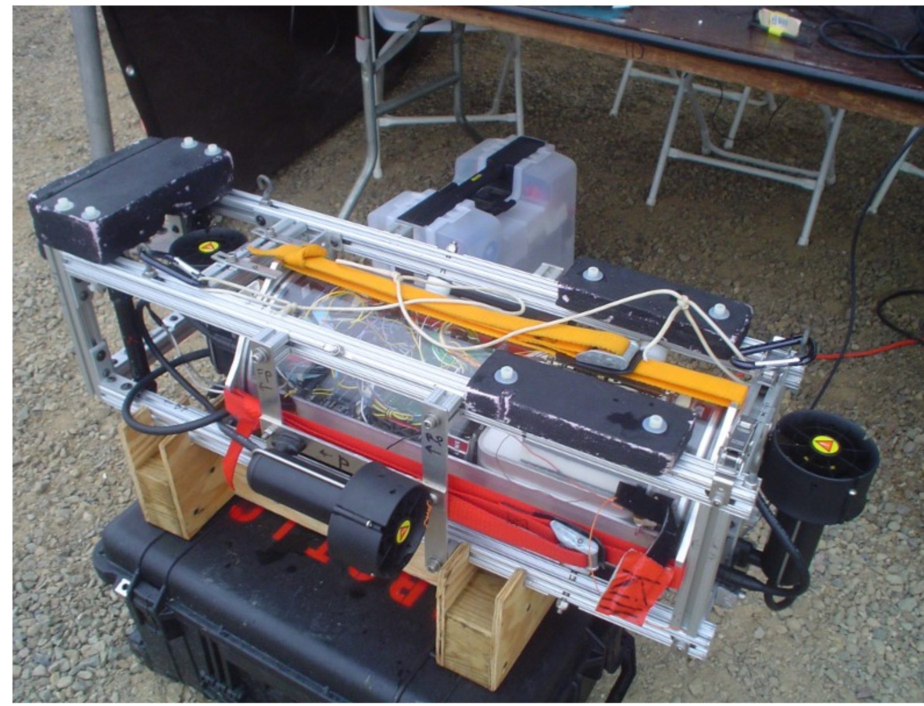
An Underwater Python: Tortuga the Python Powered Robot

Joseph Lisee

Introduction

Python Powered Robots?

- **Python**
  - ~15,000 SLOC
  - AI, GUI, Simulation, High level control
- **C++**
  - ~50,000 SLOC of C++
  - "Real time" Control, Vision, Framework
- Uses 10+ OSS libraries for support



