on my SD card.
- Check if your SD card has a write protection switch.
- Make sure you inputted the correct drive name during the flashing procedure.

### Troubleshooting

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.

### Troubleshooting

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 .

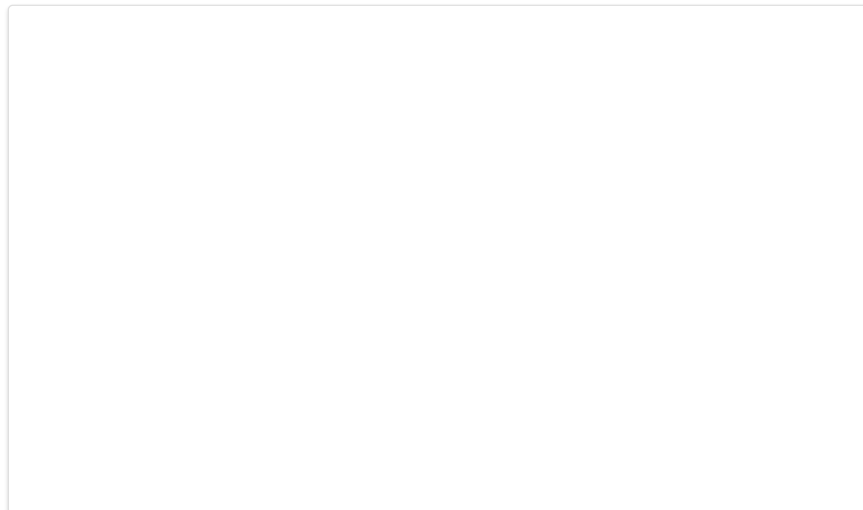
## Burn the SD card - CLI

### ⓘ Attention

Please note: You will need to specify the model of your Duckiebot when initializing the SD card.

- If you have a Duckiebot with a **2GB** Jetson Nano - the model is **DB21M**
- If you have a Duckiebot with a **4GB** Jetson Nano - the model is **DB21J**
- If you are not using a Jetson Nano, the model is the model of your Duckiebot (ex. DB19 or DBR4)

### Video Tutorial



### Step-by-Step Instructions

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. `DB21J`, `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 are `duckie` and `quackquack`, respectively.

If you plan on connecting with the Duckiebot over different networks (e.g., at home and in class), you can list all your networks like in the following example,

```
dts init_sd_card ... --wifi
duckietown:quackquack,myhomenetwork:myhomepassword,myuninetwork:myunipasswo
rd
```

#### **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): "ssid"
- PSK (Pre-shared key) protected networks: "ssid:psk"
- EAP (Extensible Authentication Protocol) protected networks:
"ssid:username:password"
```

Make sure to set your country correctly with the `--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 `init_sd_card` exist. For a full list of the options, run:

```
dts init_sd_card --help
```

After you run the `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 `Enter` .

At this point, the SD card is being flashed. On successful end of the procedure, the drives will be automatically ejected and you will be instructed to remove the SD card from the SD card reader and insert it into the Duckiebot's SD card slot.

If you experience any issues while flashing the SD card, make sure you check the [Troubleshoot - SD card flashing](#) section before asking for help on Stack Overflow.

### Troubleshoot - SD card flashing

| Troubleshooting |   |
|-----------------|---|
| SYMPTOM         | <ul style="list-style-type: none"><li>• The SD card doesn't seem to be written.</li><li>• The flashing process seemed too fast, there is no data on my SD card.</li></ul>                     |
| RESOLUTION      | <ul style="list-style-type: none"><li>• Check if your SD card has a write protection switch.</li><li>• Make sure you inputted the correct drive name during the flashing procedure.</li></ul> |

| Troubleshooting |   |
|-----------------|---|
| SYMPTOM         | The flashing procedure fails with a <code>Bad archive</code> error.   |
| RESOLUTION      | This happens when the downloaded compressed disk image file appears corrupted. You can force the re-download by adding the option <code>--no-cache</code> to the <code>init_sd_card</code> command. |

| Troubleshooting |  |
|-----------------|--|
| SYMPTOM         | The verification process fails with error <code>Please set up a token using "dts tok set"</code> . |
| RESOLUTION      | Make sure you completed the Duckietown token setup procedure .                                     |

## Setup - Booting the Duckiebot

|                    |  |
|--------------------|--|
| What you will need | <ul style="list-style-type: none"><li>• A flashed Duckiebot SD card</li><li>• A Duckiebot of the same model chosen during the SD card flashing procedure</li><li>• A fully charged battery</li></ul> |
| What you will get  | A Duckiebot successfully booted up and connected to the WiFi network.  |

You are now ready to boot up your Duckiebot!

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. While the video shows the procedure being performed on a DB21M robot, this procedure is the same on all Duckietown robot models.



### ⚠ Warning

Be sure your Duckiebattery was fully charged as shown in the assembly steps before attempting to boot.

The external power supply might not be able to provide sufficient current if the battery is low, causing the on-board computer to reboot. Should that happen during the first boot, you will likely have to burn the SD card again.

## Monitoring the First Boot

Make sure your desktop or laptop computer is connected to the same WiFi network the Duckiebot was instructed to connect to.

Then open a terminal and run the following command,

```
dts fleet discover
```

The command above will show a list of all the Duckiebots reachable on your local network. For each Duckiebot, the tool will also show the model that was used to flash the SD card, the hostname of your robot, and a status indicator.

Leave this tool open, it will refresh automatically every second, so there is no need to manually restart it.

Within a few minutes of powering up the robot with the SD card in, your Duckiebot will appear in the list with status **Booting**. If it does not appear within 5 minutes, check out the Troubleshooting guide at the end of this page.

```
dts fleet discover
NOTE: Only devices flashed using duckietown-shell-commands v4.1.0+ are supported.
-----|-----|-----|-----|-----
Type    | Model  | Status | Hostname
-----|-----|-----|-----|-----
ashby   | duckiebot | DB21J | Ready | ashby.local
kizzy   | duckiebot | DB21J | Ready | kizzy.local
sissix  | duckiebot | DB21J | Ready | sissix.local
```

Fig. 41 Output of 'dts fleet discover'

## ! Attention

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

## Software and Hardware updates

Once the status of your Duckiebot is **Ready**, you are ready to update your Duckiebot. The software update is required now, while the hardware update is conditional and also can be carried out later.

### Software update

Please update your Duckiebot's software stack with the latest by following: [Debug - Duckiebot Update](#).

### Hut Microcontroller update

The reader does not need to perform this right now. This is mentioned here so that the reader is aware of the existence and a potential need of this procedure. The Hut connects to various electronics beyond the Duckiebot computer, e.g. motors, LEDs. If you run into problems with the operations in the **Duckiebot Operations** section later, and it is related to the hardware on the HUT, you are then suggested to follow the [Debug - Re-flash Microcontroller](#) page to update the HUT firmware.

## Confirming the First Boot

Once the Duckiebot is **Ready** and **Updated** (software), you are ready to access your Duckiebot's Dashboard.

Open your browser and visit the URL <http://HOSTNAME.local/>. You will see a page similar to the following,

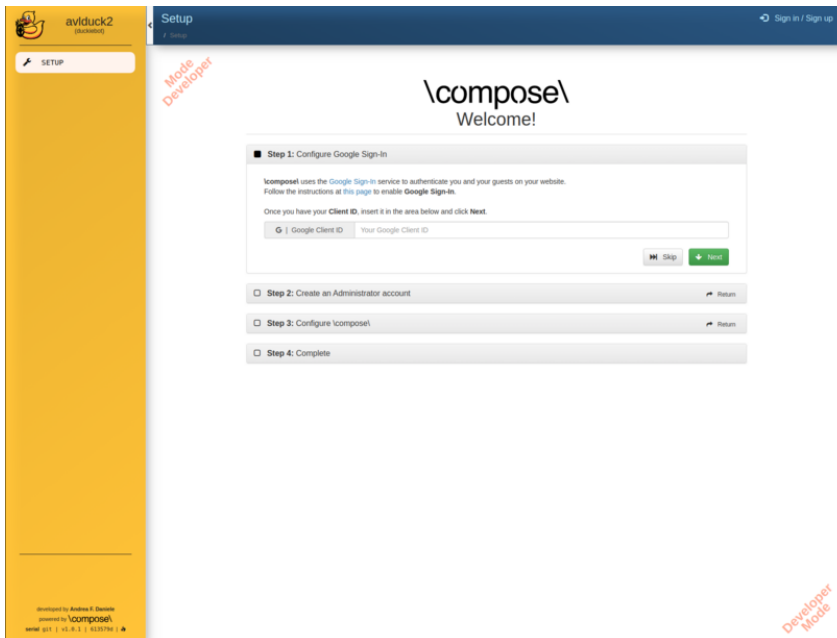


Fig. 42 Duckiebot's dashboard first setup page

This is the dashboard of your Duckiebot. The Dashboard is built using a framework called `\compose`. You configure it in [Setup - Duckiebot Dashboard](#).

If you can't access the dashboard, check out the Troubleshooting guide at the end of this page.

## Powering off 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 turn off your Duckiebot, use the command,

```
dts duckiebot shutdown HOSTNAME
```

The shutdown procedure can take up to 20 seconds.

To reboot your Duckiebot, use the command,

```
dts duckiebot reboot HOSTNAME
```

You will learn more about how to handle your Duckiebot in [Handling - Duckiebot DB21](#).

## Troubleshoot - First Boot

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | I pressed the power button on top but nothing happened.   |
| RESOLUTION | Power on your Duckiebot using the button on the side of the Duckiebattery. The top button is only for powering off. You can also learn more about how to handle your Duckiebot in <a href="#">Handling - Duckiebot DB21</a> . |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | My Duckiebot does not appear to boot after pressing the power button on the battery. I don't see a green light on the HUT or the Jetson Nano.  |
| RESOLUTION | Refer back to <a href="#">Assembly - Duckiebot DB21J</a> , and check each of your cable connections. Confirm the start and end port of each power cable from the battery. The battery must be charged fully as shown in the first assembly step. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot is getting power but does not appear to be booting. The Wifi dongle is not blinking.           |
| RESOLUTION | Make sure you flashed the SD card following the instructions in <a href="#">Setup - Duckiebot SD Card</a> . |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | My Duckiebot is getting power but does not appear to be booting. The Wifi dongle is not blinking.  |
| RESOLUTION | Make sure that you correctly specified the model of your Duckiebot when initializing the SD card.<br><br>If you have a Duckiebot with a 2GB Jetson Nano - the model is DB21M |

If you have a Duckiebot with a 4GB Jetson Nano - the model is DB21J

If you are not using a Jetson Nano, the model is the model of your Duckiebot (ex. DB19 or DBR4)

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The Duckiebot screen does not turn on even though it shows up in <code>dtS fleet discover</code> and the dashboard is accessible. The ToF and front bumper are not detected on the dashboard Components page.  |
| RESOLUTION | Disconnect the ToF sensor from the front bumper and use the long cable that originally connected the front bumper to the HUT to connect the ToF sensor directly to that same HUT port. Then reboot. This bypasses a known multiplexer issue on some bumpers. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot appears to be booted and the screen is on, but I can't see it using <code>dtS fleet discover</code> .  |
| RESOLUTION | Your Duckiebot must be connected to the same network as the computer you are using to run the <code>dtS</code> commands. Check the <a href="#">networking section</a> of the book to see if your network is set up correctly. |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I am not sure whether my Duckiebot is properly initialized.  |
| RESOLUTION | As long as the <code>fleet discover</code> tool shows ready, your Duckiebot should be ready. You can also visit the dashboard to confirm that the Duckiebot is serving its status. Generally as long as you see the Duckiebot dashboard is up, your Duckiebot should be correctly initialized. |

## Setup - Duckiebot Dashboard

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

### 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 already running on your Duckiebot after the first boot. Follow the instructions in the next step to get started.

Visit the [official documentation page](#) if you would like further information about how \compose\ works.

### Setting up the Duckiebot dashboard

Video Tutorial

## Step-by-Step Instructions

You can find your duckietown dashboard at:

```
http://![YOUR_DUCKIEBOT_NAME].local/
```

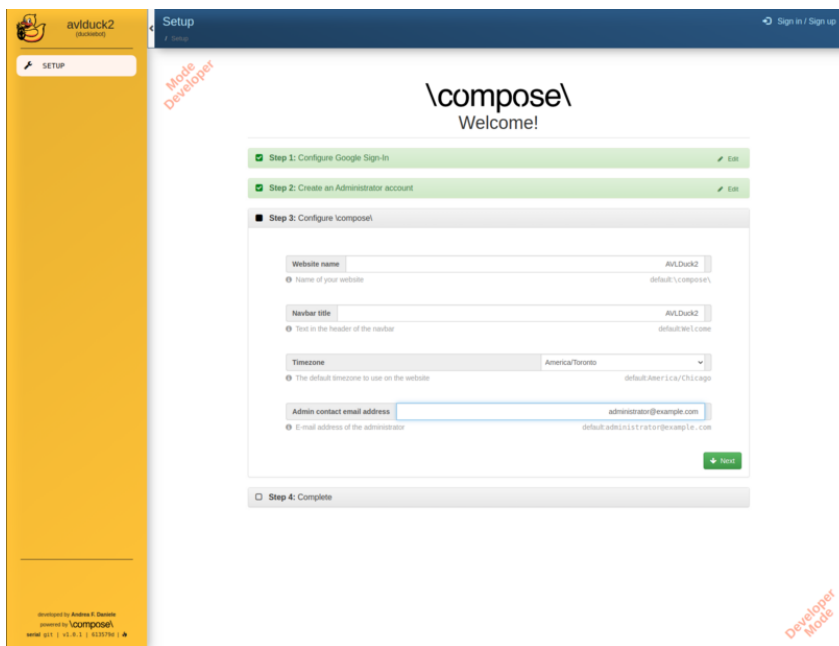
If the above address does not work, remove the `.local` part and just use

```
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.

You should be greeted by the dashboard shown here. Read the steps below before continuing through the setup page.



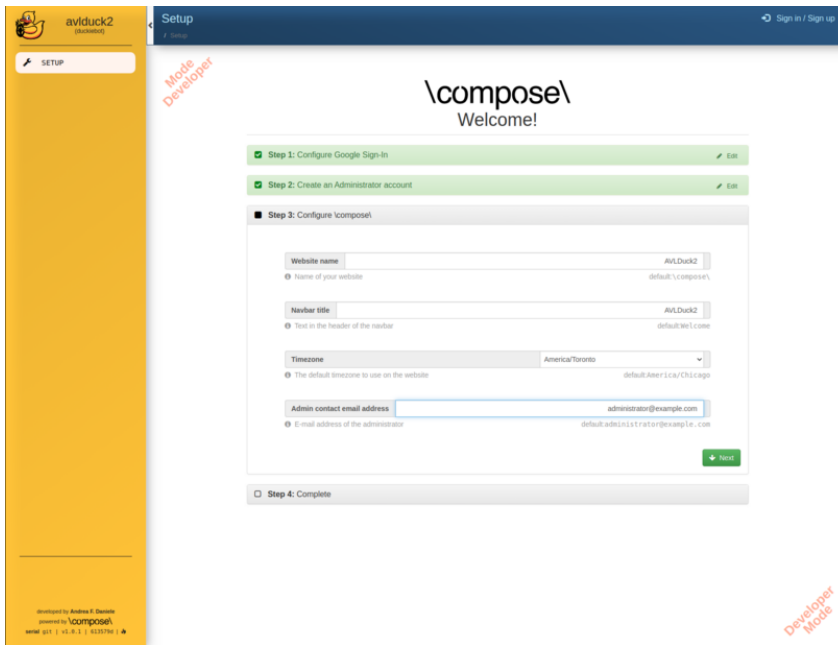
### Steps 1 and 2: Already done!

By default, `\compose\` uses Google Sign-In to authenticate the users. In Duckietown, we use authentication based on personal tokens.

You will notice that the first two steps in the dashboard already appear to be completed. Do not worry about configuring Google sign-in (Step 1) or creating an administrator account (Step 2) for now, a new administrator account will be automatically created the first time we log in using a Duckietown token later on.

### Step 3: Configure your dashboard





You can complete these fields as you please.

### Note

You can always update your choices through the **Settings** page after you finish the setup process.

