Making High-Performance Robots Safe and Easy to Use for an Introduction to Computing

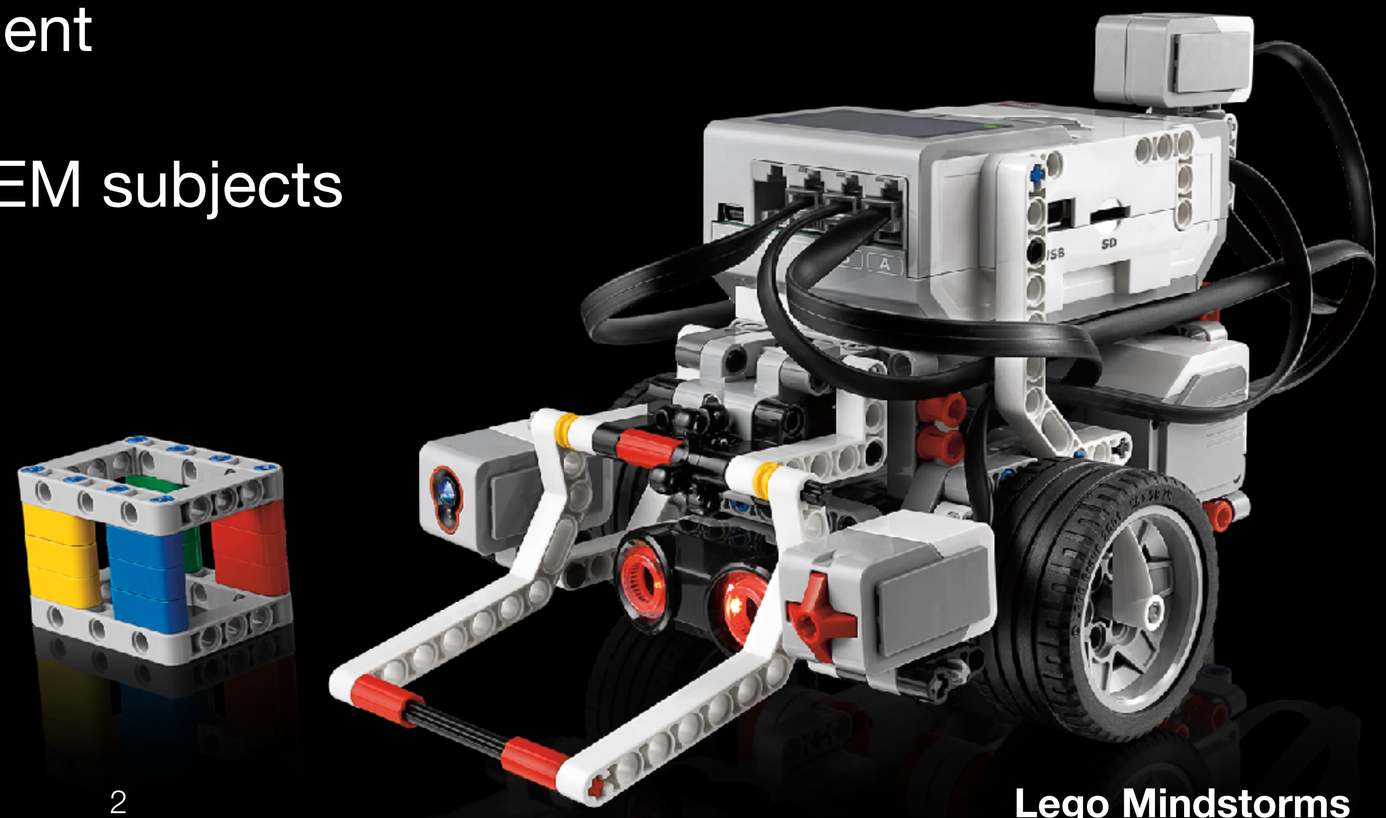
Joseph Spitzer[1], Joydeep Biswas[2], and Arjun Guha[1]