Above: *Tortuga I* at the AUVSI AUV Competition in 2007

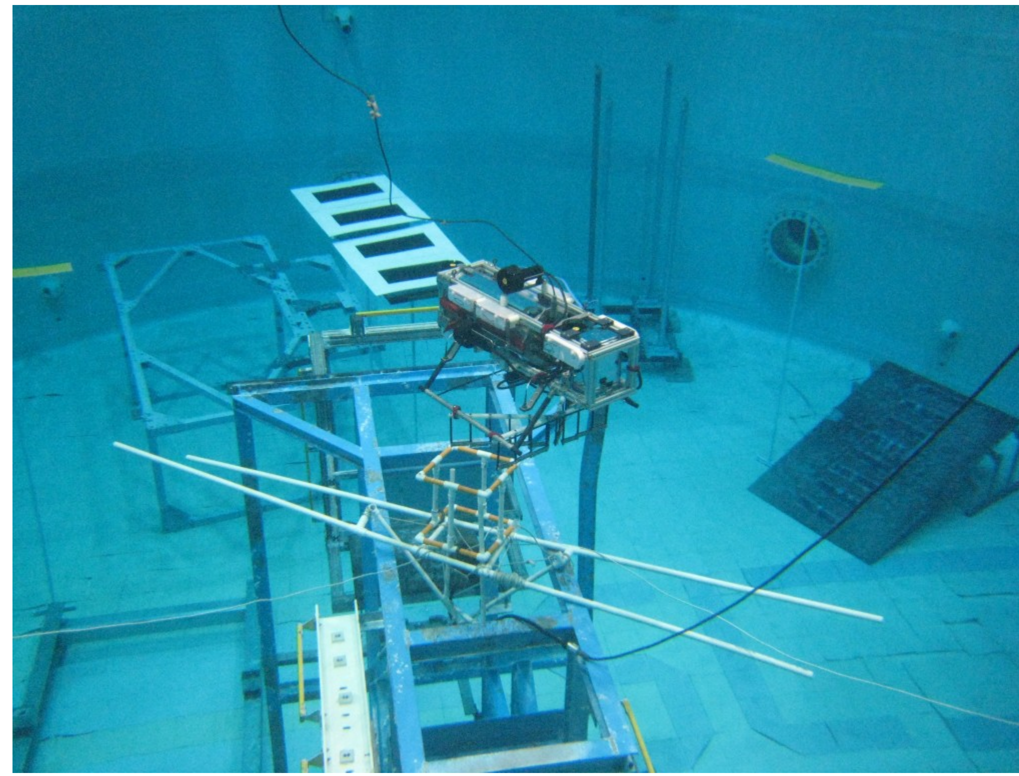
Robotics @ Maryland

Robotics @ Maryland

A highly self motivated group of students
who love to make robots

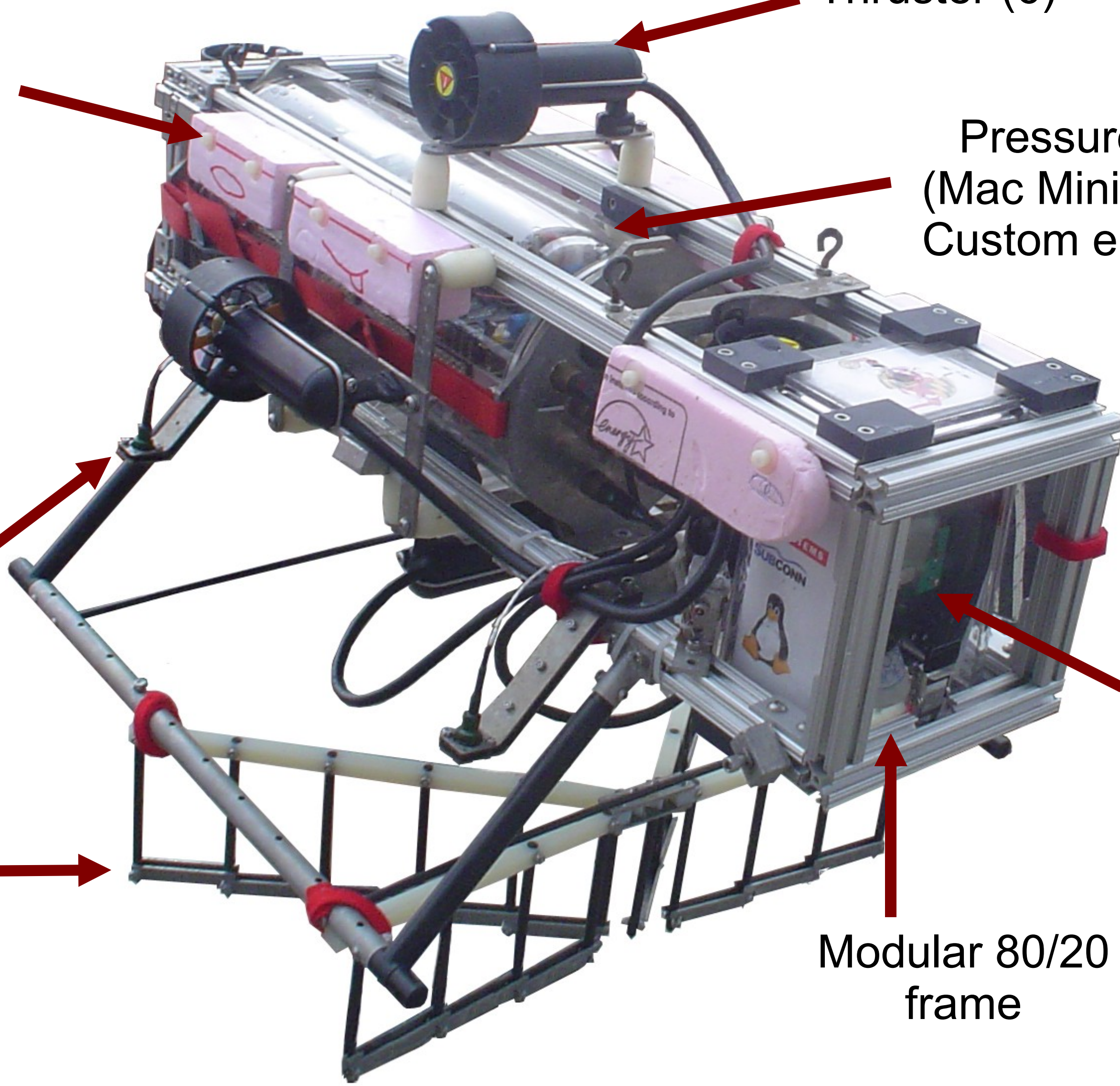
What Does Tortuga Do?

- **Main Goal:**
  - Compete in (and occasionally win!) the AUVSI International AUV Competition
- **Functionally:**
  - Not leak
  - Drive around underwater
  - See, detect & maneuver around colored objectives
  - Home in on sounds
- **Research:** undergraduate and graduate at the SSL



Above: *Tortuga II* poised to grab the safe during an autonomous testing run

How Does it Work?

