Penn Engineering

Oregon State University

A Platform for Online Monitoring of Autonomous Cars

Houssam Abbas

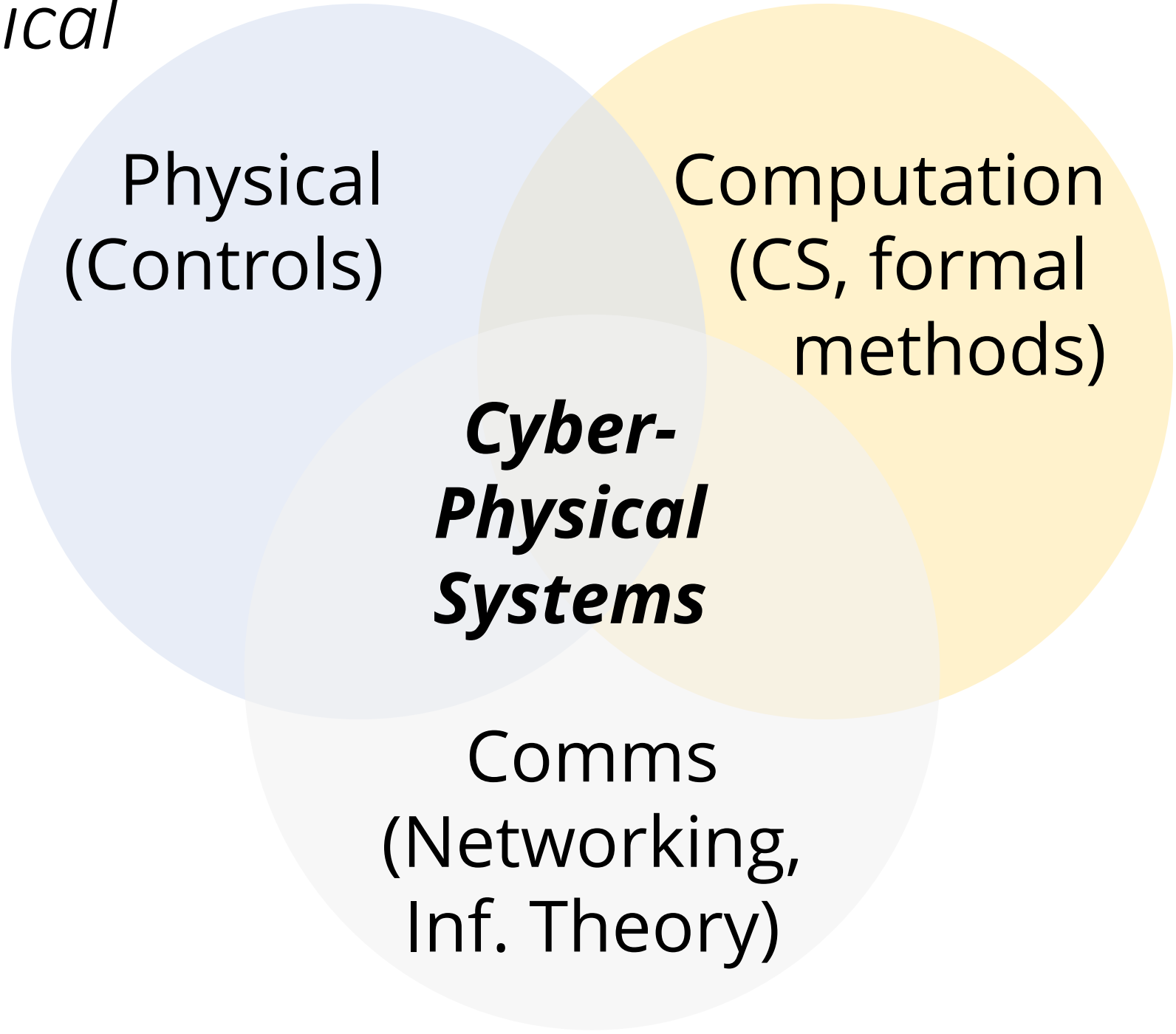
Postdoc, University of Pennsylvania

Assistant Professor, Oregon State University (Starting 2019)

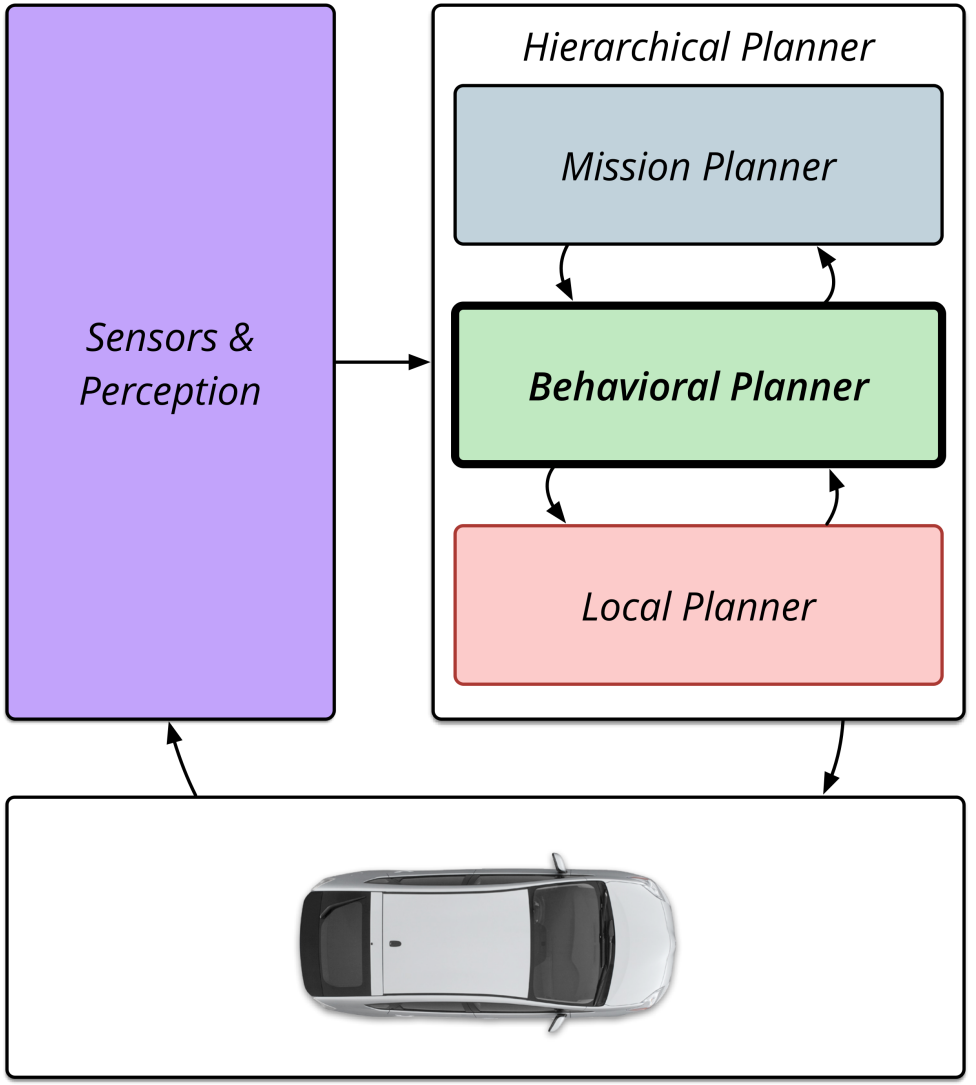
Joint work with Rahul Mangharam, Madhur Behl, and Matthew O'Kelly

Monitoring of Cyber-Physical Systems

Cyber-Physical Systems

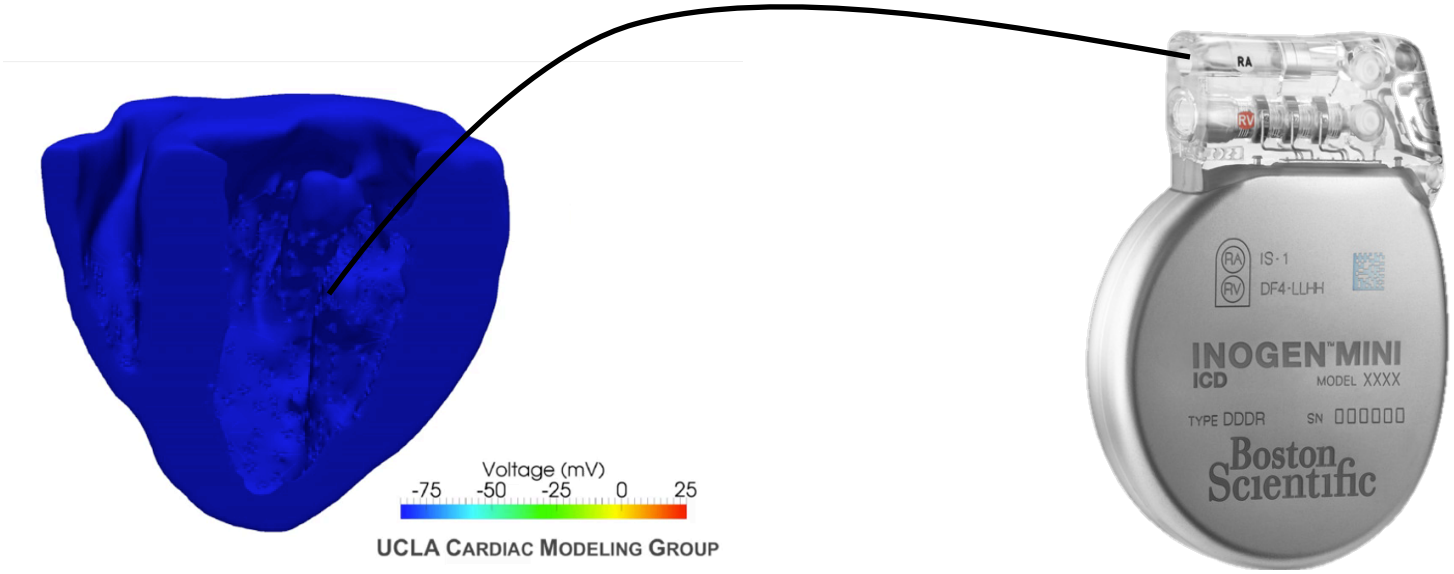


Autonomous vehicles



Smart grid

Autonomous medical devices



So you've developed a new monitor synthesis algo...

- You know the theoretical complexity

So you've developed a new monitor synthesis algo...

- You know the theoretical complexity – so what?

So you've developed a new monitor synthesis algo...

- You know the theoretical complexity – so what?
- You ran a generated monitor on your laptop – so what?

So you've developed a new monitor synthesis algo...

- You know the theoretical complexity – so what?
- You ran a generated monitor on your laptop – so what?
- You fed it recorded data – so what?

So you've developed a new monitor synthesis algo...

- You know the theoretical complexity – so what?
- You ran a generated monitor on your laptop – so what?
- You fed it recorded data – so what?

So you've developed a new monitor synthesis algo...

