Untitled

Jonathan Garcia-Mallen

??? August 2016

Contents

1 Background

1.1 This is Duckietown

COPYPASTED FROM DUCKIETOWN.MIT.EDU
""""
Duckietown is a class on advanced autonomy taught at MIT For Spring 2016, MIT has a new class about the science of autonomy at the graduate level. This is a hands-on, project-focused course focusing on self-driving vehicles and high-level autonomy. The problem: **Design the Autonomous Robo-Taxis System for the City of Duckietown.**

This is a class for makers && thinkers This is a collaborative effort: 2 Labs (CSAIL and LIDS), 3 Departments (ME, AeroAstro, EECS), and over a dozen people helped created this class, under the supervision of Prof. John Leonard and Prof. Jon How. Duckietown UROP opportunities Staff Cool videos
Duckietown is a reproducible, open-source class

This is an open-source class: all materials (hardware design, software, and teaching materials) will be released as "open source" (a free software license for code; a Creative Commons license for teaching materials) News and updates Follow us on Facebook: facebook.com/duckietown. Contacts: Please contact us at duckietown@mit.edu with any question. """"

- this is duckietown (copy from website)
- problem, and why you should care The students have to ssh into their duckiebots every time they want to test or show off their code. show in excruciating detail what has to be done in order to run a program on the bot, both remote and ssh'ing methods. Problems: it's annoying. Audience may not understand that the code is running entirely on the duckiebot, and not on the laptop. wifi isn't reliable. this has happened: senior faculty pass by quickly, you want to show them a demo, but you have to deal with your laptop. This is an autonomous vehicle. It shouldn't need a laptop. Thus, we must do everything with just a joystick

2 Requirements and Design Goals

Start a program of the student or researcher's choosing without using a separate computer to launch it. contained within duckiebot; no laptop interaction. reliable. This cannot fail when you're in front of an audience. easy and fast to use. Students shouldn't have a hard time interfacing with this, or have to push more buttons than either rosrun method. future-proof. This was not an initial need, but instead grew as a requirement as the implementation options' lifespans were reviewed.

3 Existing technologies used here

js_linux.py init.d/linux supervisord

4 Implementation

Joystick Daemon apt-get isntall joystick; jstest; python app from github joystick-daemon.py; startuptest Starting up the joystick daemon (runlevels, if need more padding) init.d rc.local supervisord

5 Conclusions and Future work

results yo. it ran like 10 times in a row, successfully. It be good. It shouldn't have taken this long. It's not worth figuring out why rc.local didn't work. Who knows. Make it prettier. Make it more user friendly.