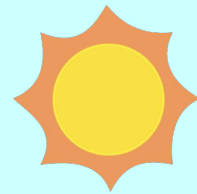
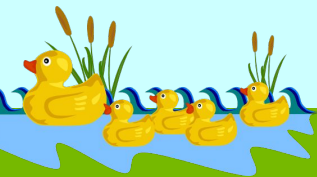
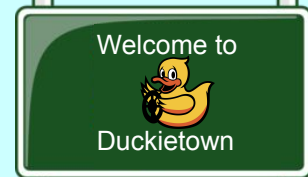


Duckietown



Software Infrastructure
for Autonomous Robotics



Considine
Breandan

Hristov
Ruslan

Hello Duckietown!

- Low-cost platform for autonomy education and research
- Robotics and machine learning for students of all ages
- Teaches concepts in perception, planning, and control with miniature autonomous vehicles in the classroom





Watchtower

April Tag

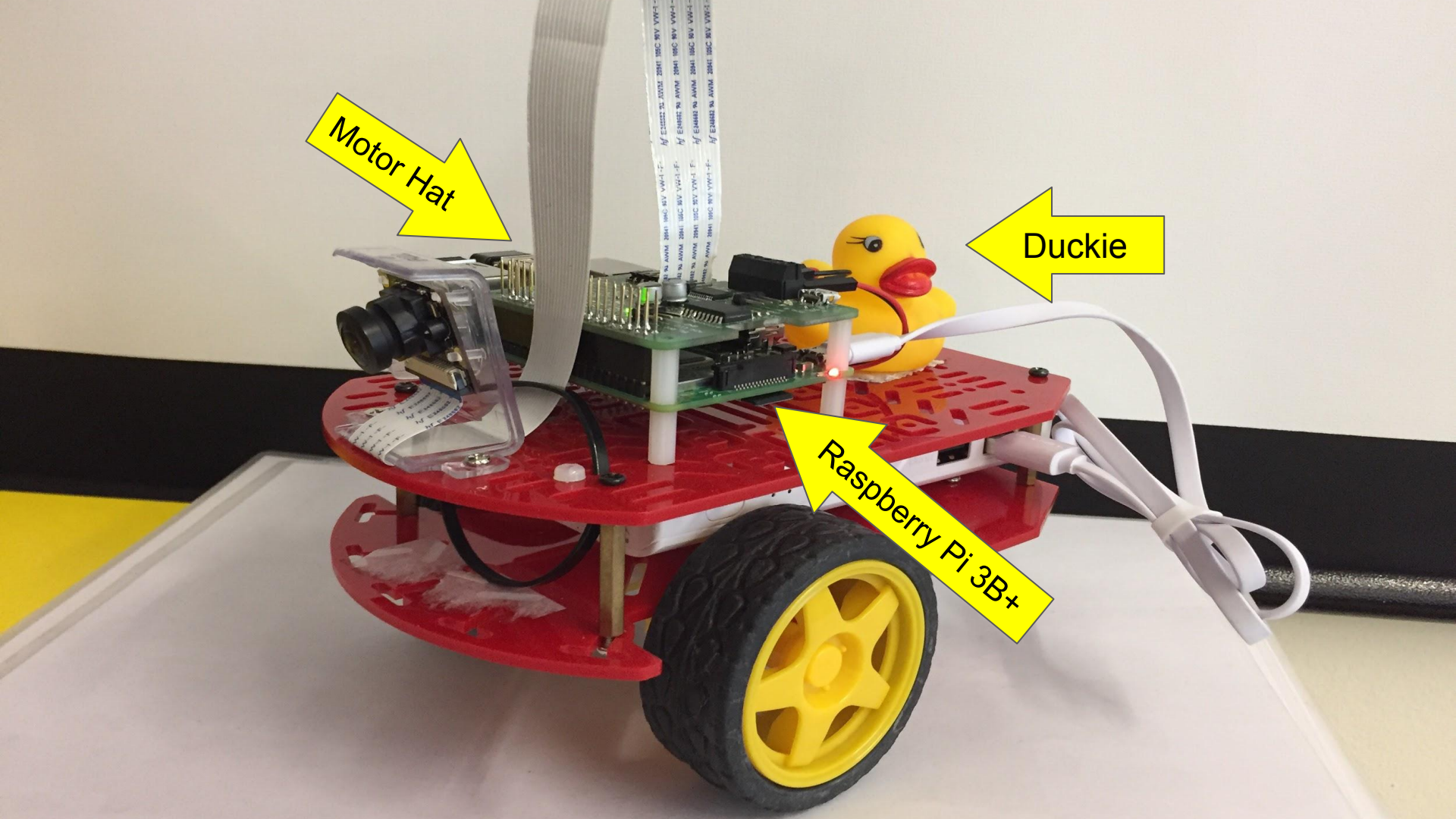
Traffic camera

Duckiebot

Motor Hat

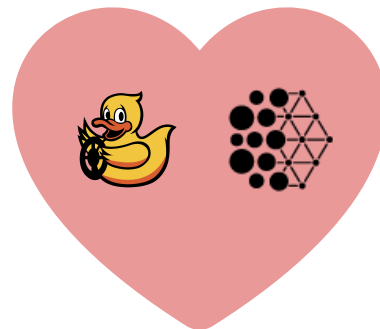
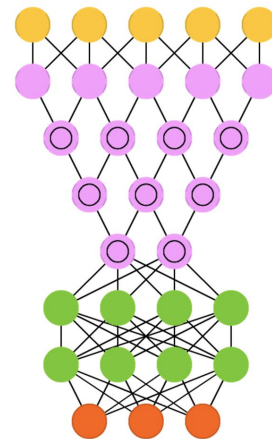
Duckie

