Open Source Aerial Vehicles
IROS 2014

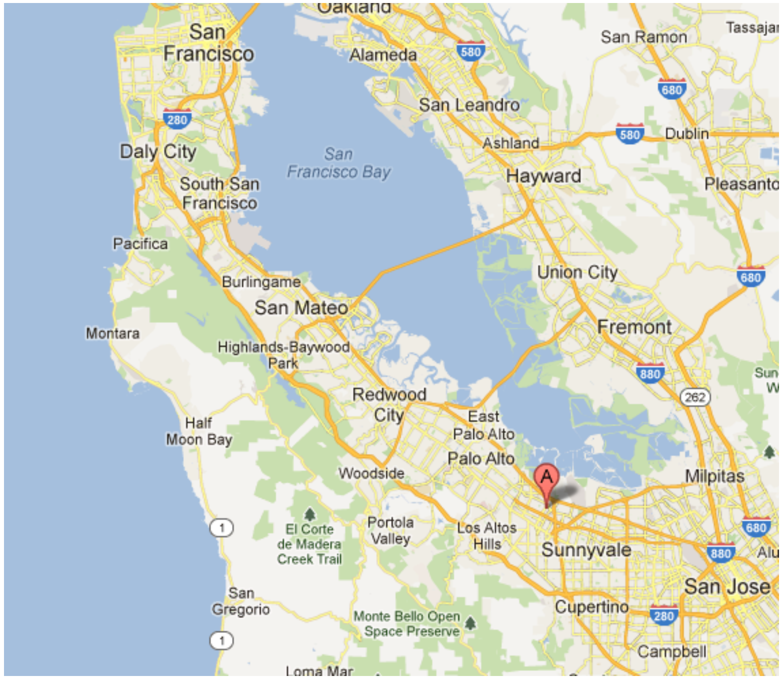
September 14, 2014

Tully Foote

Open Source Robotics Foundation

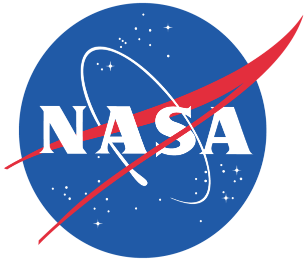
Open Source Robotics Foundation

http://osrfoundation.org



"…to support the development, distribution, and adoption of open source software for use in robotics research, education, and product development."