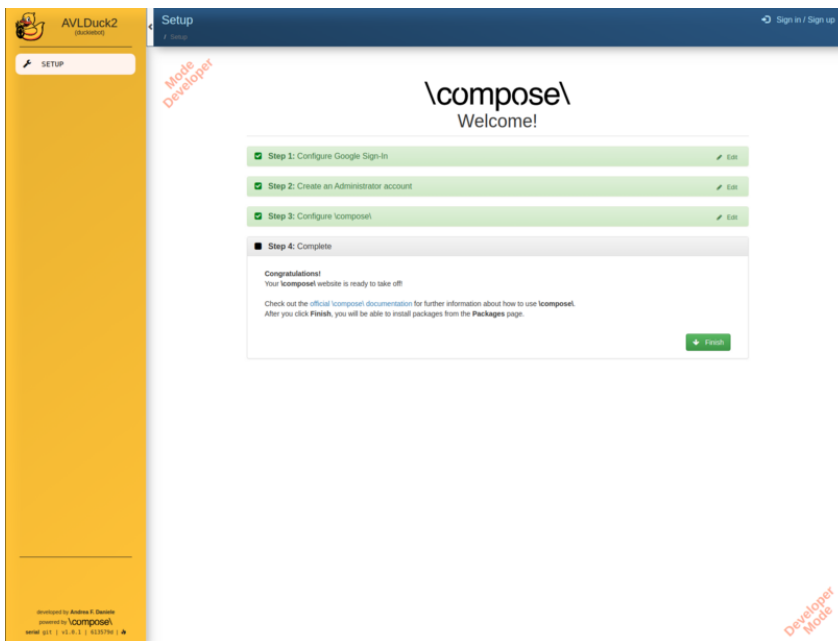
### Tip

If you are seeing an error due to permissions, take a look at the [Booting FAQs](#).

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

### Step 4: Complete the setup

The **Step 4: Complete** tab should now be open, as shown below.

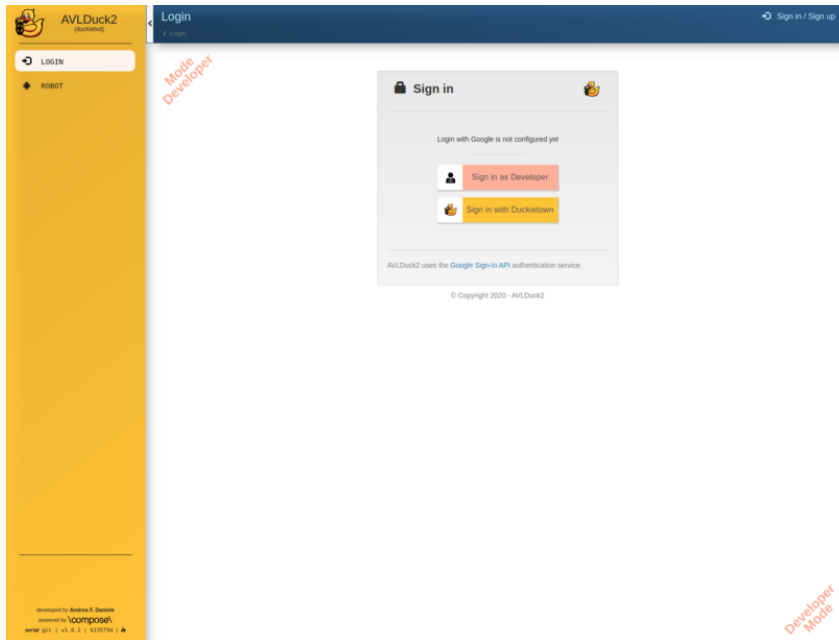


You can go ahead and press **Finish**.

## 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.



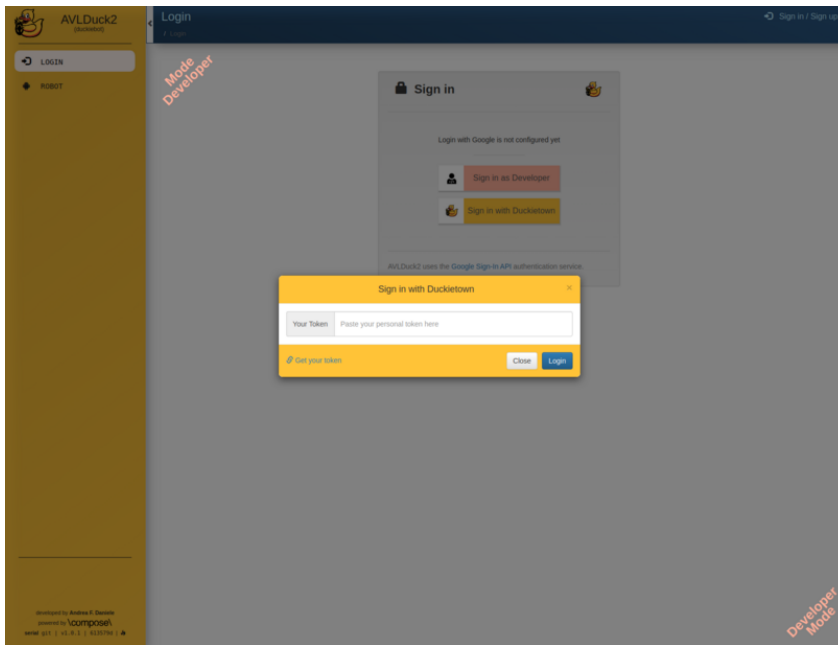
### **Note**

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.

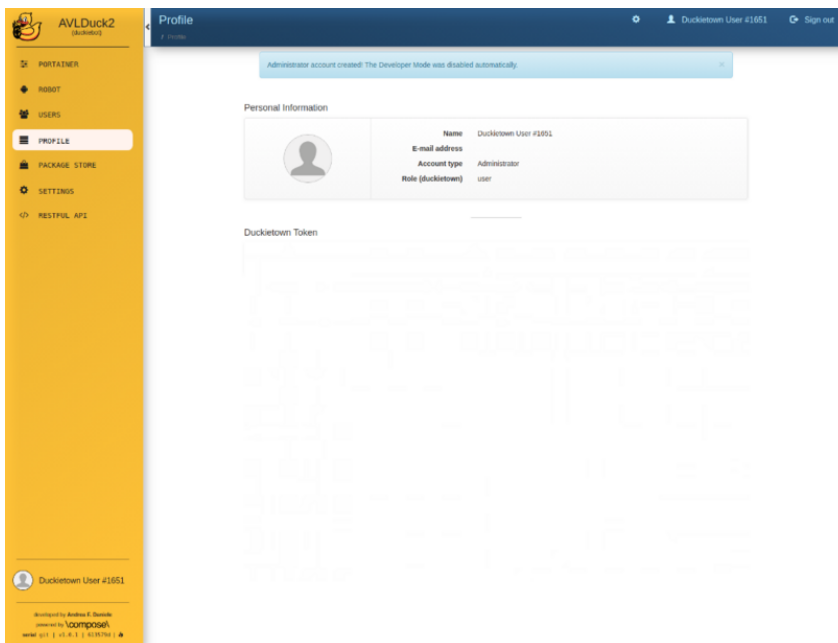
If you have not retrieved your personal Duckietown Token as described in [Setup - Accounts](#) yet, it is now time to do so. You should be able to retrieve your token by visiting the page here:

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

Once you have your personal Duckietown token, go ahead and click on the button **Sign in with Duckietown**.



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.

## Handling - Duckiebot DB21

What you will need

- An assembled **DB21** robot (e.g., **DB21M**, **DB21J**);
- An initialized **DB21** with **firmware version 1.2.2 or newer**. [Check your current firmware version](#) before proceeding.

What you will get

Knowledge on standard protocols to turn on, turn off, charge, and update the Duckiebattery software version on a **DB21** robots;

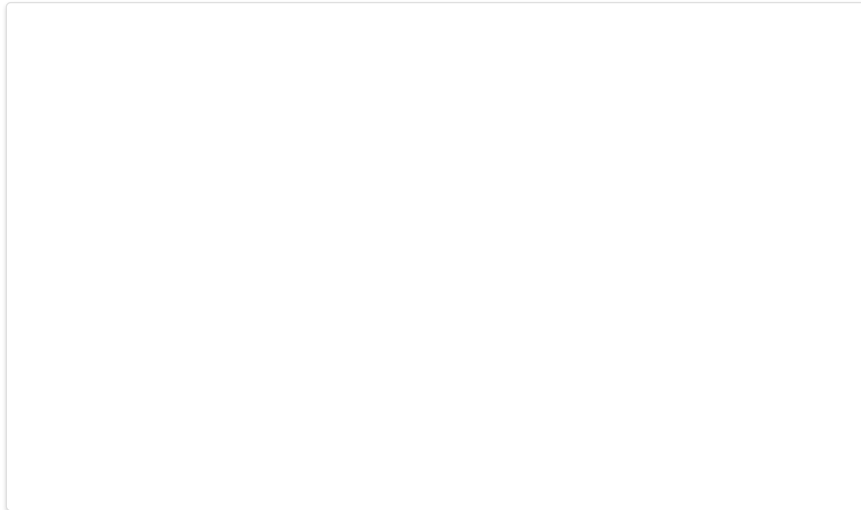
#### **Note**

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

#### **Note**

In this manual, we use **DB21** to refer to any version of the Duckiebot with prefix **DB21**, such as, **DB21J**, **DB21M**, etc.

## Duckiebot **DB21** handling tutorial video



## How to charge a **DB21**

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 **DB21** 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.

## How to power off a **DB21**

### ⚠ Warning

The proper shutdown protocol for a **DB21** requires having the Duckiebattery software version **2.0.0** or newer. 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.

There are three methods to power off a **DB21**:

1. Using the **top** button (**preferred**):

1. Press the **top** button (not the battery button) for 5 seconds and release;
2. What to expect:
  1. The top button will blink for 3 seconds;
  2. The Duckiebot front and back LEDs turn off;
  3. In about *10* seconds, the on-board computer and the fan will shut down;
3. **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. Using `dts`:

1. `dts duckiebot shutdown ![ROBOT_NAME]`
2. What to expect:
  1. In about *10* seconds, the on-board computer and the fan will shut down;
  2. If the charging cable is not attached, the front and back LEDs will also turn off;

3. Through the Duckiebot dashboard:

1. Open a browser
2. Navigate to `http://![ROBOT_NAME].local`
3. In the Top-Right corner, click on the **Power** options, and choose “**Shutdown**”. Then confirm the action.
4. What to expect:
  1. In about *10* seconds, the on-board computer and the fan will shut down;
  2. If the charging cable is not attached, the front and back LEDs will also turn off;

### ⚠ Warning

The following “hard” power shutdown should be only be used if the three methods above failed to shut down the Duckiebot, as it might lead to software corruption.

As a last resort, one could still perform a “hard” power shutdown of the **DB21**:

- `ssh duckie@![ROBOT_NAME].local sudo poweroff;`
- Unplugging the microUSB cable from the port marked as **5Vraspi** on the **HUT**;

## How to power on a **DB21**

To power on a **DB21** robot, press the button on the battery *once*.

The Duckiebot LEDs, as well as the on-board computer 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 above.

## How to SSH to the Duckiebot

Next, let us try and log in onto our robot using the SSH (Secure Shell) protocol. We can do so by running the command,

```
ssh duckie@HOSTNAME.local
```

The default password is `quackquack`.

## How to update a Duckiebattery

To update the software running on the microcontroller in the Duckiebattery, or just checking its current version, use the following instructions.

### **Important:**

1. Before the battery upgrade, please make sure the battery has at least 15% of charge.
2. Run all the 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.

1. Please update the `duckietown-shell` utility:
  1. `pip3 install --user --upgrade --no-cache-dir duckietown-shell`
  2. `dts update`
  3. `dts desktop update`
2. Update the Duckiebot:
  1. `dts duckiebot update ![ROBOT_NAME]`
3. Reboot the Duckiebot:
  1. `ssh duckie@[ROBOT_NAME].local sudo reboot`
  2. Wait until the Duckiebot reboots and the display shows information (especially about the battery).
  3. 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:
  1. `dts duckiebot battery upgrade ![ROBOT_NAME]`
    1. Note: When prompted to “double-click” on the battery button, make sure to *quickly* click twice the `battery` button.
    2. 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.
    3. Follow the instructions in the terminal.
  2. 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 'GENERIC_ERROR', please try flashing again with: dts duckiebot battery upgrade --force ![ROBOT_NAME]`
  3. 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
  1. If the battery indicates the charging states correctly, and shows the percentage number normally, proceed to `step 6`
  2. If the display shows “NoBT” (No battery detected), then **single** press the battery button, and run:
    1. `ssh duckie@[ROBOT_NAME].local sudo reboot`
    2. Wait for the reboot (as described in `step 3`)
    3. Then proceed to `step 6`
6. Verify current battery version:
  1. Method 1:
    1. `dts duckiebot battery check_firmware ![ROBOT_NAME]`
    2. Verify the battery version should be `2.0.2` or newer
  2. Method 2:
    1. Open a browser window
    2. Navigate to `http://![ROBOT_NAME].local/health/battery/info`

3. Verify the battery version should be 2.0.2 or newer

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The power button on top does not shut off the Duckiebot.   |
| RESOLUTION | The power button needs to be held for three seconds and then released. If this still does not work, run <code>dtc duckiebot update &lt;your_robot&gt;</code> and then use <code>dtc duckiebot reboot &lt;your_robot&gt;</code> . You may also need to re-flash your HUT following the procedure described in <a href="#">Debug - Re-flash Microcontroller</a> if you have not already. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | My Duckiebot has a very low battery charge and is stuck in a boot cycle.  |
| RESOLUTION | Unplug all cables from the HUT except port # that is used to charge the battery. Allow the battery to charge for at least 5 hours before plugging all cables back in their nominal positions. |

## How to update a HUT

Instructions on how to flash a Duckietown HUT board can be found [here](#).

### Note

Reflashing a HUT is rarely needed. A notable exception is for HUT version 3.15 which comes with DB21s. The HUT version can be read on the board itself.

## Operation - Use the Dashboard

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

### What is in the Dashboard?

The following video provides a brief tour of the most important features of the Duckietown Dashboard on your Duckiebot.

To see all the available dashboard components, you will need to first log in. Once logged into the dashboard, you will see a navigation panel down the left side of the page. The 7 available subpages are:

| PAGE NAME     | DESCRIPTION   |
|---------------|---|
| Portainer     | A nice GUI tool for seeing all containers running on a Duckiebot            |
| File Manager  | A file manager for managing the files on the robot                          |
| Robot         | A summary page for the robot status   |
| Profile       | Information for your duckietown account                                     |
| Package Store | A package store contain all available packages for your Duckiebot           |
| Users         | Advanced Feature: Allow multiple students to use one Duckiebot              |
| Settings      | Advanced Feature: Change configuration of your Duckiebot dashboard manually |
| Restful API   | Advanced Feature: Documentation to the RestAPI exposed by the Dashboard.    |

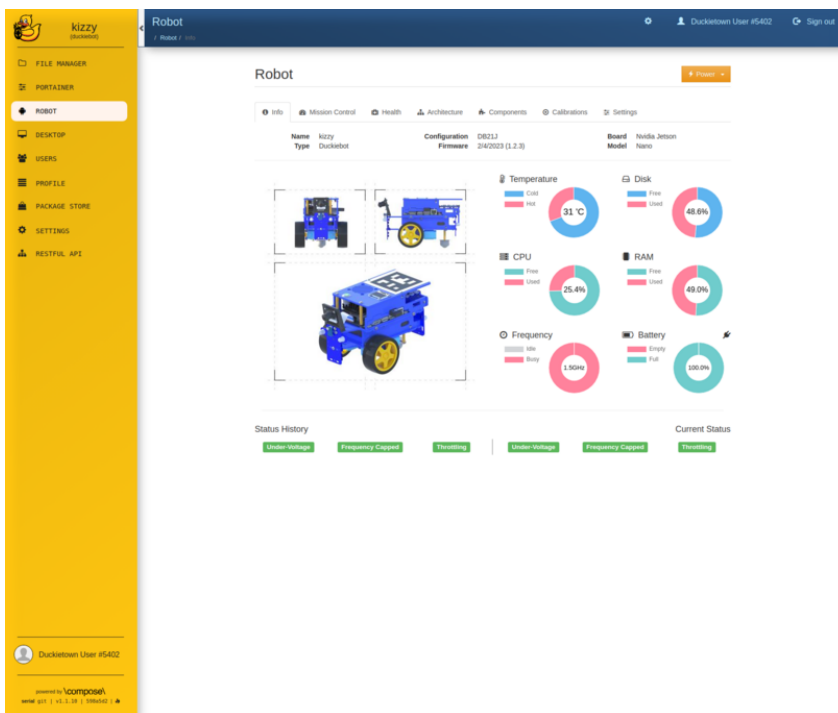
Let's dive into a few pages that you will need to get started.