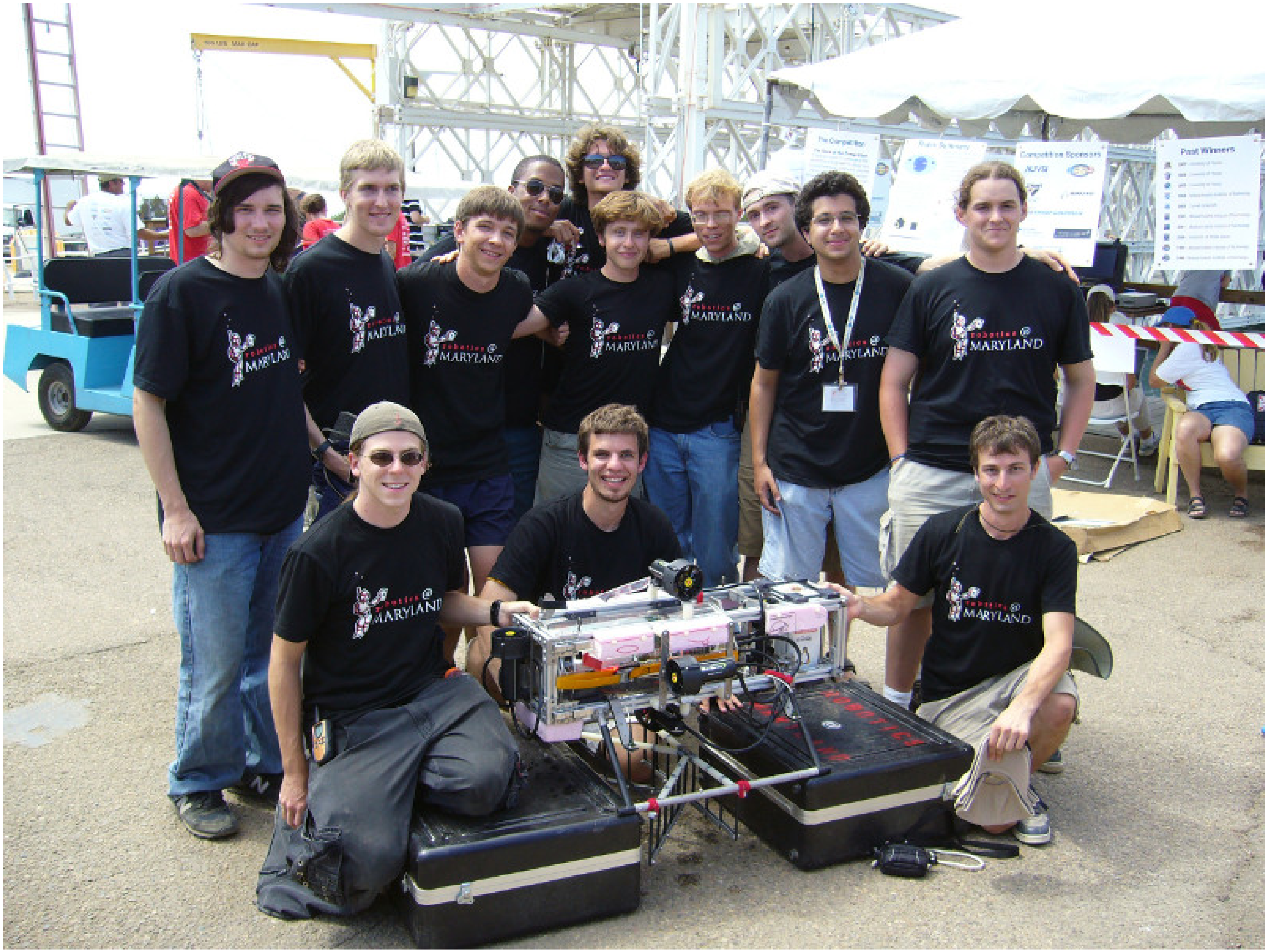


What Does Tortuga it *Really* Do?

Videos!