Raspberry Pi 3B+



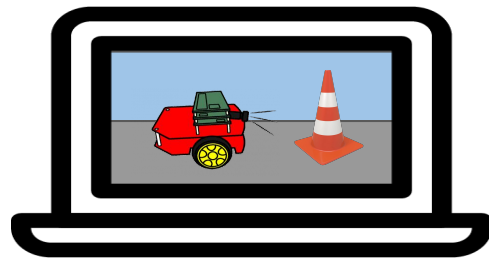
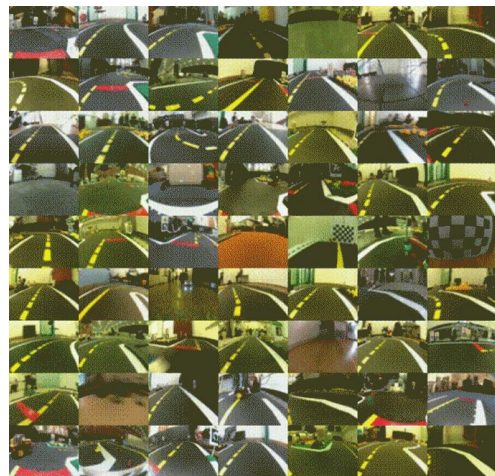
Duckiebots and Deep Learning

- Duckiebots are battery-powered vehicles
 - Camera is the only sensor
 - Classic computer vision pipeline
 - Multiple stages of hand-designed image processing
 - Proportional integral derivative (PID) controller
 - Each robot needs to be individually calibrated
- Mila is a deep learning research lab
 - Students interested in applying DL & RL
 - Teach students to teach robots how to drive



A Tale of Two Duckietowns

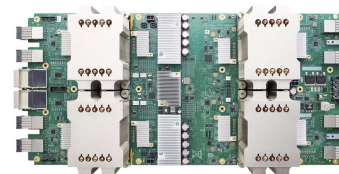
- Gathering real data is real slow
 - Time consuming and tedious to collect
 - Real robots break down, drain batteries
 - Even worse if you need a diverse dataset
 - Need to vary lighting conditions, etc.
- Simulation is an appealing alternative...
 - Can produce arbitrary amounts of data
 - Easy to augment with generated data
 - Simplifies reproducibility and verifiability



Machine learning is a changin'

- Evolving tools, frameworks, and languages
- Evolving domain models and architectures
- Evolving hardware technology and platforms