## The Robot Page

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

### Robot - 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.



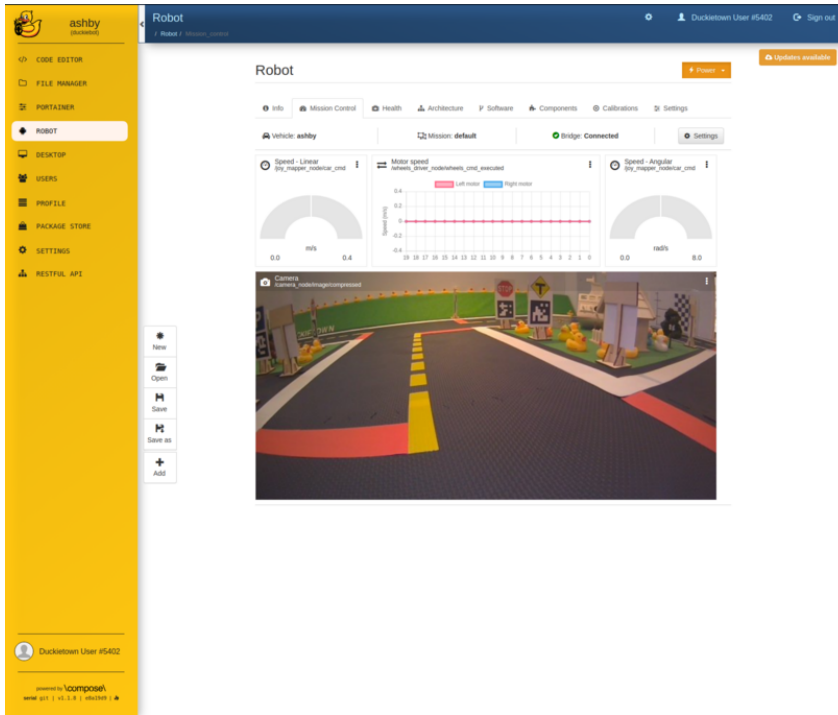
#### Note

From this page you can read the Duckiebot's firmware version, i.e., the version of the base image used during [initialization](#).

### Robot - Mission Control

This is the Mission Control tab.





In this tab you can see what the Duckiebot sees via the image stream, the 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.

**Tip**

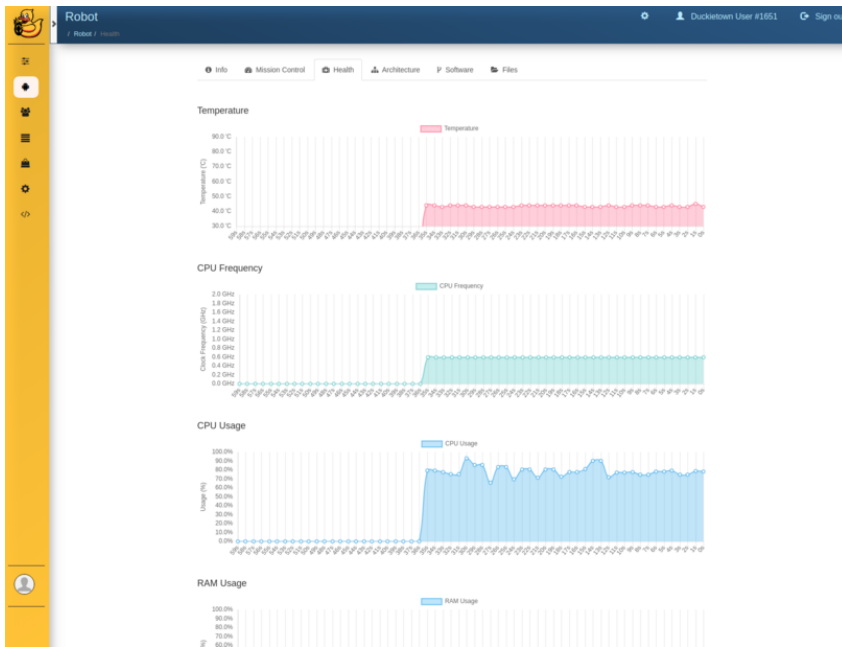
The page contains 4 blocks by default. Feel free to drag them around and rearrange them as you please. You can also use the menu button of each block to resize them.

**Troubleshooting**

|            |   |
|------------|---|
| SYMPTOM    | I do not see the camera image on the dashboard.   |
| RESOLUTION | This could be caused by a few issues. Make sure you are accessing the dashboard using <code>http://ROBOT_HOSTNAME.local/</code> instead of directly accessing the dashboard using robot IP address. Make sure the lens cap has been removed from your camera. If you still don't see an image, jump to the <a href="#">Operation - Make it See</a> page for more debugging options. |

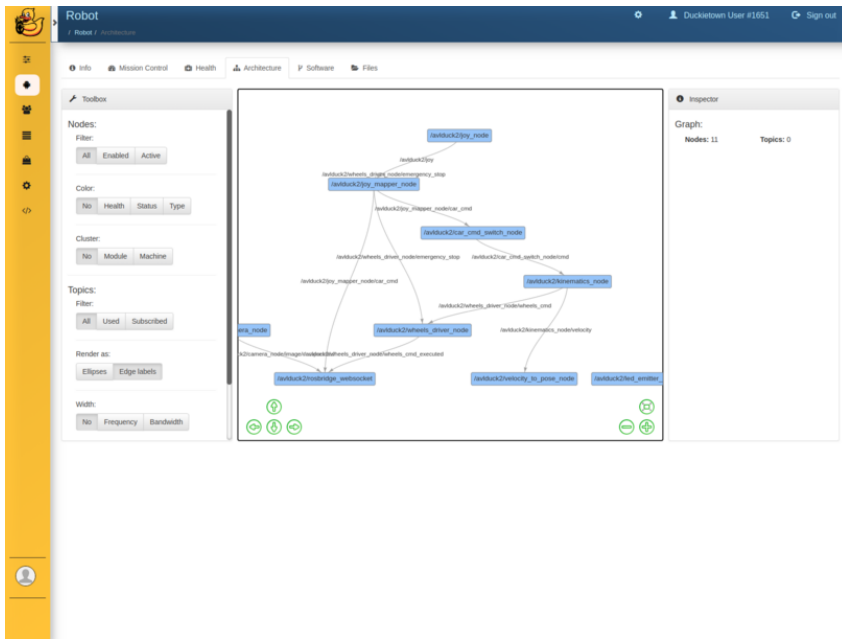
**The Health Page**

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 debugging tool to watch your code's resource usage.



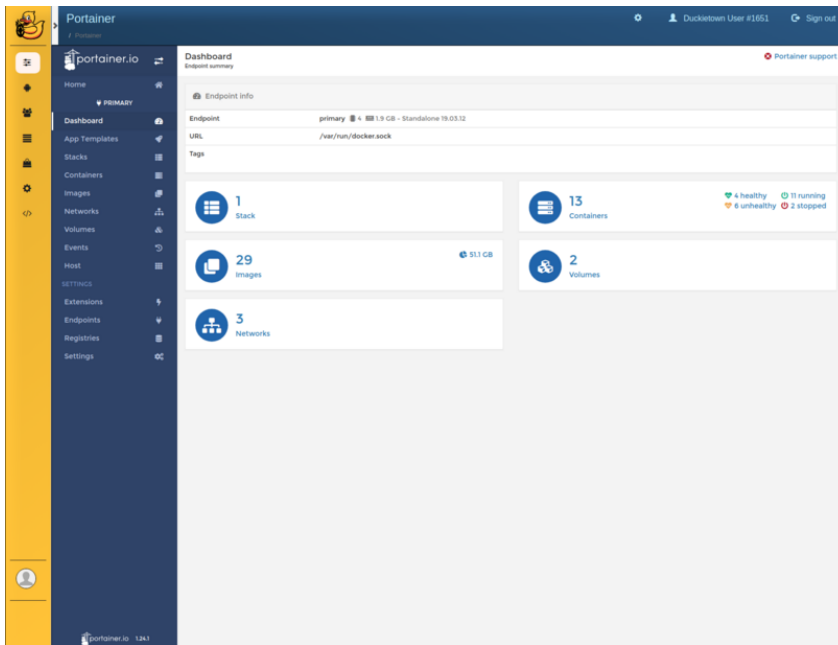
## The Architecture Page

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](#)



## The Portainer Page

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.

## Hardware Component Testing

### Note

This section is designed and tested for DB21-series Duckiebots and onwards.

This section shows how to use the Dashboard to test different individual hardware on a Duckiebot.

Assuming you have your Duckiebot assembled, the SD card prepared and the Software updated after booting. Before diving into the beautiful yet could-be-complicated world of robotics and coding, let's do a few simple individual hardware component tests to verify the sensors, actuators, computation and power units work.

The purpose of these tests is to help us be confident in the hardware, and avoid wasting time debugging software without knowing whether the hardware works.

In case you encounter any erroneous behaviors of any of your hardware components, or have any questions about the process, please check out a later sub-section on this page for [FAQs, reporting and getting help](#).

### Where to find the Hardware Tests

The tests could be accessed on the **Components** tab on the **ROBOT** page of the Dashboard.

In each component with the button **Test Hardware**, the test description and expectations would be shown in a **modal** (an on-page pop-up) when the button is clicked. Before running any test, it is recommended to watch the videos in the [next](#) section at least once to see how the robot would behave in each test.

 **Tip**

- You can always perform these tests.
- Later on, if you have problems receiving sensor readings, commanding the motion, or connecting to the robot, maybe you could run these tests again before debugging the code.

### Demos of the Hardware Tests

These videos would be what you typically need to do or would see, when performing the tests.

(You might want to manually set the video resolution to 1080P, instead of Auto.)

**Component**

**Demo Video**

---

Battery

---

Camera

---

Left motor

---

Right motor

---

Left encoder

---

Right encoder

---

Screen

---

**Component**

**Demo Video**

---

IMU

---

Power button

---

Wifi adapter

---

Front LEDs

---

Back LEDs

---

Time-of-Flight distance sensor

---

## FAQs, Reporting Problems & Getting Help

Problems could arise due to faulty hardware, misleading setup instructions, erroneous handling etc. Let's deal with them, patiently and efficiently. This part contains:

- some FAQs
- how to gather logs and enrich the context of an issue reported
- places and formats to post questions and feedback

## FAQs

Here are some common possible issues and resolutions you could already try. If the suggestions do not lead to a successful outcome, do not spend too much time trying it. Maybe we did not make it clear enough. Feel free to report to the team as mentioned in [this](#)

[part](#).

### Troubleshooting

SYMPTOM

When I click on the "Test Hardware" button:

- the button does not seem to react / is grayed-out; or
- the modal shows up, but there is no content in it.

RESOLUTION

This is likely because the `duckiebot-interface` container did not start or has not finished initialization successfully. You could go to the "Portainer" page of the Dashboard, and select **Containers** on the left nav-menu. Then, you could try these:

- First, look for the `duckiebot-interface` container, and make sure it exists. If not, try from your host computer: `docker duckiebot update ![ROBOT_NAME]`. After the command finishes, repeat this check in a few minutes.
- If it does exist, check the logs of the container ([Ref: how to check logs in portainer](#)). Ask

### Troubleshooting

SYMPTOM

Can I close the modal while the test is running?

RESOLUTION

Yes. It will be alive in the background. But you should avoid doing so. Please run only one test at a time, wait until you could check for the expected outcomes, and then close the modal once all done.

When running tests for sensors like camera or ToF, the data would be streamed. It is fine to close the modal after you could determine whether the test passed or failed. On the other hand, for tests with a progress bar, wait for the described time period. Only the LEDs might need a little longer time than written in the description (please check the next FAQ).

### Troubleshooting

SYMPTOM

My front and back LEDs seem stuck at a color when performing test. Is this normal?

RESOLUTION

The LEDs ideally should be changing brightness or color (or both) all the time during the test. However, it could occur that the frequent commands issued to the LEDs creates a temporary congestion and make the system busy. And the LEDs would appear to be neither changing color nor brightness. The situation should only last for less than 30 seconds.

- Please wait for the duration patiently to see if the test proceeds.
- If it is ever longer than that, please don't hesitate to contact our team and report the problem.

### Troubleshooting

SYMPTOM

Would it be harmful if the Duckiebot is turned off during some

|            |  |
|------------|--|
|            | test runs?   |
| RESOLUTION | No. These are very simple hardware tests. It is perfectly safe to kill any programs or power down the robot. |

## Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | Why, in the camera test, the image appears smaller than the sensor data size (see <code>start_gui_tools</code> in coming sections of this book)?                          |
| RESOLUTION | The camera stream in the Dashboard hardware test is resized, such that it fits better in the pop-up modal. The actual data size is always the same as what ROS indicates. |

## How to provide a rich context when reporting problems

You are of course welcome to ask us your questions directly anytime. In order to get back to you faster and debug efficiently, having richer context would help a lot! That is also why, in the modals you see for each hardware test, there are the buttons for:

- **Download logs of this ROS node:** by clicking this button, a text file will be downloaded from your Duckiebot to your personal computer, containing the logs of the relevant ROS node.
- **Download docker container logs:** by clicking this button, a new modal should pop up, allowing selection of the docker container(s), that is related to this test. The relevant container(s) is/are typically mentioned in the “Logs Gathering” section of the test modal.

### Tip

Here's a potential example of organizing information in one issue:

1. I am performing **hardware test via the Dashboard** for component: ...
2. I followed the instructions until this step / am waiting for this outcome: ...
3. It's not clear what to expect. / The expectation is ..., but I see ...
4. The logs for the ROS node: **file/pasted text**
5. The logs for the relevant containers: **file/pasted text**

Not only does the rich context help the team recognize the problem and provide solution faster, the community could also benefit, by knowing clearly if someone had an identical/close problem before. And you might be able to find useful answers just by searching our StackOverflow.

In the next sub-section, we recommend some tags and endpoints for communicating the issues to.

## Where to post questions & feedback

- **(Preferred)** [StackOverflow for Duckietown](#)
  - It is recommended to use these tags when posting on StackOverflow
    - `robot-setup`
    - `hwtest`, `hw tests`, `hardware-test`, `hardware-tests` (and other similar)
    - `hwtest-Name of the component`
- You could also ask on the Duckietown Slack, on the `help-robot-setup` channel. But it is more preferable to post technical issues on our StackOverflow, and any other feedback on Slack.
  - Here are some kinds of “*any other feedback*”:
    - The tests ran but the description clarity could improve **here and here**.
    - The test design would be better if the content is **this**.



# Operation - Make it Move

This section describes how to make your Duckiebot move.

## Tip

**Troubleshooting** sections are provided at the end of each operation page - start there if you run into any issues.

## Keyboard control

The easiest way to move a Duckiebot is by using the `keyboard_control` command provided in the Duckietown shell. This video shows how to drive a Duckiebot using the keyboard, through the Duckietown Shell.

### Option 1: Use the GUI (For Linux Users)

If you are using Mac OSX, [see Option 2](#).

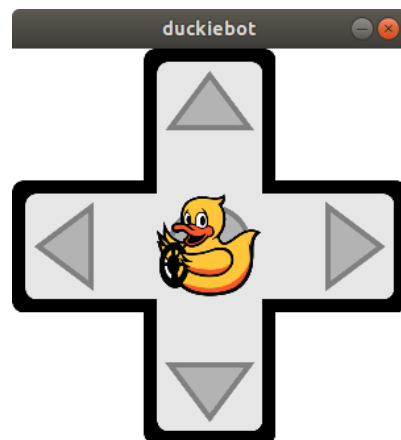
To move your Duckiebot using your computer's keyboard open a terminal and run:

```
dts duckiebot keyboard_control ![DUCKIEBOT_NAME]
```

## Attention

For all operation commands that use the Duckiebot's name - replace `![DUCKIEBOT_NAME]` with just the Duckiebot's `hostname`, do not include `.local` part that you used previously to access the dashboard.

After startup, the `keyboard_control` command will open an interface window. Make sure the window is active by selecting it, and use the keys in the table below to command your Duckiebot:



*Fig. 43* The keyboard control graphical user interface.

The following keys control the Duckiebot:

| KEY                  | Function                                |
|----------------------|---|
| Arrow Keys - ↑ ↓ ← → | Steer your Duckiebot                    |
| q                    | Quit                                    |
| a                    | Turn on lane following (see note below) |
| s                    | Stop lane following                     |
| i                    | Toggle Anti-Instagram                   |

### **Note**

The `a`, `s`, and `i` functions require the [lane following demo](#) to be running. For now, just try out the keyboard control and get your Duckiebot moving!