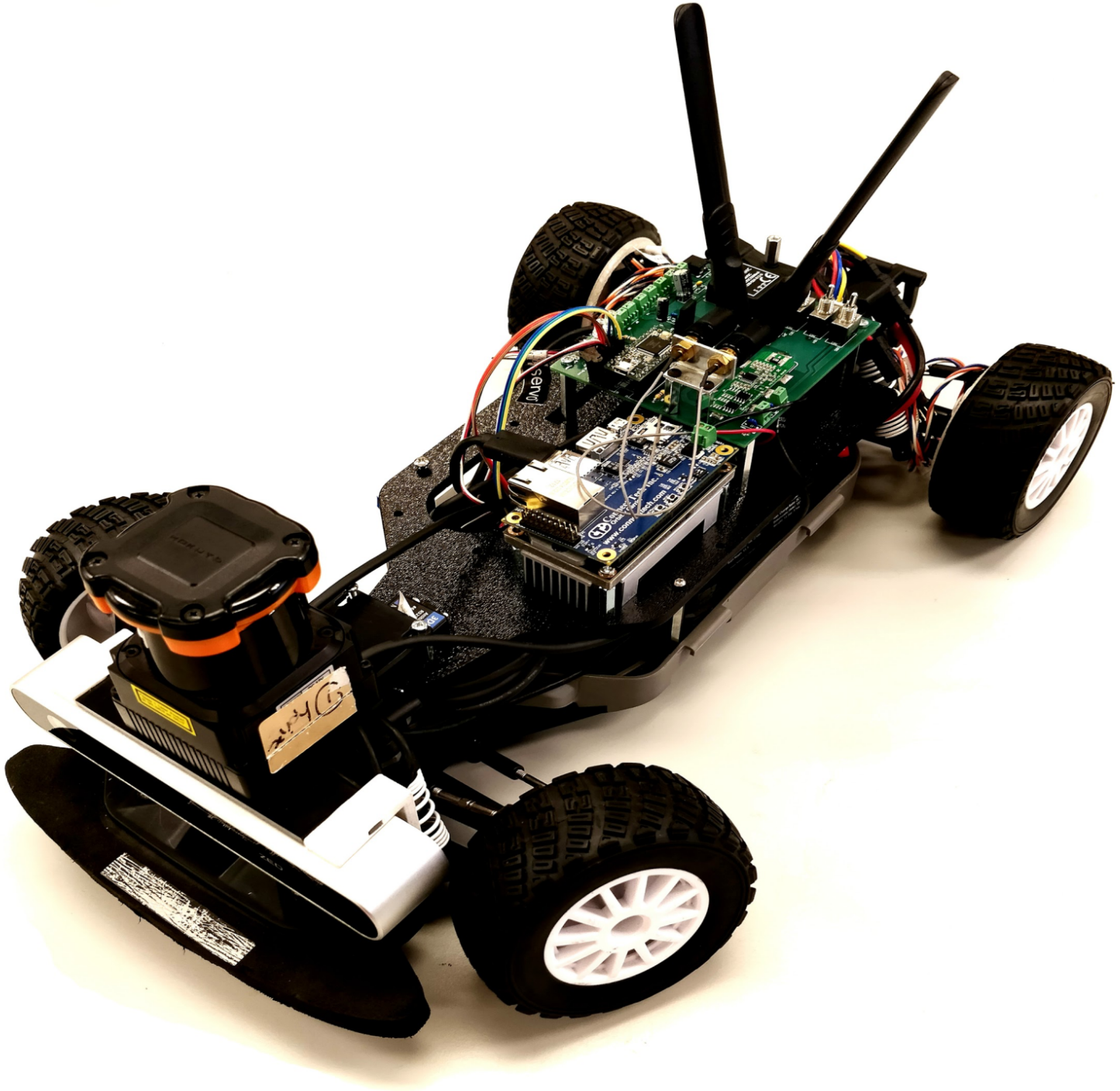
- You know the theoretical complexity – so what?
- You ran a generated monitor on your laptop – so what?
- You fed it recorded data – so what?
- You've demonstrated the code in a custom environment in your kitchen – so what?

Imagine testing your algorithm …

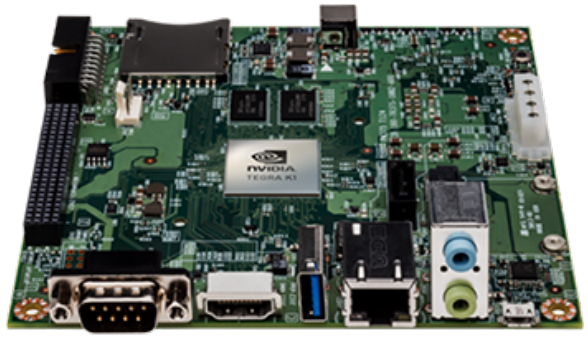
- ~~You know the theoretical complexity – so what?~~
- While measuring real runtime numbers
- ~~You ran a generated monitor on your laptop – so what?~~
- On the target hardware
- ~~You fed it recorded data – so what?~~
- With realistic software architecture to feed it data
- ~~You've demonstrated the code in a custom environment in your kitchen – so what?~~
- In an accepted and widely used environment

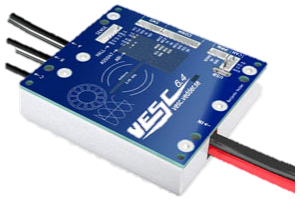
It doesn’t get better than…

Running on the real robot

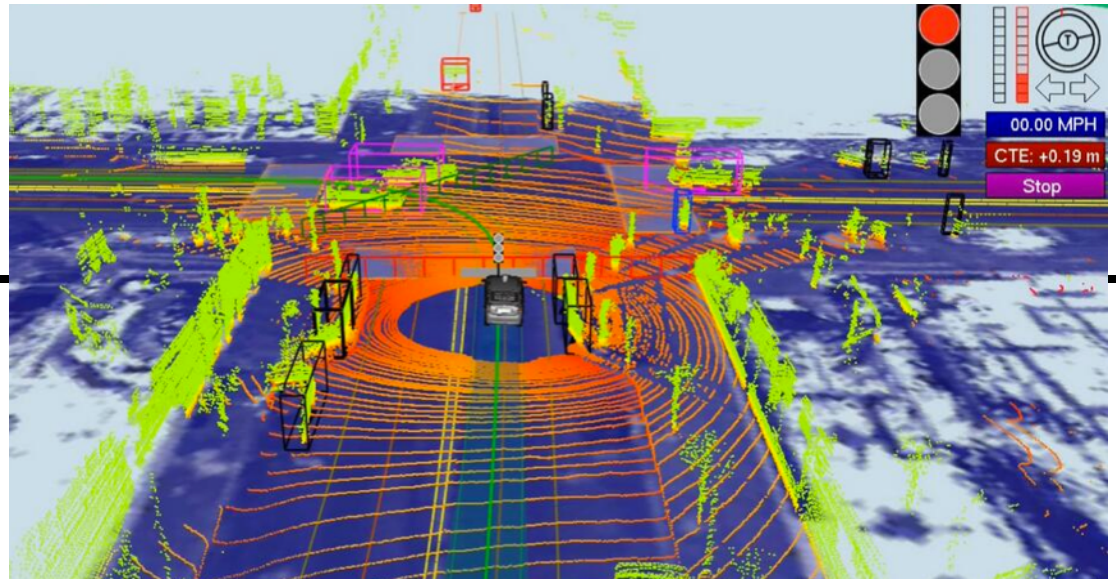


Real autonomous car!

