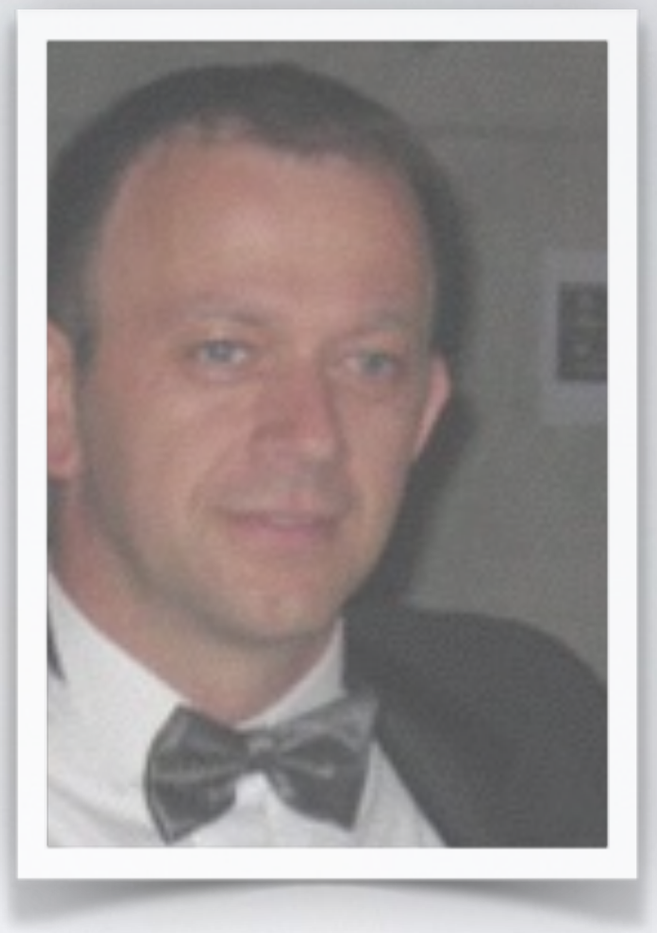
HSR CHALLENGE 2015

FAST AND FURIOUS

WOLFGANG GIERSCHE

wgiersche@gmail.com
wgi@zuehlke.com

zühlke
empowering ideas




Ultimate
Ultimate
DUNLOP

THE CHALLENGE

- Write a method that…
  - takes some numerical input
  - calculates something, and…
  - outputs a single number between 0 and 255
  - to minimise some other single numerical output

WELL, YES AND…

- Design it well
- Write and test it well
- Document it well (!!!)
- Publish it all on Github

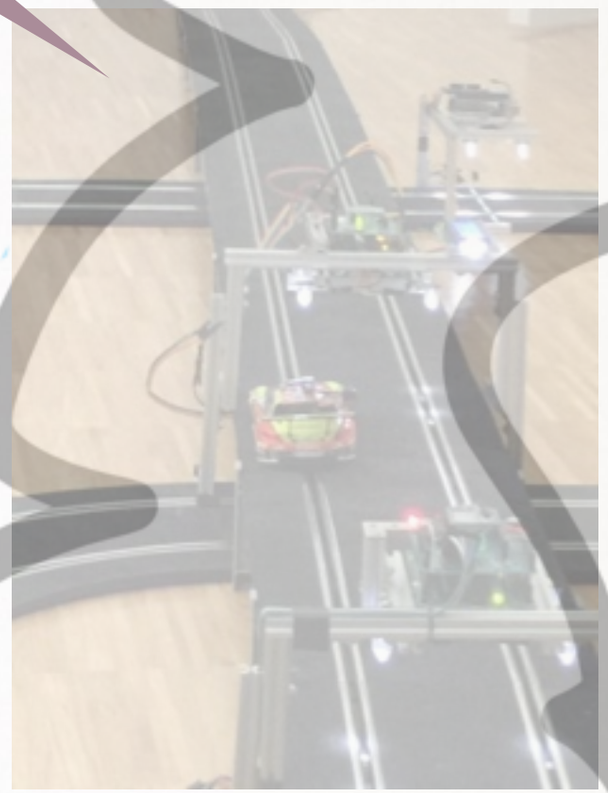
10.753s

Your auto-pilot

output

input

AMQP (rabbitmq)