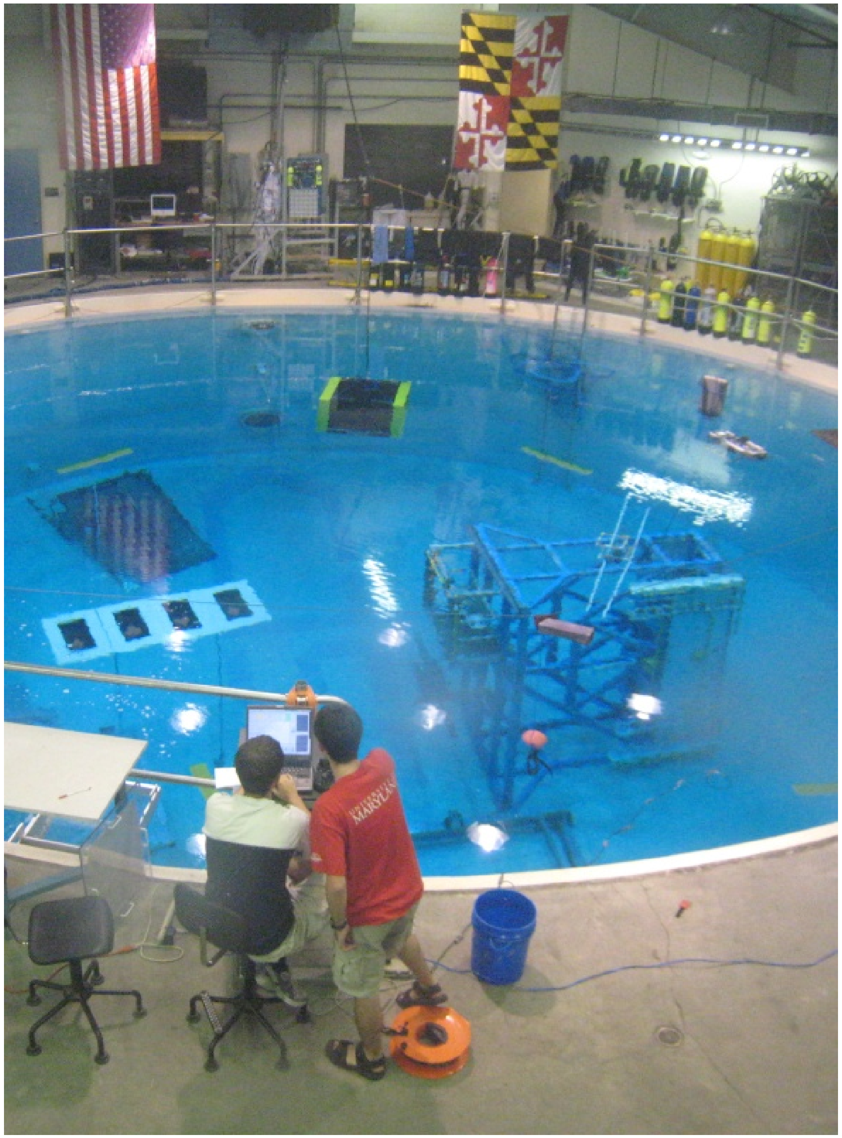
Glory: The Postives

Competition Success



1st place in 2008, our second year

Benefits Using Python In Robotics

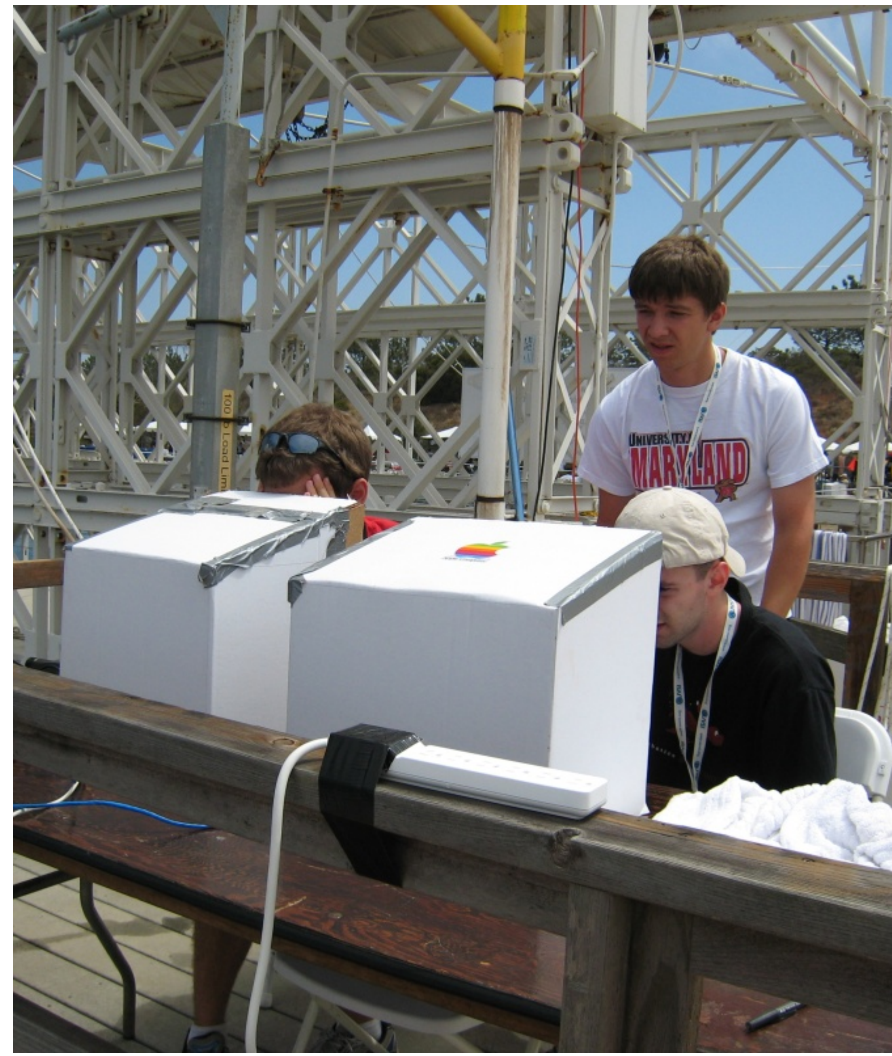


Above: Testing at ~3AM July 23rd, the night before shipping to *Tortuga II* to the competition

- Great flexibility and unit testing support
  - Allows more compact code
  - Creates greater code reuse
- Easy to learn: helps to get new members up to speed
- Batteries and third party libs speed development