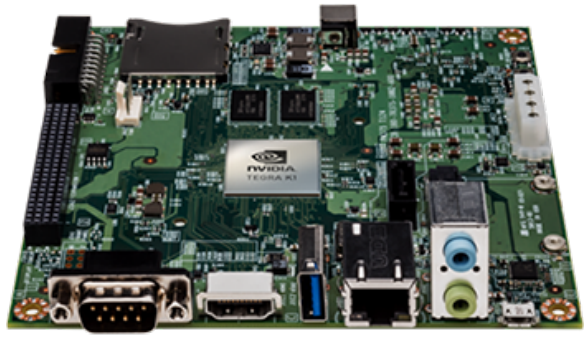


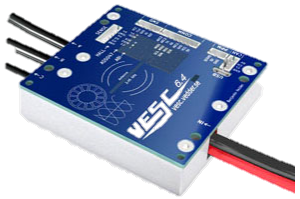


PERCEIVE → PLAN → ACT

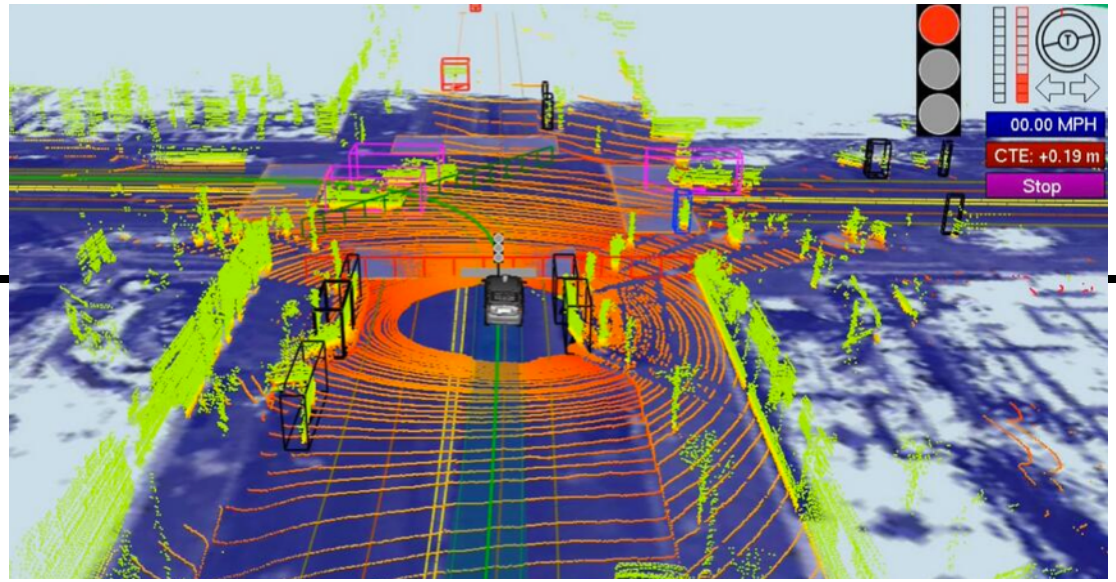


Big Bad World





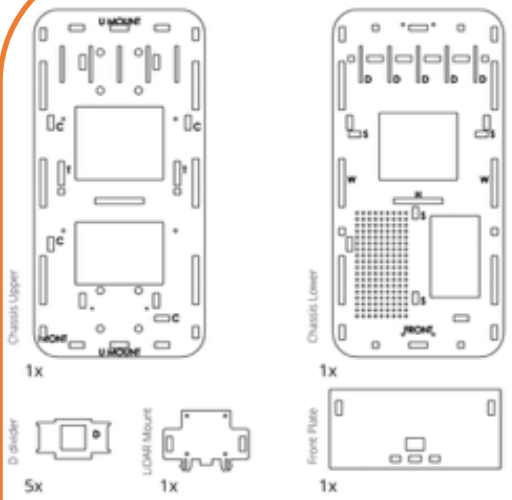
PERCEIVE → PLAN & MONITOR → ACT

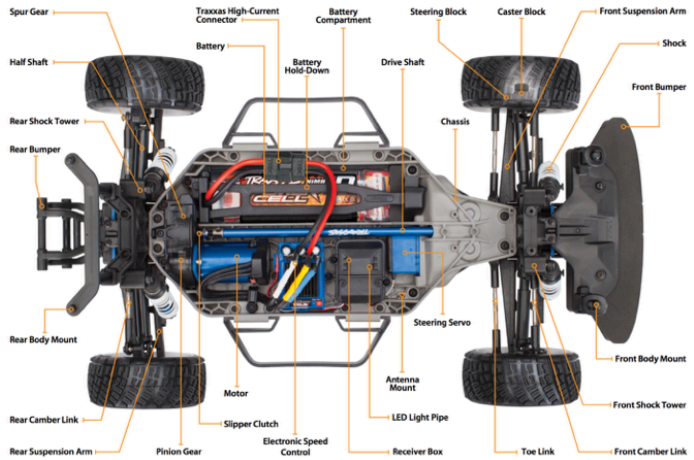


Big Bad World

The F1/10 Autonomous Race Car and Simulator

f1tenth.org





Chassis Design

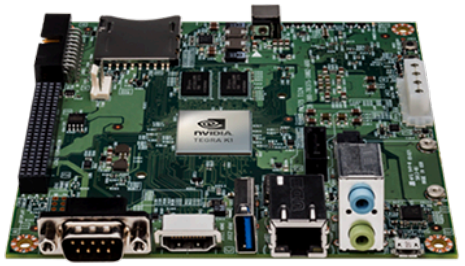
LiDAR

Camera

IMU

IR Depth Cameras

Wi-fi Telemetry

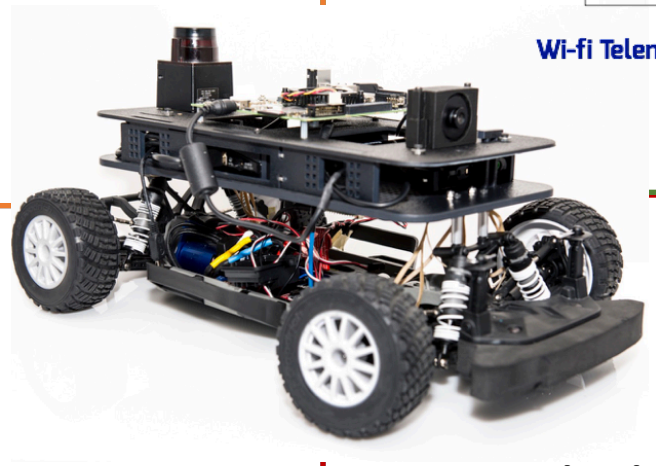


Onboard Computer

Motor Controller

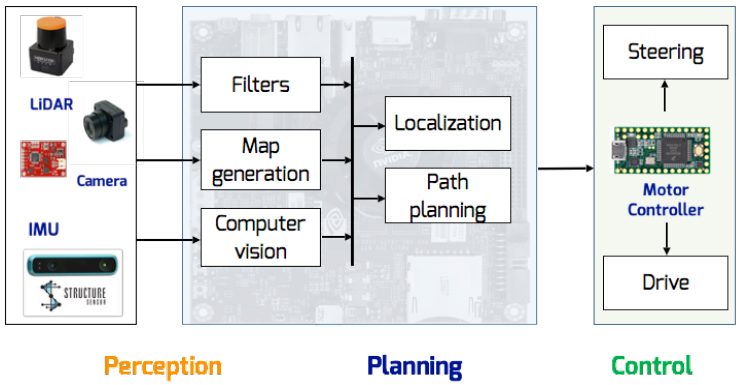
Battery

Sensor Integration

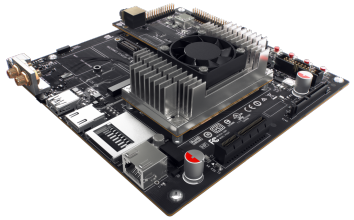


Software System Architecture

ROS



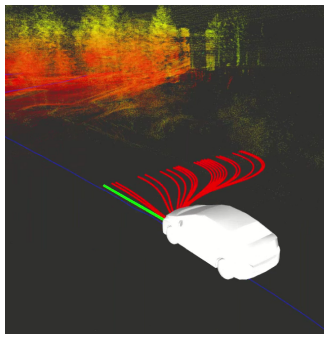
GPU accelerated libraries



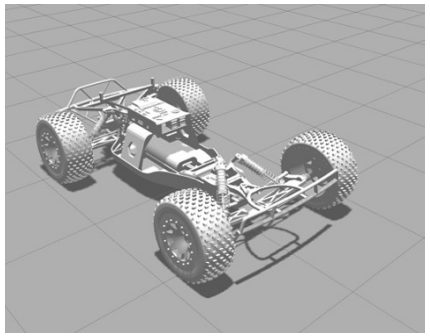
Simulation Tool

Scenario Sim

ROSbag data



Vehicle and environment models in Gazebo



F1/10 Race Car Assembly (Time Lapse)

Open Source


F1/10
Build
Drive
Race
About
Rules
Forum
Crew
Sponsor
Sign Up
F1/10
Autonomous Racing Competition
Click here to sign up for the 2018 competition at CPS Week! Otherwise subscribe to our mailing list below...
email address
Subscribe

f1tenth.org

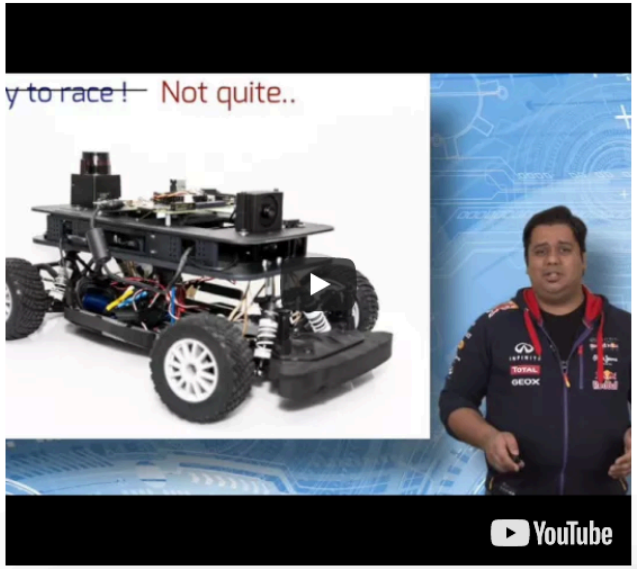
F1/10 BUILD / DRIVE / RACE About Rules Forum Crew Sponsor Sign Up

Lecture 1.1 - Course Overview

View on YouTube

Download PDF

Lecture 1.2 - Getting Started



View on YouTube

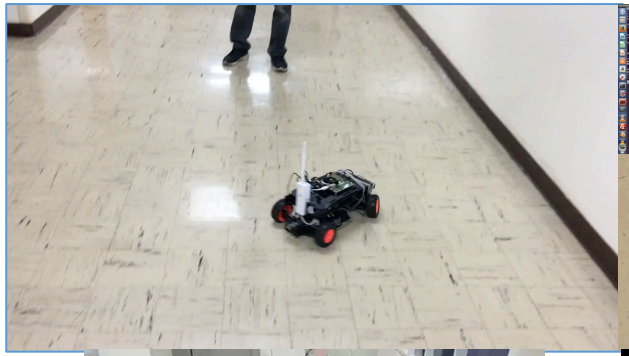
Download PDF

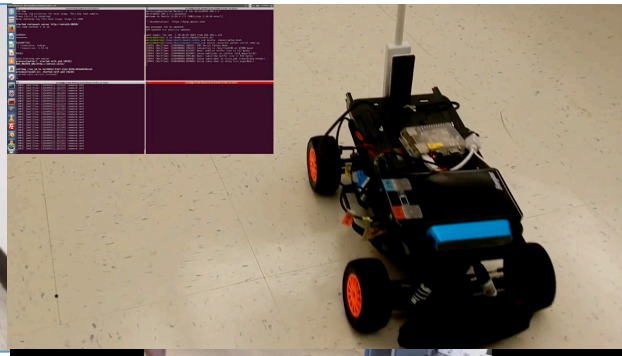
Arizona State University · The University of New Mexico · ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

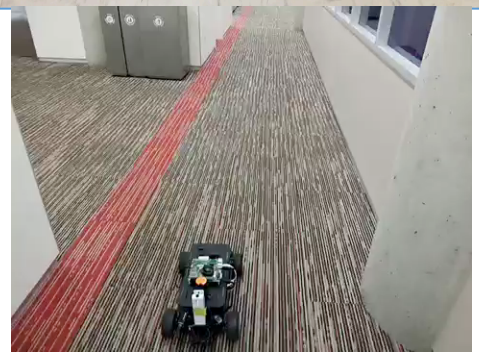
The University of Texas at Austin · SJSU San José State University · KTH Royal Institute of Technology

UC Riverside · UCLA · KAIST

UConn University of Connecticut · Clemson University · Temple University



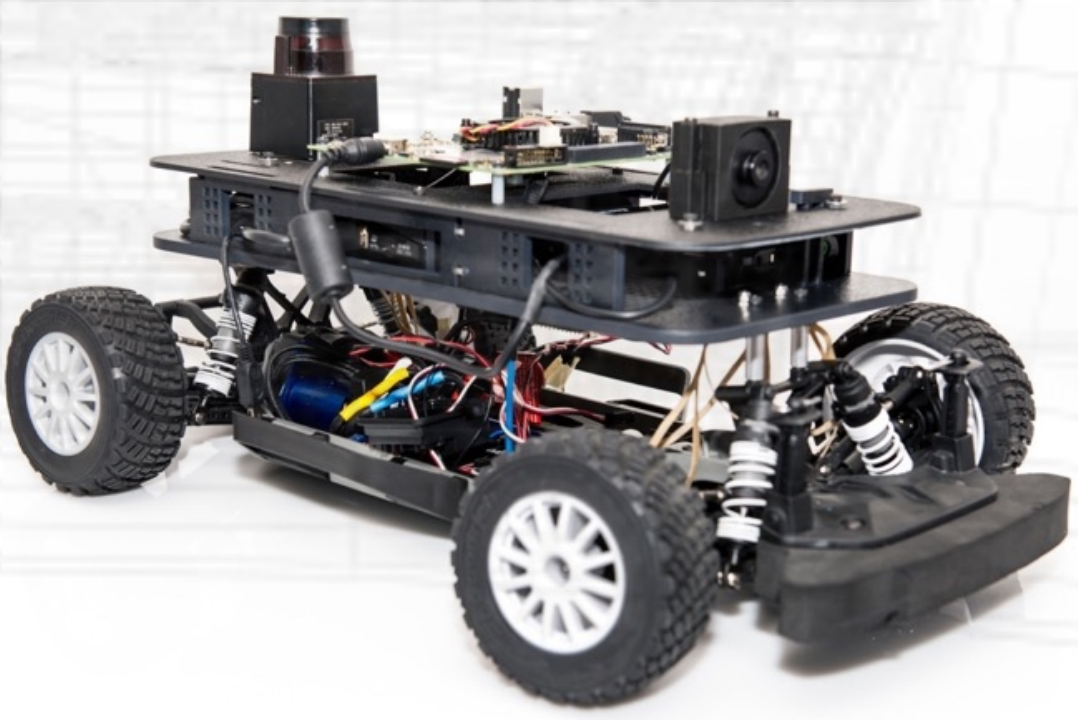




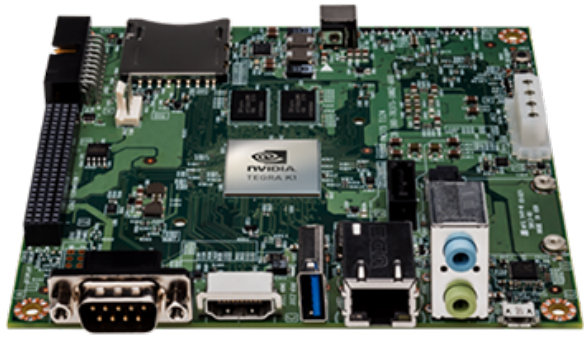
F1/10

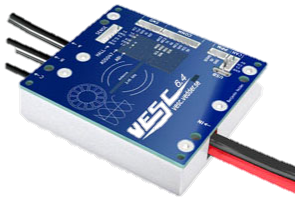
1/10th the scale.10 times the fun!