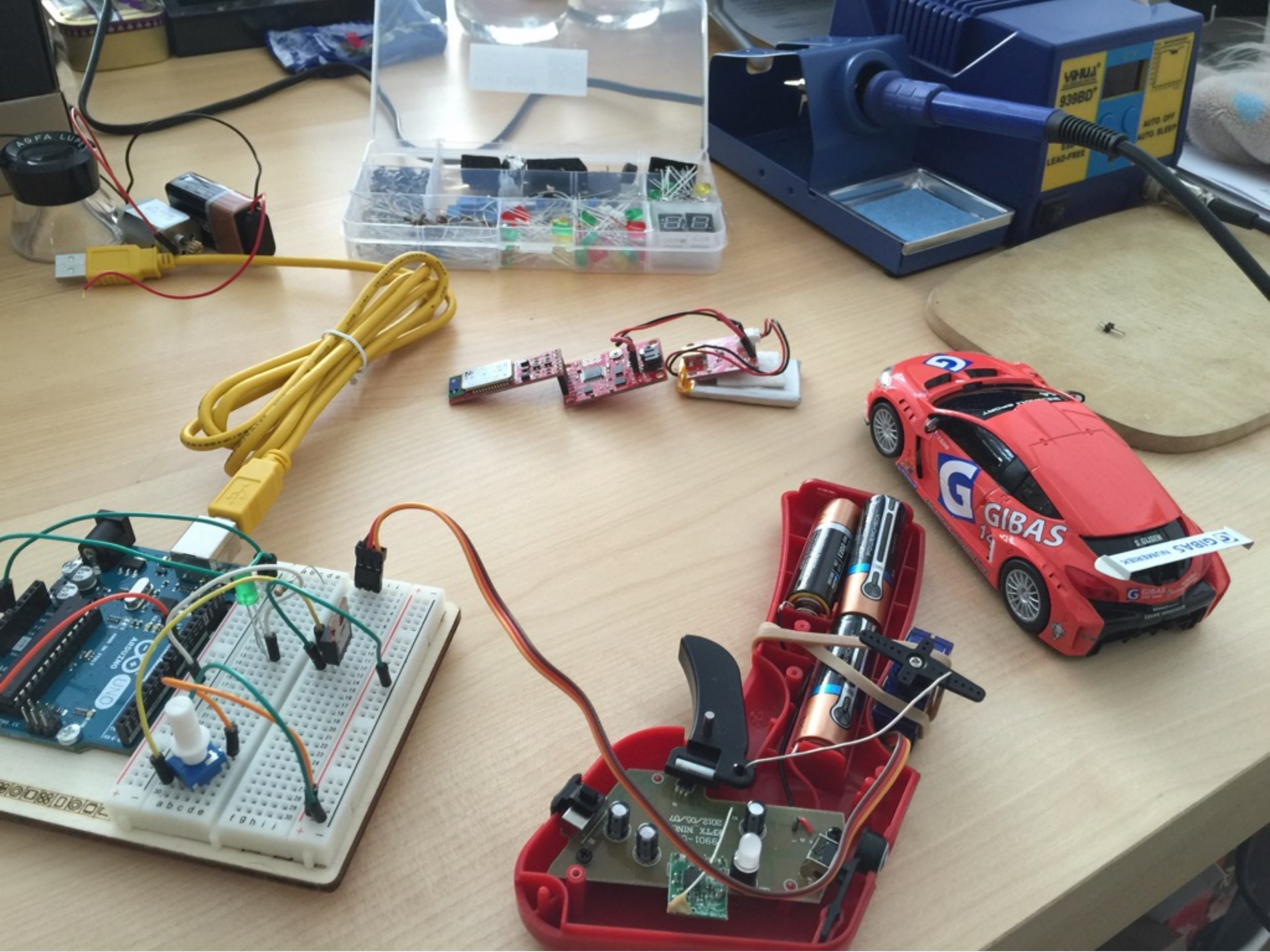


stomp over ws://

Some container

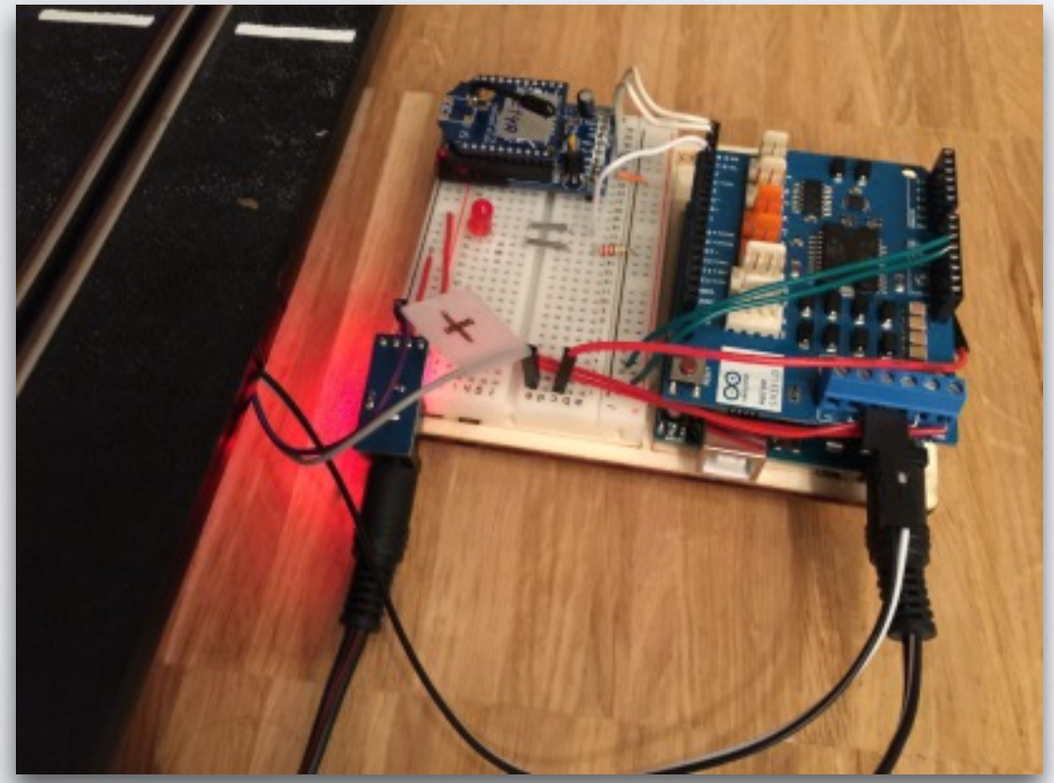
My infrastructure

```java
/**
 * Super complete, fancy interesting java doc goes here...
 */
public class MyPilot {

    private PowerApi powerApi;

    @Autowired
    public MyPilot ( PowerApi api ) {
        this.powerApi = api;
    }

    public void input ( RaceEvent event ) {

        // consider the new event
        int power = consider ( event );

        powerApi.setPower ( power );
    }
```