[1]University of Massachusetts Amherst
[2]University of Texas at Austin

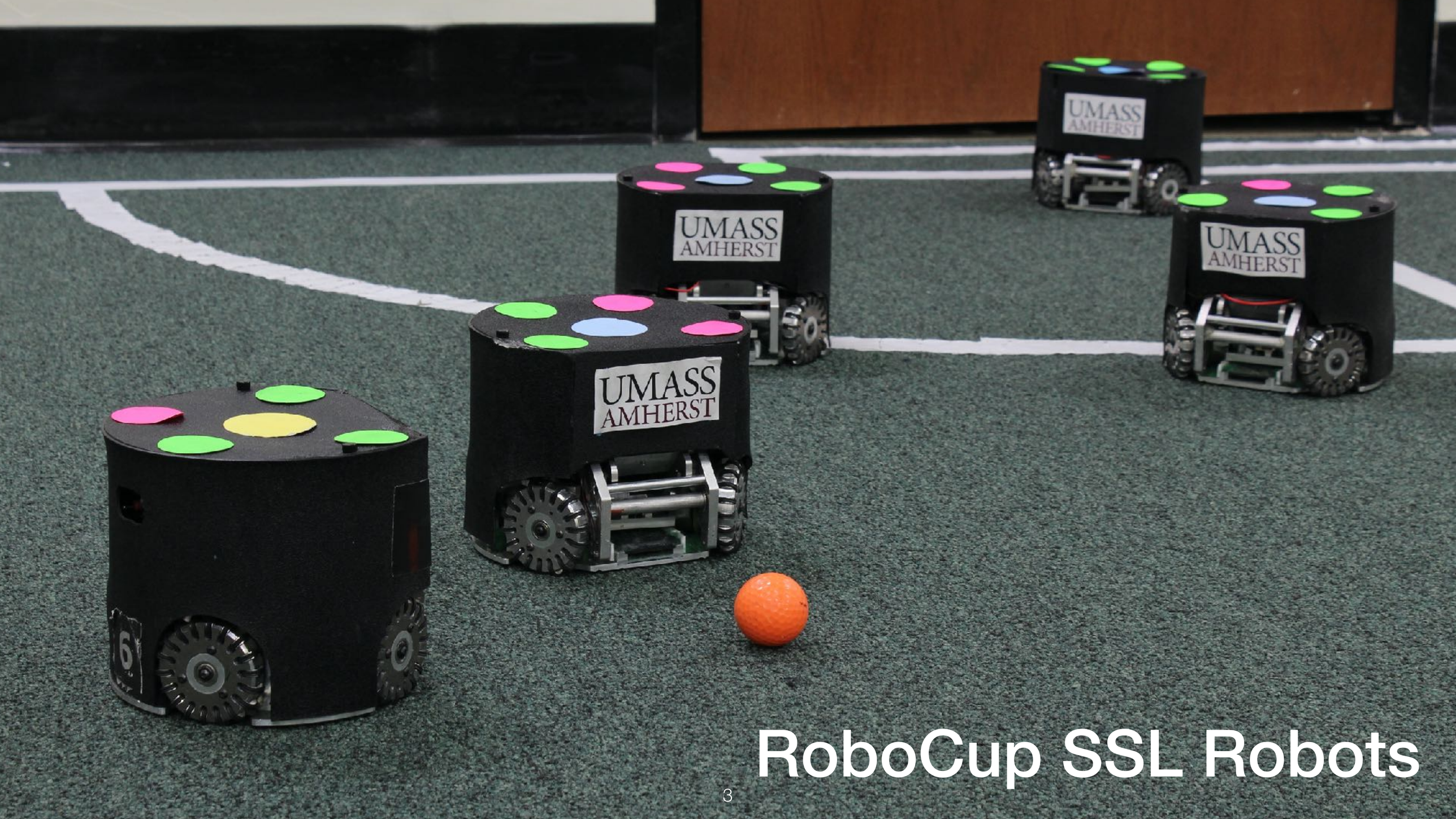
Robots for Education

Robots serve as a popular medium to introduce computing:

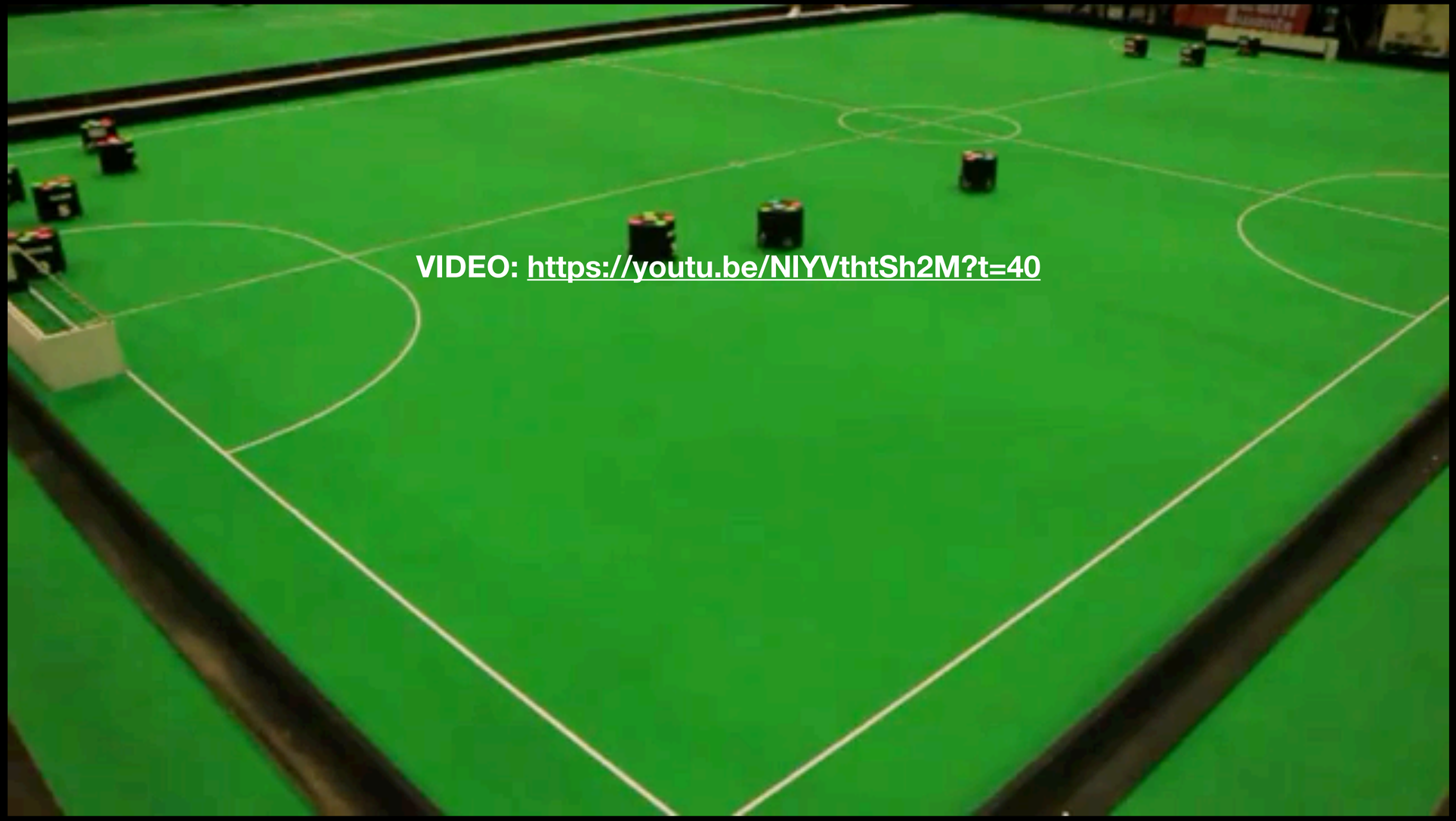
- Hands-on for increased engagement
- A tool for teaching an array of STEM subjects