Education **Research** **Competition**

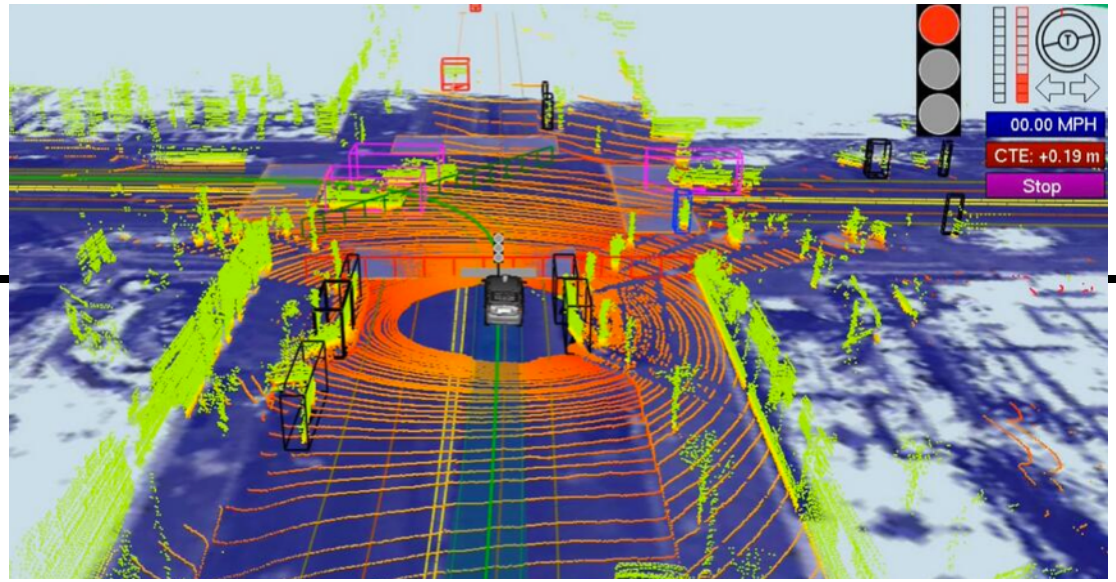



So you can’t build the car just yet…
4ever.eu

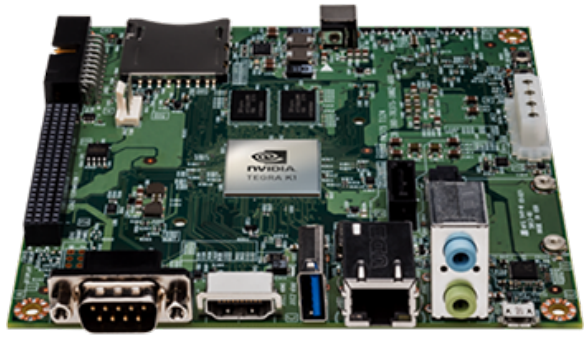


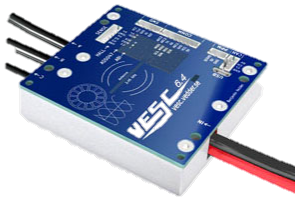


PERCEIVE → PLAN & MONITOR → ACT



Big Bad World





PERCEIVE

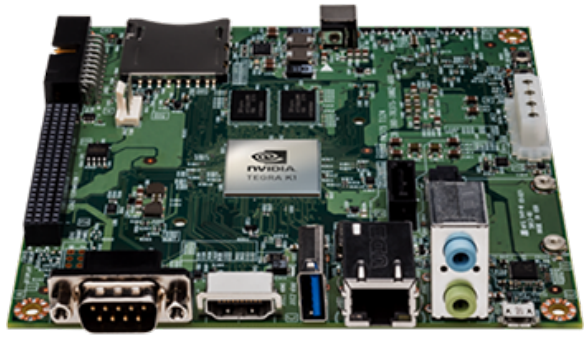
PLAN & MONITOR

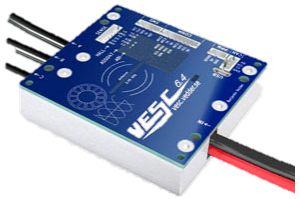
ACT

SIMULATOR

SENSOR DATA

STEERING AND ACCELERATION





PERCEIVE

PLAN & MONITOR

ACT

ACTUAL CODE THAT WILL BE DEPLOYED

SIMULATOR

SENSOR DATA

STEERING AND ACCELERATION

PERCEIVE

PLAN & MONITOR

ACT

ACTUAL CODE THAT WILL BE DEPLOYED
RUNNING ON TARGET HARDWARE

SIMULATOR

SENSOR DATA

STEERING AND ACCELERATION

PERCEIVE

PLAN & MONITOR

ACT

ACTUAL CODE THAT WILL BE DEPLOYED
RUNNING ON TARGET HARDWARE
WITHIN THE SAME SW ARCHITECTURE

SIMULATOR

SENSOR DATA

STEERING AND ACCELERATION

Starting September 2018, you will be able to do this:

(first, get a computer running Ubuntu 16.04 – or install a Virtual Machine running the same)

(and install ROS – super easy, instructions at ros.org, and they work!)

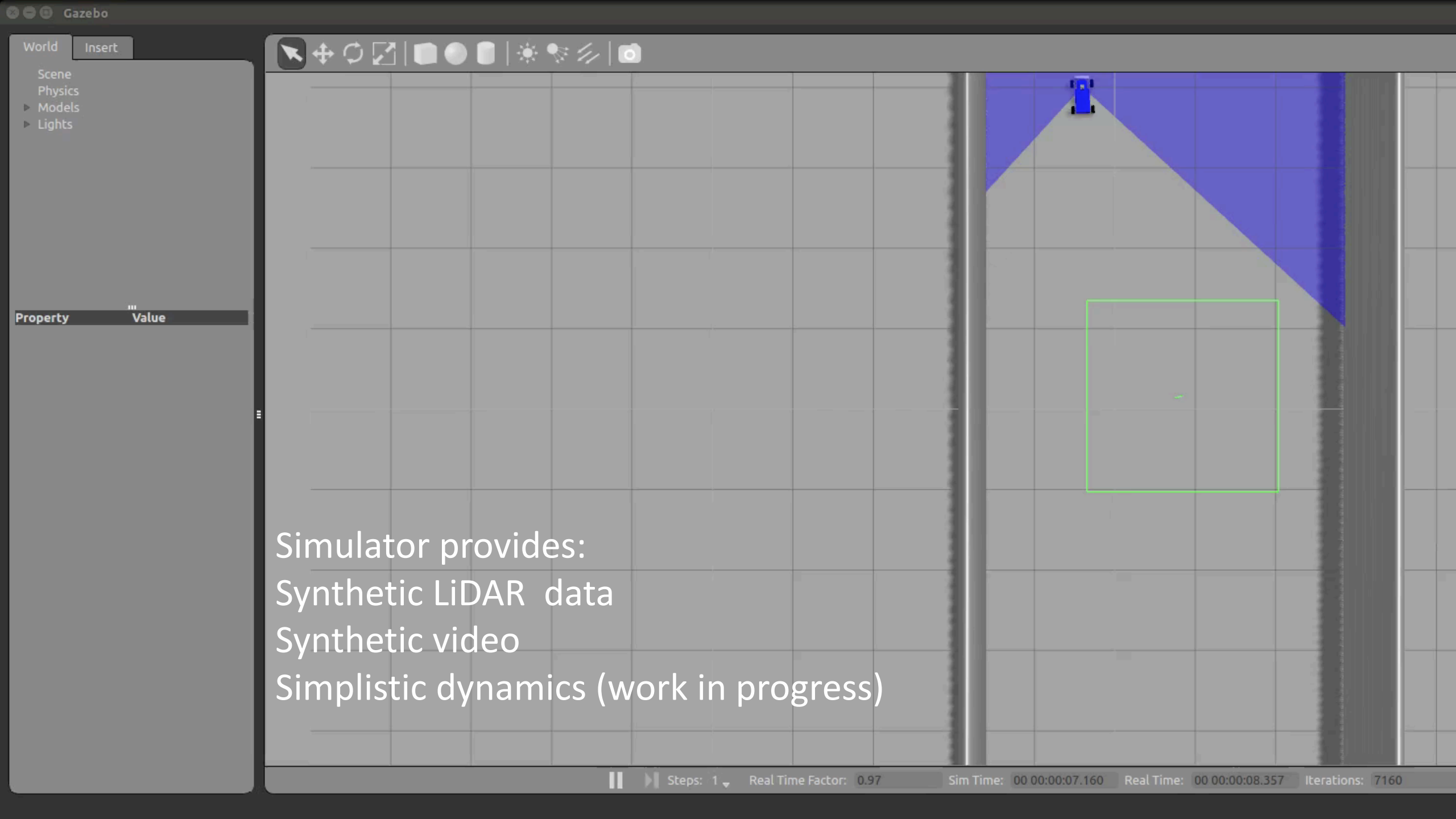
```
$ cd ~/sandbox
$ git clone \
  https://github.com/mlab-upenn/f110-upenn-course.git
$ mkdir -p sims_ws/src
$ cp -r f110-upenn-course/simulator/* sims_ws/src/
$ sudo apt-get install ros-kinetic-ackermann-msgs \
    ros-kinetic-ros-control \
    ros-kinetic-ros-controllers \
    ros-kinetic-gazebo-ros-control \
    ros-kinetic-joy
$ chmod +x sims_ws/src/simulator/wall_following/scripts/*.py
$ catkin_make

$ roslaunch wall_following wall_following.launch
```