939BD
GIBAS
UNO
ARDUINO

POWER TO THE WHEELS

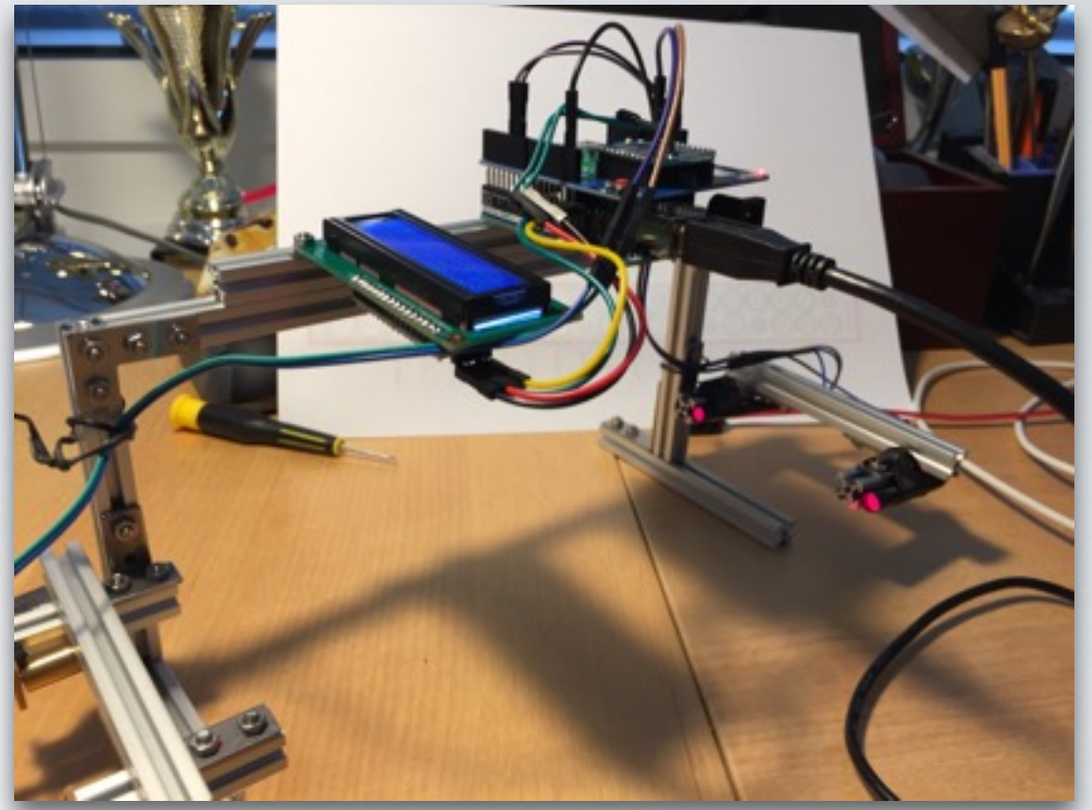
Integers [0, 255] translate into Voltage
(Pulse width modulated)