### Option 2: Use the CLI (For Mac Users)

If you are using MacOSX and find the keyboard interface is not responsive, run the stack directly on the Duckiebot and use the same keys within the command line interface as listed in the [table above](#):

```
dts duckiebot keyboard_control ![DUCKIEBOT_NAME] --cli
```

## Troubleshooting

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | The Duckiebot does not move, and I cannot see the commands being sent to the Duckiebot when looking at the <b>Dashboard &gt; Mission Control</b> page.  |
| RESOLUTION | Make sure that the keyboard gui window is active by selecting it, then try the keyboard commands again. Some keyboard configurations may require that you use <code>w a s d</code> rather than <code>↑ ↓ ← →</code> . |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I can see the commands being sent to the Duckiebot (e.g., through the <b>Dashboard &gt; Mission Control</b> ), but the Duckiebot does not move. My <b>Dashboard &gt; Robot &gt; Components</b> page shows a red alert for the <b>HUT</b> . |
| RESOLUTION | If you have a <b>HUT v3.1</b> you will stumble on this problem the first time you try to move your Duckiebot. Re-flash your <b>HUT</b> following the procedure described in <a href="#">Debug - Re-flash Microcontroller</a> .             |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I have reflashed the HUT but the joystick commands still do not work or the Duckiebot operates in a jerky manner. Additionally, the ToF sensor and front bumper are not detected on the dashboard Components page. I may also be having issues with the screen.                                    |
| RESOLUTION | Disconnect the ToF sensor from the front bumper and use the long cable that originally connected the front bumper to the HUT to connect the ToF sensor directly to that same HUT port. Then reboot. This bypasses a known multiplexer issue on some bumpers that can cause other HUT misbehaviors. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | I checked the two troubleshooting issues above, and my Duckiebot still doesn't move.  |
| RESOLUTION | Check that the <code>duckiebot-interface</code> container is running<br><br>Open <a href="#">the Portainer interface</a> and check the running containers. You should see one that has a name that contains |

```
duckiebot-interface (exact container name will depend on your robot version).
```

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.

If you don't see the container, your base image is out of date - update your Duckiebot with the command

```
dts duckiebot update ![ROBOT_NAME]
```

## Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | Duckiebot goes backwards, even though I command it to go forward.  |
| RESOLUTION | If you have a <code>DB17</code> or <code>DB18</code> , revert the polarities (plus and minus cables) of the cables that go to the motor driver ( <code>HUT</code> ) for both motors. |

## Operation - Make it See

This section describes how to see what your Duckiebot sees.

### Viewing the Image Stream on Your Laptop

The camera image is streaming from your Duckiebot by default on startup. To see it, open a terminal on your laptop and run:

```
dts start_gui_tools ![DUCKIEBOT_NAME]
```

#### ⓘ Attention

For all operation commands that use the Duckiebot's name - input just the Duckiebot's `hostname`, do not include `.local` part that you used previously to access the dashboard.

This will start a container with access to the ROS messages of the Duckiebot, including the image stream from the camera.

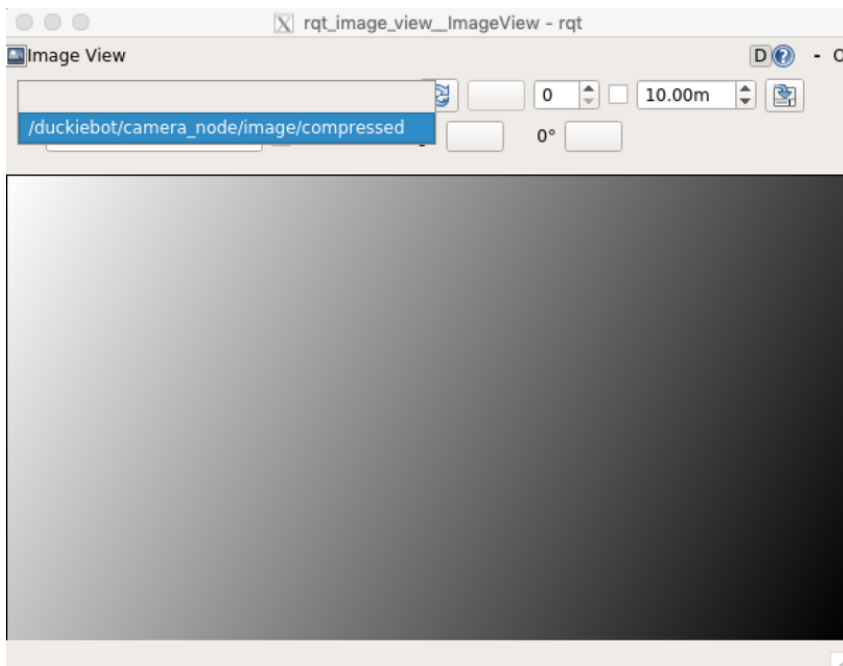
Your terminal has now turned into a command line interface running inside of that container within the Duckiebot. You can exit the container back to your normal terminal interface at any time by running the `exit` command. You will learn more about this tool in the [Operation - Tools](#) section.

To view the camera stream, run the following:

```
rqt_image_view
```

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

You will have to select the `camera_node/image/compressed` topic from the drop-down menu:



*Fig. 44* The rqt image view window with dropdown menu - select the `camera_node/image/compressed` topic.

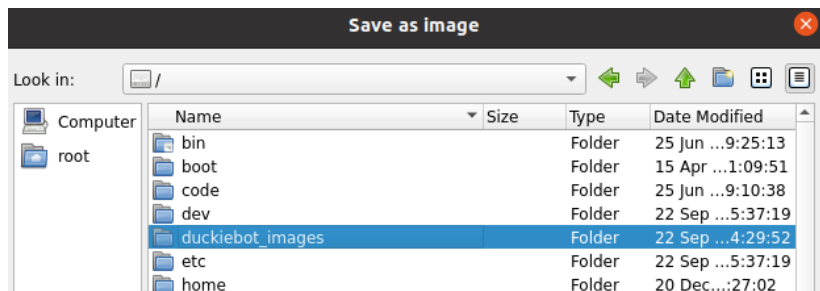
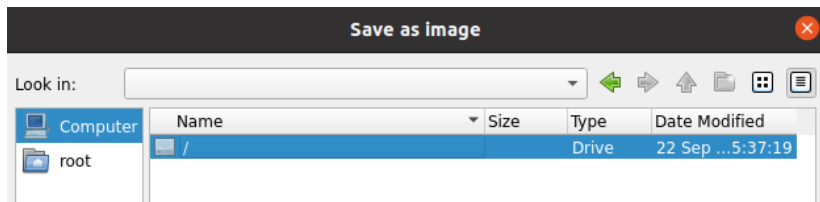
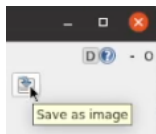
#### How to save a picture from `rqt_image_view`

On the top right of the `rqt_image_view` window, there is a button to save the current frame to an image. But don't save it yet. A little extra setup is needed to be able to view that file later.

Create a folder on you laptop for where you would like to have the image saved to, say `~/duckiebot_images/`. Then, launch the `start_gui_tools` with the following command:

```
dts start_gui_tools --mount ~/duckiebot_images/:~/duckiebot_images
[DUCKIEBOT_NAME]
```

Then, run `rqt_image_view` again, and use the top-right **“Save as image”** button to save to the `/duckiebot_images` folder. To find that folder, you might need to navigate to **Computer** and select `/` directory in the pop-up dialogue.



And the saved image from your robot's view should appear in the folder you created!

## Viewing the image stream on the Dashboard

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

Open the browser and visit the page `http://![DUCKIEBOT_NAME].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: [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.



By default, the camera stream is throttled down to 8 frames per second. This is to minimize 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.

## Viewing the image stream in no-vnc (optional)

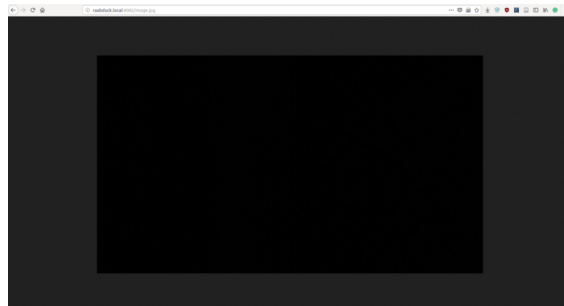
For instructions on using the no-vnc tool and viewing the image stream from the remote Desktop, see the [Operation - Software Tools](#) section.

## Troubleshooting

### Troubleshooting

#### SYMPTOM

I see a black image like this:



**Fig. 45** What you see if you leave the camera cap on.

RESOLUTION

Remove the cap of the camera.

### Troubleshooting

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.

### Troubleshooting

SYMPTOM

The images are out of focus.

RESOLUTION

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.

If the lens will not rotate, you may 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.

### Troubleshooting

SYMPTOM

I'm getting an error related to `libGL` when running `dts start_gui_tools`

RESOLUTION

If you have an error like the following 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.

### Troubleshooting

SYMPTOM

I don't see any image.

RESOLUTION

Use `rostopic hz /![DUCKIEBOT_NAME]/camera_node/image/compressed` and see if the image is being published. Images should be published at roughly 30 Hz. If the message is not being published,

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 update your Duckiebot to restart all containers with

```
dts duckiebot update ![DUCKIEBOT_NAME]
```

Or to manually start just 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
```

#### ➔ See also

For more information about `rostopic`, see [Starting non-`vnc` images](#). You can see the images as your robot sees them with `rostopic echo ![DUCKIEBOT_NAME]/camera_node/image/compressed`. Press `Ctrl + c` on the terminal once you've seen enough to confirm the messages are being populated.

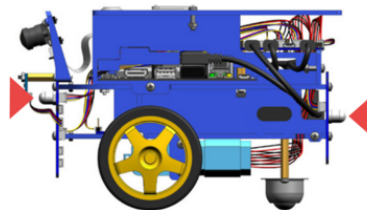
### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The camera is not detected from Duckiebot.                                       |
| RESOLUTION | (DB18, DB19 only) remove the battery pack and check the camera cable for damage. |

## Operation - Make it Shine

This section describes how to control the LEDs on your Duckiebot.

Duckiebots have four LEDs, positioned similarly to the head and tail lights on a car.



**Fig. 46** A duckiebot with the LEDs shining white and a diagram with arrows indicating the front and rear LEDs.

LEDs as actuators on a Duckiebot can be used for many purposes, including

- Indicating what mode or mission the Duckiebot is running
- Communicating state changes in the controller
- Signaling upcoming turns or other navigation plans
- Expressing character and personality
- And simply lighting the driving environment

## LED control

You can update the LEDs on your Duckiebot manually by bringing up the LED Widget using the `led_control` command provided by the Duckietown Shell.

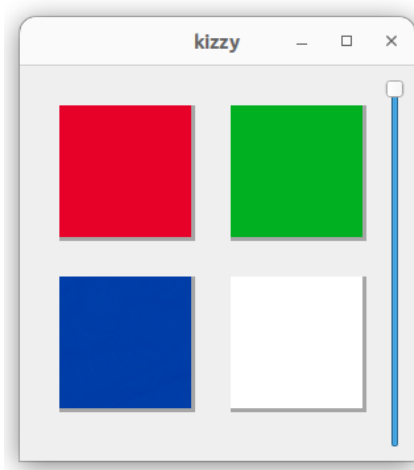
Open a terminal and run:

```
dts duckiebot led_control ![DUCKIEBOT_NAME]
```

### Attention

For all operation commands that use the Duckiebot's name - replace `![DUCKIEBOT_NAME]` with just the Duckiebot's `hostname`, do not include `.local` part that you used previously to access the dashboard.

After startup, the `led_control` command will open an interface window. Make sure the window is active by selecting it, and press the buttons to update the color and intensity of your Duckiebot's LEDs.



*Fig. 47* The LED control interface

## Troubleshooting

### Troubleshooting

#### SYMPTOM

When I press the buttons, the LEDs do not update. My **Dashboard > Robot > Components** page shows a red alert for the **HUT**.

#### RESOLUTION

If you have a **HUT v3.1** you will stumble on this problem the first time you try to move your Duckiebot. Re-flash your **HUT** following the procedure described in [Debug - Re-flash Microcontroller](#).

### Troubleshooting



|            |  |
|------------|--|
| SYMPTOM    | I have reflashed the HUT but the led commands still do not work. Additionally, the ToF sensor and front bumper are not detected on the dashboard Components page. I may also be having issues with the screen and joystick control.  |
| RESOLUTION | Disconnect the ToF sensor from the front bumper and use the long cable that originally connected the front bumper to the HUT to connect the ToF sensor directly to that same HUT port. Then reboot. This bypasses a known multiplexer issue on some bumpers that can cause other HUT misbehaviors. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | I checked the two troubleshooting issues above, and my Duckiebot still doesn't respond.   |
| RESOLUTION | <p>Check that the <code>duckiebot-interface</code> container is running</p> <p>Open <a href="#">the Portainer interface</a> and check the running containers. You should see one that has a name that contains <code>duckiebot-interface</code> (exact container name will depend on your robot version).</p> <p>You can also determine this by running:</p> <pre>docker -H ![ROBOT_NAME].local ps</pre> <p>and look at the output to find the <code>duckiebot-interface</code> container and verify that it is running.</p> <p>If you don't see the container, your base image is out of date - update your Duckiebot with the command</p> <pre>dts duckiebot update ![ROBOT_NAME]</pre> |

## Operation - Software Tools

This section describes how to use available software tools for your Duckiebot including the `start_gui_tools` interface, ROS tools, and the `no-vnc` Desktop interface.

### Using `start_gui_tools`

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

```
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 and have access to all ROS messages streaming on your Duckiebot.

#### ! Attention

Note again that in these commands you input the Duckiebot `hostname`; do not include `.local` part.

### ⚠ Warning

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

## Starting `no-vnc` images

If you would rather use an image that runs `no-vnc` to provide you with a Desktop environment connected to the Duckiebot, type:

```
dts start_gui_tools --vnc ![DUCKIEBOT_NAME]
```

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. Click on the `Terminal` Desktop icon to open the command line interface.

## Using the ROS utilities

Once you've opened a terminal connected to the Duckiebot container using one of the two methods above, you can use any of the commands below within that interface to check the data streams in ROS.

### List topics

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

```
rostopic list
```

See also: For more information about `rostopic`, see [the official ROS wiki here](#).

You should see at least the following topics:

```
#![DUCKIEBOT_NAME]/camera_node/camera_info  
#![DUCKIEBOT_NAME]/camera_node/image/compressed  
/rosout  
/rosout_agg
```

There might be other topics if you have already started other demos.

### Show topics frequency

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

```
rostopic hz ![DUCKIEBOT_NAME]/camera_node/image/compressed
```

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

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

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

### Show topics data

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

```
rostopic echo ![DUCKIEBOT_NAME]/camera_node/image/compressed
```

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

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

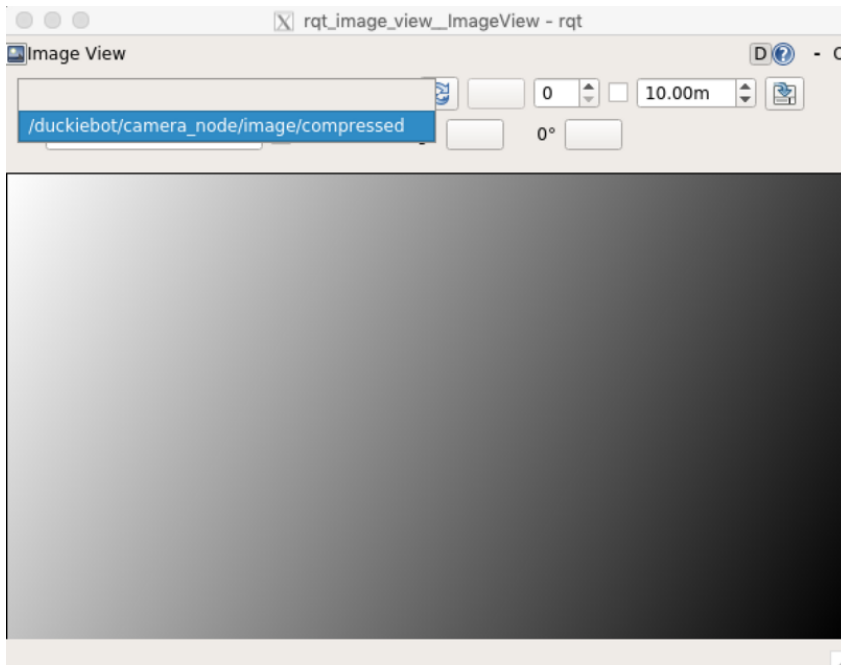
### rqt\_image\_view tool

To see what your Duckiebot sees, you can run

```
rqt_image_view
```

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

You will have to select the `camera_node/image/compressed` topic from the drop-down menu:



**Fig. 48** The rqt image view window with dropdown menu - select the `camera_node/image/compressed` topic.

Or if you are using the `no-vnc` interface, click on the `RQT Image View` application icon on the desktop. You will see the same `rqt_image_view` interface start up

### 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.

## ROS Troubleshooting

| Troubleshooting |  |
|-----------------|--|
| SYMPTOM         | My ros commands are not working. I cannot use tab to auto complete ROS commands.                                 |
| RESOLUTION      | You can fix that by sourcing <code>devel/setup.bash</code><br><pre>source /code/catkin_ws/devel/setup.bash</pre> |

## Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I cannot connect to ROS master.  |
| RESOLUTION | Go to <b>Dashboard &gt; Portainer</b> to make sure the following containers are running without errors: <ul style="list-style-type: none"> <li>• ROS</li> <li>• car-interface</li> <li>• duckiebot-interface</li> </ul> <p>If they are not running, refer to <a href="#">Docker troubleshooting</a> for instructions to resolve the issue.</p> |

## Calibration - Camera

This section describes the intrinsic and extrinsic calibration procedures. **You will need to complete both the Camera and Wheels calibration processes before running any Duckietown demos.**

|                    |  |
|--------------------|--|
| What you will need | You can see the camera image on the laptop.  |
| What you will get  | Your camera intrinsics and extrinsics are calibrated and stored on the Duckiebot. You will be able to access these calibrations via the dashboard. |

### 0) Required Materials

Calibration board

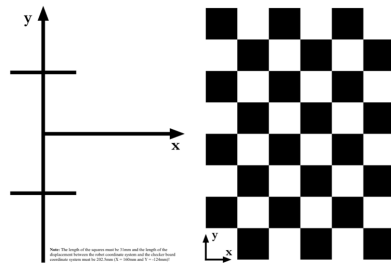


Fig. 49 Calibration checkerboard

If you do not have one already:

- Download the [Calibration checkerboard pdf](#)
- Print it with **A3 Format**
- Fix the checkerboard to a rigid planar surface that you can move around

#### **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.
- In case your squares are not of the right size, make sure your printer settings are on A3 format, no automatic scaling, 100% size.

