Introduction to Cascar | TSFS12

Pavel Anistratov & Theodor Westny

LINKÖPING UNIVERSITY

Hardware platform

- Repurposed RC car
- Raspberry Pi4 & Arduino
- Sensors:
  - RPlidar A2
  - Hall sensors (wheel ticks)
  - MPU6050 6-dof IMU
  - Raspicam (optional)

Software platform

- Ubuntu Mate 20.04 Lts
- ROS Noetic
- Python3
- C++
- https://gitlab.liu.se/thewe60/cascar_pkgs

ROS

- ROS solves one of the more tedious problems in robotics – communication
- Provides software libraries and a framework for implementation
- If you are not familiar with ROS…
  - Not a requirement, but check out: *http://wiki.ros.org/ROS/Tutorials*

What we provide for you

- Necessary code to **move** the car.
- Programs to handle sensors and publish data
- Simple GUI for steering the car
- Skeleton code, and examples for your own development!

https://gitlab.liu.se/thewe60/cascar_pkgs

Let’s take a look at the code...

https://gitlab.liu.se/thewe60/cascar_pkgs

Useful ROS tools

- rqt_graph (Node and topic config)
- rqt_plot (tune the IMU)
- Rviz
- *Roscore*
- *Roslaunch*
- *Rosrun*

Remember to source the environment and export IP!

Suggestions for implementation

- Running linux on your own computer?
  - Install ROS
- Windows 10?
  - (Install ROS experimental) or use ssh
  - Virtual Machine
- Mac?
  - Ssh?
- Development on the RPI4

Suggestions for implementation

- Clone the provided repository
- Create your own repo on gitlab
  - Put it within the src folder

Proposed projects

- Improved state estimation
  - Kalman filtering, SLAM
- Formula student competition
  - Map generation + lap speed
- Real-time planning and Control
  - RRT, Lattice.. PP, LQ, MPC
    - Parking, obstacle avoidance


LINKÖPING
UNIVERSITY