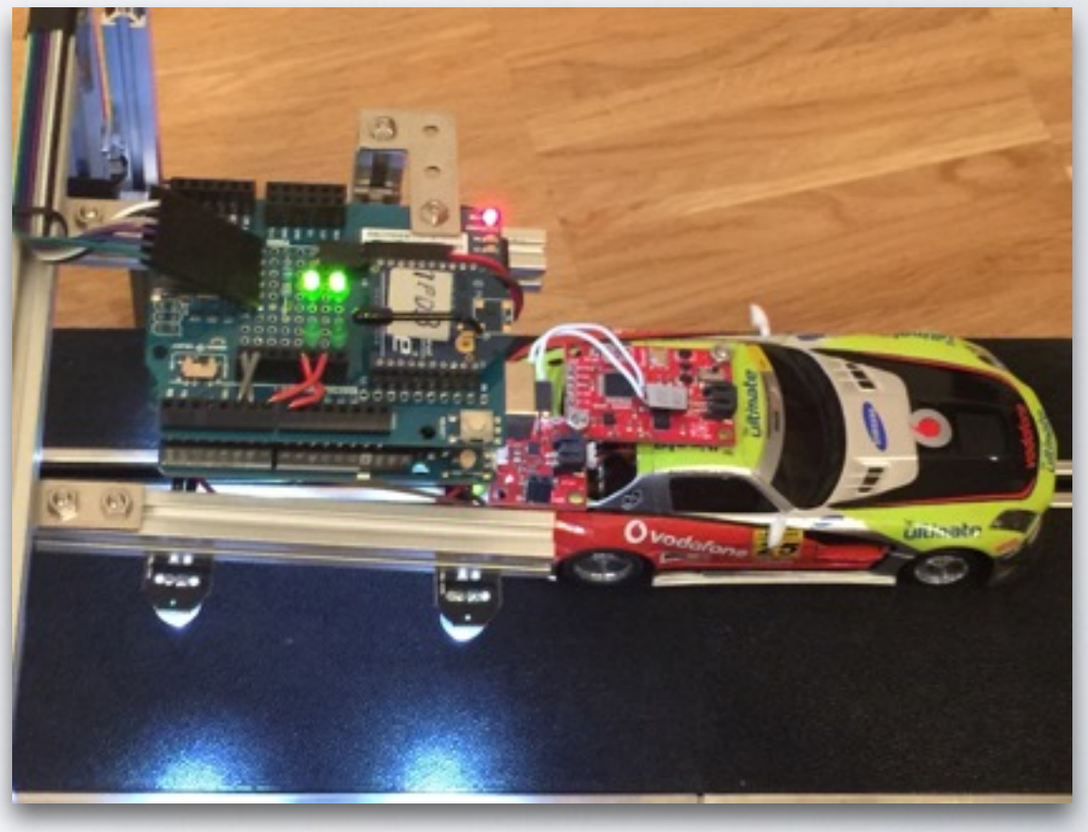


VELOCITIES AND PENALTIES

Light barriers provide

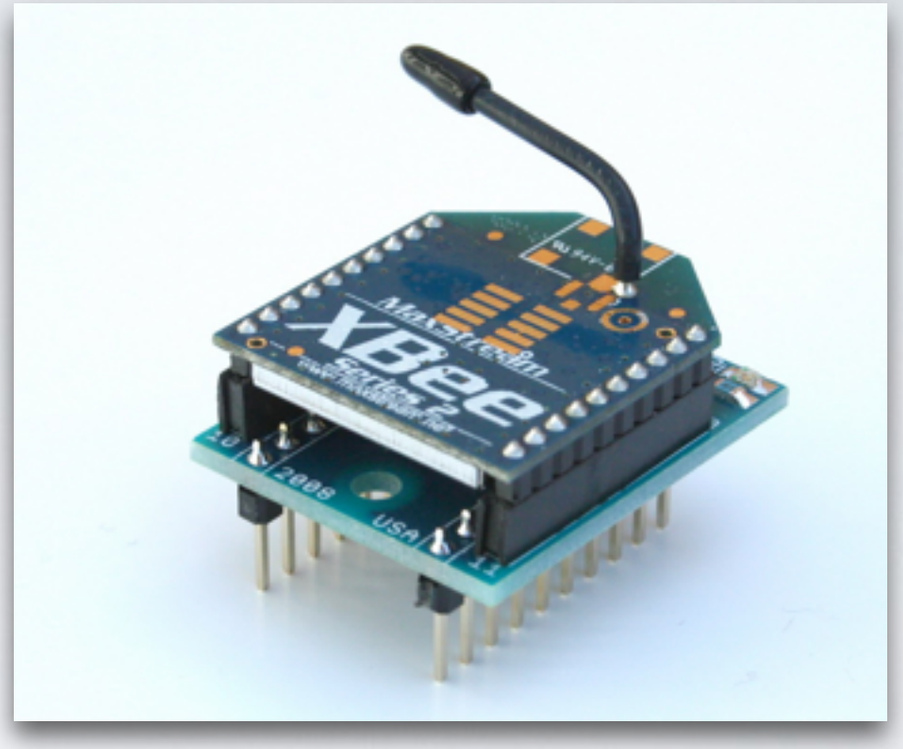
- velocity data
- out-of-tarmac simulation by applying penalty brakes
- round time information