#### **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.

## Optional material

This is not necessary, but you could also use a “lane” during the extrinsic calibration procedure.

## 1) 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, see [our theory slides](#). And the procedure is basically a wrapper around the [ROS camera calibration tool](#).

### Launch the intrinsic calibration application

You can launch the intrinsic calibration program with:

```
dts duckiebot calibrate_intrinsics [your_duckiebot_hostname]
```

You should see an application launching, similar to the following figure, on the laptop.

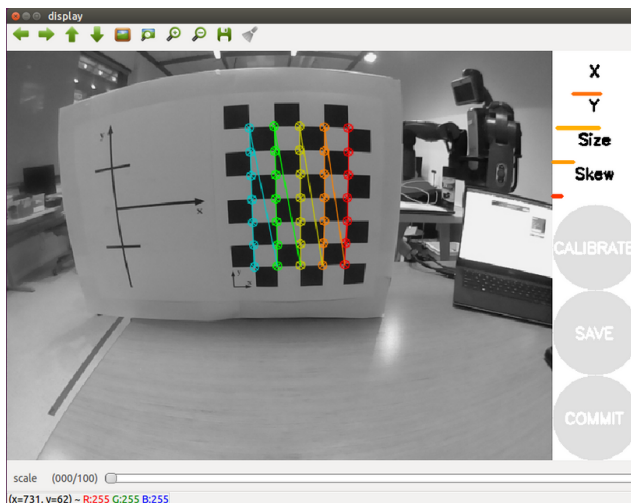


Fig. 50 Intrinsic camera calibration tool

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | Only a black window starts up  |
| RESOLUTION | Try resizing the window manually once using cursor, and you should see the window content correctly. |

### Do the 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.

Make sure to focus the image by rotating the mechanical focus ring on the lens of the camera until you can clearly read the x and y labels.

#### **Warning**

Do not adjust the focus again after this process, as it will invalidate the calibration.

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

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.

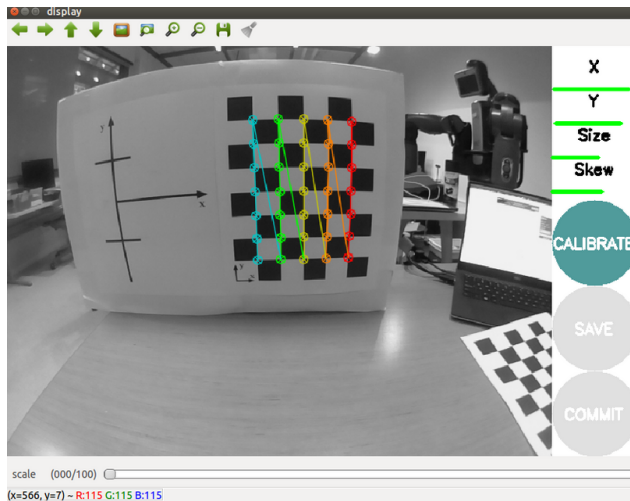
The indicator bars will fill as you collect enough data along each axis.

Once you have collected enough data, all four indicator bars will turn green. Then press the "CALIBRATE" button in the sidebar.

**Tip**

If you are having a difficult time getting the indicator bars to turn green, try slowly increasing the extremes at which you present the checkerboard to the camera, focusing on one indicator bar at a time. (Move further left/right along the x axis only, then the y axis, etc.)

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



*Fig. 51* Calibration step

**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.)

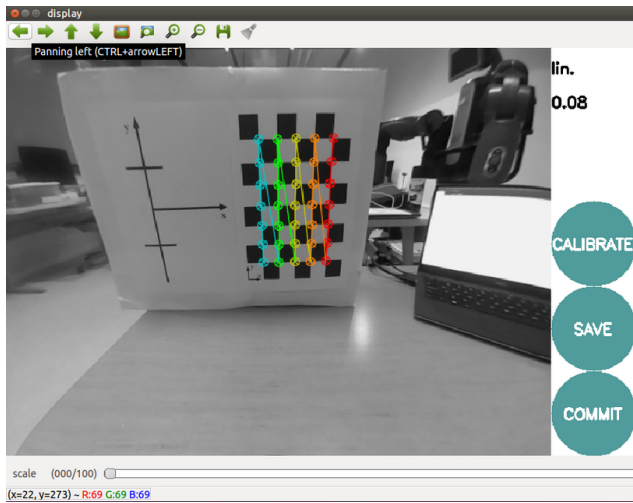


Fig. 52 Committing the calibration

This will automatically save the calibration results on your Duckiebot:

```
/data/config/calibrations/camera_intrinsic/[your_duckiebot_hostname].yaml
```

You can view or download the calibration file using the Dashboard running at [http://\[your\\_duckiebot\\_hostname\].local](http://[your_duckiebot_hostname].local) under **File Manager** in the sidebar on the left, navigating to `config/calibrations/camera_intrinsic/`.

### Confirm the calibration

Use the Dashboard to confirm that both calibration the intrinsic file has been saved.

### Keeping your calibration valid

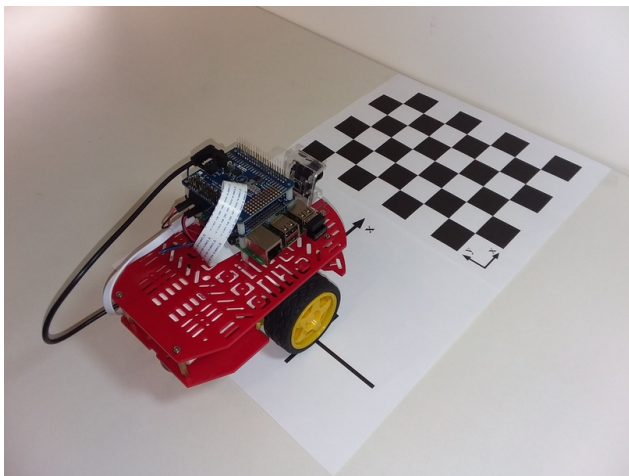
#### ⚠ Warning

- Do not change the focus during or after the calibration, otherwise your calibration is no longer valid.
- Do not use the lens cover anymore; removing the lens cover may change the focus.

## 2) Extrinsic Camera Calibration

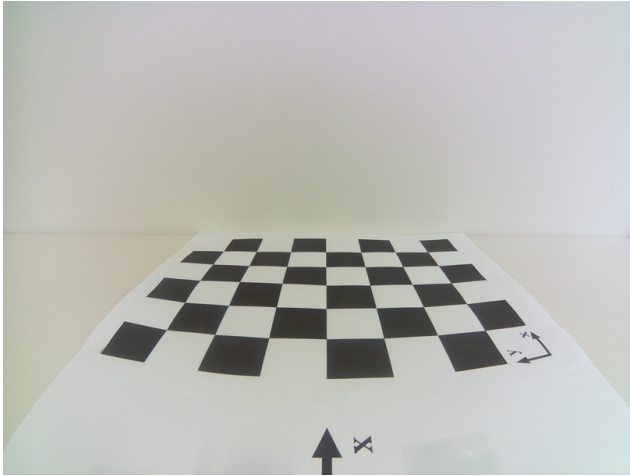
### Setup

Arrange the Duckiebot and checkerboard according to [Fig. 53](#). Note that the axis of the wheels should be aligned with the y-axis.



**Fig. 53** Extrinsic calibration setup

[Fig. 54](#) shows a view of the calibration checkerboard from the Duckiebot. To ensure proper calibration there should be no clutter in the background.



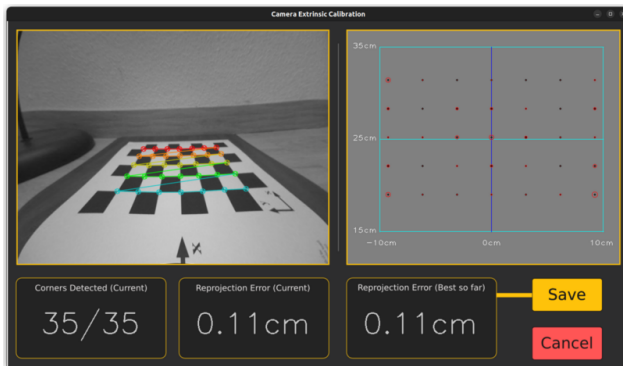
**Fig. 54** Extrinsic calibration view

Launch the extrinsic calibration pipeline

Run:

```
dts duckiebot calibrate_extrinsics [your_duckiebot_hostname]
```

You will see the calibration window pop up:



**Fig. 55** Extrinsic calibration GUI

Press **Save** to automatically save the calibration results on your Duckiebot. You can find it at the path:

```
/data/config/calibrations/camera_extrinsic/[your_duckiebot_hostname].yaml
```

Similarly to the intrinsic calibration, you can also view or download the calibration file through the Dashboard.

Once the calibration has been saved a validation window will open. Here you should be able to see the checkerboard from a top-down view obtained using the computed homography matrix.



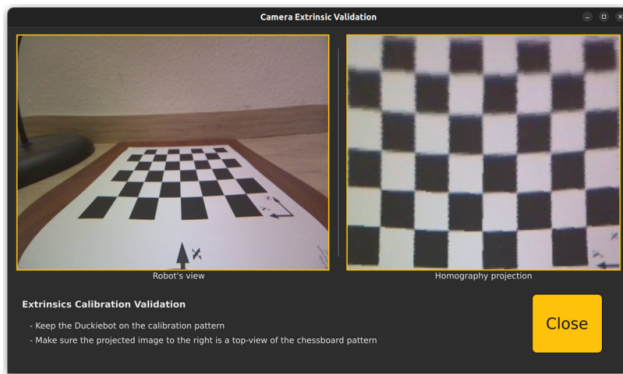


Fig. 56 Extrinsic calibration validation GUI

### Attention

If you do not see a saved extrinsic calibration file, your Duckiebot was not able to detect the checkerboard and generate a valid calibration.

This could be due to

- lack of evenly bright overhead lighting
- A patterned background in the environment
- A textured background wall or the shadow on the background wall

Try changing the calibration environment to match [Fig. 54](#) before running the `calibrate_extrinsics` command again.

### Confirm the calibration

Use the Dashboard to confirm that both calibration the intrinsic and extrinsic files have been saved.

### Troubleshooting

| Troubleshooting |   |
|-----------------|---|
| SYMPTOM         | You see a long complicated error message that ends with something about <code>findChessBoardCorners failed</code> .   |
| 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. |

### Calibration - Wheels

|                    |   |
|--------------------|---|
| What you will need | You can make your robot move  |
| What you will get  | You can calibrate the wheels of the Duckiebot such that it goes in a straight line when commanded so. You can set the maximum speed of the Duckiebot. |

### Note

In the following documentation on this page, `[hostname]` refers to the hostname of your duckiebot. For example:

```
ping [hostname].local
```

with a Duckiebot named `myrobot` would be:

```
ping myrobot.local
```

Also, replace other fields in a command with `[]` around with the corresponding values.

## Step 1: Make your robot move

Follow instructions in [Operation - Make it Move](#) to make your robot move with keyboard control, and keep the terminal open.

## Step 2: See how the robot really moves

There is a lot going on between pressing `↑` (up arrow) on the keyboard and your Duckiebot moving. To get a better view of what is going on, we need another terminal, but ran closer to where the action is happening:

```
dts start_gui_tools [hostname]
```

Duckietown uses [ROS](#) to move data around. To determine if the command above worked, type:

```
rostopic list
```

You should see a list of ROS *topics* currently active on your Duckiebot. If you know ROS, here you can use ROS commands at will. If you are not familiar with ROS, note that each of these *topics* might carry *messages*, i.e., actual data. You can, e.g., “listen” to the data inside each topic. For example:

```
rostopic echo /[hostname]/camera_node/image/compressed
```

will show you incoming images as the Duckiebot sees them! You won't be able to read bytes or make sense of pixel values, though. Now, press `Ctrl+C` to stop reading the camera stream.

### Note

Keep this terminal open. We will use it to perform the wheel calibration. All the `ros...` commands below are executed in such a terminal.

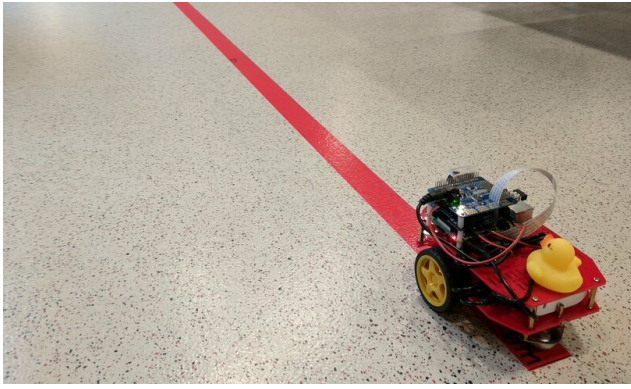
## Step 3: Perform the calibration

### 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 (in the `start_gui_tools` terminal from the previous step):

```
rosparam set /[hostname]/kinematics_node/trim [trim_value]
```

Use some tape to create a straight line on the floor ([Fig. 57](#)).



**Fig. 57** Straight line useful for wheel calibration

Place your Duckiebot on one end of the tape. Make sure that the Duckiebot is perfectly centered with respect to the line.

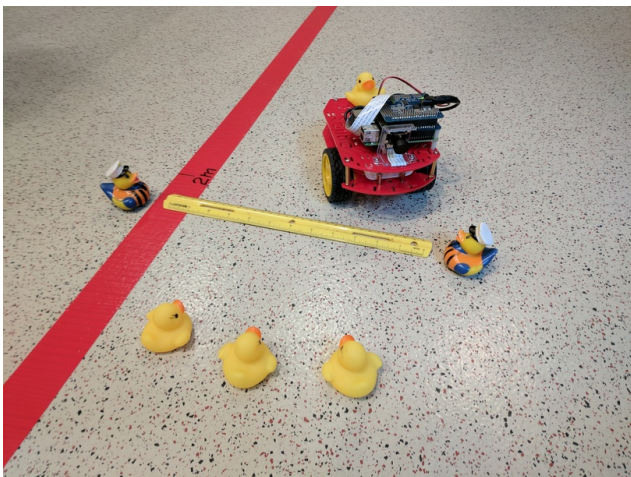
Command your Duckiebot to go straight for about 2 meters. Observe the Duckiebot from the point where it started moving, and annotate on which side of the tape the Duckiebot drifted ([Fig. 58](#)).



**Fig. 58** Left/Right drift

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 ([Fig. 59](#)).

Make sure that the ruler is orthogonal to the tape.



**Fig. 59** 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:

```
rosparam 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:

```
rosparam set /[hostname]/kinematics_node/trim 0.1
```

Repeat this process until the robot drives straight.

### 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]
```

### 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 calibration file

### 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](http://[hostname].local) under **File Manager** in the sidebar on the left, navigating to `config/calibrations/kinematics/`.

### Additional information

There are additional parameters you can play around with to get a better driving experience. You can learn about odometry and odometry calibration here: [video](#), [theory](#), [activities and exercises](#).

## Operation - Taking and verifying a log

### This section is outdated

Note that this section is outdated. We are working on updating it.

Requires: [Operation - Make it Move](#)

Requires: [Calibration - Camera](#)

Result: A verified log.

### Preparation

### **Note**

Use note directives for basic highlighting.