Get code

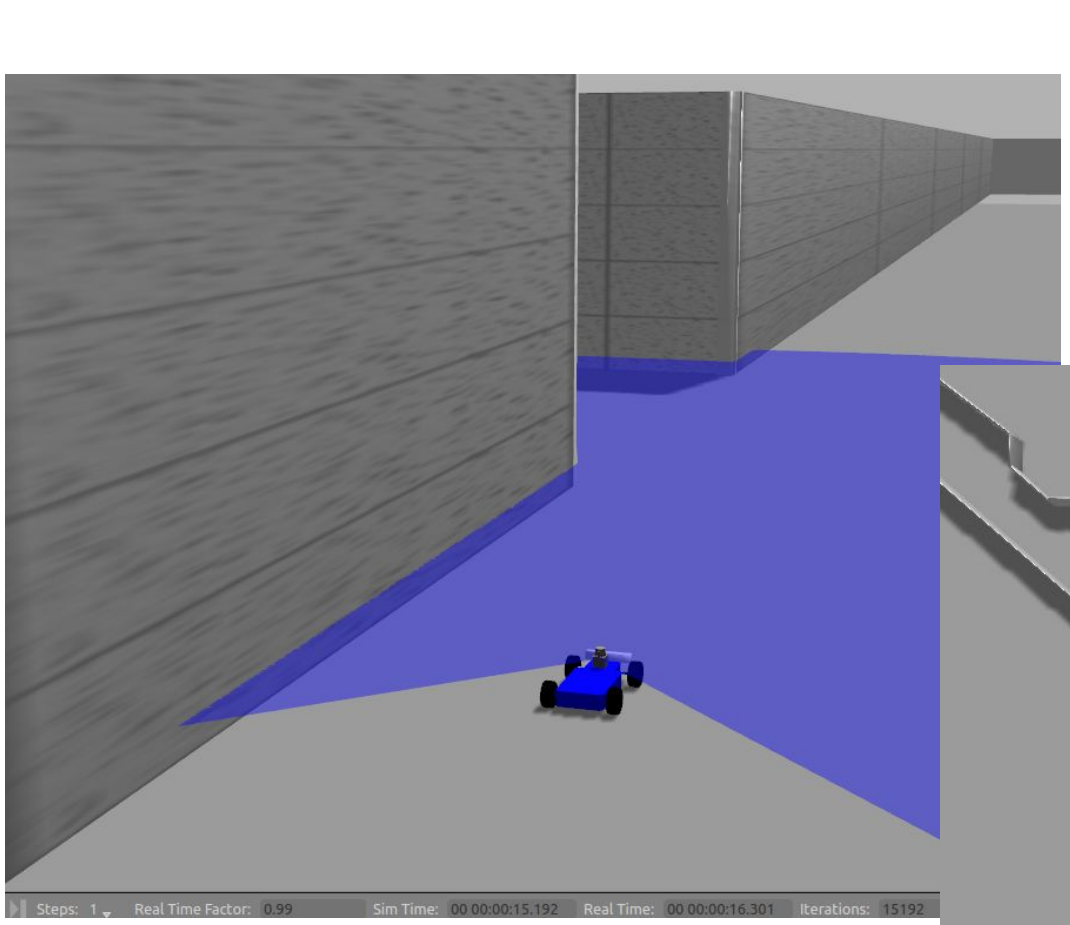
Get some packages

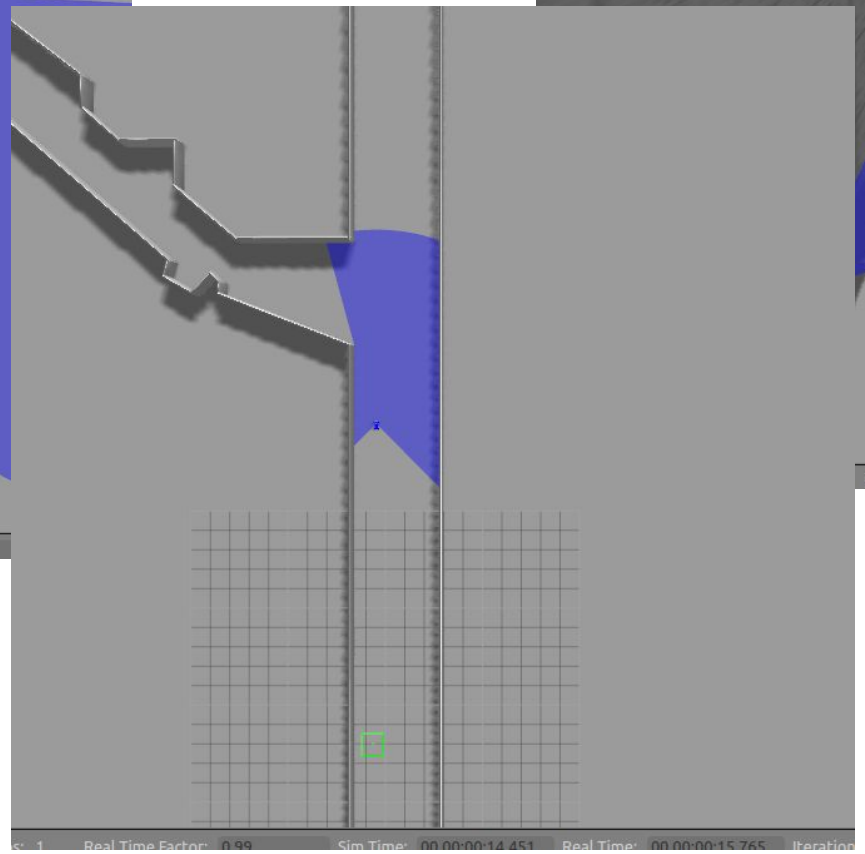
Compile

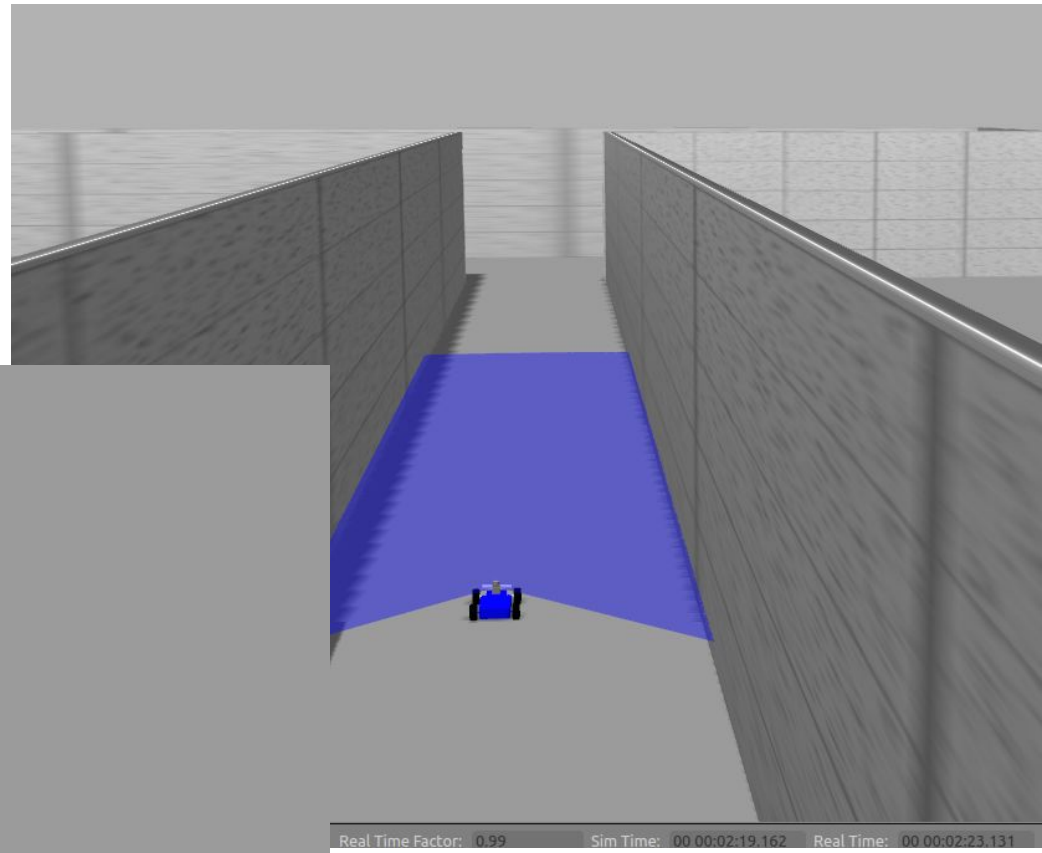
Run


Gazebo
World
Insert
Scene
Physics
Models
Lights
Property
Value
Steps: 1
Real Time Factor: 0.97
Sim Time: 00 00:00:07.160
Real Time: 00 00:00:08.357
Iterations: 7160
Simulator provides:
Synthetic LiDAR data
Synthetic video
Simplistic dynamics (work in progress)