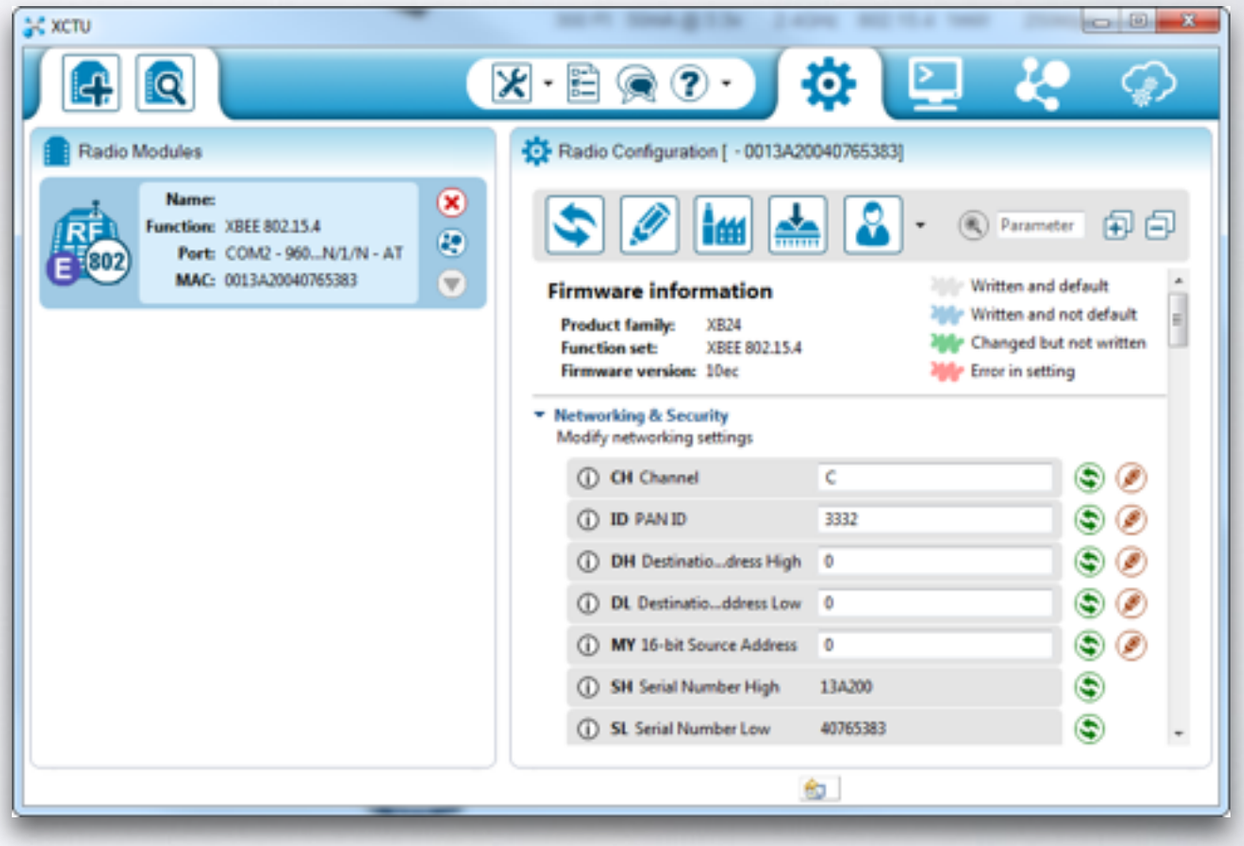




XBEE RADIO DATA

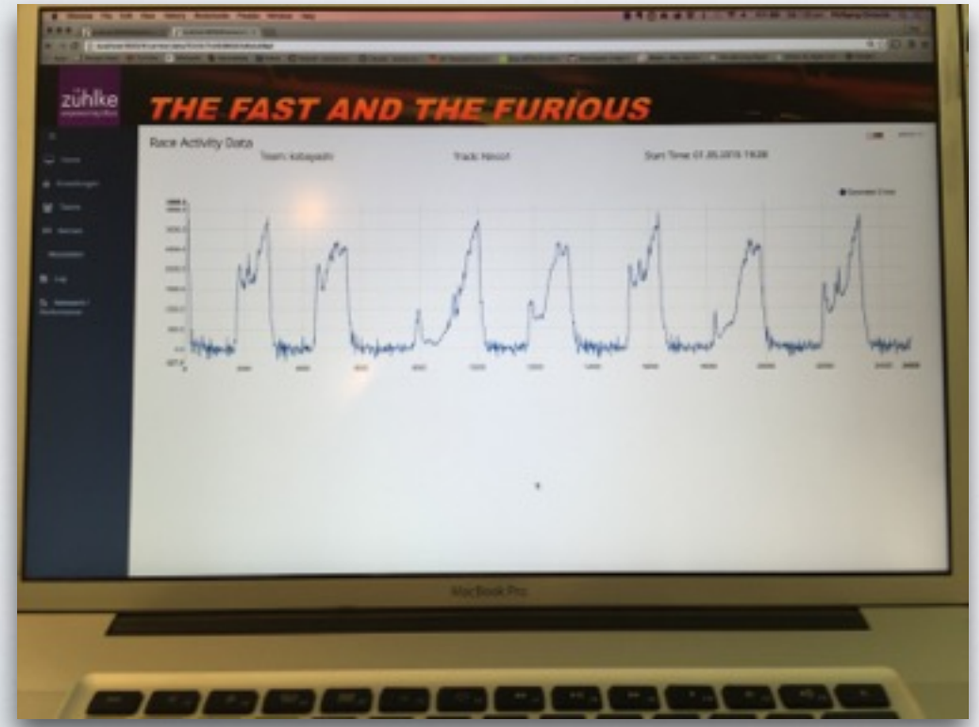
collecting speed information over the air





