Flying Penguins

Embedded Linux applications for autonomous UAVs
stabilization
telemetry
missions
failsafes
AUTO PILOT ≠ AUTONOMOUS
“system finds its own goal positions”
“system finds its own goal positions”

where to go
“system finds its own goal positions”

where to go
how to get there
“system finds its own goal positions”

where to go
how to get there
what to do next
SO MANY ALGORITHMS, SO LITTLE MEGAHERTZ

http://ra3ndy.deviantart.com/art/Sad-Panda-69204875
Autopilot runs on Linux

- Real-time kernel
- Hardware drivers (SPI, I2C, CAN, UART)
- Device trees
- Programmable real-time units
- PixHawk Fire Cape
- BeaglePilot project
Autopilot *talks to* Linux

- Linux runs on a companion computer
- RS-232 serial interface to autopilot
- Treat the autopilot as a peripheral
- This is what I’m talking about today
ODROID-XU3 Lite

- Samsung Exynos5422 **octa core**
  - 4x Cortex™-A15 2.0GHz
  - 4x Cortex™-A7 1.4GHz
- 2 GB RAM
- 32+ GB flash
- 4x USB 2.0 + 1x USB 3.0
Middleware

- **DroneAPI**
  - Python
  - Go to Kevin Hester’s talk tomorrow

- **ROS + mavros**
  - Python, C++, Lisp (really)
  - Access to a wealth of robotics research and tools
ROS CRASH COURSE
Robot Operating System

“ROS is an open-source, meta-operating system for your robot.”

– http://wiki.ros.org/ROS/Introduction
Nodes

- Process / address space
- ROS applications composed of many small nodes
- “Do one thing and do it well”
- Modular
- Reusable
- Separation of concerns
Topics

- Publish / subscribe message bus
- Strongly-typed messages
- Peer-to-peer message passing
- Centralized name registry (master node)
Services

- Similar to topics, but with request / reply semantics
- Think of it as RPC
but that's not all...

parameters
transformations
record/playback
visualization
logging
mavros
Topics

- /mavros/global_position/global
- /mavros/local_position/local
- /mavros/imu/data
- /mavros/state
- /mavros/setpoint_position/local_position
- /mavros/setpoint_velocity/cmd_vel
Services

- /mavros/cmd/arming
- /mavros/cmd/land
- /mavros/cmd/takeoff
- /mavros/set_mode
- /mavros/set_stream_rate
Event-driven programming

• “Don’t call me, I’ll call you”

• Your application code responds to events
  • Message arrival
    • “my position is \((x, y, z)\)”
  • Timer expiry
    • “it’s time to run the control loop”
Example Application

Yet Another Precision Lander
Nodes

• **Tracker**
  - Processes video stream, looking for landing pad
  - Publishes target position/velocity messages

• **Commander**
  - Subscribes to vehicle state and position messages
  - Controls vehicle velocity
(TODO: code snippets)
Simulations
HITL

• Hardware in the loop
• Flight software runs on flight hardware
• Simulated sensor and control inputs
SITL

- Software in the loop
- Flight software runs on (Linux) desktop
- Simulated sensor and control inputs and HAL
ArduPilot SITL
PX4 SITL

- 3D simulation with Gazebo
- TODO
Practical Considerations
Connections

- UART recommended
- USB works for development
Power

• UBEC

• ODROID + USB camera + WiFi + 3S LiPo = 5 hours
Launch files

- ROS feature that makes it easy to start and manage multiple nodes and their parameters

- `roslaunch mavros apm.launch`

- `rosparam load ~/tracker.yaml /tracker`
Startup

• use ubuntu’s upstart to launch ROS + mavros + application nodes

• robot_upstart
Telemetry

- MAVLink + radio
- WiFi
  - Ad-Hoc mode (man wireless)
  - `sudo apt-get remove wpasupplicant`
- GSM
Coordinate Frames

• Global / Local
  • NED
  • NEU
  • ENU
• Body-fixed
• tf library
What will you make?
ardupilot.com
pixhawk.org/start
ros.org
github.com/mavlink/mavros
github.com/claymation/lander