Duckieracer Guide

Hardware Assembly

Shopping List

- Jetson Nano Developer Kit
- Complete Duckiebot chassis including wheels and motors
- 64GB micro SD card
- Raspberry Pi V2 Camera Module with 160° wide angle lens attachment
- Standard Duckiebot Hut
- 5V 3A battery pack (standard Duckiebot battery is fine)
- 8265.NGWMG.NV Intel WiFi card
- 2x Molex 2042811100 Wifi antenna
- Noctua NF-A4x20 5V PWM fan (or any other 5V fan)
- Assorted screws and jumper cables

Assembly and Notes [TODO: Take pictures for each step]

- Mount the motors and the camera mount to the chassis according to standard Duckiebot procedure.
- Insert the Wifi card and antennas into the Jetson Nano.
- Attach the Jetson Nano to the top plate of the chassis, with the slot for the camera cable facing the camera mount.
- Assemble the complete chassis and feed the LED and motor cables through suitable spaces in the chassis.
- Attach the wide angle lens to the camera, screw the camera to the camera mount, and attach the connector cable to both the camera and the Nano (pay attention to the orientation).
- Insert the Duckiebot Hut onto the GPIO pins of the Nano (the only possible orientation is the correct one). Connect the motor and LED cables.
- Slide the battery pack into the open space inside the Duckiebot chassis (this can be a bit tricky depending on the size of the battery).
- Screw the fan to the heat sink of the Nano and secure the WiFi antennas, using the sticker on the back, in a way which is convenient.

Software Setup

Prerequisites

For the software setup you will need:

- 64GB micro SD card
- An Ubuntu OS laptop or desktop
- An Internet connection

Setup

Flash the JetPack 4.2.3 image onto an 64GB SD card (do not use the newest image, the camera pipeline does not work with it yet). It can be found under "https://developer.nvidia.com/jetson-nano-sd-card-image-r3223" or in the JetPack software archive: "https://developer.nvidia.com/embedded/jetpack-archive"

Attach a monitor, keyboard, and mouse to the Jetson Nano.

Insert the SD card into the Nano and boot it. Upon first boot you will be able the choose name, region, keyboard layout etc. Follow the configuration steps until you arrive at the desktop.

Create the default calibration files on the bot:

ssh into your bot or access it locally and run the following commands:

```
$ cd data/
$ mkdir config
$ cd config/
$ mkdir calibrations
$ cd calibrations/

$ mkdir camera_extrinsic
$ cd camera_extrinsic/
$ sudo vim default.yaml
# copy/paste this file into the new one:
https://github.com/duckietown/duckietown-shell-
commands/blob/daffy/init_sd_card/calib_cam_ext_default.yaml

$ cd -  #should reroute you to /data/config/calibrations
$ mkdir camera_intrinsic
$ cd camera_intrinsic/
$ sudo vim default.yaml
# copy/paste this file into the new one:
https://github.com/duckietown/duckietown-shell-
commands/blob/daffy/init_sd_card/calib_cam_int_default.yaml

$ cd -
$ mkdir kinematics
$ cd kinematics/
$ sudo vim default.yaml
# copy/paste this file into the new one:
https://github.com/duckietown/duckietown-shell-
commands/blob/daffy/init_sd_card/calib_kin_default.yaml
```

You should now have three folders in your calibrations folder, each containing one file called default.yaml. Check that this is correct.

The following steps can either be completed locally or via ssh:

Install pip, pip3, and cython:

```
$ sudo apt update
$ sudo apt install python3-pip
$ sudo apt install python-pip
$ sudo pip3 install cython
```

To be able to access the GPU from within containers we need to update Docker to 19.03.:

```
($ sudo apt update)
$ sudo apt install curl
$ curl -sSL https://get.docker.com/ | sh
$ sudo docker version
```

The last command should (among other things) display "Version: 19.03.".