THE SLOT CAR

- RAZOR 9DOF Sensor
  3 x mag, 3 x gyro, 3 x acc
- Bluetooth
- 5V Lithium Battery



RabbitMQ

mongoDB

Spring Cloud
Config Server

Kobayashi

Relay

Race Console

spring boot

YOU

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

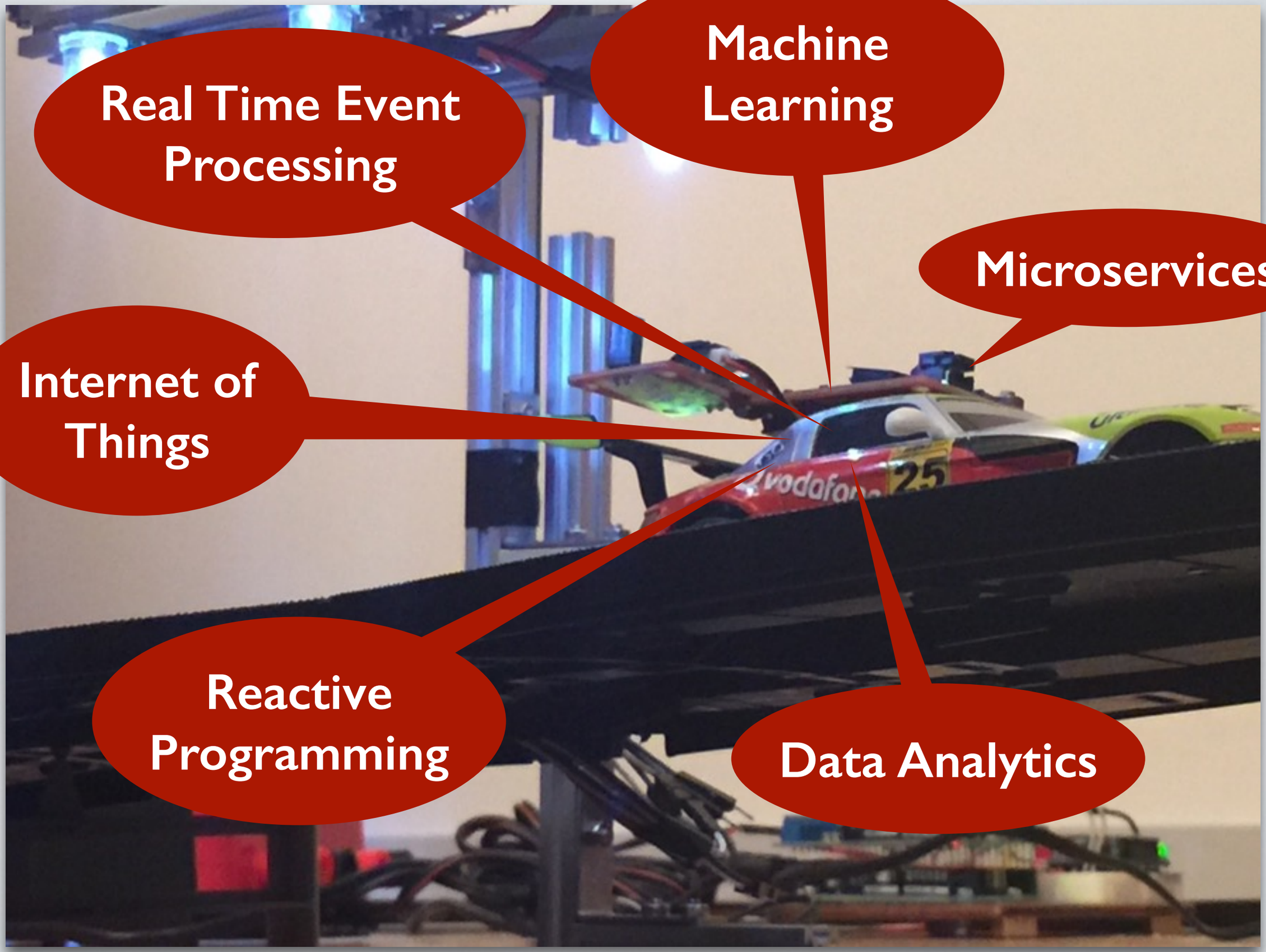
MySQL

Competitions

JHipster

Team Server

JHipster


Real Time Event Processing
Machine Learning
Microservices
Internet of Things
Reactive Programming
Data Analytics