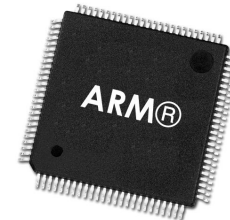
Cloud TPU



Top Frameworks



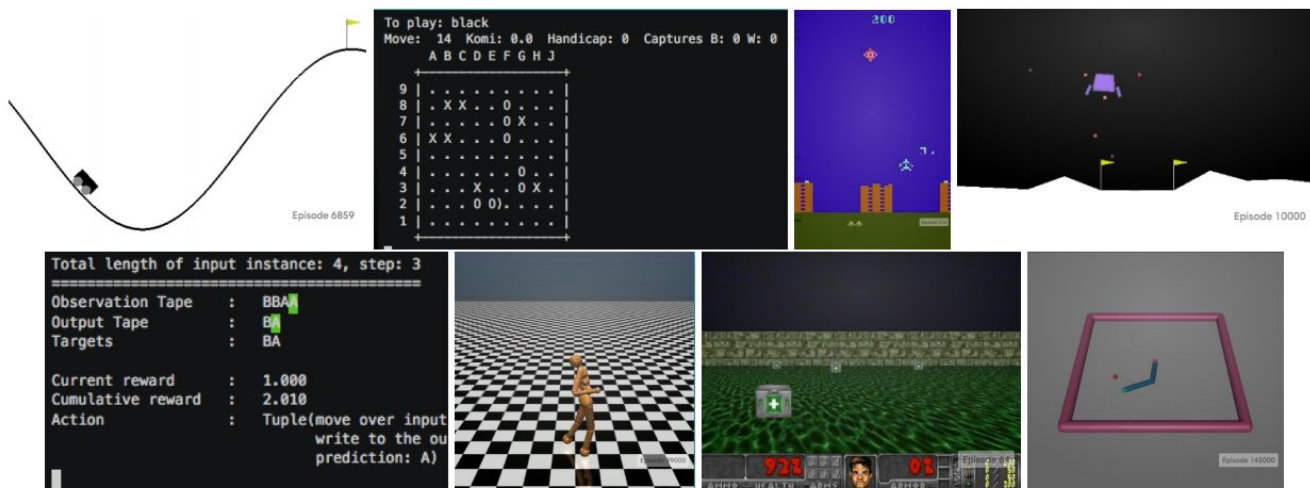
Programming languages



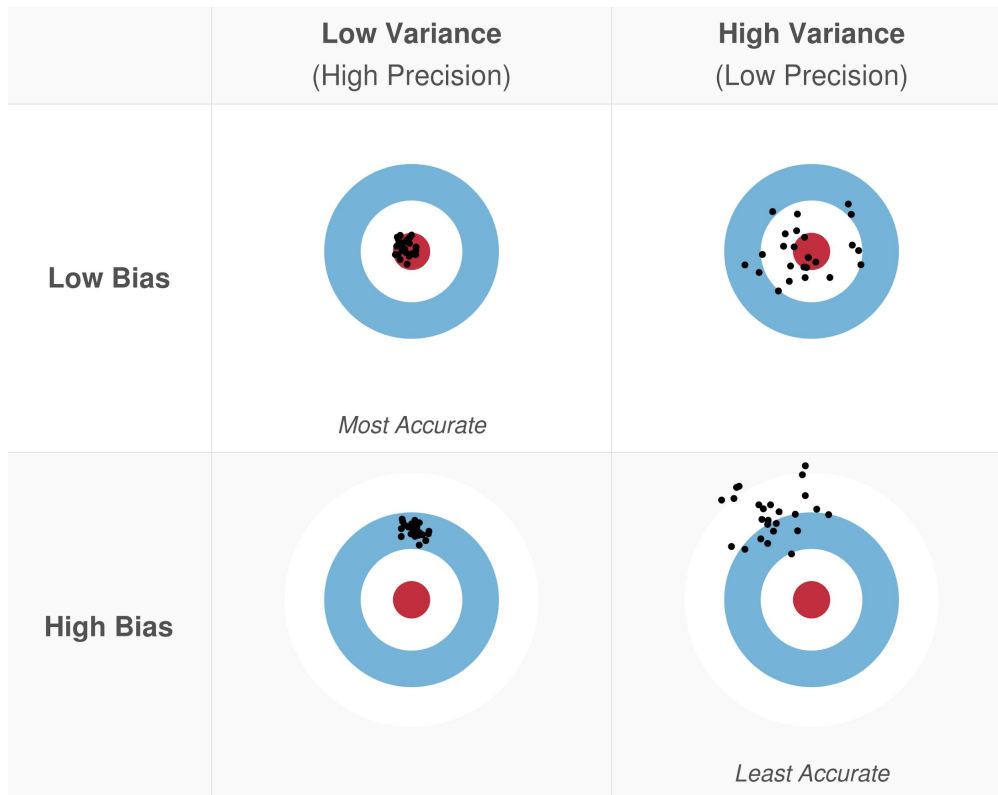
Simulators and environments

Brockman, Greg, et al. "Openai gym." arXiv:1606.01540 (2016).

<https://arxiv.org/pdf/1606.01540.pdf>



Problem: Bias-Variance and Overfitting



Software is a changin'

- Karpathy, Andrej. **"Software 2.0."** (2017).

<https://medium.com/@karpathy/software-2-0-a64152b37c35>

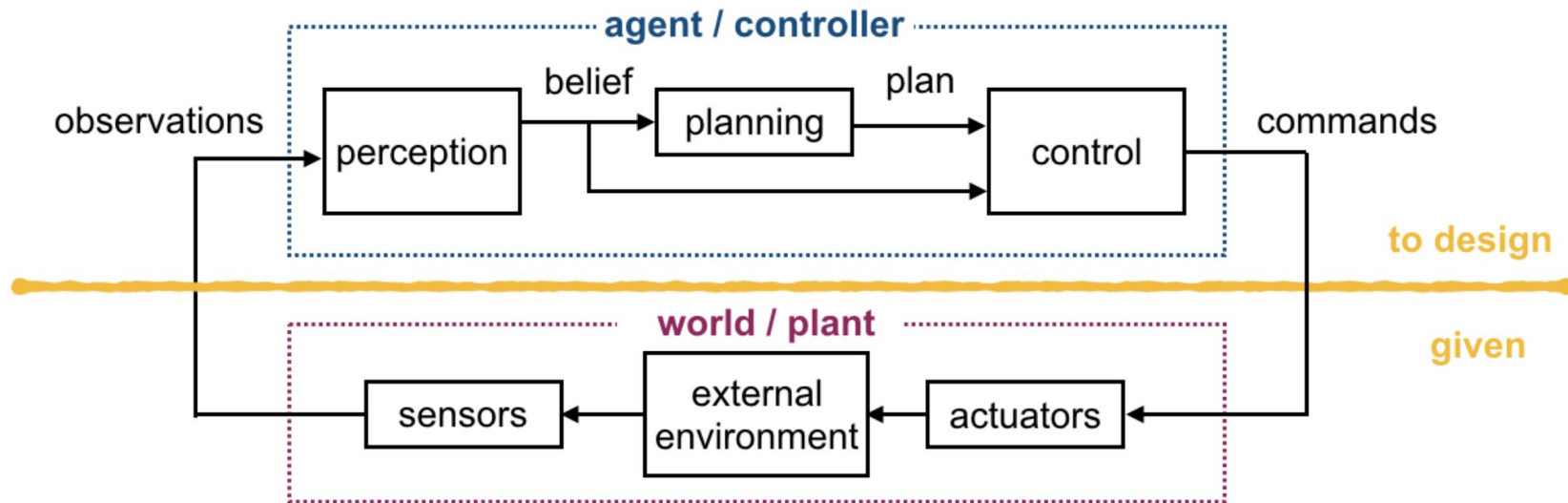
Benefits

- Computationally homogeneous
- Portability & runtime characteristics
- Predictable latency/accuracy tradeoffs
- Modularity, portability, agility

(Current) Limitations

- Low interpretability
- Unintuitive failure modes
- Software stack is immature
- Difficult to train and test

Robotics is a changin'