OSRF Sponsors



Sandia
National
Laboratories

YUJIN ROBOT

BOS
Invented f

rethink
robotics

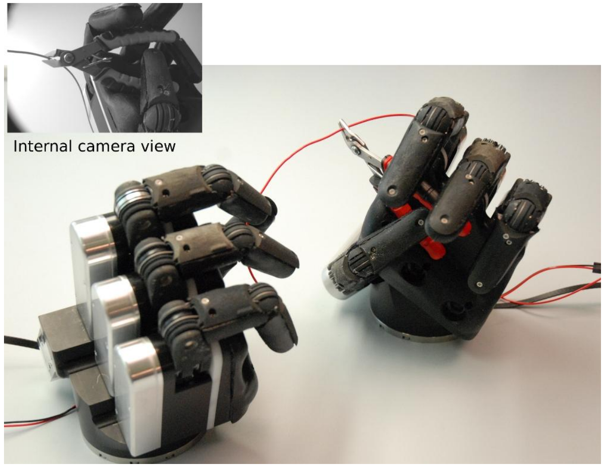
ROS
industrial
consortium

OSRF Projects

ROS

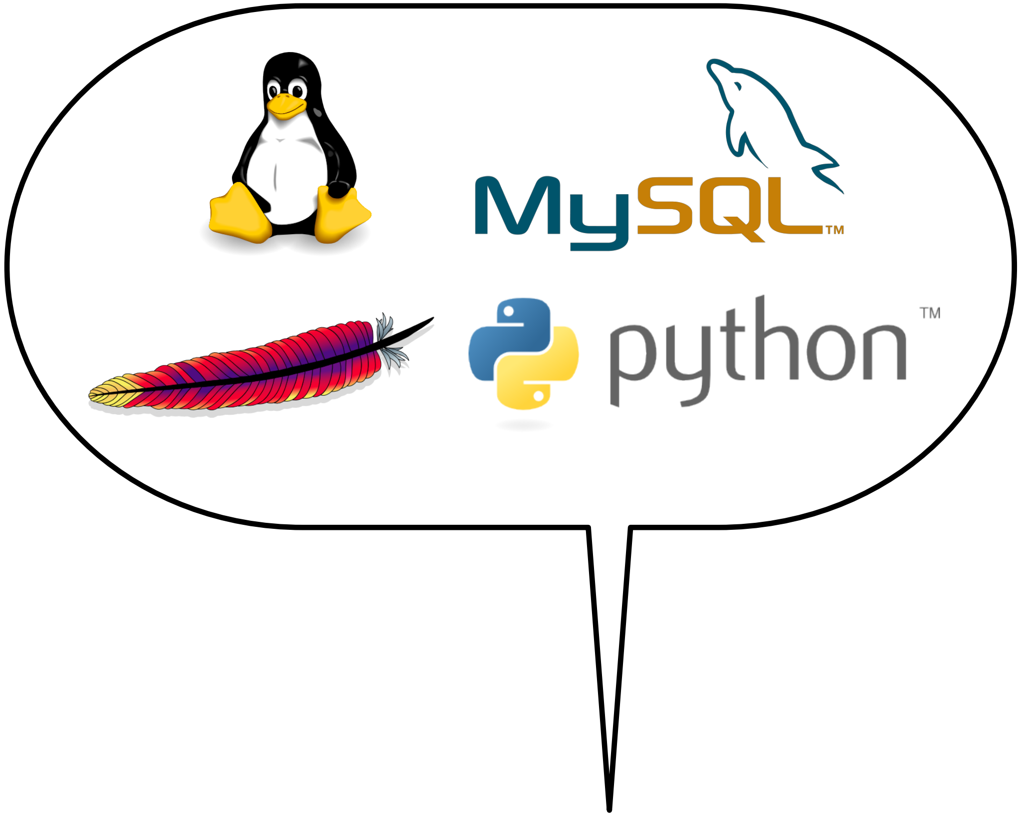
Gazebo: 3-D simulation

CloudSim



Electronics and firmware

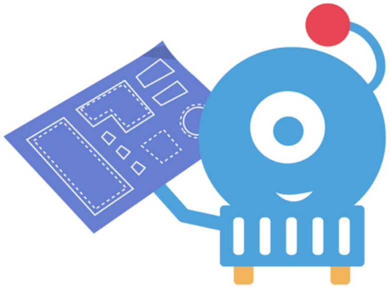
ROS



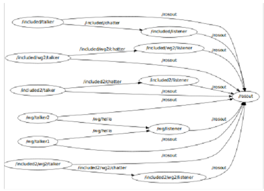
*.com

ROS

and friends

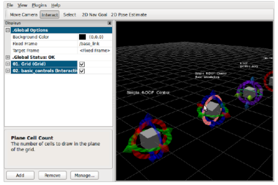


ROS =



Plumbing

+



.Global Options
Background Color (0,0,0)
Fixed Frame /base_link
Target Frame <Fixed Frame>
.Global Status: OK
01. Grid (Grid) ☑
02. basic_controls (Interacti ☑

Plane Cell Count
The number of cells to draw in the plane of the grid.

Tools

+

Capabilities

+

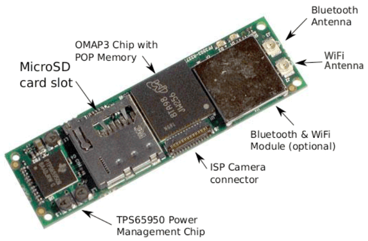
Ecosystem

Open Source Robotics Foundation

Movie: 5 years of ROS

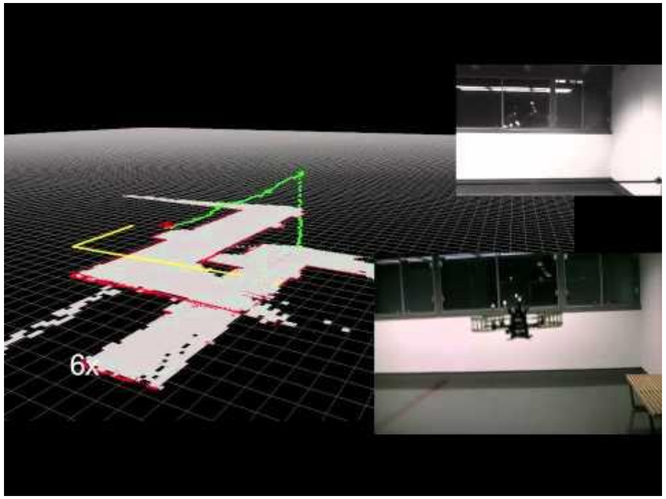
Common Platforms

- Turtlebot
- Clearpath Husky
- Nao
- PR2
- Embedded:
  - Gumstix
  - Raspberry Pi