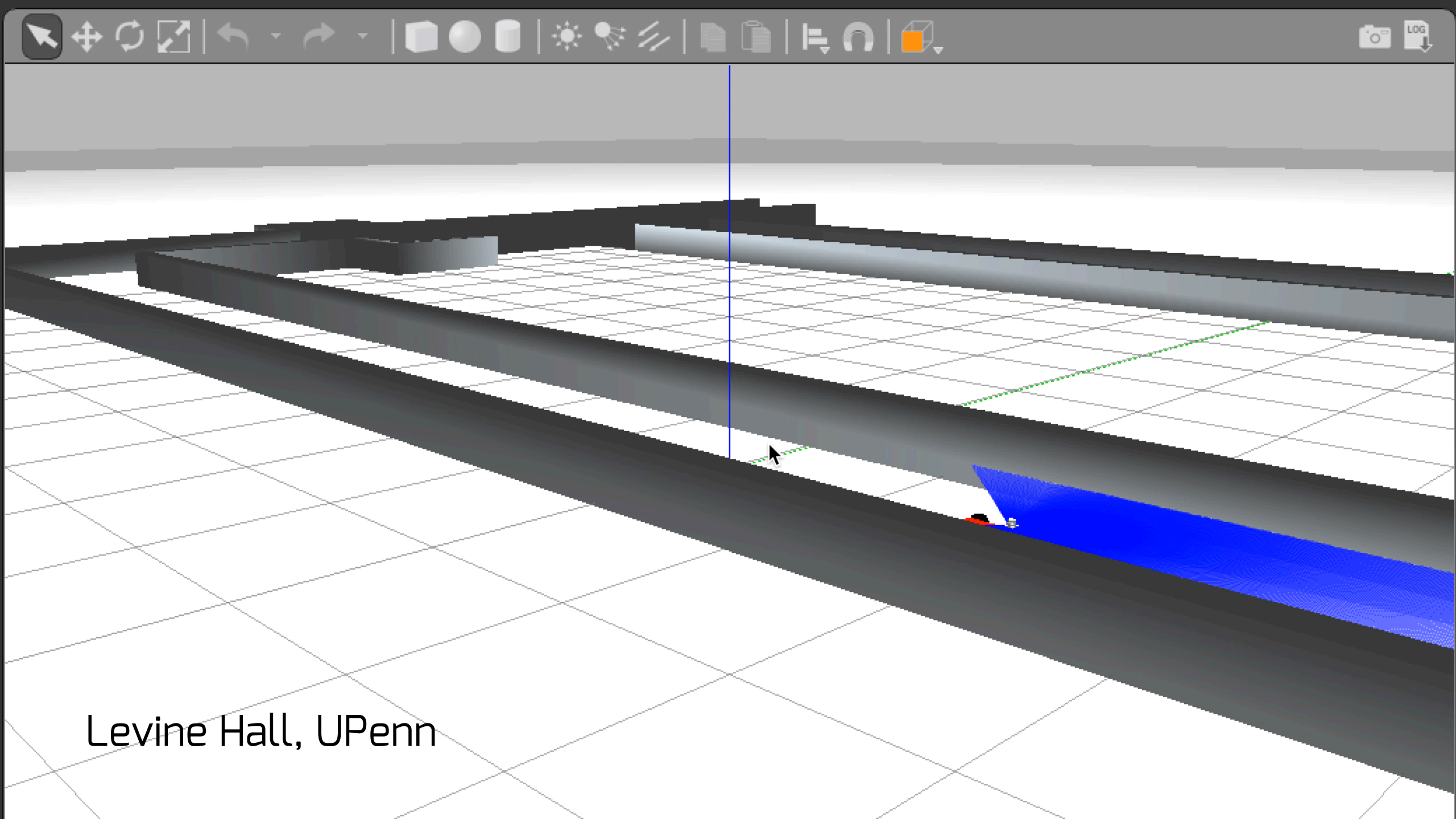
Choice of map

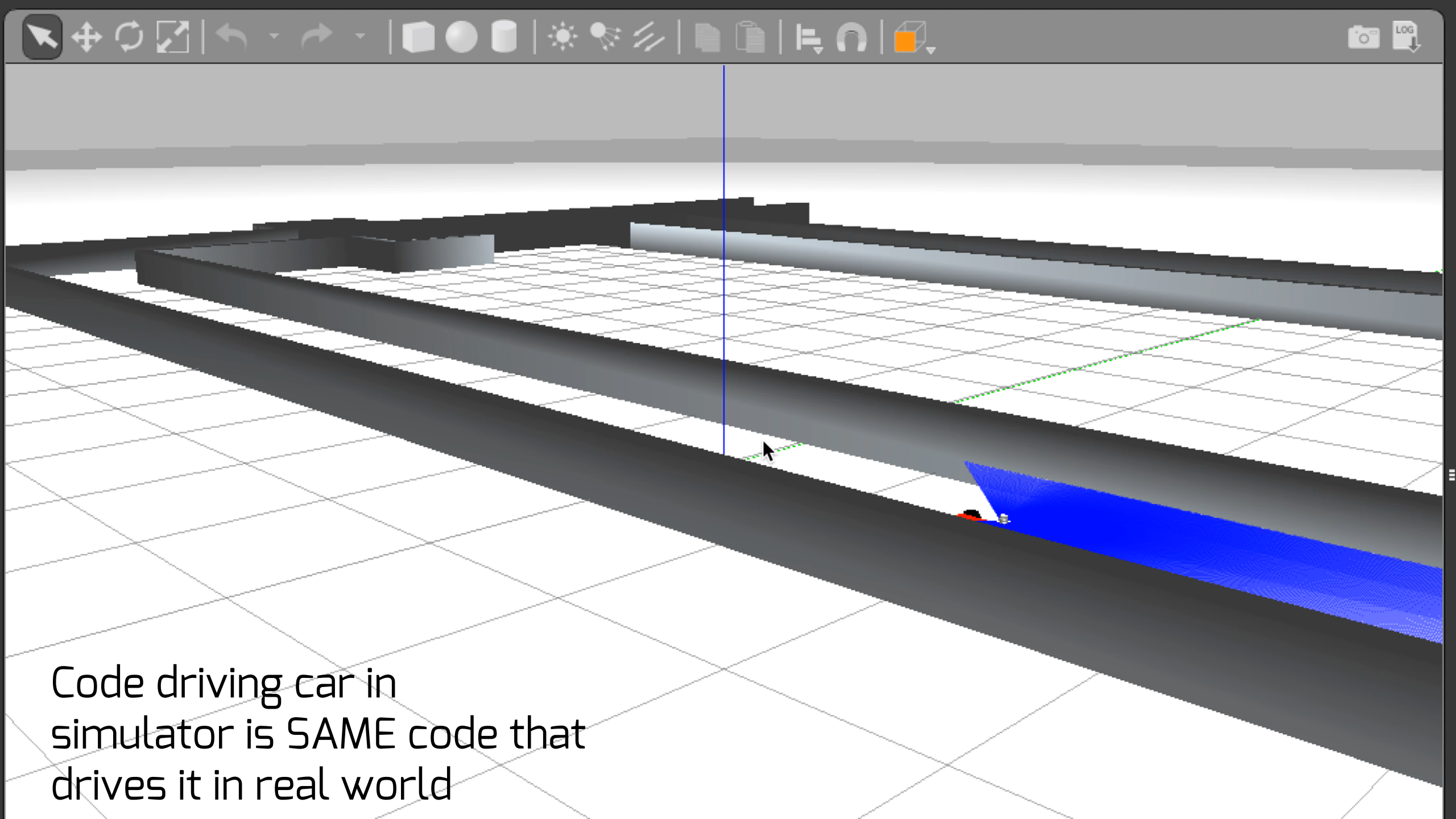


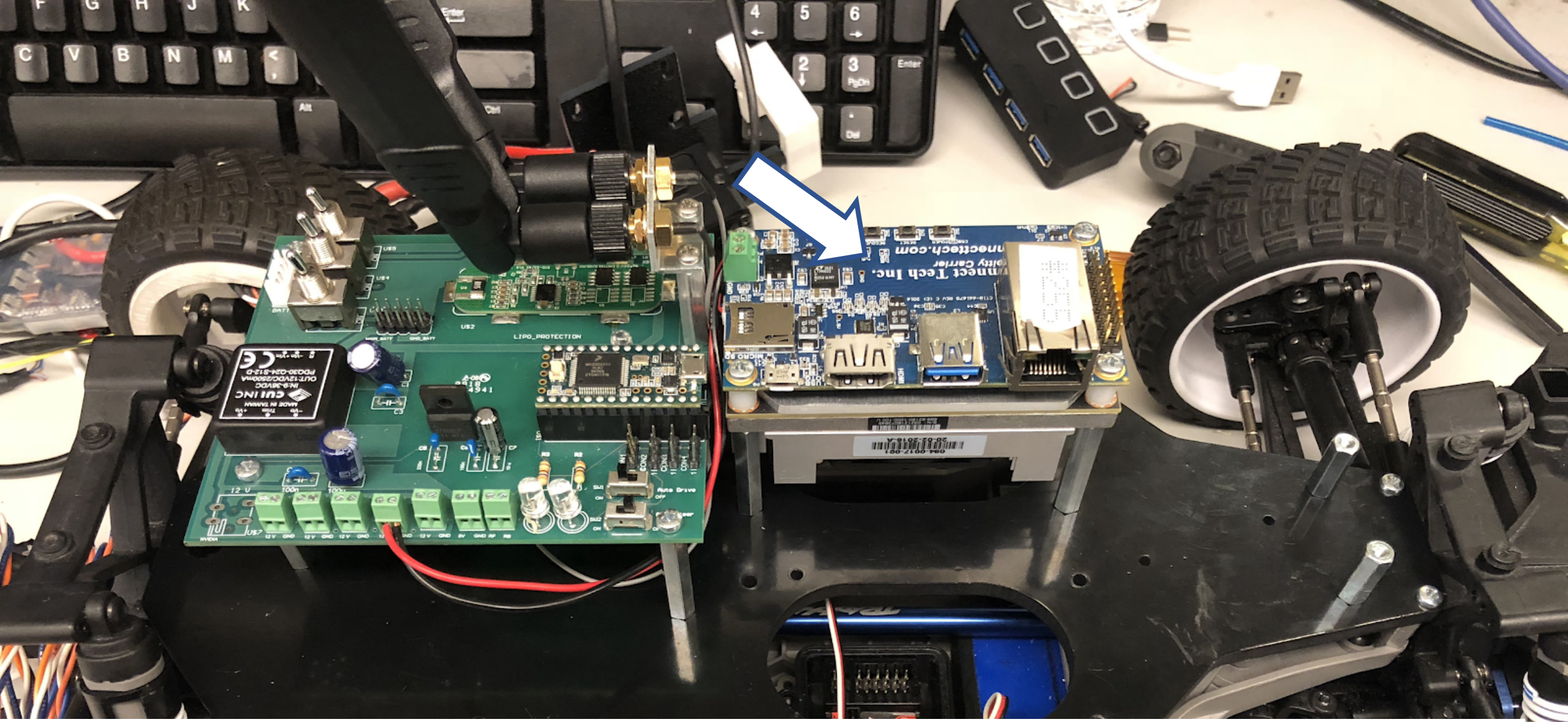




MIT Tunnels


Levine Hall, UPenn


Code driving car in
simulator is SAME code that
drives it in real world

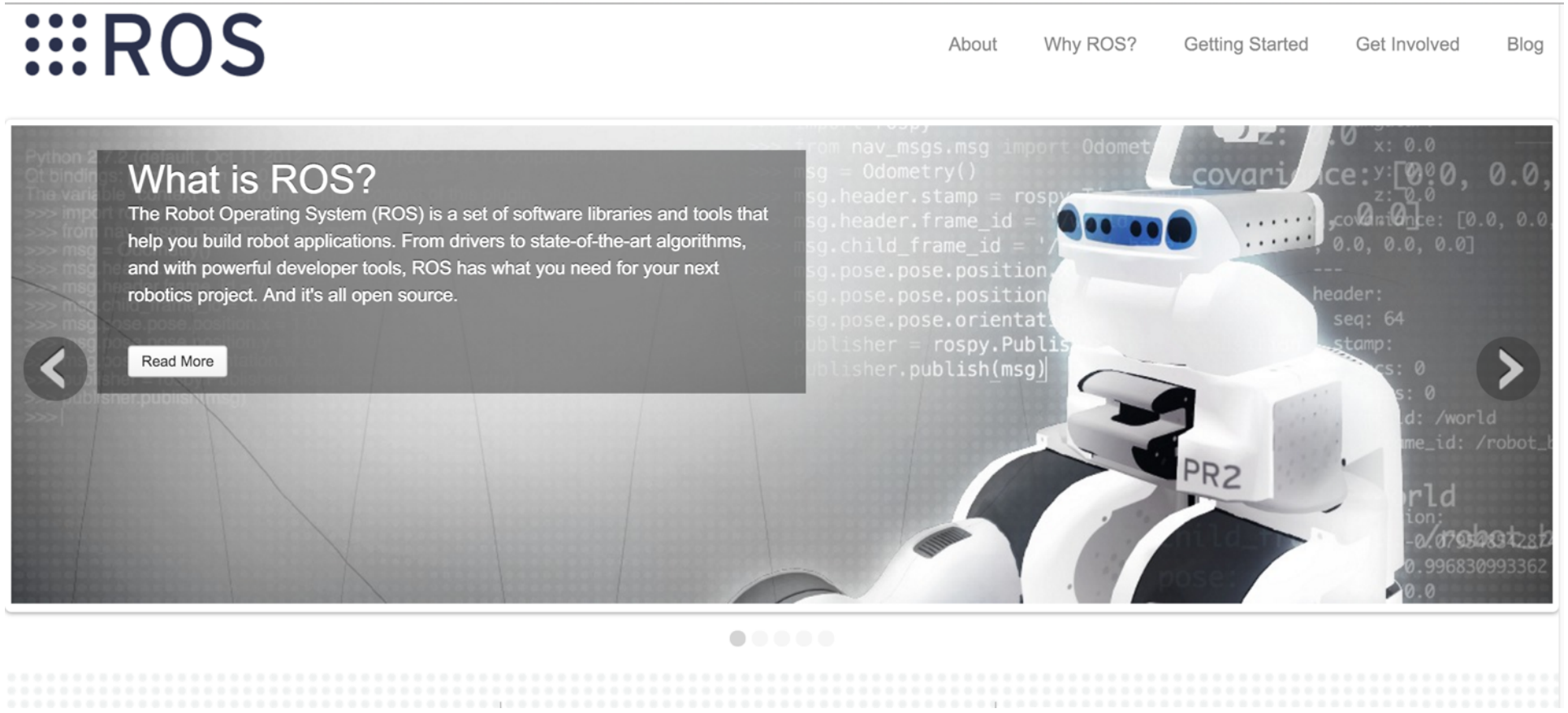


That same code runs on the computer (TX1) that will be on-board the car

Imagine testing your algorithm …

- While measuring real runtime numbers on the platform hardware
- With realistic software architecture
- Implemented in a supported language
- In an accepted and widely used environment

ROS: Robot Operating System

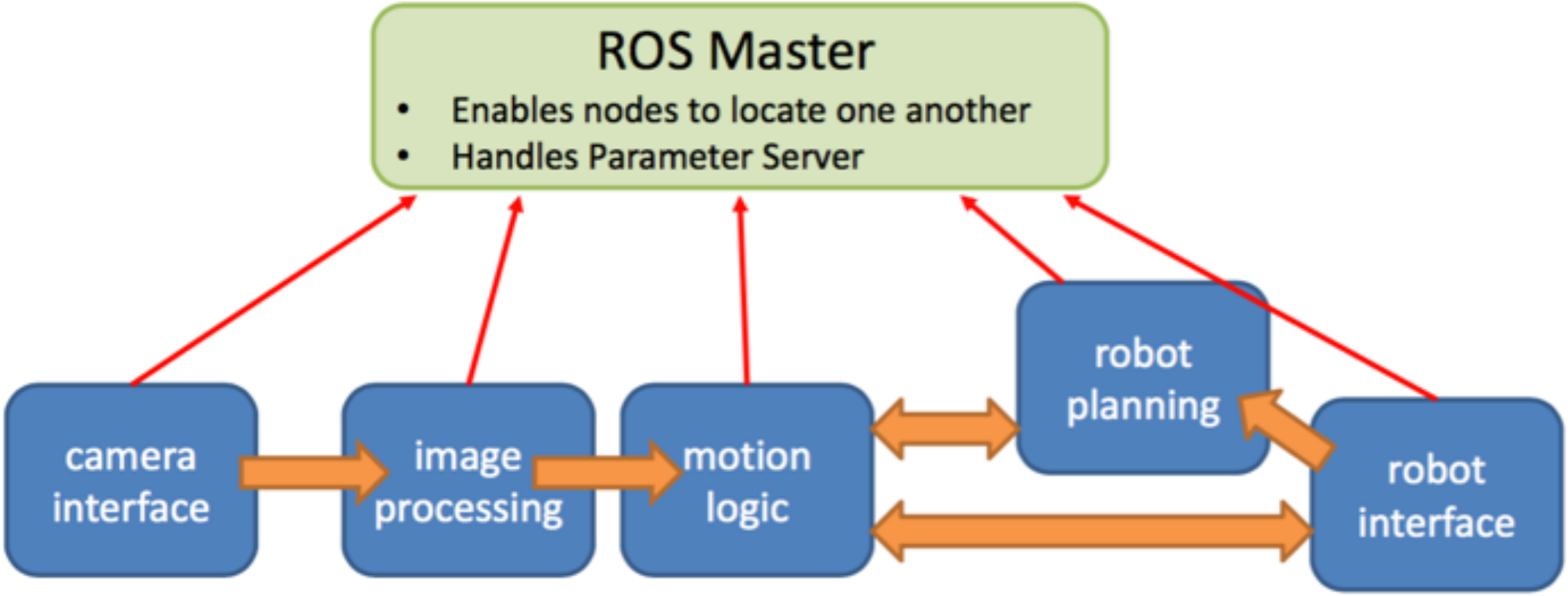


ROS.org

Open Source Robotics Foundation

ROS master and nodes

Node: a single process with a specific functionality



Communication between nodes

Nodes communicate messages via topics
in an asynchronous publish-subscribe model

LaserScan (Message)

scan (topic)

Message1
Message2
...