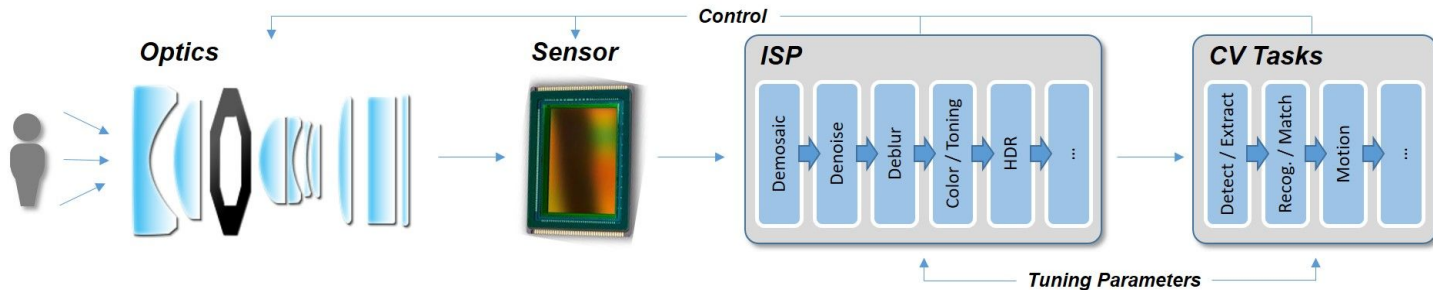


Idealized Agent



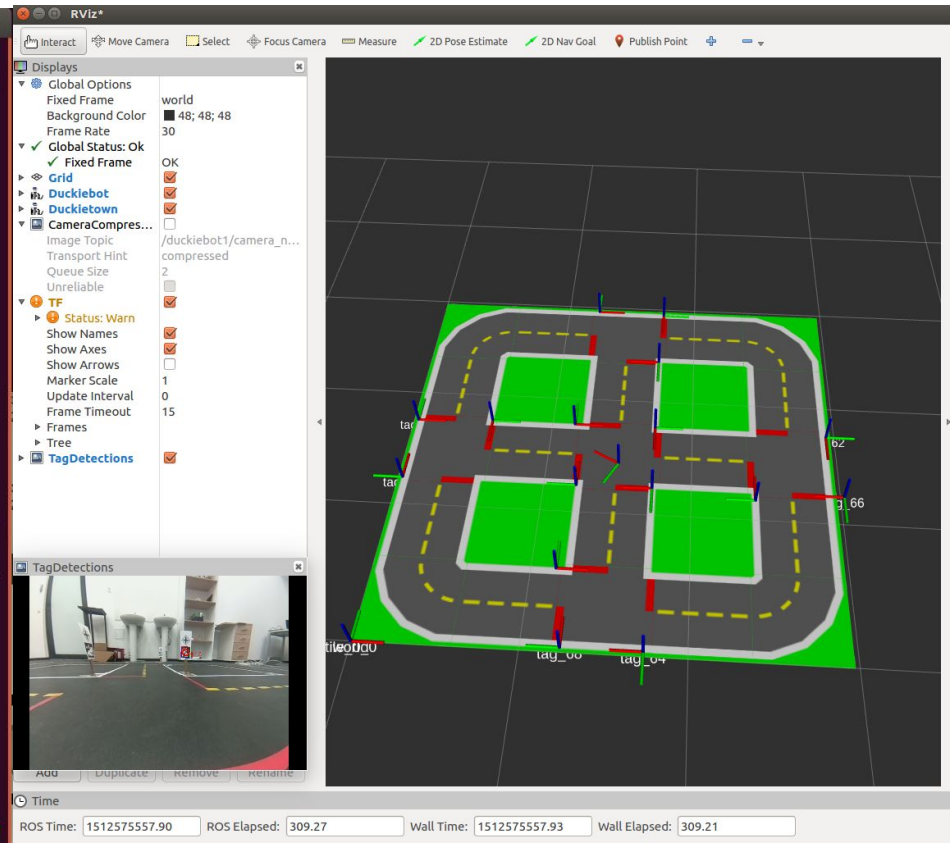
Environment

Reality



ROS / Duckietown Tools (rviz, rqt_*, sh)

```
niklasstolz@duckietown10: ~  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
e  
153 packages can be updated.  
0 updates are security updates.  
Last login: Wed Jun 13 13:01:33 2018 from 192.168.1.195  
megaduck@megabot02:~$  
18 packages can be updated.  
1 update is a security update.  
Last login: Wed Jun 13 09:29:15 2018 from 192.168.1.63  
megaduck@megabot03:~$  
135 packages can be updated.  
0 updates are security updates.  
Last login: Wed Jun 13 13:01:52 2018 from 192.168.1.195  
megaduck@megabot16:~$  
[xpanes-151:02-15682* "duckietown10" 13:02 13-Jun-18
```



Let's try to make installs more repeatable

Step 1. Partition hard drive

Step 2. Install Ubuntu

Step 3. Install ROS

Step 4. Install Python stuff

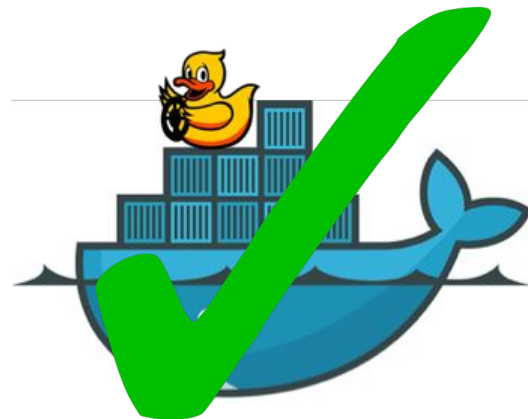
Step 5. Configure network

...

Step 98. `source environment.sh`

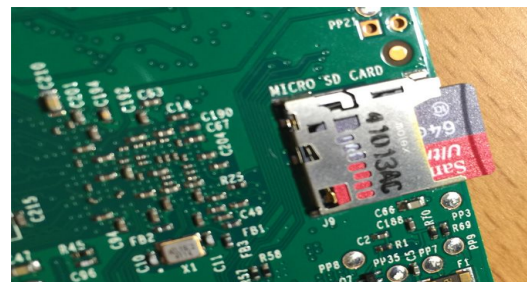
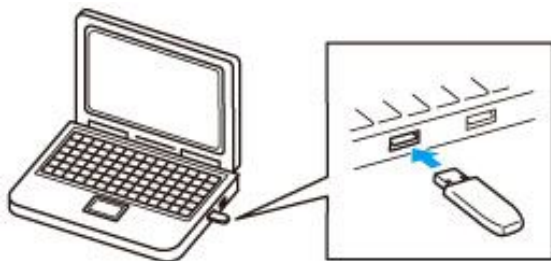
Step 99. `catkin_make -C ...`

```
$ dts init_sd_card
```