Lego Mindstorms



RoboCup SSL Robots

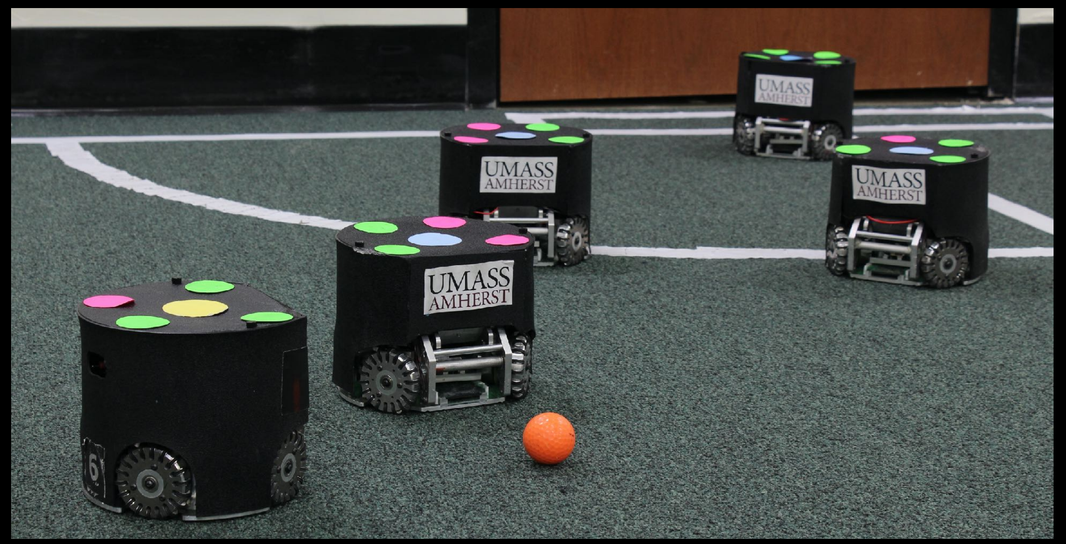

VIDEO: https://youtu.be/NlYVthtSh2M?t=40

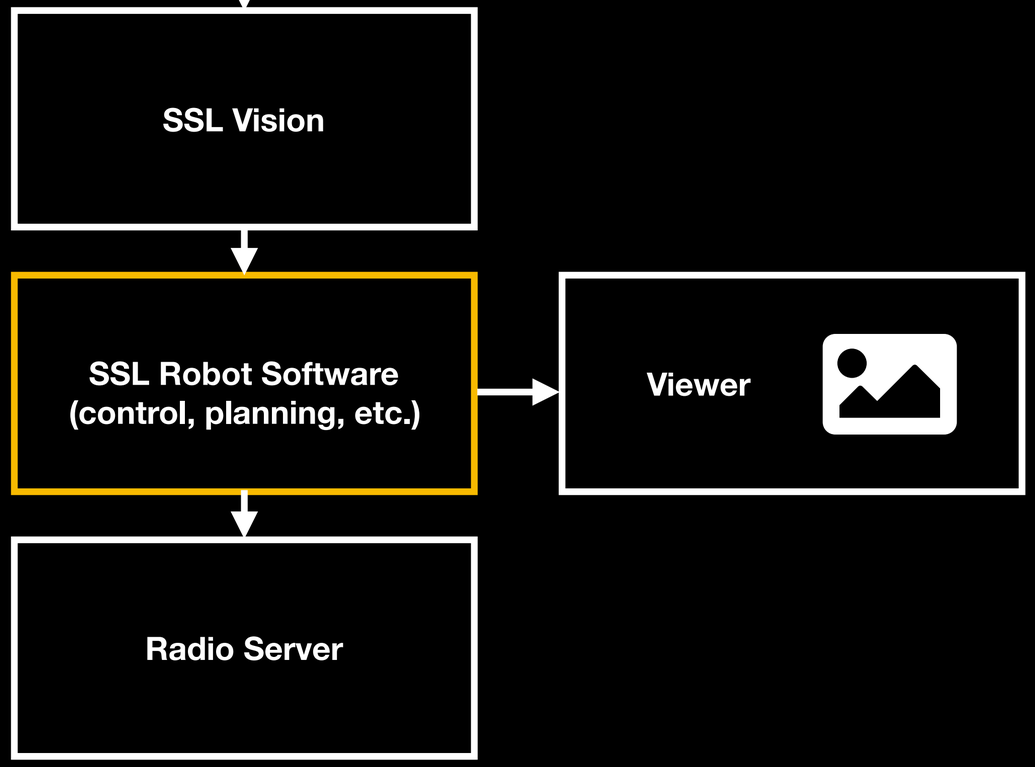
SSL Robot Software

- Optimized, high-performance C++
- Multi-threading and minimal dynamic memory allocation for speed

This presents a steep learning curve for even experienced programmers, and significantly more so for beginners.