Pull the Portainer image and run it.

```
$ docker pull portainer/portainer
$ docker volume create portainer_data
$ docker run -d -p 9000:9000 -p 8000:8000 --name portainer --restart always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer
```

Enable remote access to the Docker daemon:

```
$ sudo mkdir -p /etc/systemd/system/docker.service.d
$ sudo vim /etc/systemd/system/docker.service.d/options.conf
```

Copy/Paste the following into the file, make sure it looks the same, save and exit.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H unix:// -H tcp://0.0.0.0:2375
```

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

Set up the GSTREAMER pipeline using v4l2loopback:

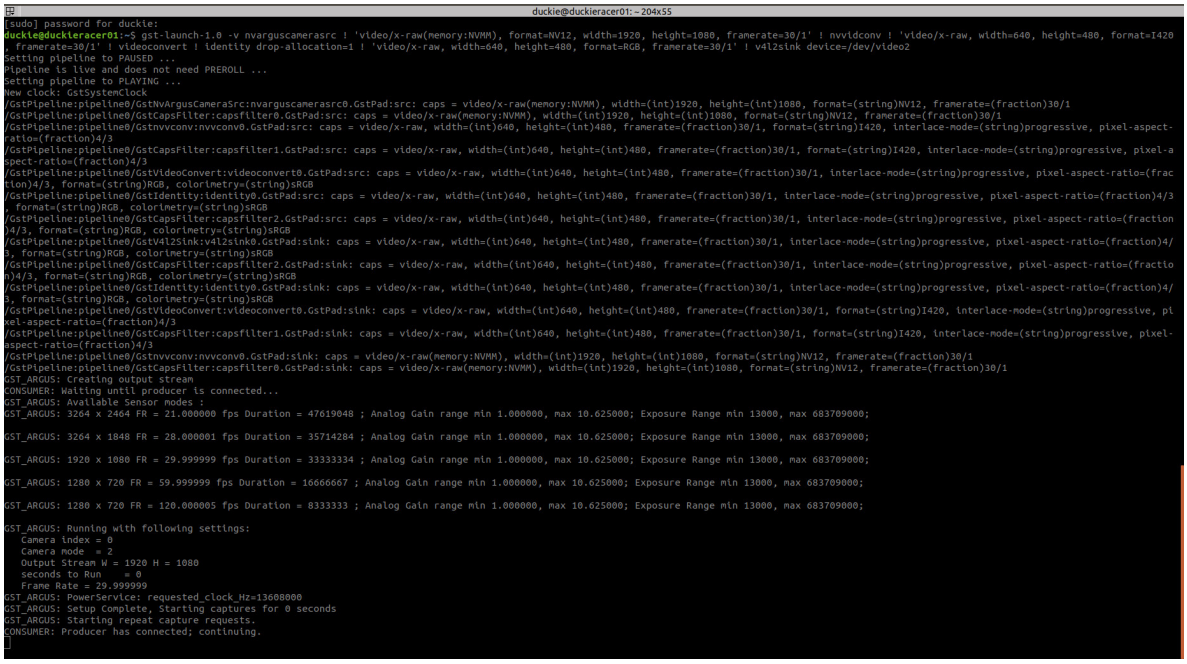
```
$ sudo su
$ cd /usr/src/linux-headers-4.9.140-tegra-ubuntu18.04_aarch64/kernel-4.9
$ mkdir v4l2loopback
$ git clone https://github.com/umlaeute/v4l2loopback.git v4l2loopback
$ cd v4l2loopback
$ make
$ make install
$ apt-get install -y v4l2loopback-dkms v4l2loopback-utils
$ modprobe v4l2loopback devices=1 video_nr=2 exclusive_caps=1
$ echo options v4l2loopback devices=1 video_nr=2 exclusive_caps=1 >
/etc/modprobe.d/v4l2loopback.conf
$ echo v4l2loopback > /etc/modules
$ update-initramfs -u
```

Check that it works:

In one terminal run:

```
$ gst-launch-1.0 -v nvarguscamerasrc ! 'video/x-raw(memory:NVMM), format=NV12,
width=1920, height=1080, framerate=30/1' ! nvvidconv ! 'video/x-raw, width=640,
height=480, format=I420, framerate=30/1' ! videoconvert ! identity drop-
allocation=1 ! 'video/x-raw, width=640, height=480, format=RGB, framerate=30/1'
! v4l2sink device=/dev/video2
```

It should look something like this:



Leave this terminal open.

To check if your pipeline is working, in a new terminal on the Nano, run:

```
$ gst-launch-1.0 v4l2src device=/dev/video2 ! 'video/x-raw, width=640,
height=480, format=RGB, framerate=30/1' ! videoconvert ! xvimagesink
```

It should open a window with a live image of your camera stream. This command can always be used locally to check if your camera feed is up and running.

Inspect the image of the stream. Can you spot a pink tint around the edges of the feed? If so, close the stream window and run these commands:

```
$ wget https://www.waveshare.com/w/upload/e/eb/Camera_overrides.tar.gz
$ tar zxvf Camera_overrides.tar.gz
$ sudo cp camera_overrides.isp /var/nvidia/nvcam/settings/
$ sudo chmod 664 /var/nvidia/nvcam/settings/camera_overrides.isp
$ sudo chown root:root /var/nvidia/nvcam/settings/camera_overrides.isp
```

This adds a filter into the default settings of the camera which fixes the problem. Although the tint may not seem significant in itself, it can cause significant problems in lane following after the camera feed is run through the anti-instagram filter.

Add the pipeline to be started on boot

Locally on the Nano, run

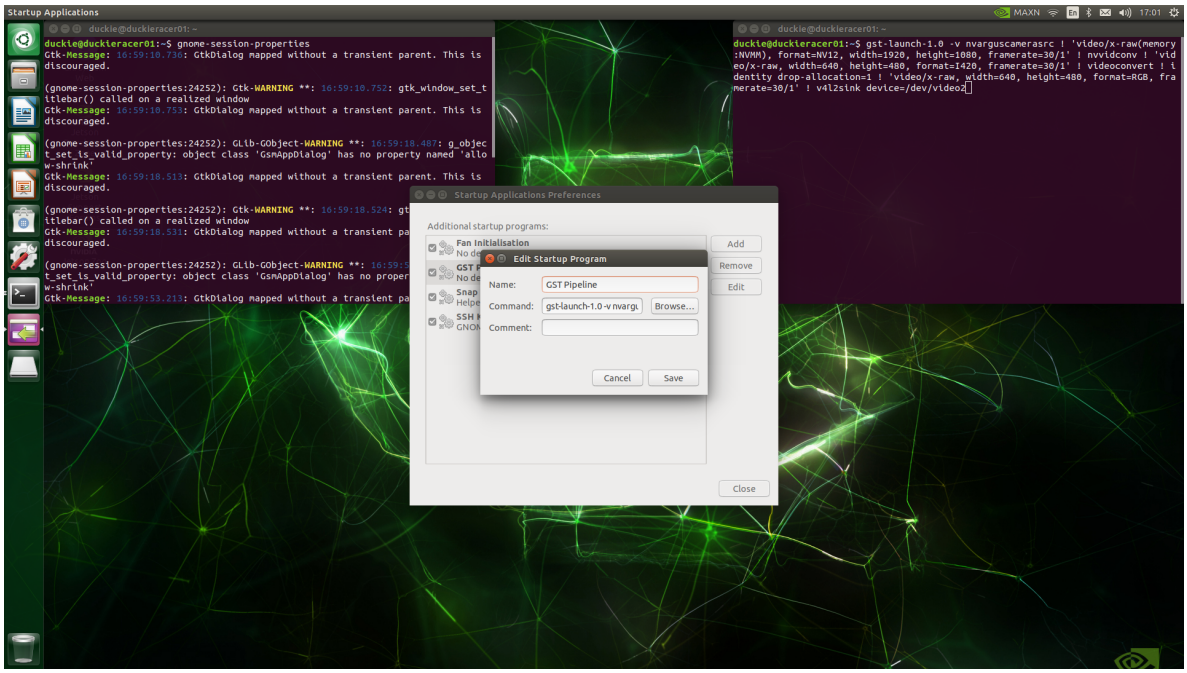
```
$ gnome-session-properties
```

It should open a window where you can add commands to be run on boot.

Click on `Add` and enter the below GST Pipeline command into the *Command:* window.