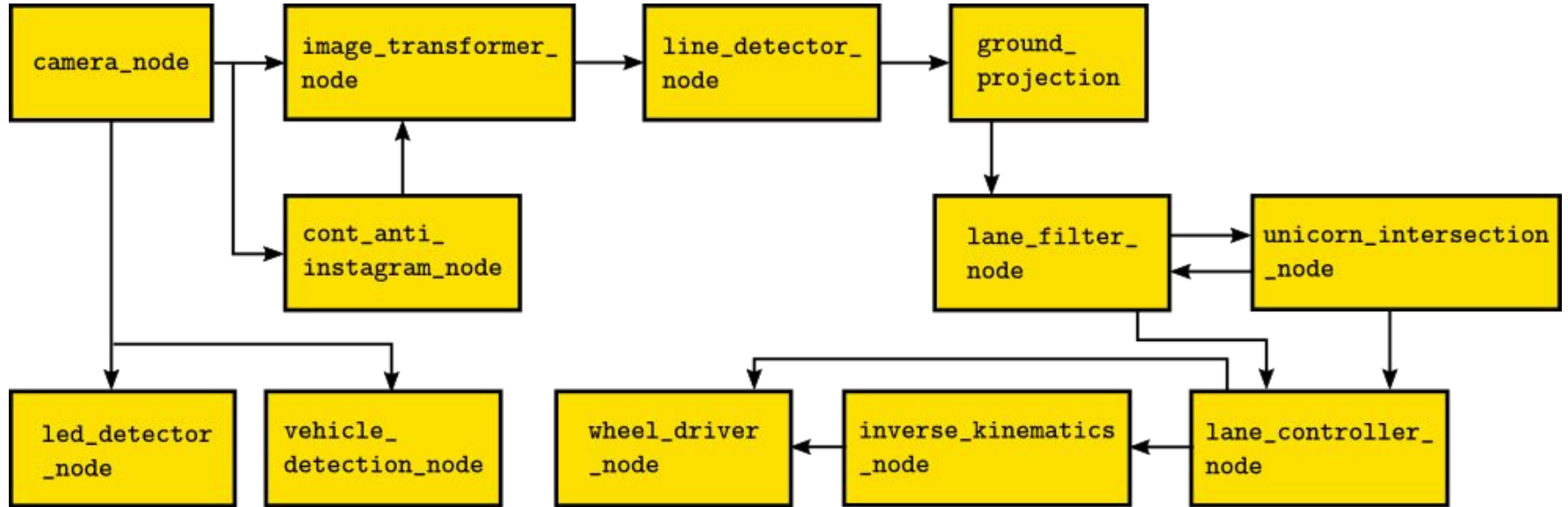


Containerization: A User Story

1. Type a short command, e.g. `$ dts init_sd_card`
2. Follow the installation wizard to flash an image.
3. Transfer flashed SD card to Duckiebot and boot.
4. Open a URL, e.g. <http://duckiebot.local:9000/>
5. Start or download a container, e.g. `duckietown/rpi-duckiebot-base`, `duckietown/gym-duckietown-agent`, `duckietown/rpi-joystick-demo`
6. Open a browser console and run, e.g. `roslaunch joystick ...`

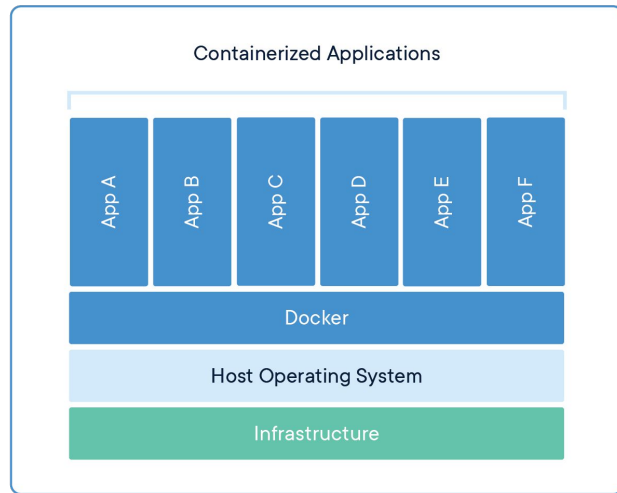


Duckietown ROS Nodes



Reproducible Builds & Containerization

- Benefits of containerization:
 - Reproducible build and deployment artifacts
 - Specified, documented software environments
 - Reusable, multi-platform applications
- Disadvantages:
 - Learning curve for Docker containers
 - GUI applications and X11
 - Migration complexity