THE CHALLENGE

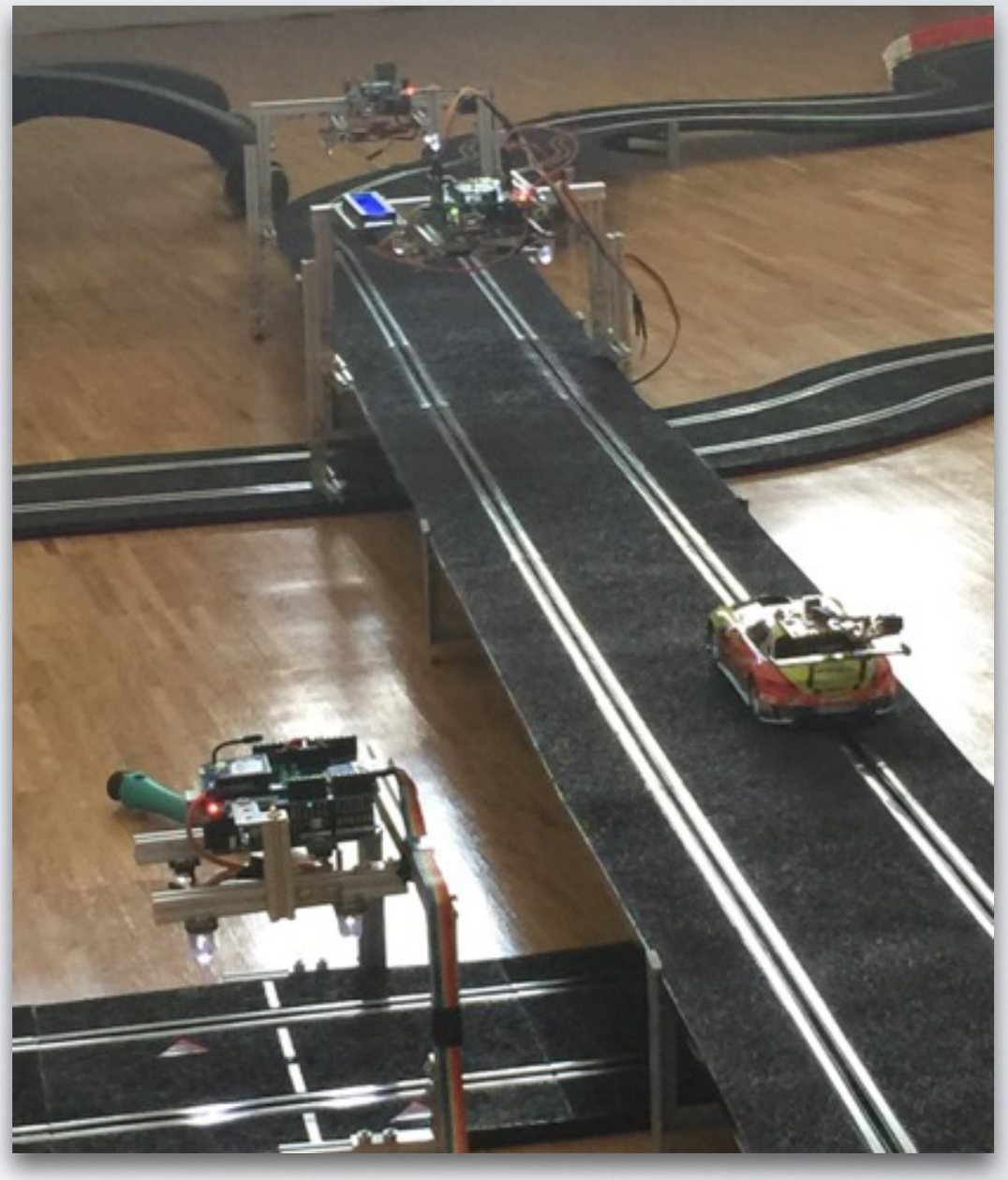
- Concurrency
- Fault Tolerance
- Noise
- Unpredictable latency
- Little time to learn
- Physical modelling

AGAIN: THE CHALLENGE

- Write a method that…
  - takes some numerical input
  - calculates something, and…
  - outputs a single number between 0 and 255
  - to minimise some other single numerical output