Bluetooth Antenna
OMAP3 Chip with POP Memory
WiFi Antenna
MicroSD card slot
Bluetooth & WiFi Module (optional)
ISP Camera connector
TPS65950 Power Management Chip

Photos: from manufacturer websites

Uses of ROS in Aerial Vehicles: Videos



Berkeley

UPenn

CCNY

Uses of ROS in Aerial Vehicles: Packages

- mavlink2ros
  - https://github.com/posilva/mav2rosgenerator
- mav_tools
  - http://wiki.ros.org/mav_tools
- CRATES
  - https://bitbucket.org/asymingt/crates
- roscopter
  - https://code.google.com/p/roscopter/
- hector_quadcopter
  - http://wiki.ros.org/hector_slam
- rospilot
  - https://github.com/rospilot/rospilot
- asctec_mav_framework
  - http://wiki.ros.org/asctec_mav_framework

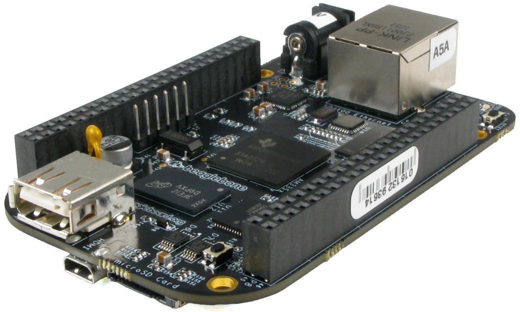
Challenges of Aerial Robotics

- Small payloads, minimal computation available
  - Newer SBC options much higher power
  - To get the minimum functionality often ends up with custom small implementation
- Many very different configurations
  - Basic controls are not standardized/generalized
- Black box interfaces from hardware manufacturers

Binary ARM Packages Coming Soon

Thanks to the sponsorship of Qualcomm we will be setting up official Ubuntu ARM packages to support the new generation of ARM SBC.

Announced yesterday at ROSCon: http://www.osrfoundation.org/open-source-robotics-foundation-to-extend-ros-support-to-qualcomm-snapdragon-processors.html



Upcoming features in ROS 2.0

Features of interest to aerial robotics

- Modern API, minimal dependencies, and better portability
- Benefits of underlying DDS middleware
  - Reliability QoS settings
  - UDP Multicast, shared memory, TLS over TCP/IP
  - Real-Time capable
  - Master-less discovery
  - Minimal dependencies (Current DDS vendors have none)
- Easier to work with multiple nodes in one process
- More dynamic run-time features like topic remapping and aliasing
- Lifecycle management and verifiable systems
- And so many other things we don’t have time to cover here…
  - Dynamic parameters
  - Synchronous, scheduled execution of nodes (the ecto problem)
  - More efficient package resource management

Insights as a human pilot

- Very standard maps
    - Shared reporting points, shared references
    - Never fly without a map
    - Map has a bunch more information than just collision info
        - Radio frequencies, notes, reference points
- Very standard anti-collision protocols
    - Automatic tools, ADS-B
    - ATC/ Self reporting mechanism
    - Traffic patterns
- Completely standard controls inputs
    - Stick + collective + pedals or stick + throttle + rudders
    - Pilot can move between vehicles
        - Local knowledge
        - FBO chatting
- Differences are in standard procedures written down completely (startup/shutdown checklists, emergency procedures)

Examples of standardization in other fields

Mobile Platform:

CoordinateFrames:

http://www.ros.org/reps/rep-0105.html

Generic 2d navigation interface:

http://wiki.ros.org/nav_core

http://wiki.ros.org/navigation/Tutorials/RobotSetup

Humanoid Coordinate Frames:

http://www.ros.org/reps/rep-0120.html

My questions for the community

- What would be required to standardize?
- What common interfaces can be defined?
- Can there be a standard Hardware Abstraction Layer?
- Can we standardize enough to setup unit tests, continuous integration, and possibly performance tests in simulation?
- How can we promote communication and visibility for collaboration?
- What would be the flying LAMP?