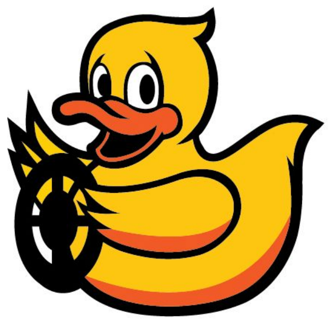
Lab 04 - February 26



Goals for this lab:

1) **Everybody knows how to make a package in ROS**
2) **Everybody understands the conventions about where to put files and what to call them.**
3) **Everyone knows how to push to their own branch and generate a pull request**

- Main channel for this event is: **#lab04-feb26**
    - Other relevant channels:
        - #help-git
        - #help-ros
- Important documents for this lab:
    - Setup Step 4.0 - Creating your own ROS package
    - Setup Step 4.1 Integrating your new package into the duckietown infrastructure
    - Source Code Management Rules and Conventions
    - Checklist - conforming ROS contribution
    - The progress spreadsheet

Editing policy: Staff and students: you are welcome to edit and add materials into this document, or any of the setup (or other) docs, but please **put it in dark purple like this**, and then it will be turned into black by the instructors after the change is announced. This will avoid us getting confused regarding what is the starting vegitrsion and what are the changes.

Reminder: Color key: blue is what students should read; orange is important/needs attentions; **red** is urgent / past due; purple is the status survey; green is done (yay!), **dark purple are changes to the instructions (by students and staff)**

Exercise 1 - Pushing your branch

1) make a branch M02_RCDP-<handle> (if you haven't already)
    a) git pull
    b) git branch
        - what branch are you on? if M02_RCDP then skip ahead to (d) if on M02_RCDP-<handle> then skip ahead to (f)
    c) git checkout M02_RCDP
    d) git branch M02_RCDP-<handle>
    e) git checkout M0**2**_RCDP-<handle>
    f) (if you haven't started, just create a file: "lab04-<handle>.txt"

        g) git add + git commit that file (otherwise your work so far)
        h) git push --set-upstream origin M02_RCDP-<handle>
2) marvel at our git graph

Note 1: if you pushed to master by mistake: here's what we are going to do in the future: we are going to fix it in a way that is transparent to the others, but it will involve you having to remove ~/duckietown and recreate it. (as described in the Source Code Management Rules and Conventions)

Note 2: The honor code: …

- **Status update! Please edit column D in the lab 04 status**
    - Are you stuck? Please put your name and problem here:
        - Erlend: How to "marvel" at the git graph?

Suggestion box / tips and tricks (please write your name and things that you found useful to get through the exercise):

Exercise 2 - Package creation

Open Setup Step 4.0 - Creating your own ROS package and follow the steps to create a "virtual_mirror_<handle>" (if you have not already done so).

Note: if you have already created the package "virtual_mirror_<handle>" make sure that it is at the right location according to the conventions laid out at the beginning of Setup Step 4.0. You can do this by just moving things.

- **Status update! Please edit column E in the lab 04 status**
    - Are you stuck? Please put your name and problem here:
        -

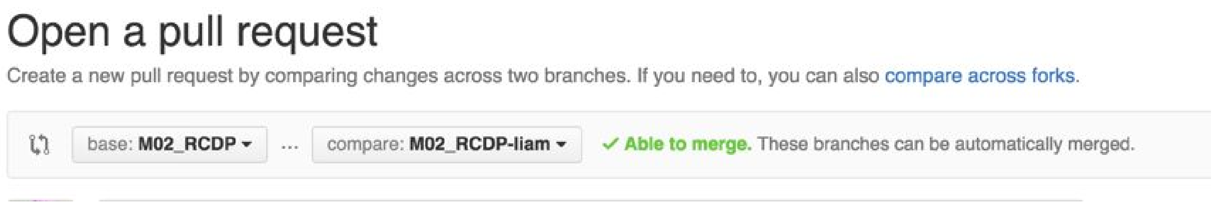
Suggestion box / tips and tricks (please write your name and things that you found useful to get through the exercise):

Exercise 3 - Pull request

1) First step should be to merge M02_RCDP into M02_RCDP-<handle>:
    a) Make sure that you're on M02_RCDP-<handle> branch by
        $ git branch
    b) Merge the M02_RCDP into your M02_RCDP-<handle> by:
        $ git fetch
        $ git merge origin/M02_RCDP
        This makes the the pull request much easier to handle.

2) Go to github duckietown/Software page
3) Click “New Pull Request” green button

Set the base and compare as M02_RCDP and M02_RCDP-<handle> respectively:


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: M02_RCDP
compare: M02_RCDP-liam
✓ Able to merge. These branches can be automatically merged.

3) Click “Create Pull Request”
4) Post a message to #lab04_feb26 and someone will check accept your merge.

- **Status update! Please edit column F in the lab 04 status**
    - Are you stuck? Please put your name and problem here:
        -

Suggestion box / tips and tricks (please write your name and things that you found useful to get through the exercise):

Exercise 4 - Duckietown - Engineering folder conventions

Open Setup Step 4.1 Integrating your new package into the duckietown infrastructure.

In this exercise we will bring your virtual_mirror-<handle> package into compliance and also add a config to be read, a message to be created, and parameter to be updated.

Note: If you have not completed Module 02 you can continue to work on it now. The following tasks will be part of the next module.

(a) Make a virtual_mirror_<handle>_node (it should have the functionality described in Module M02_RCDP - due Sunday Feb 28 **plus** it should read in a parameter `flip_direction` which can take the value `vert or horz`. The parameter should be loaded in according to the conventions in Setup Step 4.1. For loading in the parameters in the node look at basically any node in the duckietown code base (for example `lane_filter_node.py`)
(b) Create a new msg (according to the conventions) and use it to publish at 1Hz the value of the config param that you read in (you will need to set up a Timer for this - look in joy_mapper_node.py). This message type should be an enumeration. For an example see: `Segment.msg` in `$(DUCKIETOWN_ROOT)/catkin_ws/src/duckietown_msgs.`

(c) Make it so that the value of the param read in can be updated from the command line using `rosparam set /<vehicle_name>/<virtual_mirror-<handle>_node/flip_direction <value>`
Hint: you also need to update the code in the node - for clues look in `joy_mapper_node.py` in the `joy_mapper` package.
(d) Write a stub tester node with provided default images. The stub tester should evaluate the correctness for all 2 possible values of the parameter.
(e) Make the elemental launch file
(f) Make the stub test launch file
(g) Make the system level launch file
(h) Make the unit test launch file

- **Status update! Please edit column G-M in the lab 04 status**
    - Are you stuck? Please put your name and problem here:
        -

Suggestion box / tips and tricks (please write your name and things that you found useful to get through the exercise):

- Teddy: If your images are not passing the test even though they look correct, it could be the compression is messing the the pixel values. Try using '.png' instead of '.jpeg'.

Timeline

By 11:59 Thursday:
AC: polish the initial git policy for students
SY: should polish the package creation doc
LP: should polish the launch file and conventions doc

Friday:

9am:
LP + SY + MC + HZ arrive at 226

9:15:
Move to Beaverworks

9:30:
LP + HZ: set up the network
SY: Connect laptop to the projector

LP + SY: Final tests

Equipment checklist

- Network
  - all the switches we can find
  - 5 super long (orange?) ethernet cable
  - 30 long (orange) ethernet cable
  - 30 short (green) ethernet cable
  - 1x airport express
  - 1x extra airport express
- Power
  - all the power strips and power extenders we can find
- Duckietops for loan
- Extra mouse
- Robots? Megaman + Pontiac + ?