```
gst-launch-1.0 -v nvarguscamerasrc ! 'video/x-raw(memory:NVMM), format=NV12,
width=1920, height=1080, framerate=30/1' ! nvvidconv ! 'video/x-raw, width=640,
height=480, format=I420, framerate=30/1' ! videoconvert ! identity drop-
allocation=1 ! 'video/x-raw, width=640, height=480, format=RGB, framerate=30/1'
! v4l2sink device=/dev/video2
```

Click `Save` to finish.



Pull the necessary ROS images:

```
$ sudo docker pull ros:kinetic-ros-base-xenial
$ sudo docker pull duckietown/dt-ros-kinetic-base
$ sudo docker pull duckietown/dt-ros-commons
```

Pull the unmodified dt containers:

```
$ sudo docker pull duckietown/dt-car-interface:daffy-arm32v7
$ sudo docker pull duckietown/dt-core:daffy-arm32v7
```

Install the dts and its dependencies:

(This step is not absolutely necessary as everything can also be built remotely, however I find it useful to have.)

```
$ sudo apt install -y git git-lfs
$ sudo adduser `whoami` docker
$ pip3 install --no-cache-dir --user -U duckietown-shell
```

Pull and build the racer version of dt-duckiebot-interface:

```
$ git clone https://github.com/carmen-sc/dt-duckiebot-interface.git
$ cd dt-duckiebot-interface
$ dts devel build -f
```

Running Software on the Duckieracer

When running software on the Duckieracer it is important to keep a few things in mind:

- The Nano tends to run hot pretty quickly and will throttle itself. To avoid this activate the fan on the heat sink using the `$ sudo jetson_clocks` command. It always runs on max speed and I have never had issues with heat while it was running. To turn it off simply call the same command again.
- The Nano has two power modes; a 5W and a 10W mode. You can switch modes either on the desktop by clicking the NVIDIA icon on the top right, or via command line with the `$ sudo nvpmodel -m 1` or `$ sudo nvpmodel -m 0` commands, for 5W and 10W respectively.
- If at any point in time the Nano tries to pull more power than the battery pack can provide, it will not throttle but instead turn off. As such you need to pay attention to your battery and preferably only operate on the 10W mode if your battery isn't running low on charge.
- Per default, Docker containers may not utilize the GPU of the Jetson board, only the CPU. To enable GPU access, add the `--gpus all` flag to the container launch.

Initialization

After you have completed the software setup, it makes sense to run through these steps to see if everything initialized correctly and to set up the containers.

If you have installed a fan, run

```
$ sudo jetson_clocks
```

If you haven't rebooted since software setup, initialize the GST pipeline:

```
$ gst-launch-1.0 -v nvarguscamerasrc ! 'video/x-raw(memory:NVMM), format=NV12,
width=1920, height=1080, framerate=30/1' ! nvvidconv ! 'video/x-r
aw, width=640, height=480, format=I420, framerate=30/1' ! videoconvert !
identity drop-allocation=1 ! 'video/x-raw, width=640, height=480, format=RGB,
framerate=30/1' ! v4l2sink device=/dev/video2
```

Run dt-duckiebot-interface, dt-car-interface, and dt-core. Include the flag `--gpus all` on dt-core. Check the logs of all containers to look for errors.

Open GUI tools to check if the camera pipeline is working and all ROS topics are publishing correctly.

```
$ dts start_gui_tools <DUCKIERACER_HOSTNAME>
        $ rqt
```

Open keyboard control to see if the motors are running and connected correctly. Try running basic lane following.

```
$ dts duckiebot keyboard_control <DUCKIERACER_HOSTNAME>
```

Follow the standard intrinsic camera calibration procedure. (https://docs.duckietown.org/daffy/opmanual_duckiebot/out/camera_calib.html)

Running Software

It should be possible to run any Duckietown demo on the Duckieracer. It is important to remember to include the flag `--gpus all` after the dts command on all containers you want to be able to access the GPU.

Troubleshooting

A list of known problems and fixes:

Problem: Nothing related to Docker is running / Portainer does not work.
Fix: Docker does not always start properly on boot. Check if the Docker daemon is running with `$ systemctl status docker`. If it isn't flagged as "active" run `$ sudo systemctl restart docker`. If it is active check if the Portainer container is up and running with `$ docker ps`.

Problem: rqt_image_view displays a still image (not a feed).
Fix: Stop all dt containers and the GST pipeline. Then run the `$ sudo systemctl restart nvargus-daemon` command. Re-initialize the GST pipeline. Start dt-duckiebot-interface. Start everything else.

Problem: The nano is very slow.
Fix: Check if the fan has been activated. Without it, the Nano will run hot very fast and throttle itself. to activate it run the `$ sudo jetson_clocks` command.