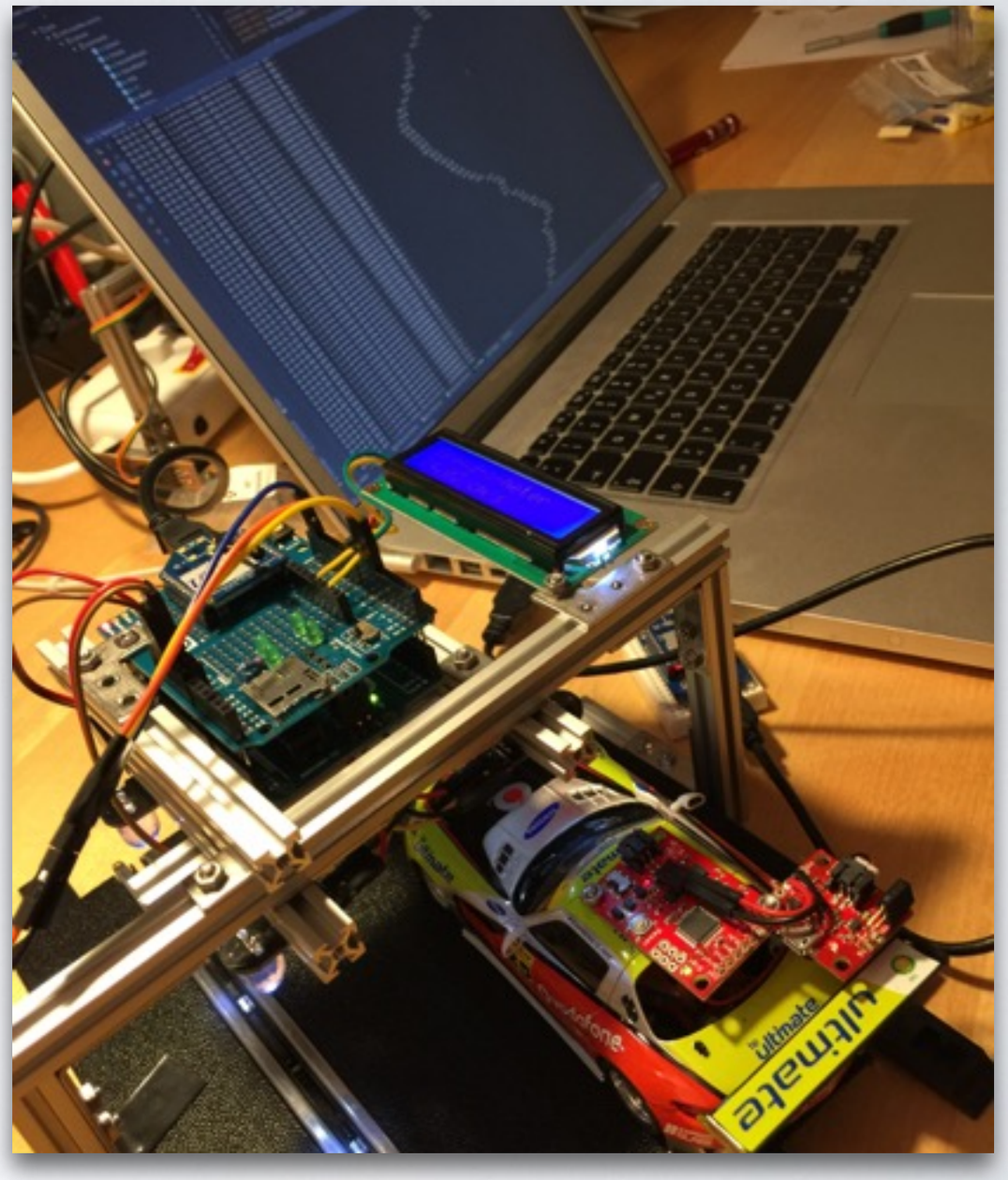
INPUT: RACE EVENTS

- Race start
- Race stop
- Velocity measurement
- Penalty messages
- Sensor readings at 50Hz
- Round passed



RESILIANCE AND CONCURRENCY

- Actors or similar primitives
- Streams
- Event based design
- Starter Kit: Typesafe Akka



DESIGN CONSIDERATIONS

- First comes race track recognition
- Read about "unsupervised" learning, it may help a bit.
- "Learning" should be parallel to driving
- Strategies may develop during race time
- The street looks different at higher speed.
- Parametrise your algorithms

RECOMMENDATIONS

- Get your hands dirty on data visualisation
- Make yourself comfortable with a minimum of statistics and math (average, standard deviation,…)
- Do NOT use classic concurrency primitives, such as @Async, synchronised, etc - unless you really love trouble.
- Consider using state-of-the-art technology, even distributed ones R Studio, Mathlab, Spark, Storm, Akka, RXJava, ReactJS…
- …but be careful - they're big!

TECHNICAL ENTRYPOINTS

- Akka Starter kit: http://github.com/FastAndFurious/AkkaStarterKit