Great Built In Unit Testing

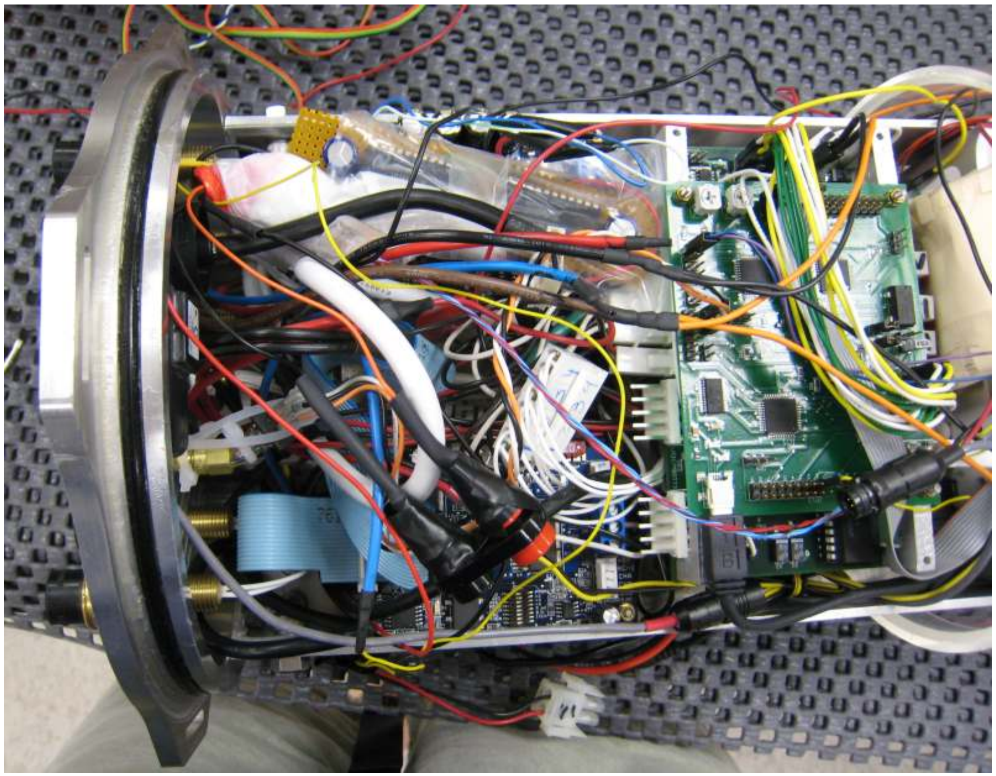
- No 3rd party library to install and manage
- Dynamic nature of python allowed high unit test code reuse
- Allowed refactoring as code scope increases
- Gave us actual confidence in our code (a rare thing in robotics)



Above: Joe L., Steve M., and Mike L. sweat bullets while testing at the competition

Trials & Tribulations: The Negatives

C++ Integration Woes



Above: Spaghetti mess of wiring in *Tortuga I*, similar to the elegance of our C++ integration

- Boost.Python & Py++ are powerful, but complex
- Overhead for such wrappers is large in terms of dependencies, disk space, and compile time
- Small bugs and compiler incompatibilities lead to "fragile" bindings