Mapping Node

Publisher Node

Subscriber Node

Communication between nodes

Nodes communicate messages via topics
in an asynchronous publish-subscribe model

LaserScan (Message)

scan (topic)

Message1
Message2
...

Mapping Node

Monitor Node

Publisher Node

Subscriber Nodes

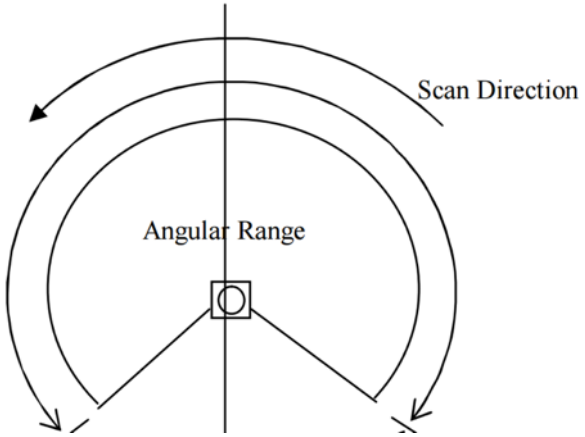
Messages: data structures

LaserScan [Message]

scan [topic]

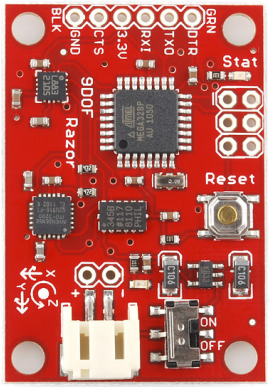
Message1
Message2
...

Monitor Node



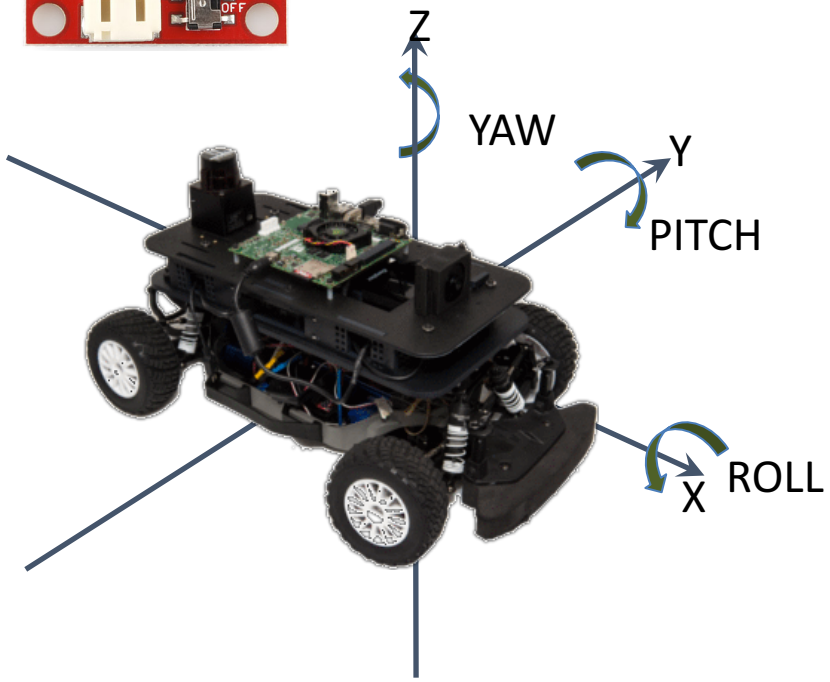
std_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities

Messages: data structures



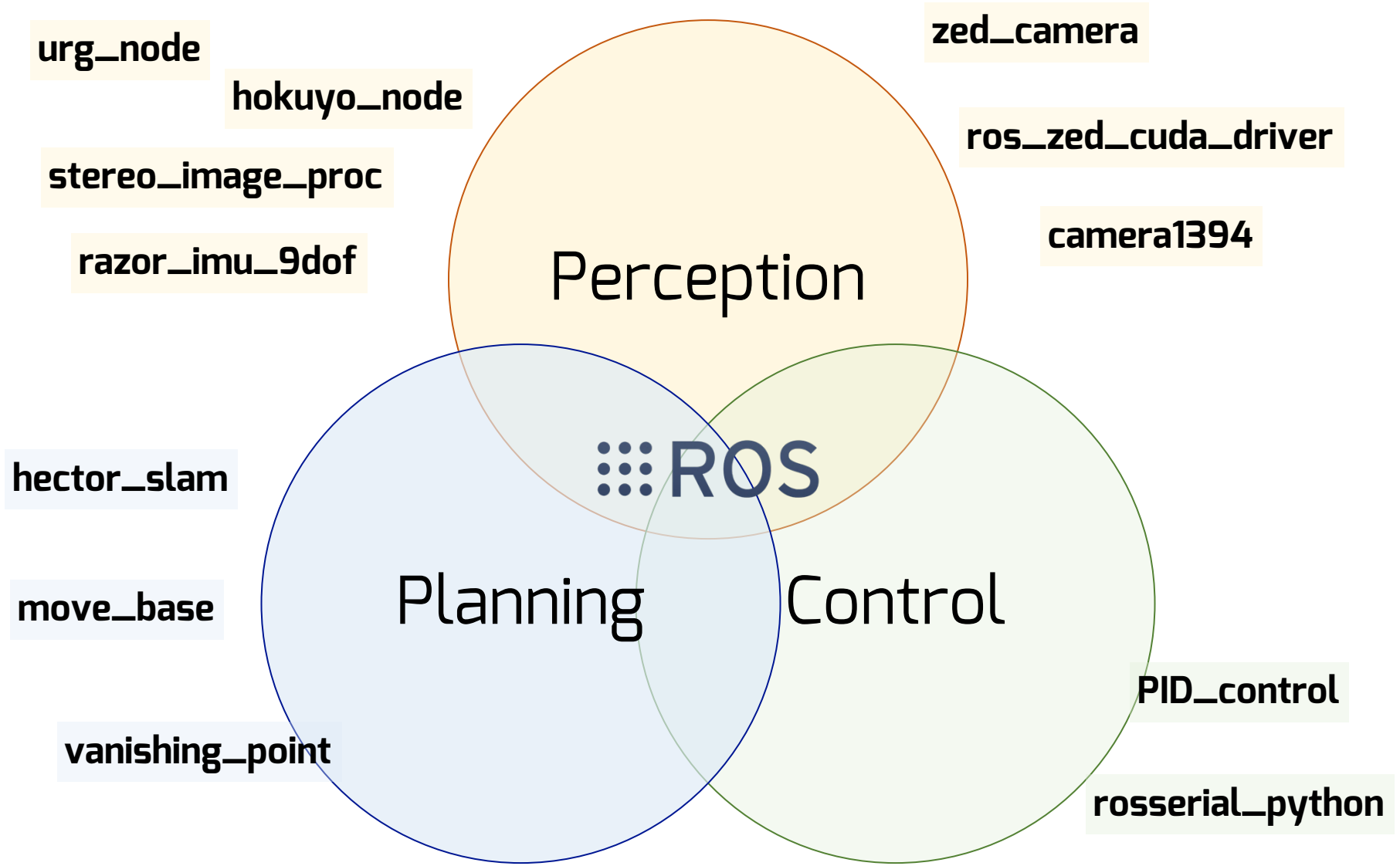
Message1
Message2
...

Localization Node




```
# Message
sensor_msgs/imu
geometry_msgs/Quaternion orientation
float64[9] orientation_covariance

geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance

geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance
```

ROS Capabilities



Imagine testing your algorithm …

- While measuring real runtime numbers on the platform hardware
- With realistic software architecture
- In an accepted and widely used environment

Creating a new monitor manually

- Create .cpp or .py monitor file
- Edit two configuration files
- Compile
- Voilà!

Starting September 2018, you will be able to do this:

(first, get a computer running Ubuntu 16.04 – or install a Virtual Machine running the same)

(and install ROS – super easy, instructions at ros.org, and they work!)

```
$ cd ~/sandbox
$ cp -r \
  f110-upenn-course/algorithms/runtime_monitoring/ \
  sims_ws/src/

$ catkin_make

$ roslaunch wall_following wall_following.launch
# in a new terminal
$ source devel/setup.bash
$ rosrun runtime_monitoring moussa_sim_monitor
```

Get code

Compile

Run

Monitor synthesis

- Work in progress, with Dogan Ulus, on monitor-synthesis-to-ROS
- Study performance of synthesized monitors in the ROS environment

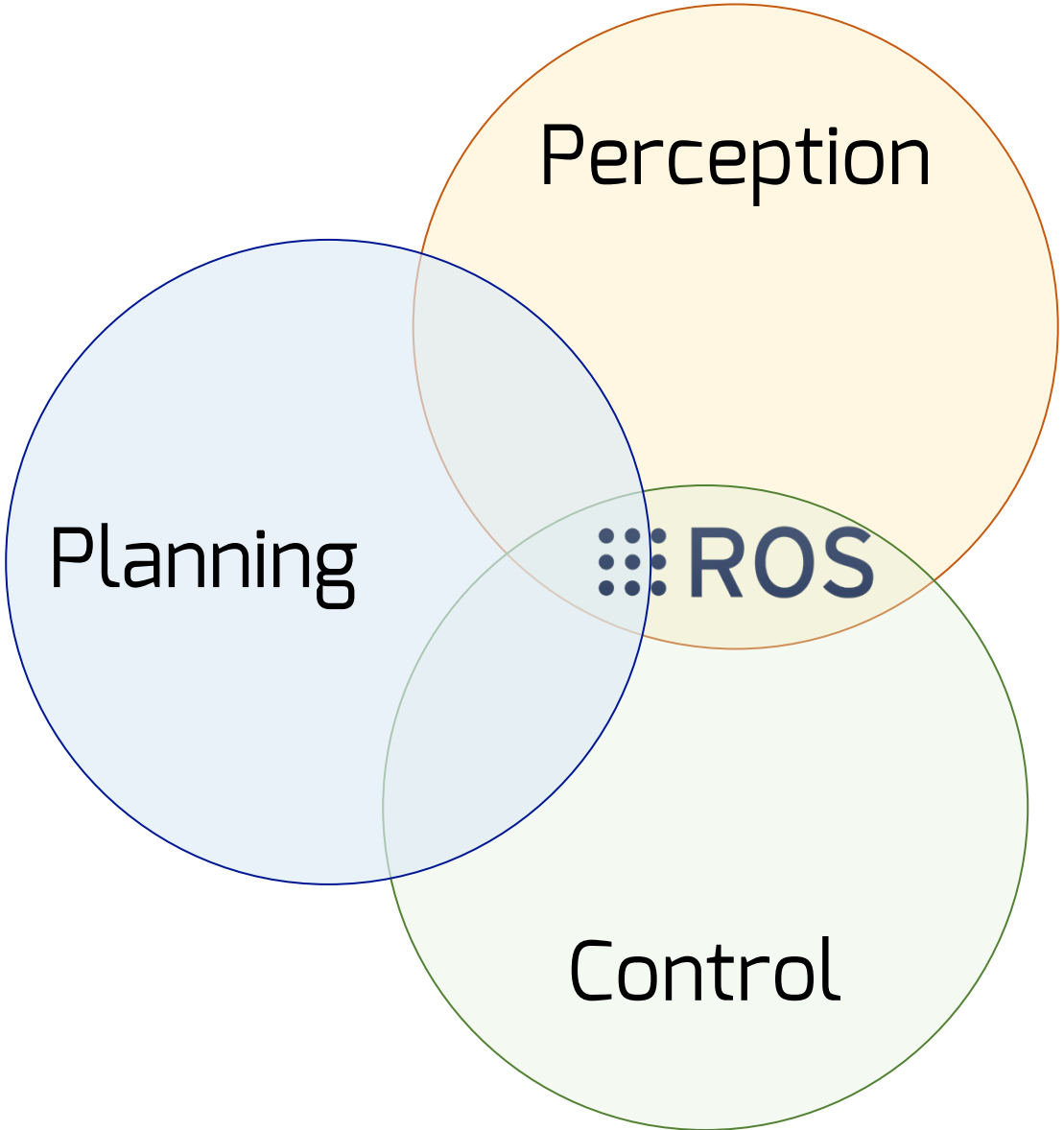
```
$ cd ~/sandbox/sims_ws/
$ tl2cpp -with-headers “always[1,4] gt(x:float,4)” \
   --outdir src/runtime_monitoring/include

# Edit CMakeLists.txt and package.xml

$ catkin_make

$ roslaunch wall_following wall_following.launch
# In a new terminal
$ source devel/setup.bash
$ rosrun runtime_monitoring ree_tl_1
```