PRIMARY

Dashboard

App Templates

Stacks

Services

Containers

Images

Networks

Volumes

Configs

Secrets

Swarm

PORTAINER SETTINGS

Endpoints

Registries

Settings

Containers

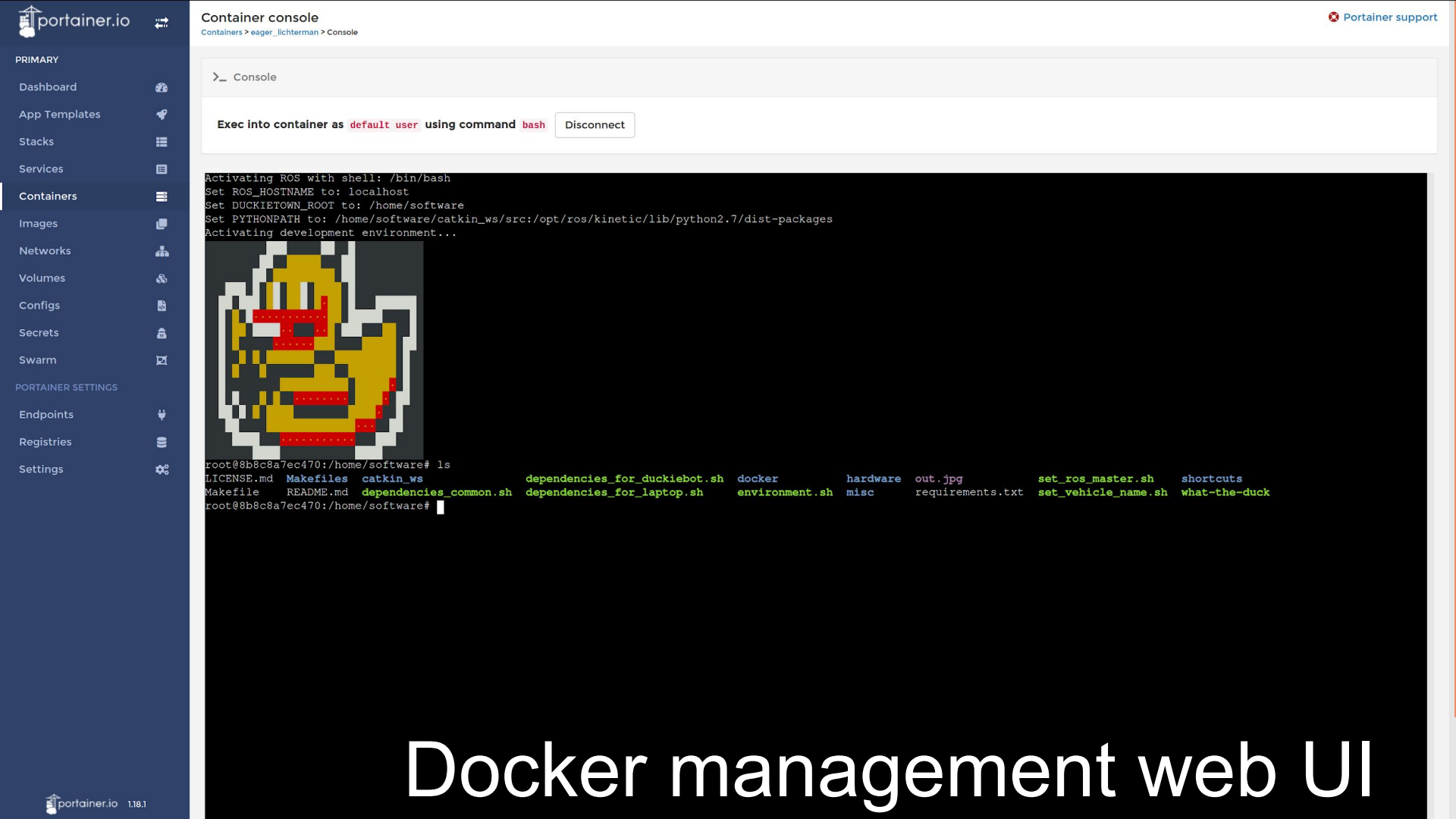
Search Settings

Start Stop Kill Restart Pause Resume Remove Add container

<input type="checkbox"/>	Name	State Filter	Quick actions	Stack	Image	IP Address	Published Ports
<input type="checkbox"/>	eager_lichterman	running	Logs Exec Terminal Info	-	duckietown/software	172.17.0.3	-
<input type="checkbox"/>	portainer.1.raa07xtbrhxc6d9m...	running	Logs Exec Terminal Info	-	portainer/portainer:linux-arm	172.17.0.2	9000:9000

Items per page 10

Docker management web UI



>_ Console

Exec into container as default user using command bash

Disconnect

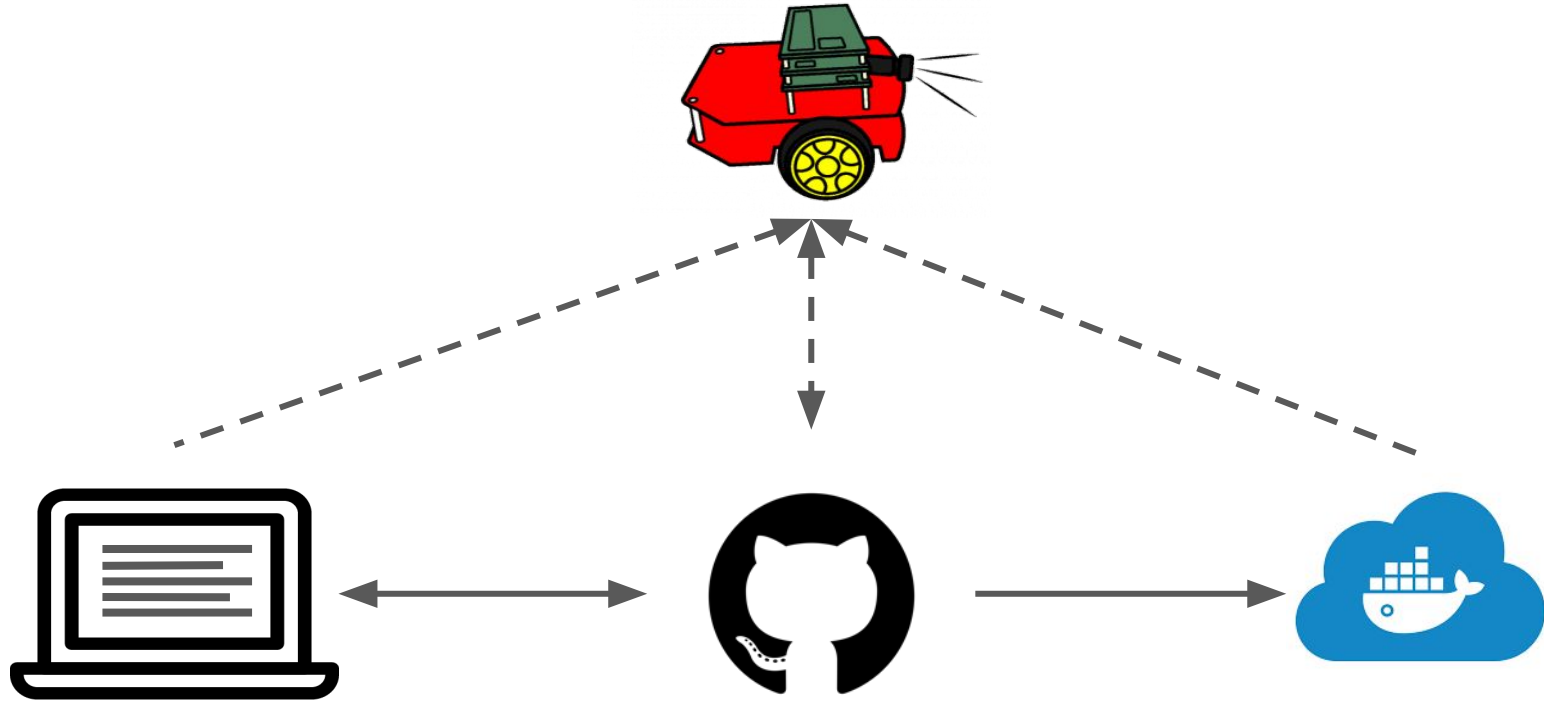
```
Activating ROS with shell: /bin/bash
Set ROS_HOSTNAME to: localhost
Set DUCKIETOWN_ROOT to: /home/software
Set PYTHONPATH to: /home/software/catkin_ws/src:/opt/ros/kinetic/lib/python2.7/dist-packages
Activating development environment...
```



```
root@8b8c8a7ec470:/home/software# ls
LICENSE.md  Makefiles  catkin_ws      dependencies_for_duckiebot.sh  docker      hardware  out.jpg      set_ros_master.sh  shortcuts
Makefile    README.md  dependencies_common.sh  dependencies_for_laptop.sh    environment.sh  misc      requirements.txt  set_vehicle_name.sh  what-the-duck
root@8b8c8a7ec470:/home/software#
```