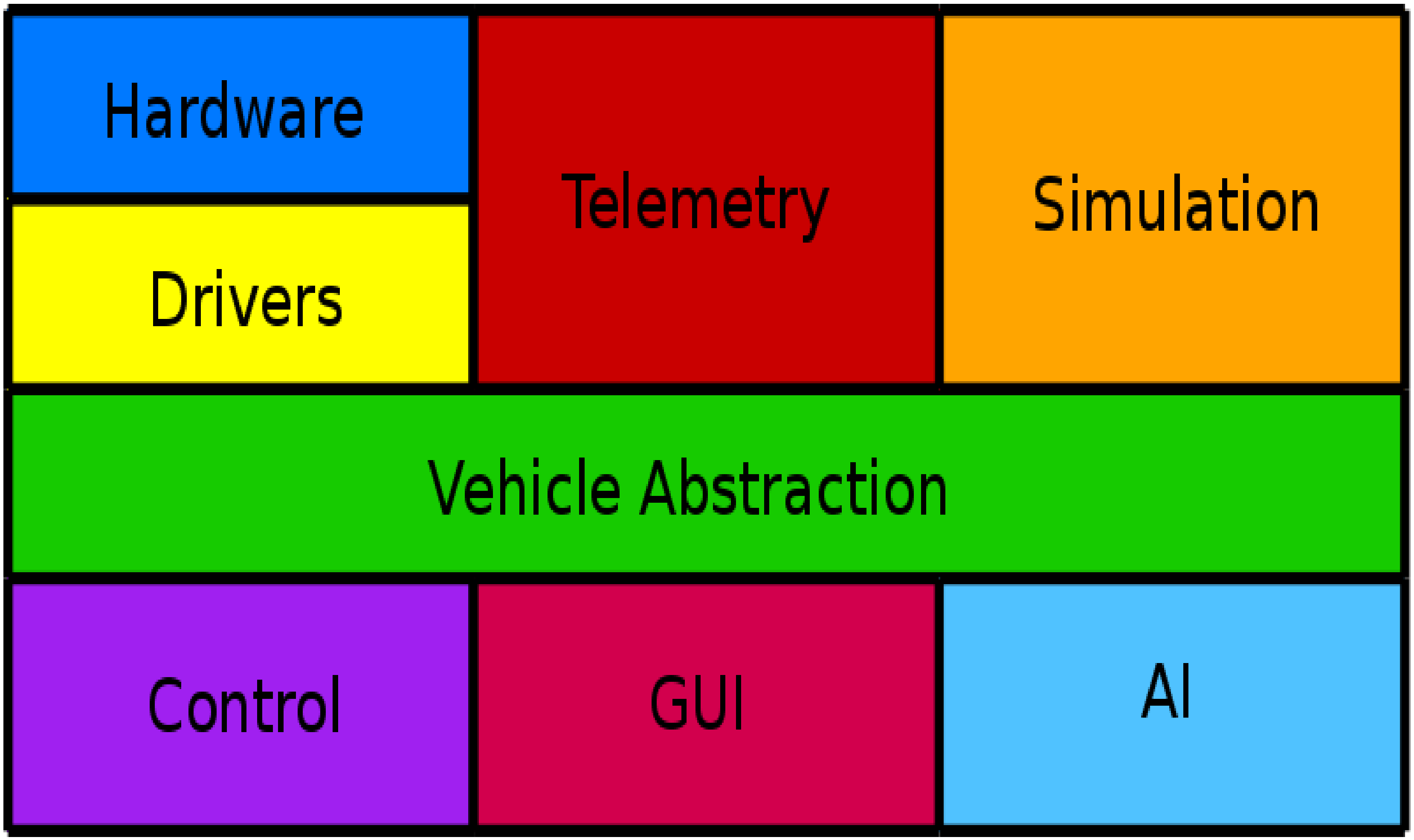
The GIL

- Inflexible nature greatly constrains concurrent system design
- Forced the core of our software into C++
- C++ calling back into python is especially likely to run afoul of the GIL

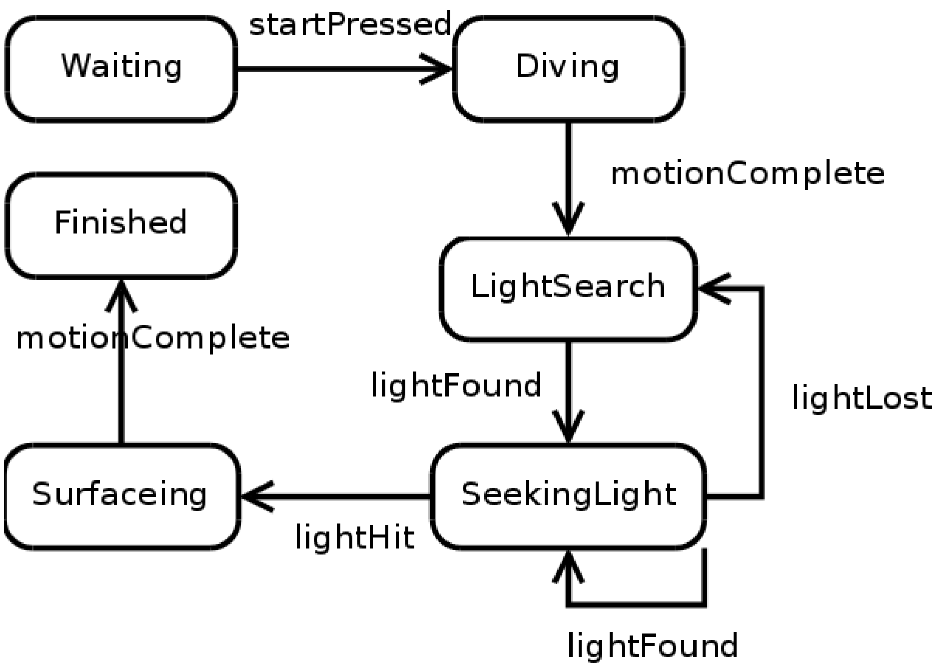
Above: The polar opposite of the GIL, Dave the judge, frustratingly flexible in his interpretation of the rules