Generate code

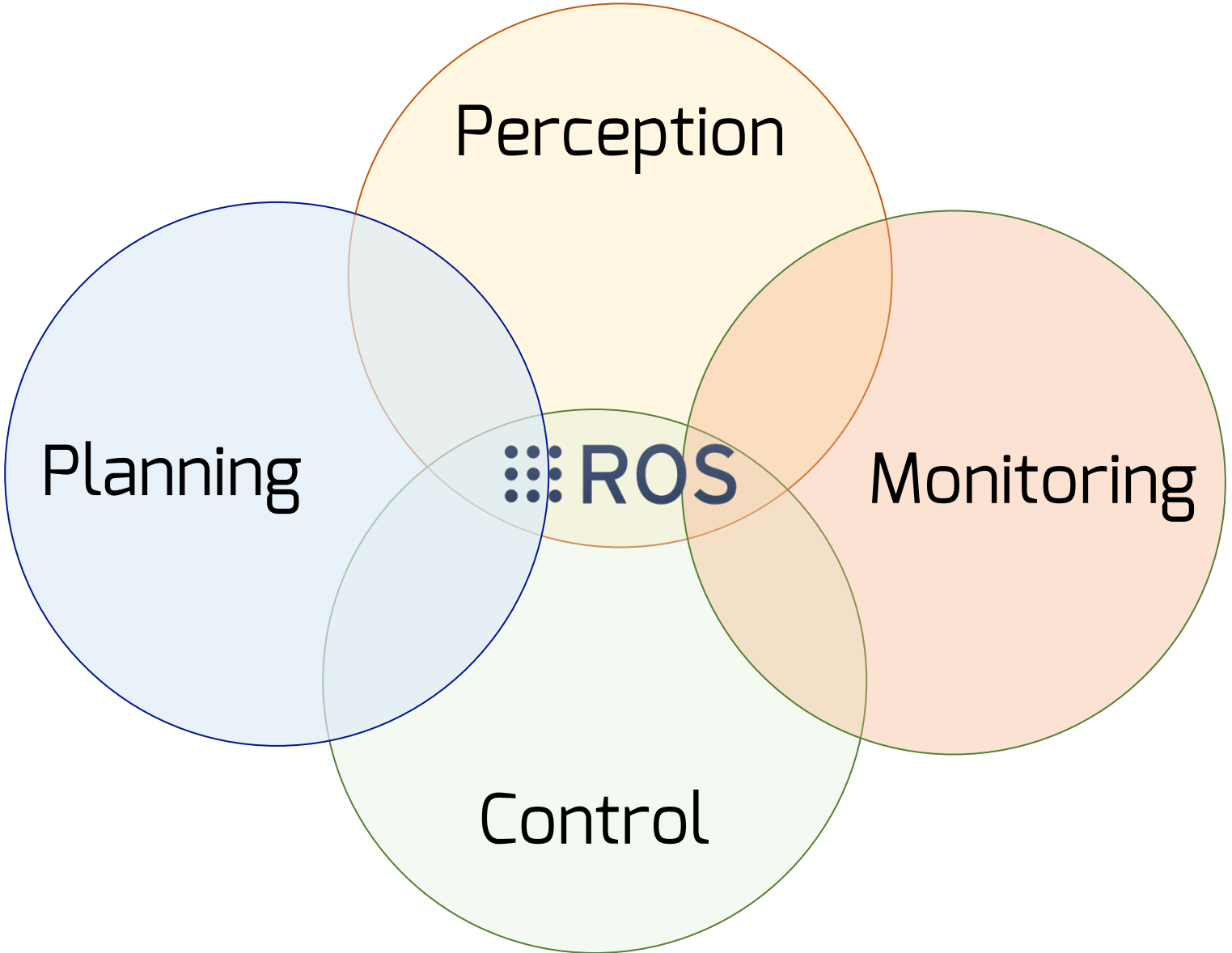
Compile

Run

ROS Capabilities



ROS Capabilities



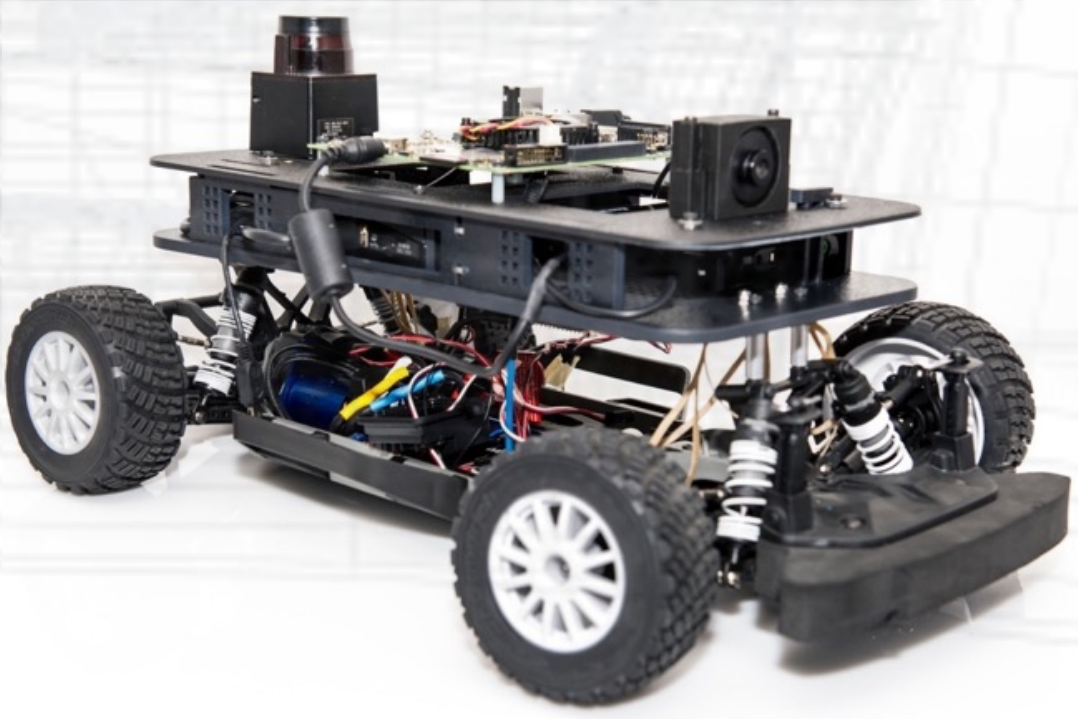
Plans

- What you saw should be released in September

F1/10

1/10th the scale. 10 times the fun!

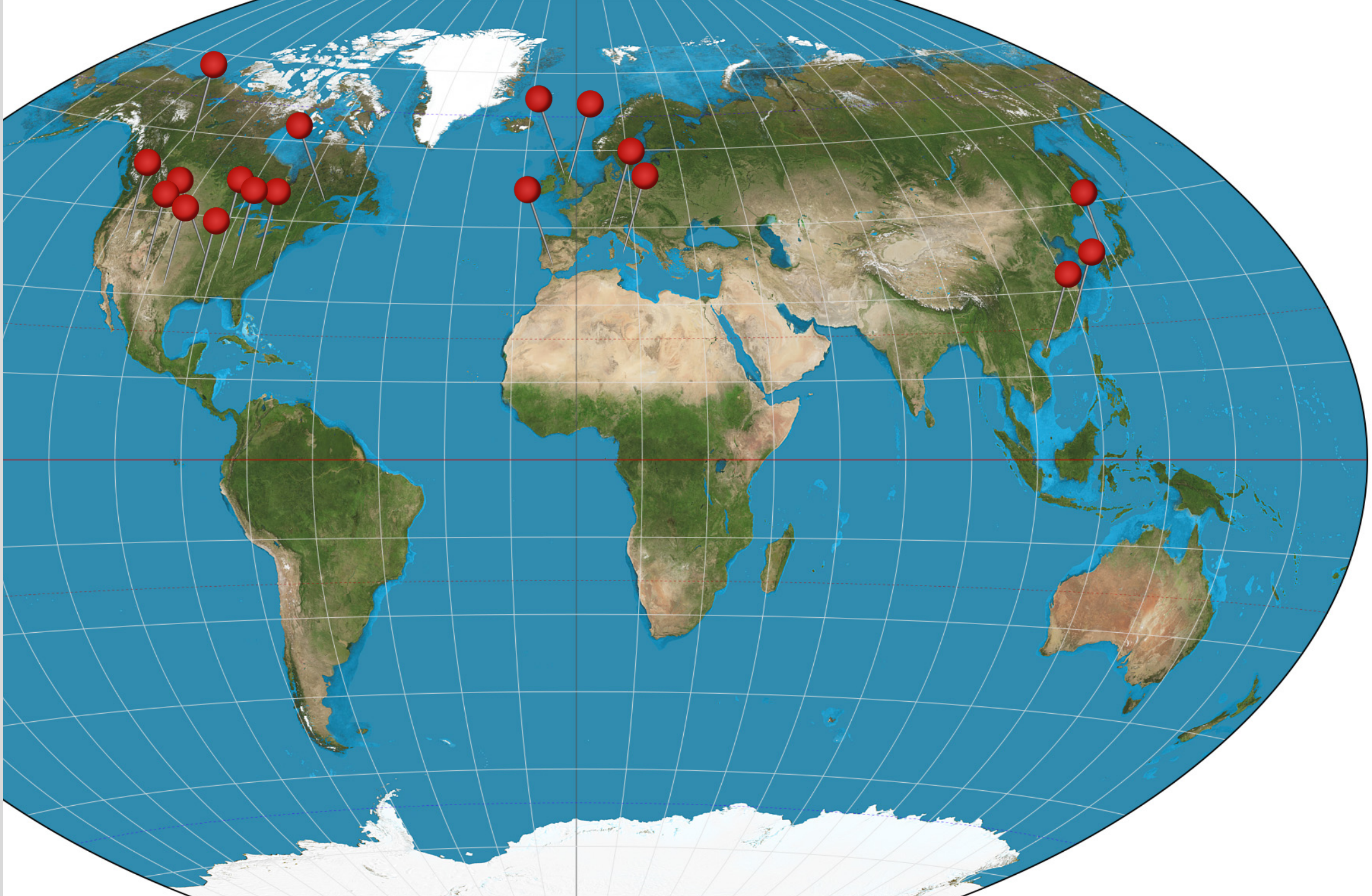
Education **Research** **Competition**



Organizers
Pennsylvania
Virginia
Italy

Racers
Arizona
Pennsylvania
Korea
Sweden
Italy
Virginia
Czech Republic

Community
California
Denmark
Portugal
Texas
Japan
North Carolina
Tennessee
South Carolina
New York
Canada
Austria
Hong Kong



Organizers
Pennsylvania
Virginia
Italy

Racers
Arizona
Pennsylvania
Korea
Sweden
Italy
Virginia
Czech Republic

Community
California
Denmark
Portugal
Texas
Japan
North Carolina
Tennessee
South Carolina
New York
Canada
Austria
Hong Kong

RV GAMES?