This assumes that you have the folder `/data/logs` on the Duckiebot. If not, 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.

## Record the log

### 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.

### Option: Full Logging on the Duckiebot

To log everything that is being published, run the base container on the Duckiebot:

```
dts duckiebot demo --demo_name make_log_full_docker --duckiebot_name !  
[DUCKIEBOT_NAME] --package_name duckietown_demos
```

## 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.

## Getting the log

### 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/`.

### Using a USB drive

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

### 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.

## Verify a log

### Note

This procedure requires `roscpp` to be installed. If you have not installed that already, you can do so via:

```
sudo apt-get install python3-roscpp
```

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

```
roscpp 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 joystick was publishing at a rate of around **26Hz**.

An example of the output looks like this:

```
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
```

## Introduction to Demos

This section lists all Duckietown *demos*. By *demos* we mean behaviors that are easily executed, but not explained. Demos are useful for doing rather than learning, e.g., to quickly test a behavior or showcase it in outreach circumstances.

- **Supported:** these are polished demos that are actively maintained. You should be able to follow the instructions and obtain the expected outcomes without errors.
- **Legacy:** these demos are either under development or worked at some point in the past. This content is not actively supported at this point in time but might work if special attention is provided. We leave this non-polished content here to provide ideas on projects or classes. We welcome contributions to these materials. If you would like to see some of these demos moved to the “supported” section do not hesitate to let the Duckietown team know.

## Duckiebot Supported Demos

This section contains all the *supported* demos that can be run on one or more Duckiebots.

What you will need

- A computer with all steps and checkpoints completed from the [Software Setup section](#)

What you will get

- An assembled and calibrated Duckiebot
- A Duckiebot following lanes in your Duckietown!

## Lane following (LF)

What you will need

- An assembled and initialized Duckiebot;
- [Wheels calibration](#) completed;
- [Camera calibration](#) completed;
- [Joystick demo](#) has been successfully launched;
- A Duckietown city loop, as detailed in the [appearance specifications](#).

What you will get

- A Duckiebot driving autonomously in a Duckietown city loop, without other vehicles, intersections, or obstacles.

## Expected results

**i** Outcome of the lane following demo.

[A Duckiebot following the lane.](#) from [Duckietown](#) on [Vimeo](#).

## Duckietown setup notes

Before getting started, make sure your Duckietown is ready to go:

- The layout adheres to the [The Duckietown Operation Manual](#);
- The lighting is “good”: ideally white diffused light. This demo relies on images from the camera and color detections, so avoid colored lights, reflections or other conditions that might confuse or blind the onboard image sensor;

## Duckiebot setup notes

- Duckiebot in configuration [DB21M, DB21J, or DB19](#).
- The Duckiebot is powered on and connected to your network. You should be able to successfully ping it from your base station;
- You are able to [see what the Duckiebot sees](#);
- You are able to [remote control](#) your Duckiebot;
- The [camera is calibrated](#);
- The [wheels are calibrated](#).

## Pre-flight checklist

- Place the Duckiebot inside a lane (driving on the right-hand side of the road), making sure it sees the lane lines;

- Make sure no containers are running on the Duckiebot which use either the camera or the joystick. We will run these ROS nodes in a new container.

## Demo instructions

### Start the lane following demo

To start the lane following (LF) demo:

```
dts duckiebot demo --demo_name lane_following --duckiebot_name !
[DUCKIEBOT_NAME] --package_name duckietown_demos
```

It will take a minute for the demo to launch. You can check Portainer if all the containers started successfully, and their logs if any issues arise.

### Start driving autonomously

Run the keyboard controller (not necessary if you have a joystick set up instead) and press the indicated key to initialize the driving behavior:

```
dts duckiebot keyboard_control ![DUCKIEBOT_NAME]
```

| Controls             | Joystick  | Keyboard |
|----------------------|-----------|----------|
| Start Lane Following | <b>R1</b> | a        |
| Stop Lane Following  | <b>L1</b> | s        |

Table 2 Lane following demo start and stop commands

In case intersections and/or red lines are present in the city layout, they will be neglected. The Duckiebot will drive across them like it is a normal lane.

You can regain control of the Duckiebot at any moment by stopping the demo using the (virtual) joystick. The demo can be resumed by pressing the start button.

### Visualize the detected line segments

This step is optional, and it provides a visualization of the line segments that the Duckiebot detects, and is useful to debug eventual weird behaviors.

- Make the `lane_filter_node` publish all the image topics. The `start_gui_tools` will provide a shell that is connected to the ROS agent:

```
dts start_gui_tools ![DUCKIEBOT_NAME]
```

- Run `rqt_image_view` and select the `/ROBOT_NAME/line_detector_node/debug/segments/compressed`. You should see something like this:



**i Outcome of the line detector node.**

[Line segment detections](#) from [Duckietown](#) on [Vimeo](#).

- In `rqt_image_view` by selecting the `/ROBOT_NAME/ground_projection_node/debug/ground_projection_image/compressed` you should see a top-down view of the detected segments:



**Fig. 60** Line segments projected on the ground

## Troubleshooting

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | The demo does not respond when pressing the start or stop buttons |
| RESOLUTION | Make sure the keyboard controller window is actively selected     |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The Duckiebot does not move  |
| RESOLUTION | <ul style="list-style-type: none"><li>• Check if you can manually drive the Duckiebot</li><li>• Try re-launching <code>dts duckiebot keyboard_control ![hostname]</code></li><li>• Check if ROS messages are received on the robot on the <code>![hostname]/joy</code> topic</li></ul> |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | The Duckiebot does not stay in the lane, or overall exhibits a bad driving behavior   |
| RESOLUTION | <p>This can be due to a number of issues, e.g.: bad robot calibrations, bad perception of lines due to inappropriate lighting or non-respected appearance specifications of the town, and/or bad tuning of the lane PID controller. The best way to debug this is to:</p> <ul style="list-style-type: none"><li>• Check <code>rqt_image_view</code> and look at <code>image_with_lines</code>.</li><li>• Check if you see enough segments. If not enough segments are visible, reset the Anti-Instagram filter.</li></ul> |

- 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.

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | The Duckiebot does not drive nicely through intersections   |
| RESOLUTION | For this demo, there should not be any intersections in the city layout. Duckiebots will interpret intersections as “broken” lanes, perceiving less salient features, potentially compromising the state estimate hence causing the driving troubles. |

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | The Duckiebot does not drive nicely through intersections   |
| RESOLUTION | For this demo, there should not be any intersections in the city layout. Duckiebots will interpret intersections as “broken” lanes, perceiving less salient features, potentially compromising the state estimate hence causing the driving troubles. |

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | The Duckiebot cuts white line while driving on inner curves (advanced)   |
| RESOLUTION | This might be due to wrongly constructed lanes or bad Duckiebot calibrations. Fortunately, feedback control should take care of most of these problems.<br><br>While running the demo modify the PID controller gains from the base station through: |

```
rosparam set ![
[DUCKIEBOT_NAME]/lane_controller_node/k_d -45
rosparam set ![
[DUCKIEBOT_NAME]/lane_controller_node/k_theta -11
```

## Duckiebot Legacy Demos

This section lists legacy Duckiebot demos.

|                    |  |
|--------------------|--|
| What you will need | An internet connection; About 10 minutes; A computer with the Duckietown Shell command <a href="#">installed and correctly setup</a> ; |
| What you will get  | Duckietown token correctly set up;   |

## 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.

What you will need

- A Duckiebot in [DB18](#) configuration that is [initialized](#)
- Laptop configured, according to .

What you will get

- A behavior executed on your Duckiebot.

## Start demos

In the [Duckietown dt-core](#), some ROS packages serve as building blocks for complex demos. Each package contains node-specific launch files. The `duckietown_demos` package contains demo launch files that combine multiple node launch files and adds the necessary connections 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
```

This command will start the `DEMO_NAME.launch` launch file in the `PACKAGE_NAME` package from the `duckietown/![IMAGE]:daffy` Docker image on the `DUCKIEBOT_NAME` Duckiebot.

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.

## 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.

## Lane following with obstacles

### Warning

This demo is archived and not currently maintained. These instructions might be obsolete.

This demo is build on the [lane following demo](#) by allowing multiple Duckiebots on the same lane. Duckiebots will detect and maintain a distance from other vehicles. This demo is sometimes referred to as “follow the leader”, when ran outside a city, or more simply as “traffic management”.

What you will need

- Several assembled and initialized Duckiebot;
- [Wheels calibration](#) completed on all Duckiebots;
- [Camera calibration](#) completed on all Duckiebots;
- [Joystick demo](#) has been successfully launched;

What you will get

- A Duckietown city loop, as detailed in [The Duckietown Operation Manual](#).
- One or more Duckiebot driving autonomously in lanes while preventing crashes with other Duckiebots on the same lane.

Video of expected results {#demo-lane-following-with-obj-det-expected}

[Video Link \(Real\) - Lane following with vehicles](#)

Duckietown setup notes {#demo-lane-following-with-obj-det-duckietown-setup}

To run this demo, you can setup a quite complex Duckietown. The demo supports a variety of road tiles, straight, complex turns, etc. It also supports dynamic and static vehicles and can robustly avoid them. That makes it a level more difficult than the lane following demo.

You also need a wireless network set up to communicate with the robot or to `ssh` into it. The demo is robust to varied lighting conditions but take care that the duckietown is not too dark to hinder with the object detections.

- Add one (or possibly more) Duckiebot in duckiebot configurations.
- Care must be taken that duckiebot is rigid and there are no loose parts.
- The camera must be upright so that it has a proper field of view.
- Make sure the calibration(both intrinsic and extrinsic is done meticulously). The performance is sensitive to it.
- Make sure the battery is fixed in place and fully charged.

Pre-flight checklist {#demo-lane-following-with-obj-det-pre-flight}

- Check that every Duckiebot has sufficient battery charge and that they are all properly calibrated.
- Turn the duckiebot on and wait for it to boot. To check if it's ready, try to `ssh` into it.
- Place the duckiebot in the right lane. The algorithm may not allow it to recover if it is kept in the wrong ie, the left lane.
- You can use `portainer` to see what all containers are running on the duckiebot. In this demo, we will run a new container.

Demo instructions {#demo-lane-following-with-obj-det-run}

[Github Link to the Package for the Demo](#)

## Setup

Fork and clone udem-fall19-public:

```
git clone https://github.com/charan223/udem-fall19-public.git
```

If you are using fork, create an upstream:

```
git remote add upstream https://github.com/duckietown-udem/udem-fall19-public.git
```

Then pull from upstream:

```
git pull upstream master
```

Update submodules

```
git submodule init
git submodule update
git submodule foreach "(git checkout daffy; git pull)"
```

To run in simulation

From udem-fall19-public directory run:

```
docker-compose build
docker-compose up
```

You can open the notebook just like before by copying the url that looks like:

```
http://127.0.0.1:8888/?token={SOME_LONG_TOKEN}
```

After that, open terminal in the notebook and run below commands:

```
jupyter-notebook $ catkin build --workspace catkin_ws
jupyter-notebook $ source catkin_ws/devel/setup.bash
jupyter-notebook $ ./launch_car_interface.sh
```

Run your launch file using below commands:

```
jupyter-notebook $ roslaunch purepursuit purepursuit_controller.launch
```

Visualize it in the simulation at <http://localhost:6901/vnc.html> in your browser, with **quackquack** as the password. Run **rqt\_image\_view** in the terminal to visualize the image topics.

### To run on real bot

Go to package's base directory and do

```
dts devel build --push
```

Pull the docker image for LFV on the duckiebot:

```
ssh ![DUCKIEBOT_NAME].local
docker pull charared/charan_ros_core:v1-arm32v7
```

To run LFV on the duckiebot:

```
dts duckiebot demo --demo_name purepursuit_controller --duckiebot_name !
[DUCKIEBOT_NAME] --package_name purepursuit --image
charared/charan_ros_core:v1-arm32v7
```

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.

Portainer can be accessed at below link:

```
https://![DUCKIEBOT_NAME].local:9000/#/containers
```

You can place your duckiebot on the lane after you see in the logs that the duckiebot can see the line segments. The duckiebot stops if it sees an obstacle(duckiebot) ahead and continues following the lane after the obstacle is removed.

Stopping the container `demo_purepursuit_controller` in the portainer will stop the lane following.

### Implementation details

#### Description of the Demo and the Approach.

The task is to follow the lane and avoid dynamic vehicles. For Lane Following, we use the Pure-Pursuit Controller. It seems to work better than the vanilla PID controller. For Lane Following with Dynamic Vehicles (LFV), we consider two types of objects, static Duckies, and dynamic Duckiebots. We follow a simple approach to avoid vehicles. First, we do Object Detection. We tried two methods for this

- Supervised Learning using Faster RCNN
  - HSV Thresholding based detection Upon Detection of a nearby obstacle, in this demo, we just stop the Duckiebot till the obstacle makes way or is removed. This approach can be expanded to incorporate more complex *Rule-Based Maneuvering*
- 

## Lane Following with the Pure-Pursuit Controller

### Basics of Pure Pursuit Controller

We have used the purepursuit controller for the duckiebot to follow the lane. Follow point is calculated ahead in the lane using the white and yellow segments detected.

### Our implementation of the controller

We have used a trust segment approach, where we consider only that particular color segment if they are more in number than others in calculating the follow point. If both color segments are equal in number, we take an average of the centroids. Using this kind of approach avoids a jittery path of the duckiebot.

### Dynamic velocity and omega gain for the duckiebot

We have implemented a dynamic velocity and omega gain for the duckiebot using follow point. If the y coordinate of the follow point is towards the farther right or left side, it implies that there is a right or left turn there, and we slow down the duckiebot and increase the omega gain to make the turn smoothly. We also vary velocity based on how farther the x coordinate of the follow point is (i.e., increase velocity if the road is straight for a long distance).

---

## LFV: Object Detection using HSV Thresholding

The most basic way to detect objects which can work in both simulation and the real environment is doing HSV thresholding and then doing opencv operations to get the bounding box for the object. A brief description is as follows

- We have performed HSV thresholding for both simulation and the real environment separately
- The `pure_pursuit` node on receiving an image processes it and determines if an object is there.
- HSV thresholding for reds and yellows is done to nicely detect the duckiebots and duckies, to form a mask. We used OpenCV trackbars to find optimal HSV values.
- Dilation is applied to form good blobs.
- Then we find contours using `cv2.findContours` to aggregate the blobs. We use those contours to get a nice bounding box around the detection

Now after the object is detected, we need to compute how far it is from our duckiebot and whether the stopping criterion is satisfied. This is described next:

### Ground Projections

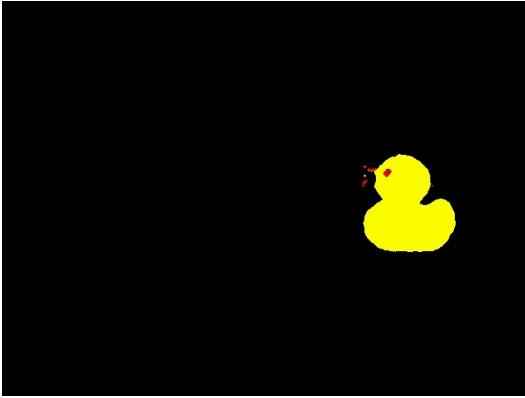
We have written our custom `point2ground` function to transform the detected bounding box coordinates(bottom two points of the rectangle) on the ground. We have initially normalized point coordinates to original image coordinates and calculated the ground point coordinates by performing dot product of homography matrix and the point coordinates. Now the projected points tell us how far the detected object is from our robot.

### Vehicle Avoidance

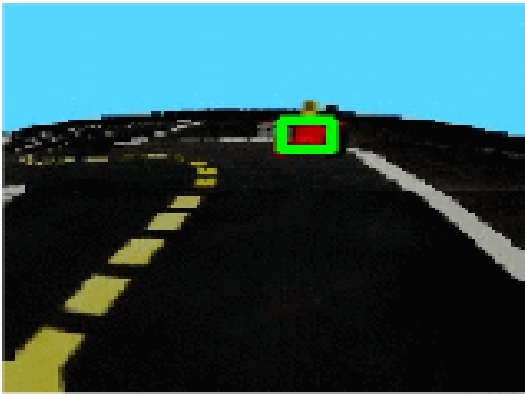
We use a very simple approach of just stopping on spotting an object which will collide with the current trajectory. The stopping criterion is very simple. If the object is within some deviation of our lane and within a threshold distance, then we stop, else we ignore. This distance is obtained by using the ground projection node described before. This threshold is fine-tuned for optimal performance.

## Qualitative Results in the Simulation

The following is the results obtained from doing hsv thresholding in the simulation. We can see that it does a good job of detecting all the objects.



Qualitatively we can also see the detection and the consequent stopping of the duckiebot after the thresholding in simulation.