Docker management web UI

Containerization: Deployment Models



Containerization: Classic (ROS) Stack



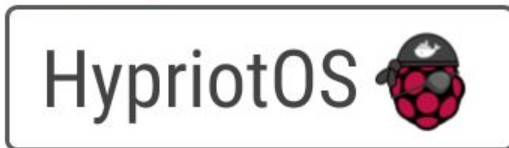
Demos and ROS nodes



Based on ARM32v7, ROS Kinetic Kame, Ubuntu Xenial Xerus, Python 2.7



Docker Client (via get.docker.com)



Lightweight base operating system



ARM-based single board computer (SBC)

Containerization: Laptop / Cloud Stack



Demos and ROS nodes



Based on ARM32v7, ROS Kinetic Kame, Ubuntu Xenial Xerus, Python 2.7



Docker Client (via get.docker.com)



Any major OS (Windows/macOS/Deb)



Any x86 compatible architecture will do



DuckieOS



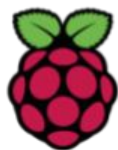
DuckieOS



docker



HypriotOS



POWERED BY

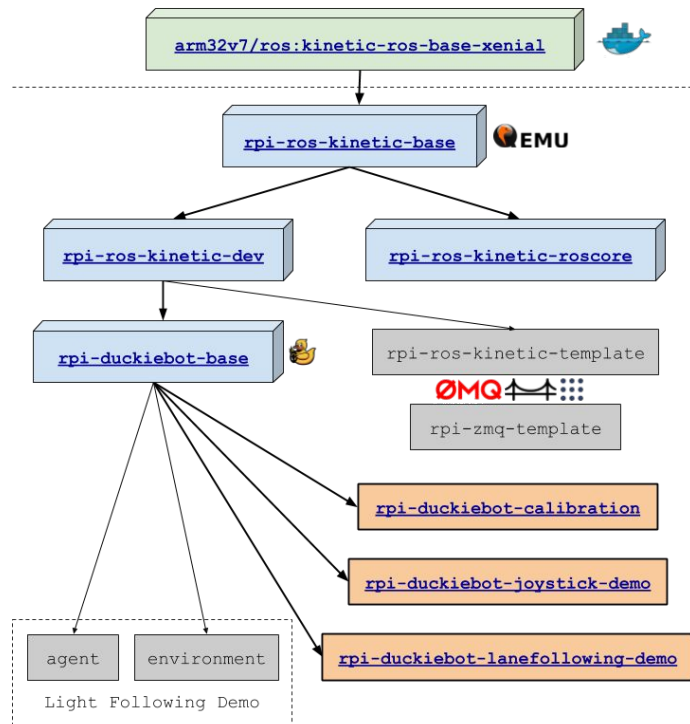
Raspberry Pi



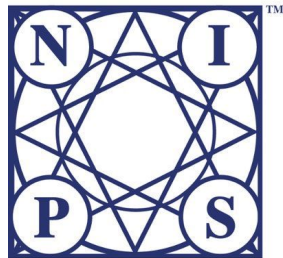
(x86)

Lessons Learned

- Be careful with Docker inheritance
 - Rebuilds can play havoc on a large tree
- Use a versioning scheme from the outset
- Don't try to over automate
 - There is still value in teaching manual commands
- Compile nodes and run tests in the build
 - Prevents changes from propagating downstream
- Utilize emulator tools for cross-building
- Don't compile libraries unnecessarily
 - PiWheels et al. have precompile binaries for ARM
 - Long builds will slow down your development
- Utilize caching whenever possible



AI Driving Olympics




Three principal challenges:

- Lane following
- Lane following with obstacles
- Fleet management / mobility on demand

We evaluate your submission in a simulator and run it on a real Duckiebot!

Coming to NIPS 2018 and ICRA 2019. Register today at duckietown.org

 gym-duckietown-agent `[:gpu]` 



gym-duckietown 



docker



HypriotOS



ARM Stack



x86 Stack



NVIDIA.