Software

Overall System Breakdown



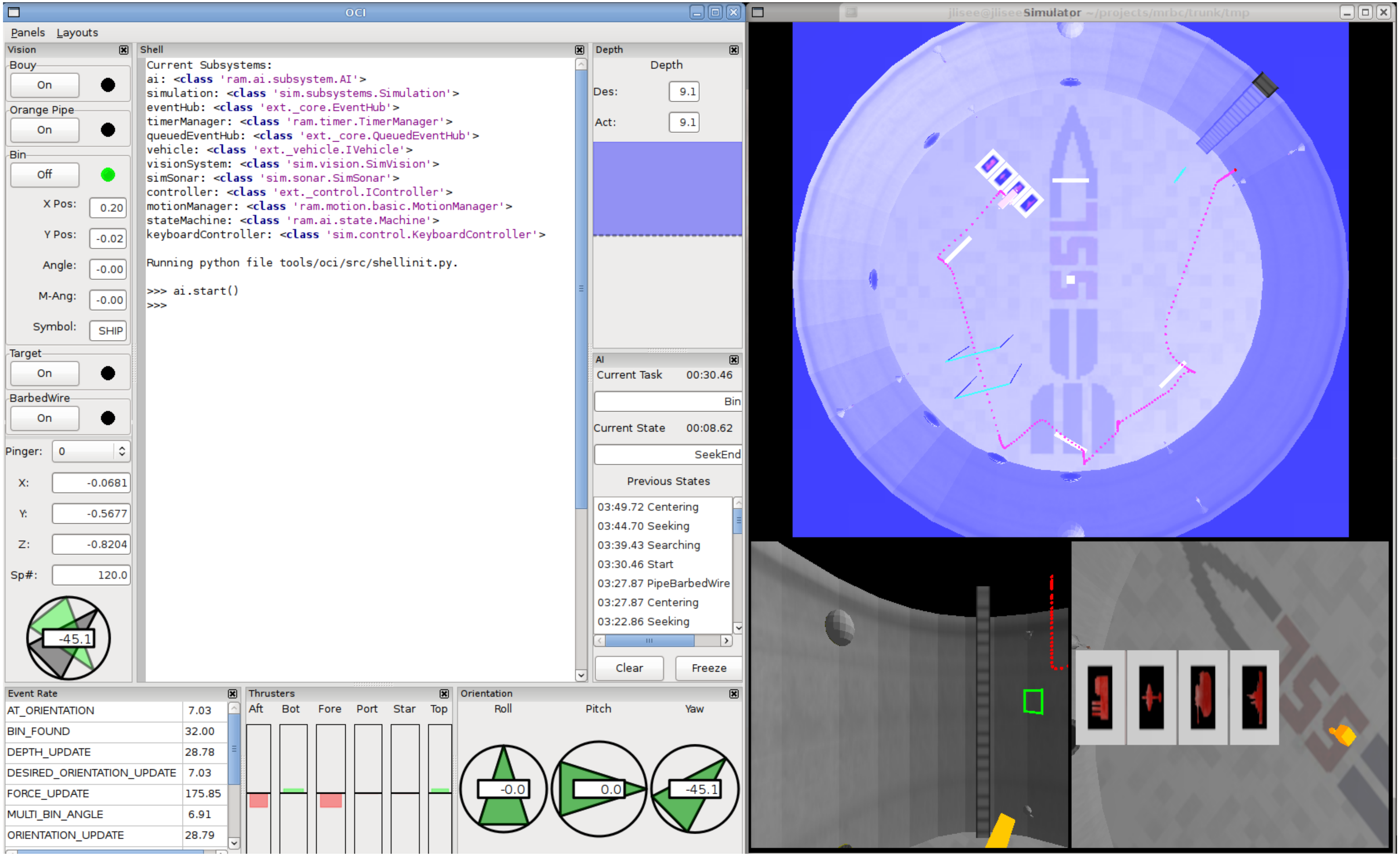
Artificial (Semi)Intelligence



Above: Diagram of basic state machine which seeks and hits a light

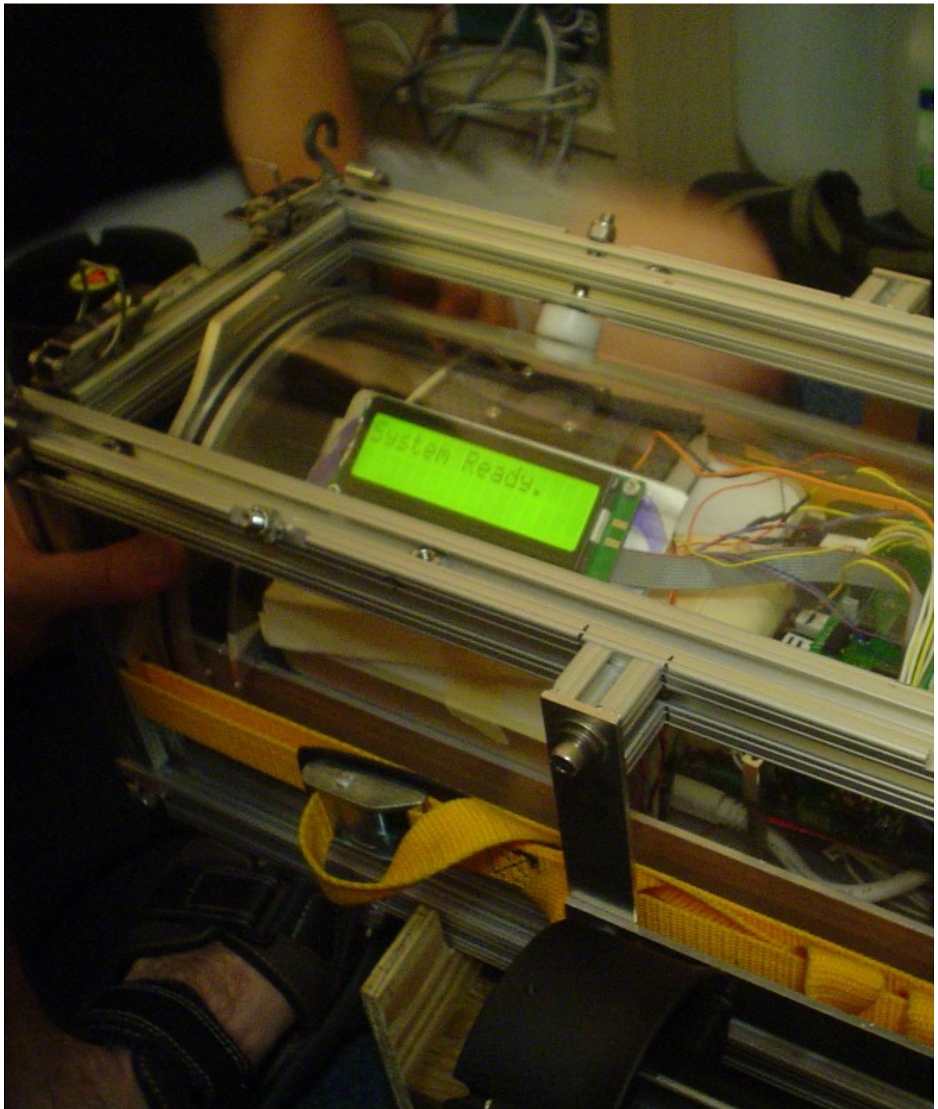
- Encoded in a pure python state machine
- Blocks are states, arrows are event driven transitions
- Easy to adapt and change after testing
- Release as the StatePy on PyPi!