[Full Video Link \(Simulation\) - Lane following with vehicles](#)

## Qualitative Results in the Environment

Note that in the environment, the HSV thresholding values are different.



[Full Video Link \(Real\) - Lane following with vehicles](#)

[Full Video Link \(Real\) - Lane following](#)

---

## LFV: Object Detection using Deep Learning

This approach is mainly targetted at leveraging the available Dataset and compute to use DL to do Object Detection in Real Environment. We use logs from the real world, and Deep Learning-based Faster RCNN with ResNet-50 and Feature Pyramid Network (FPN) backbone.

### Dataset

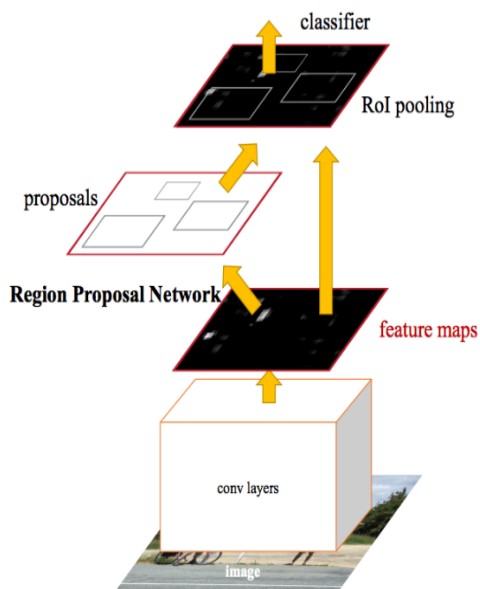
- Dataset has images from logs of a Duckiebot in a Duckietown in a variety of lighting conditions with objects like cones, signs, duckiebots, duckies etc.
- There are annotations for all of these objects. We only make use of the annotations for duckiebots and duckies since we assume only these in our environment.
- Dataset as approximately 3k images
- This dataset was provided by Julian Zilly, ETH Zurich. Contact him for a copy of the dataset. [jzilly@ethz.ch](mailto:jzilly@ethz.ch)
- TODO: Preprocessing script for this dataset can be found at,

### Faster RCNN

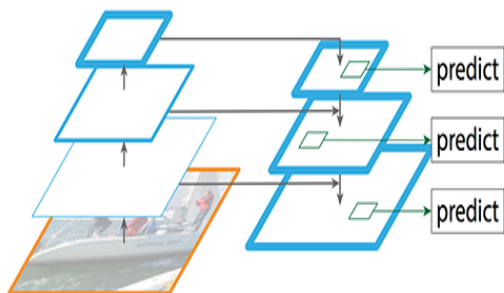
- Faster RCNN is a popular object detector model that first uses a Region Proposal Network to give candidate regions based on the image statistics, and for each of those regions, there is a classifier for the object type(also has background class) and a regressor for the bounding boxes.
- There are numerous tricks that are used to do efficient computation of the Feature Maps for Proposals using Roi pooling trick.
- The backbone used in this network is the 51 layered ResNet-51.
- For more details you can refer to <https://arxiv.org/abs/1506.01497>
- On just using the Faster RCNN, we noticed that the smaller duckies and the far away duckiebots were not detected properly.
- So to do detection at different scales for more fine-grained detections, we use the Feature Pyramid Network. For more details look at <https://arxiv.org/abs/1612.03144>



- We used the implementation by Facebook AI research in their Detectron 2 framework.
- Link to our implementation here: [Colab Link](#)



In Faster R-CNN, a single CNN is used for region proposals, and classifications. Source: <https://arxiv.org/abs/1506.01497>.



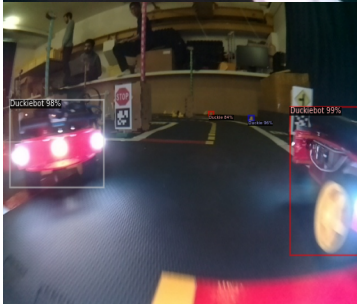
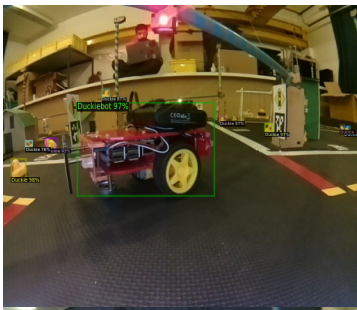
#### Training Details

- We used an 80:20 Train, Val Split.
- Approx 500 Duckiebot instances and 5k duckie instances were used.
- For the backbone we use pre-trained weights from MS-COCO. So we need to only fine-tune to our dataset. Which is time-efficient as well as allows us to train on our small-dataset.

#### Obtained Results

- Mean Average Precision(MAP) for duckies was 52.180 and for duckiebot was 52.011
- Inference Time for the trained model is 0.05 sec/img
- The detections are robust to different lighting conditions, as can be seen from the results below.

We present some qualitative results which clearly show that this method performs really well and is able to accurately detect all the duckiebots and all the ducks which are important.



The initial plan was to use this Deep Learning-based classifier in the real environment using a GPU. But the process of setting up the Duckietown framework such that the inference is done on the remote and the detections are communicated via the network is non-trivial, and hence it was left for future work. We attempted to run our code on CPU, but the inference times ranged between 5 to 10 seconds. So it wasn't feasible to run it on the raspberry pi for which the inference timing would have been even longer.

## Performance

**RUNNER UP** during LFV Challenge at NeurIPS, Vancouver 2019

**RUNNER UP** during LF Challenge at UdeM, Montreal 2019

AI Driving Olympics Leaderboard

| Challenge                 | Position |
|---------------------------|----------|
| aido3-LFV-real-validation | 2        |
| aido3-LFV-sim-testing     | 6        |
| aido3-LFV-sim-validation  | 8        |
| aido3-LF-real-validation  | 8        |
| aido3-LF-sim-testing      | 10       |
| aido3-LF-sim-validation   | 10       |

## References

- Mask R-CNN, Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, <https://arxiv.org/abs/1703.06870>
- Feature Pyramid Networks for Object Detection, Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie
- <https://github.com/facebookresearch/detectron2>

Maintainer: Contact Charan Reddy (UdeM/ Mila), Soumye Singhal (UdeM/ Mila), via Slack for further assistance.

## Introduction to Learning Experiences

This section lists all Duckietown learning experiences (LXs):

- **Supported:** these are polished LXs that are actively maintained. You should be able to follow the instructions and obtain the expected outcomes.
- **Experimental:** these LXs are either under development or worked at some point in the past. This content is not actively supported at this point in time but might work with some particular attention. We leave this non-polished content here to provide ideas on projects or classes. We welcome contributions to these materials.

## Supported Learning Experiences

This section lists the supported learning experiences available for Duckiebots.

|                    |   |
|--------------------|---|
| What you will need | An internet connection; About 10 minutes; A computer with the Duckietown Shell command <a href="#">installed and correctly setup</a> ; Duckietown token correctly set up; |
| What you will get  | Knowledge!  |

- [Braitenberg](#)
- [Collision Checker](#)
- [Odometry and PID Control](#)
- [Visual lane servoing](#)
- [Object Detection](#)
- [State Estimation](#)
- [Planning](#)

## 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.

|                    |  |
|--------------------|--|
| What you will need | <ul style="list-style-type: none"> <li>• A Duckiebot that is initialized</li> <li>• Laptop configured, according to .</li> </ul> |
|--------------------|--|

- That you are able to submit a challenge according to .

What you will get

## Video Tutorial



## Getting Started

Fork the [duckietown-lx](#) 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/mooc-exercises.git
```

Now to pull anything new from the original repository you can do:

```
git pull upstream daffy
```

Enter the `mooc-exercises` folder that you just cloned do:

```
cd mooc-exercises
```

In here you will see a number of folders. Each folder corresponds to an exercise.

## The Anatomy of an Exercise

Exercises should contain all of the following:

`config.yaml`

This contains information about the exercise. Example:

```
# 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.

### `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`.

### `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.

### `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.

### `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 `ctrl+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]
```

### 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*.

### 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`.

### The Exercises API

In the following we will describe the current commands that are supported within `dts exercises` and how they are used.

### 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.

### 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 `duckietown_baseline`, 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.

### 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]
```

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

```
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

```
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

```
exit
```

## Experimental Learning Experiences

This section lists the experimental learning experiences available for Duckiebots.

|                    |  |
|--------------------|--|
| What you will need | An internet connection; About 10 minutes; A computer with the Duckietown Shell command <a href="#">installed and correctly setup</a> ; |
| What you will get  | Duckietown token correctly set up;   |

## Creating a Simulator ROS Wrapper

### Note

This exercise is likely out of date

|                    |   |
|--------------------|---|
| What you will need | The simulator installed according to <a href="#">these instructions</a> |
| What you will get  | The ability to run a ROS agent using the simulator as if it was a robot |

How would we be able to exploit the powerhouse of dt-core in this simulator? By creating a ROS interface to the simulator! This will allow us to run the same code that we run on the duckiebot on the simulator.

In this exercise, you will create a ROS wrapper that maps wheel commands to actions and observations to camera images on a ROS topic.

To do so, you will leverage the skills you have obtained in the previous exercises where you used a ROS package template and created your own publisher and subscriber. This time, we encourage you to again use the ROS package template and to create a node which can both publish and subscribe to topics.

[This link](#) contains some important files that will be required to properly test your ROS wrapper. The `docker-compose.yaml` file spins up several containers at once through the simple command `docker-compose up` from the directory where the file resides. If you take a peek at the file you will see that these containers have familiar names. They are used to provide functionality to your Duckiebot, and in this scenario they are still needed to allow you to run demos in the simulator, to use the virtual joystick, etc. Inside `docker-compose.yaml` there are some lines which you will have to modify. You have to make sure that the data folder that you have downloaded from the link above is mounted on the containers so that the simulator is able to use your calibration and other configuration files.

Since now we are running our code on a fake robot (which is really our local machine) we need to modify a few things. In your `/etc/hosts` file, you will have to add the line `127.0.0.1 fakebot.local`. At the end of the Dockerfile in your ROS project (based on the Duckietown template) add the line: `ENV VEHICLE_NAME fakebot`.

Some apt packages you will need are: `freeglut3-dev, xvfb`



You will also need the `duckietown-gym-daffy` pip3 package

Finally, to ensure your publishers and subscribers parse the same ROS messages as the rest of the Duckietown pipeline, you might want to make use of `duckietown_msgs` (which is just a ROS package defined in [dt-ros-commons](#)).

Since your containers don't have a display, you will want to run these lines of bash code inside your container before running the wrapper.

```
dt-exec Xvfb :1 -screen 0 1024x768x24 -ac +extension GLX +render -noreset
export DISPLAY=:1
```

## Troubleshooting

### Troubleshooting

|            |   |
|------------|---|
| SYMPTOM    | Despite following the above instructions, when I run my container I get an error like <code>pygame.canvas.xlib.NoSuchDisplayException: Cannot connect to "None"</code>  |
| RESOLUTION | It could be that display <code>:1</code> is in use or cannot be used by the docker container. Try to change the display number to a higher number (e.g. <code>:33</code> ). Check out <a href="#">this post</a> for more details. |

## Operation - Networking

|                    |   |
|--------------------|---|
| What you will need | <ul style="list-style-type: none"><li>• A Duckiebot that is initialized according to .</li><li>• Patience (channel your inner Yoda)</li></ul> |
| What you will get  | <ul style="list-style-type: none"><li>• A Duckiebot that you can connect to and that is connected to the internet.</li></ul>                  |

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).

### 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
}
```

## 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:

```
duckiebot $ sudo curl google.com
```

which will try to download the Google homepage. If it is successful, you should see an output like:

```
<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>
```

## 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.

## 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.

### 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:
  1. Network Settings... (or run the command `nm-connection-editor`)
  2. Click "Add"
  3. Type -> Ethernet

4. Connection Name: "Shared to Duckiebot"
5. Select "IPV4" tab
6. Select Method
7. Select "Shared to other computers"
8. Click apply.

Now, you should be able to SSH to your Duckiebot:

```
$ ssh ![hostname]
```

#### **Note**

The next three commands should be executed on your Duckiebot through SSH

Check whether you can access the internet from your Duckiebot:

```
$ sudo curl google.com
```

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
```

## Mac

Untested instructions [here](#)

duckiebot-network-push = ## 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.

## Troubleshooting

I cannot ping the Duckiebot

### Troubleshooting

**SYMPTOM** I cannot ping the Duckiebot (`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.

## Troubleshooting

### SYMPTOM

When I run `ssh ![robot_name].local` I get the error `ssh: Could not resolve hostname ![robot_name].local`.

### RESOLUTION

Make sure that your Duckiebot is ON. Connect it to a monitor, a USB mouse and a keyboard.

Let's try restarting the services for the mDNS (`.local`) hostname resolution. Please run these commands on the Duckiebot:

```
$ sudo systemctl restart avahi-daemon
```

```
$ sudo reboot
```

If the issue persists, please try following these steps to ensure the service status is normal, and the configuration is correct.

(With the monitor, keyboard and mouse connected) On the duckiebot run:

```
$ sudo service avahi-daemon status
```

You should get something like the following:

```
• avahi-daemon.service - Avahi mDNS/DNS-SD Stack
Loaded:                               loaded
      (/lib/systemd/system/avahi-daemon.service; enabled;
 vendor preset: enabled) Active: active (running)
 since Sun 2017-10-22 00:07:53 CEST; 1 day 3h ago
 Main PID: 699 (avahi-daemon) Status: "avahi-daemon
 0.6.32-rc      starting      up."      CGroup:
 /system.slice/avahi-daemon.service      └─699
 avahi-daemon: running [![robot_name_in_avahi].local
 └─727 avahi-daemon: chroot help
```

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
```

## Troubleshooting

### SYMPTOM

I can SSH to the Duckiebot but not without a password

### RESOLUTION

Check the file `~/.ssh/config` and make sure you add your `ssh` key there, in case it doesn't exist.

The `init_sd_card` procedure should generate a paragraph in the above file in the following format:

```
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 exist.

## Troubleshooting

### 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
```

## Troubleshooting

### 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.

### RESOLUTION

A possible cause is that your computer's firewall is blocking

the incoming traffic from the robot. 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.

## Debug - Re-flash Microcontroller

### What you will need

- A Duckiebot of [configuration DB18](#) or above.
- A stable network connection to your Duckiebot.

### What you will get

- A flashed microcontroller (not SD card) on the HUT board, with the latest code version.

### Warning

This procedure is needed only if your Duckiebot does not recognize the HUT (Dashboard > Robot > Components). Although often unnecessary, it is safe to perform on any HUT of version **2.0** and above.

## 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 **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.

## How to flash the microcontroller - Method 1: with **dts**

On your computer

```
dts duckiebot hut_upgrade [ROBOT_NAME]
```

There are instructions on-screen to guide you through the process.

During the process, you will be asked twice to compare the command output against some expected outputs. For example,

```
avrdude: verifying ...
avrdude: 2832 bytes of flash verified

avrdude: safemode: Fuses OK (E:FF, H:DF, L:E2)

avrdude done. Thank you.

=====
====
=== (Above) command output
=====

=====
====

=== (Below) expected output
=====
=====
=====

avrdude: verifying ...
avrdude: 2832 bytes of flash verified

avrdude: safemode: Fuses OK (E:FF, H:DF, L:E2)

avrdude done. Thank you.

Did the command output match the expected output? (Y/N, default N):
```

Once confirmed, you could type 'Y' and press the "Enter" key to continue.

## How to flash the microcontroller - Method 2: step-by-step via SSH

SSH into your Duckiebot by running:

```
ssh duckie@[ROBOT_NAME].local
```

### **Note**

All of the following instructions are run on Duckiebot through the SSH terminal

Install the packages needed to compile the microcontroller firmware:

```
sudo apt-get update
sudo apt-get install bison autoconf flex gcc-avr binutils-avr gdb-avr avr-
libc avrdude build-essential
```

Clone the firmware for the microcontroller using the following command:

```
git clone https://github.com/duckietown/fw-device-hut.git
```

Navigate inside the repository you cloned :

```
cd fw-device-hut
```

Warning: read the next passages carefully. Do not just copy and paste every line of code!

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:

```
sudo cp _avrdudeconfig_jetson_nano/avrdude.conf /etc/avrdude.conf
```

**else**, if you have a Raspberry Pi based Duckiebot, use:

```
sudo cp _avrdudeconfig_raspberry_pi/avrdude.conf /etc/avrdude.conf
```

**Then**, test the `avrdude` and set the low-level configuration with:

```
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.

To complete the procedure (in all cases, whether or not a warning was issued), remove all temporary files by running:

```
make clean
```

Compile the firmware and upload it to the microcontroller:

```
make
```

The resulting output should be:

```
```none
.....