GPU Stack

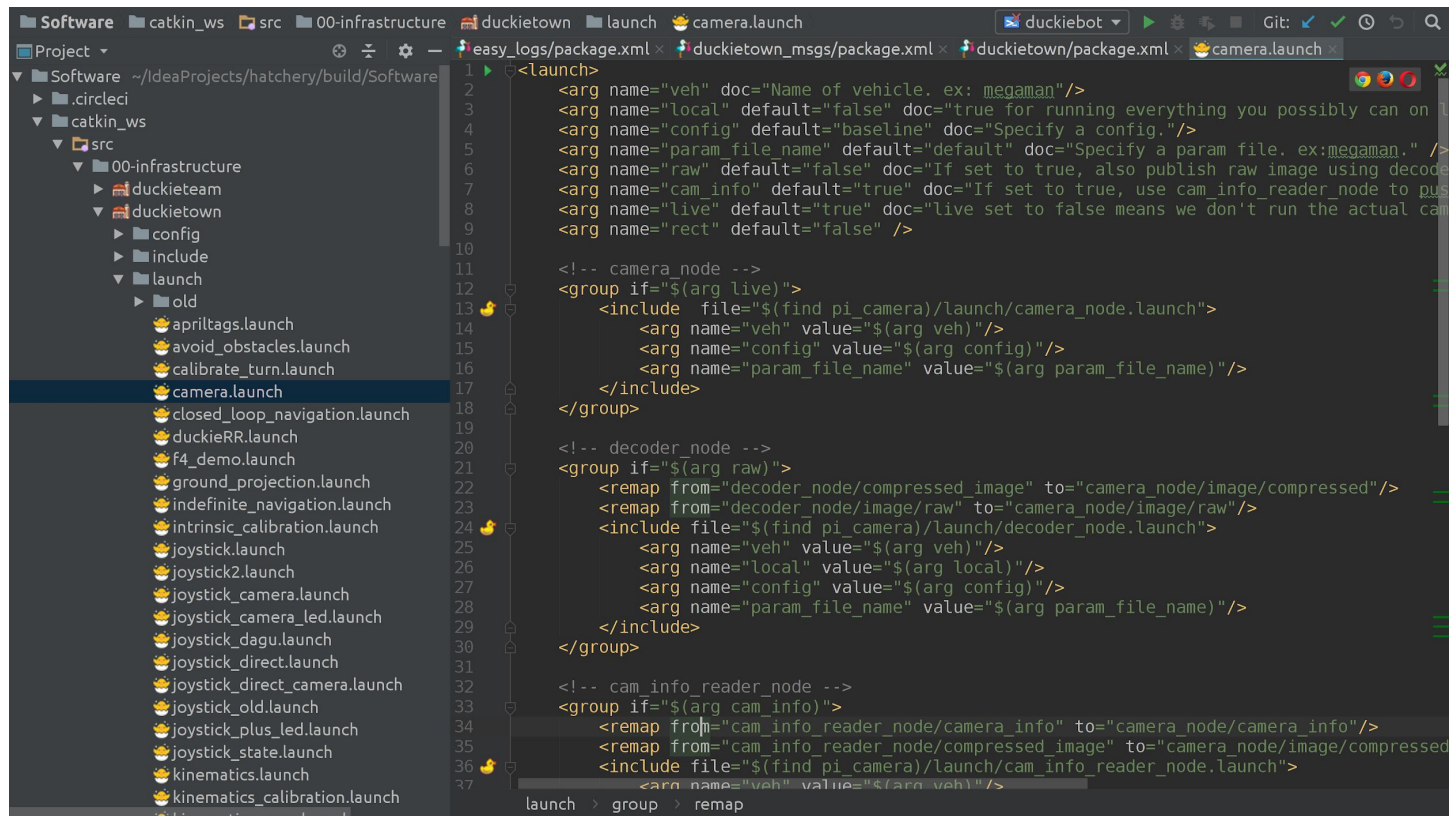
← auto migration

← user migration

Recap and future work

- Two groups of users: researchers and students
 - One wants reproducibility, both want user friendly tooling
- Need to facilitate comparability with baseline implementations
- Simplify deployment model to fleet with easy rollbacks
- Gracefully degrade services and exert precise control over QOS
- Docker helps us achieve this, but it is not a silver bullet for reproducibility
- By utilizing emulation, we can gradually deploy and fail early
- Hardware-in-the-loop testing can give us further predictability

Help Wanted: Hatchery, a ROS IDE



The screenshot displays the Hatchery ROS IDE interface. On the left, a project tree shows the directory structure: `Software` (containing `catkin_ws` and `00-infrastructure`), `src`, `duckietown`, `launch`, and `camera.launch`. The `launch` directory is expanded, showing various launch files like `apriltags.launch`, `avoid_obstacles.launch`, `calibrate_turn.launch`, `camera.launch` (highlighted), `closed_loop_navigation.launch`, `duckieRR.launch`, `f4_demo.launch`, `ground_projection.launch`, `indefinite_navigation.launch`, `intrinsic_calibration.launch`, `joystick.launch`, `joystick2.launch`, `joystick_camera.launch`, `joystick_camera_led.launch`, `joystick_dagu.launch`, `joystick_direct.launch`, `joystick_direct_camera.launch`, `joystick_old.launch`, `joystick_plus_led.launch`, `joystick_state.launch`, `kinematics.launch`, and `kinematics_calibration.launch`.

The main editor area on the right shows the content of `camera.launch`. The file starts with a `<launch>` tag and includes several arguments for configuration. It then defines three main groups of nodes: `camera_node`, `decoder_node`, and `cam_info_reader_node`. Each group is conditional based on arguments like `live`, `raw`, and `cam_info`. The `camera_node` group includes the `camera_node.launch` file. The `decoder_node` group includes the `decoder_node.launch` file. The `cam_info_reader_node` group includes the `cam_info_reader_node.launch` file. The file ends with a `</launch>` tag.

```
<launch>
1
2   <arg name="veh" doc="Name of vehicle. ex: megaman"/>
3   <arg name="local" default="false" doc="true for running everything you possibly can on l
4   <arg name="config" default="baseline" doc="Specify a config."/>
5   <arg name="param_file_name" default="default" doc="Specify a param file. ex:megaman." />
6   <arg name="raw" default="false" doc="If set to true, also publish raw image using decode
7   <arg name="cam_info" default="true" doc="If set to true, use cam_info_reader_node to pub
8   <arg name="live" default="true" doc="live set to false means we don't run the actual cam
9   <arg name="rect" default="false" />
10
11   <!-- camera_node -->
12   <group if="$(arg live)">
13     <include file="$(find pi_camera)/launch/camera_node.launch">
14       <arg name="veh" value="$(arg veh)"/>
15       <arg name="config" value="$(arg config)"/>
16       <arg name="param_file_name" value="$(arg param_file_name)"/>
17     </include>
18   </group>
19
20   <!-- decoder_node -->
21   <group if="$(arg raw)">
22     <remap from="decoder_node/compressed_image" to="camera_node/image/compressed"/>
23     <remap from="decoder_node/image/raw" to="camera_node/image/raw"/>
24     <include file="$(find pi_camera)/launch/decoder_node.launch">
25       <arg name="veh" value="$(arg veh)"/>
26       <arg name="local" value="$(arg local)"/>
27       <arg name="config" value="$(arg config)"/>
28       <arg name="param_file_name" value="$(arg param_file_name)"/>
29     </include>
30   </group>
31
32   <!-- cam_info_reader_node -->
33   <group if="$(arg cam_info)">
34     <remap from="cam_info_reader_node/camera_info" to="camera_node/camera_info"/>
35     <remap from="cam_info_reader_node/compressed_image" to="camera_node/image/compressed
36     <include file="$(find pi_camera)/launch/cam_info_reader_node.launch">
37       <arg name="veh" value="$(arg veh)"/>
38     </include>
39   </group>
40 </launch>
```

Learn more at github.com/duckietown/hatchery

Special Thanks

Rusi Hristov

Liam Paull

Andrea Censi