GUI & Simulator



Above: Sim & Control interface done in wxPython & Python-Ogre

System Buildout



Above: *Tortuga I* at end of the 4 hour assembly at the 2007 competition

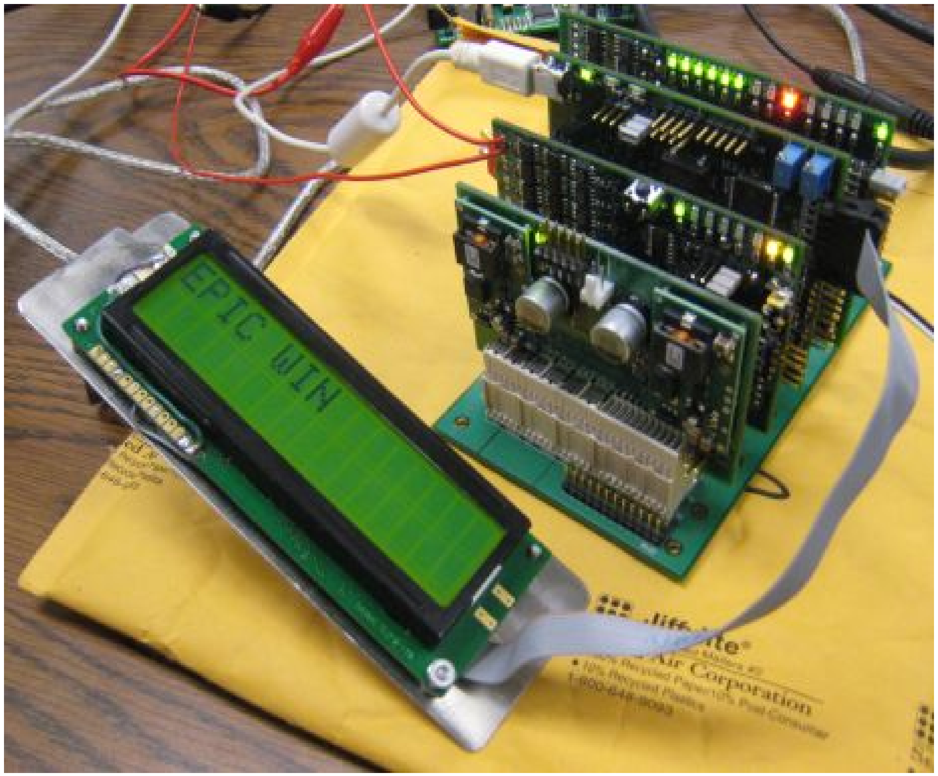
- Reduces start-up on new systems, a huge barrier for new developers
- Relies on pre-built dependencies for each platform
- Minimal use of native OS packages
- Places all files into a single directory (usually /opt/ram/local)
- Done with a single buildit based program upon first checkout

Dependency Management

- R@M uses ~20 open source tools and libraries
- SVN vendor branches
  - All deps in source R@M tree
- Build and package each dependency with builtit
- Manual upload to the server in platform based file tree
- Dependence on target platforms to be kept to a minimum
- Makes OS upgrades easy because you keep the same version of almost all dependencies

Above: Steve M. uses a screen-less laptop to remotely use his keyboard-less laptop to reprogram electronics

Conclusion



Above: First successful test of our custom electronics (before we fried them in the vehicle)

- Dynamic languages great fit for dynamics problems
- Python let us develop more functionality then a pure C++ would of
- Python let new developers contribute faster
- Competitions and Robotics are lots of fun, but so much more work then normal software

Thanks To These OSS Projects

OGRE

PYTHON-OGRE

python™

FANN

FFTW

FFmpeg

boost
C++ LIBRARIES

wxWidgets
Cross-Platform GUI Toolkit

wxPython
http://wxPython.org/
Cross-Platform GUI Library

OpenCV

libUSB

libdc1394

PyYAML

Thanks To The R@M Sponsors

A. James Clark School of Engineering
Department of Aerospace Engineering

BAE SYSTEMS

University of Maryland
Department of Computer Science

University of Maryland
Department of Electrical & Computer Engineering
A. James Clark School of Engineering

HOKUYO

Northrop Grumman
Defining the Future

EKFOX
Engineering. Ingenuity.

MEMSense

Lockheed Martin
We never forget who we're working for®

A. James Clark School of Engineering
The Institute for Systems Research

NSF

University of Maryland Student Government Association

J. Craig Venter™ Institute

Space Systems Laboratory
SSL