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:

```
cd .. && rm -rf fw-device-hut
```

and finally reboot the Duckiebot:

```
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.

## Debug - Duckiebot Update

|                    |   |
|--------------------|---|
| What you will need | <ul style="list-style-type: none"><li>• A Duckiebot that has been initialized</li><li>• A computer with an <b>Ubuntu OS</b>,</li><li>• Duckietown Shell, Docker, etc., as configured in <a href="#">Setup - Laptop</a>.</li><li>• Duckietown Token set up as in <a href="#">Setup - Accounts</a>.</li><li>• An internet connection to the Duckiebot, configured as in <a href="#">Operation - Networking</a>.</li></ul> |
| What you will get  | *An up-to-date Duckiebot!   |

## Understand what is the difference between OTA update and Release Update

The update method described in this page will allow you to receive an **Over 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.

## 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.

```
$ dts duckiebot update ![DUCKIEBOT_NAME]
```

You will see a prompt similar to this:

```
frank@ChudeQian-Desktop-Duckietown: ~
File Edit View Search Terminal Help
frank@ChudeQian-Desktop-Duckietown:~$ dts duckiebot update avlduck2
INFO:dts:duckietown-shell 5.1.6

dts : Problems with a command?
:
:   Report here: https://github.com/duckietown/duckietown-shell-commands/issues
:
:   Troubleshooting:
:
:   - If some commands update fail, delete ~/.dt-shell/commands
:
:   - To reset the shell to "factory settings", delete ~/.dt-shell
:
:   (Note: you will have to re-configure.)
INFO:dts:Commands version: daffy-new-deal
INFO:duckietown-challenges:duckietown-challenges 5.1.5
INFO:zj:zuper-lpce 5.3.0
INFO:zuper-typing:zuper-typing 5.3.0
INFO:zuper-commons:zuper-commons 5.0.11
INFO:dts:duckietown-shell-commands 5.0.2
INFO:dts:Fetching software status from your Duckiebot...
INFO:dts:Found 14 Duckietown software modules. Looking for updates...

duckietown/dt-code-api:daffy-arm32v7:          update available
duckietown/dt-core:daffy-arm32v7:             update available
duckietown/dt-duckiebot-interface:daffy-arm32v7: update available
duckietown/dt-car-interface:daffy-arm32v7:     update available
duckietown/dt-rosbridge-websocket:daffy-arm32v7: update available
duckietown/dt-ros-commons:daffy-arm32v7:       update available
duckietown/dt-device-proxy:daffy-arm32v7:      update available
duckietown/dt-device-health:daffy-arm32v7:     update available
duckietown/dt-files-api:daffy-arm32v7:         update available
duckietown/dt-device-online:daffy-arm32v7:     update available
duckietown/dt-system-monitor:daffy-arm32v7:    update available
duckietown/dt-commons:daffy-arm32v7:          update available
duckietown/dt-base-environment:daffy-arm32v7:  update available
duckietown/dt-device-dashboard:daffy-arm32v7:  update available

WARNING:dts: 14 module(s) will be updated.
Do you confirm? [y]: █
```

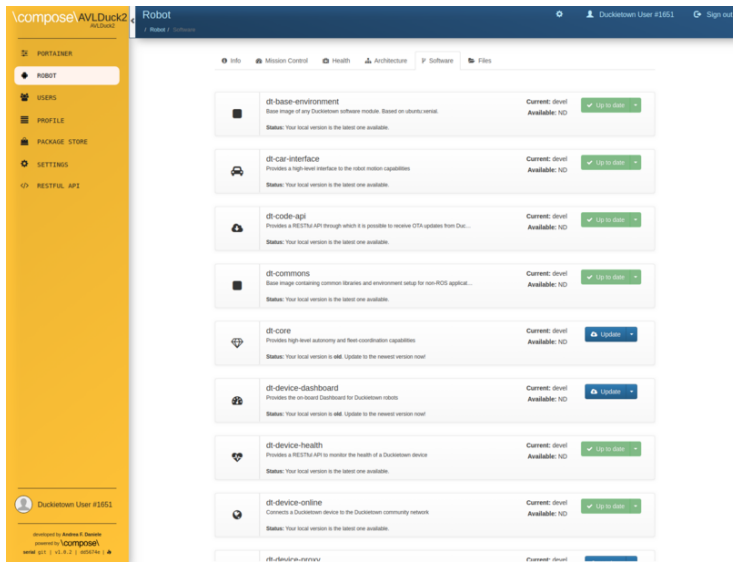
*Fig. 61* Auto Update Duckiebot Container

Type in `y` for yes to continue updating. If you would like to abort, you can use `Ctrl - C` to stop the update.

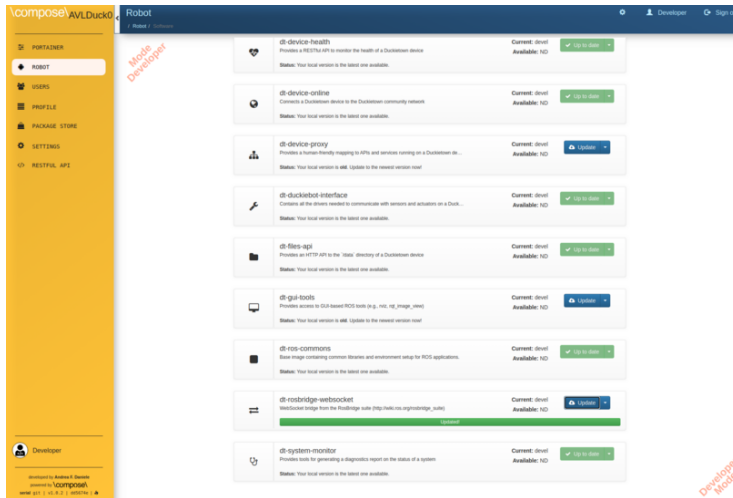
Note: This process is expected to take a while to complete.

## Update Duckiebot container using Dashboard

To use the Dashboard for updating the Duckiebot container, first navigate to the Dashboard software page. If you are not sure how to do that, refer to the [Setup - Duckiebot Dashboard](#) page. If you have containers that need to be updated, you will see the update button next to the container:



The update will proceed upon click. If update is successful, you will see the image below:



## Requesting a Support Connection

### When and why should I run this procedure?

If your Duckiebot ends up in a unique state that is difficult to debug over Slack or other help channels, a Duckietown support engineer may ask you to open a remote support connection.

This connection allows the Duckietown team to take logs from your Duckiebot that will help track down any issues.

The connection will only remain open for as long as you leave the command running.

#### **Warning**

Because this process enables access to your Duckiebot, you should only follow these steps **when asked by a Duckietown support engineer**.

It is also important to ensure that the connection is closed after taking the logs - you will agree ahead of time on a time to open and complete the support process.

### Step 1: Create a request

When instructed to do so, you can create a support request by running

```
dts duckiebot support request <your_robot>
```

This will result in the following message:

```
=====
Your Duckiebot is now starting a support tunnel.
Leave this command running until debugging is finished.

> Support ID - your_robot.support_address_will_appear_here

Press Enter when you are ready to exit.
=====
```

### ! Attention

You will notice that the running process does not exit at this point. It is important to leave it running in your terminal until instructed to close the support connection.

## Step 2: Send the request address

Send the address corresponding to `your_robot.support_address_will_appear_here` in **fig:support-code** to Duckietown support. They will use this address to retrieve the logs from your Duckiebot.

## Step 3: Close the support connection

Once the debugging process is finished, the Duckietown support engineer will let you know that you can close the connection.

Simply exit the command using `Ctrl+C` in your terminal.

## Reset Dashboard

Sometimes the Dashboard's database can get corrupted due to an improper shutdown of the robot. This can have different kinds of effects on the usability of the dashboard, going from flat out **404** errors, to **bad gateway**, to **permission errors**. In many cases, a hard-reset of the database is enough for the problem to get fixed.

### Reset database

#### ! Warning

Resetting the database will delete all custom data from the dashboard and you will be asked to perform the first setup again.

The dashboard stores all its databases inside a Docker volume. Use the following commands to force a reset by removing such volume,

```
docker -H ROBOT_NAME.local stop dashboard
docker -H ROBOT_NAME.local rm dashboard
docker -H ROBOT_NAME.local volume rm duckietown_compose-data
dts duckiebot update ROBOT_NAME
```

The last step will remake the dashboard container and regenerate a database in the process.

## Version Control with Git

### Background reading

See: [Github tutorial](#)

See: [Github workflow](#)

## 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).

##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]"
```

## Git tips

### 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.

### 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.

### Move between branches

You can move to a different branch using the command,

```
git checkout ![destination-branch]
```

### 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

```
git branch
git status
```

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).

### 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]
```

## 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.

## Delete branches

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

```
git branch -d ![branch-name]
```

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

```
git push origin --delete ![branch-name]
```

## 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.

## 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:

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

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

## 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.

## Git troubleshooting

### 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
```

### 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]
```

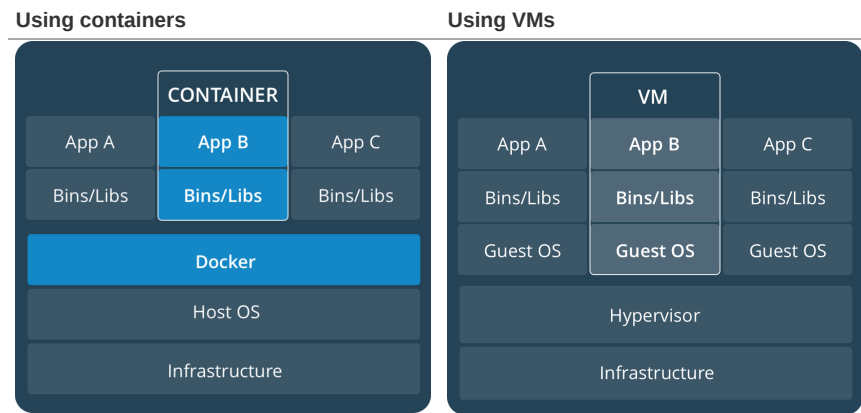
## Docker Basics

### 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.

## 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.

## 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:

```
docker image prune
```

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.

## 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:

```
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.

```
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]
```

## Running images

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

[Table of Docker Command Flags](#) shows a summary of the options we use most often in Duckietown. Below, we give some examples

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

Table 3 Table of Docker Command Flags

#### **Note**

Most of this is hidden from the Duckietown user when running commands because it is contained within the Duckietown Shell commands.

#### Examples

Set the container name to `joystick`:

```
--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
```

## Other useful commands

### 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.

### 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:

```
docker volume create portainer_data
docker run -d -p 9000:9000 --name portainer --restart always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer
```

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.

## 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's official Get Started tutorial](#);
- [Docker Curriculum](#);
- *Docker Deep Dive*, by Nigel Poulton.

## Docker Troubleshooting

### Troubleshooting

|            |  |
|------------|--|
| SYMPTOM    | I got a permission denied error while trying to connect to the Docker daemon socket                                    |
| RESOLUTION | If this is on your laptop, that means when you set up your environment you did not grant your user account right to do |

environment 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 - it should now be fixed.

### Troubleshooting

**SYMPTOM** The container I am trying to run does not start - `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

### Troubleshooting

**SYMPTOM** Docker exits with the message `tls: oversized record received`

**RESOLUTION** 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 `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 [here](#).

### Troubleshooting

**SYMPTOM** I can't run a container because I get `exec user process caused "exec format error"`

**RESOLUTION** Despite not being very descriptive, an error like this 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 endpoint of the container by adding `--entrypoint=qemu3-arm-static` to options when running it.

## Secure shell (SSH)

Requires: Time: 5 minutes.

Results: You will know about some useful shortcuts.

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.

## SSH aliases

Instead of using

```
ssh duckie@[ROBOT].local
```

You can set up SSH so that you can use:

```
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:

```
Host ![ROBOT]
  User duckie
  Hostname ![ROBOT].local
```

Note that this does **not** let you do

```
ping ![ROBOT]
```

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

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

## Handling circuits and batteries

What you will need

- Nothing

What you will get

- Preliminary knowledge of circuits and power source properties useful in Duckietown

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

### The Duckiebattery (DB-C-DBatt)

This battery has been the standard battery for the Duckiebots since 2021. For example, models `DB21M`, `DB21J`, and `DBR4` use this battery.

The Duckiebattery is special because it is a programmable, smart battery, designed and manufactured specifically for Duckiebots. Using this battery will allow your Duckiebot to monitor the state of charge and other diagnostics, and shutdown via software. Moreover, it enables advanced features such as auto-charging in Autolabs, because it will guarantee power to the onboard computer when being plugged in or out of charge.



**Fig. 62** The Duckiebattery.

### 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), max 2.5A on a single port
- Charge time: 0-100% takes about 5h and 0-90% about 4h with a 2A power supply
- Weight: 189g (fully charged)

#### **⚠ Caution**

Lithium-ion batteries like the Duckiebattery are potentially dangerous and must be handled with care.

Here are some things to do and not to do.

#### **⚠ Warning**

Keep reading the following safety precautions: do not skip this section.

### Handling: things to do

#### **➡ Things to do**

- Dispose of the battery pack immediately if it has been subject to moisture and/or the case is eminently damaged.
- In case of fire use a CO2 extinguisher.
- Store preferably in a cool, dry, and ventilated area subject to moderate temperature change.
- Storage at high temperatures (>50 C) should be avoided.

### Handling: things not to do

#### **⚠ Things not to do**

- 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 its 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 it to direct sunlight for long periods.



## LED description

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



**Fig. 63** LEDs indicate the state of charge of the Duckiebattery.

### **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.



**Fig. 64** Wake up the battery by pressing the button once.

## Charge the battery

After setting the battery in *idle* mode, charge it by connecting a 5V 2A power adapter. Note that using a higher amperage charger will not damage the unit. The LEDs will be flashing at 1 Hz, showing the battery is receiving charge.



**Fig. 65** Charging the Duckiebattery.

When the battery's state of charge is particularly depleted (e.g., as soon as you receive the battery), the LEDs might be unresponsive for up to 30 minutes while receiving charge.

## 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 cells from low-voltage discharge.

When a certain low cell voltage level is detected, the battery microcontroller, 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 being charged. This does not mean the battery is dead, just “hibernating”.

## USB outputs

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



**Fig. 66** 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.

## 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 is 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.

### **Troubleshooting**

#### SYMPTOM

My battery does not charge.

#### RESOLUTION

There can be several reasons why a charge is not being accepted. Below are the most common issues:

- The input voltage is too low or too high. Make sure you 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.

## Troubleshooting

### SYMPTOM

One or both USB outputs are not working

### RESOLUTION

There can be several reasons why the USB output is not working. Below are the most common issues:

- 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 more than 30 minutes 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.
- An 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.

## The Duckie-power-bank

The Duckie-power-bank (or Duckiebattery version 1) is the standard power source for Duckiebots in **DB18** and **DB19** configurations. Duckie-power-banks are easily recognizable:



**Fig. 67** The Duckie Power Bank is the first version of the Duckiebattery, used in **DB18** and **DB19** Duckiebots.

The Duckiebattery is equipped with 2 USB type A outputs (ports A and B) and 1 Micro USB connector for charging.



**Fig. 68** The Duckie Power Bank ports.

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:

| LED | State of Charge |
|-----|-----------------|
| D1  | 3-25%           |
| D2  | 25-50%          |
| D3  | 50-75%          |
| D4  | 75-100%         |

**Table 4** Duckiebattery LED charge indicators

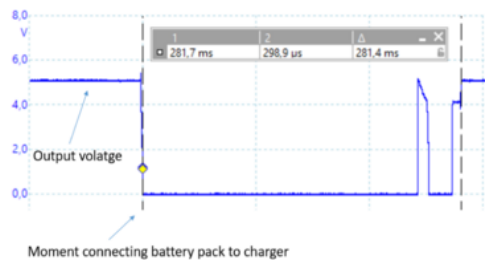
If D1 is flashing (0.5Hz) while not being charged, the battery pack is at a critical low charge (less than 3%).

### 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 ports A and B will turn off for around 280ms:



**Fig. 69** The Duckie Power Bank output voltage drops when plugging in and removing the charger.

This is an unwanted effect that 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 warring of the Raspberry

Pi. This will put the Raspberry Pi in a throttling mode limiting its performance.

## Discharging

The output ports A and B have an unloaded output voltage of around 5.1V. To turn the outputs on simply plug in a load on port A/B, for example, a Raspberry Pi and a Duckietown Hut, 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:

| <b>Load</b> | <b>Efficiency</b> | <b>Autonomy</b> |
|-------------|-------------------|-----------------|
| 1 A         | 91%               | 6h 44m          |
| 1.5 A       | 88%               | 4h 33m          |
| 2 A         | 85%               | 3h 09m          |
| 2.5 A       | 79%               | 2h 21m          |

*Table 5* Duckie Power Bank discharge statistics