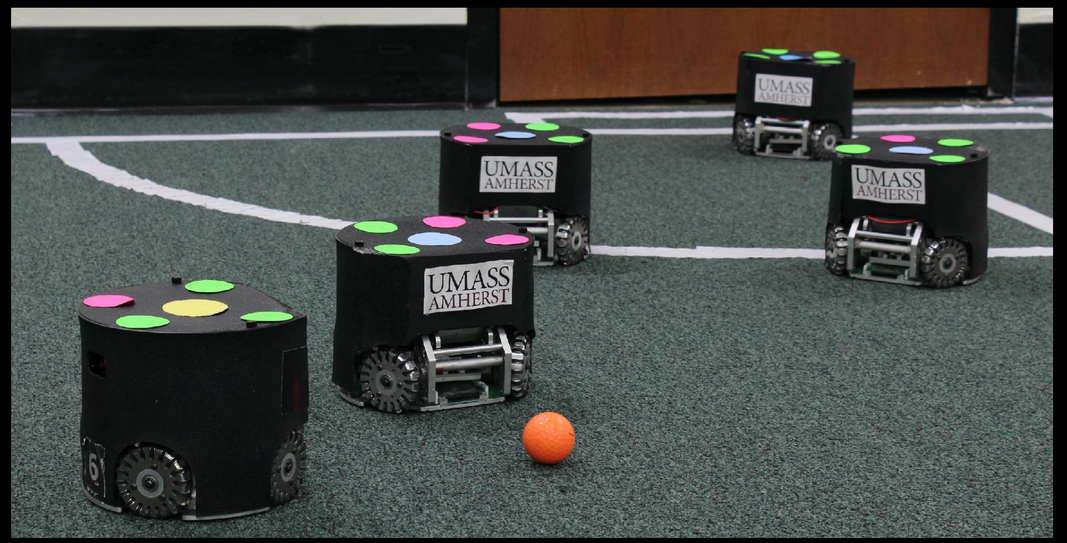
Current System





Ideal System

Current System



Why JavaScript?

- One of the most widely used languages
- Dynamic types
- **Only needs a browser**

Making SSL Robots Safe and Easy

- Physical protection of the users and robots

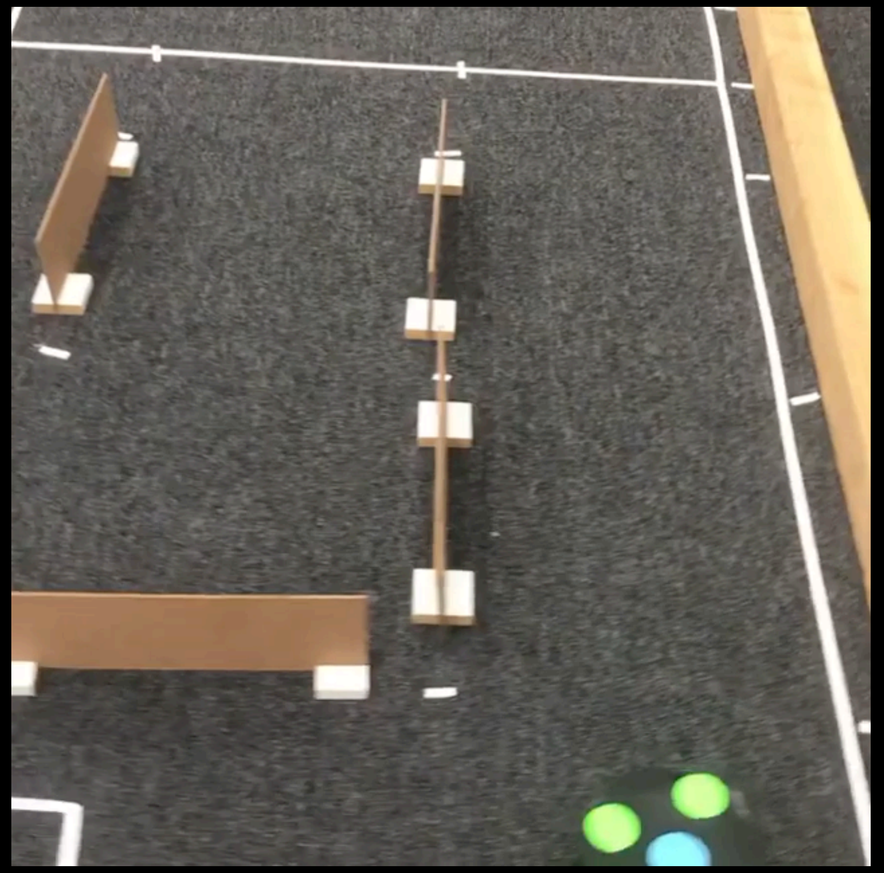
Making SSL Robots Safe and Easy

- Physical protection of the users and robots
- Simplification of the development experience

Robot Safety

- **Reduced Motion Model:** From in excess of 4m/s to at most 1m/s
- **Crash Prevention Buffer:** A uniform safety margin preventing collisions

Robot Safety



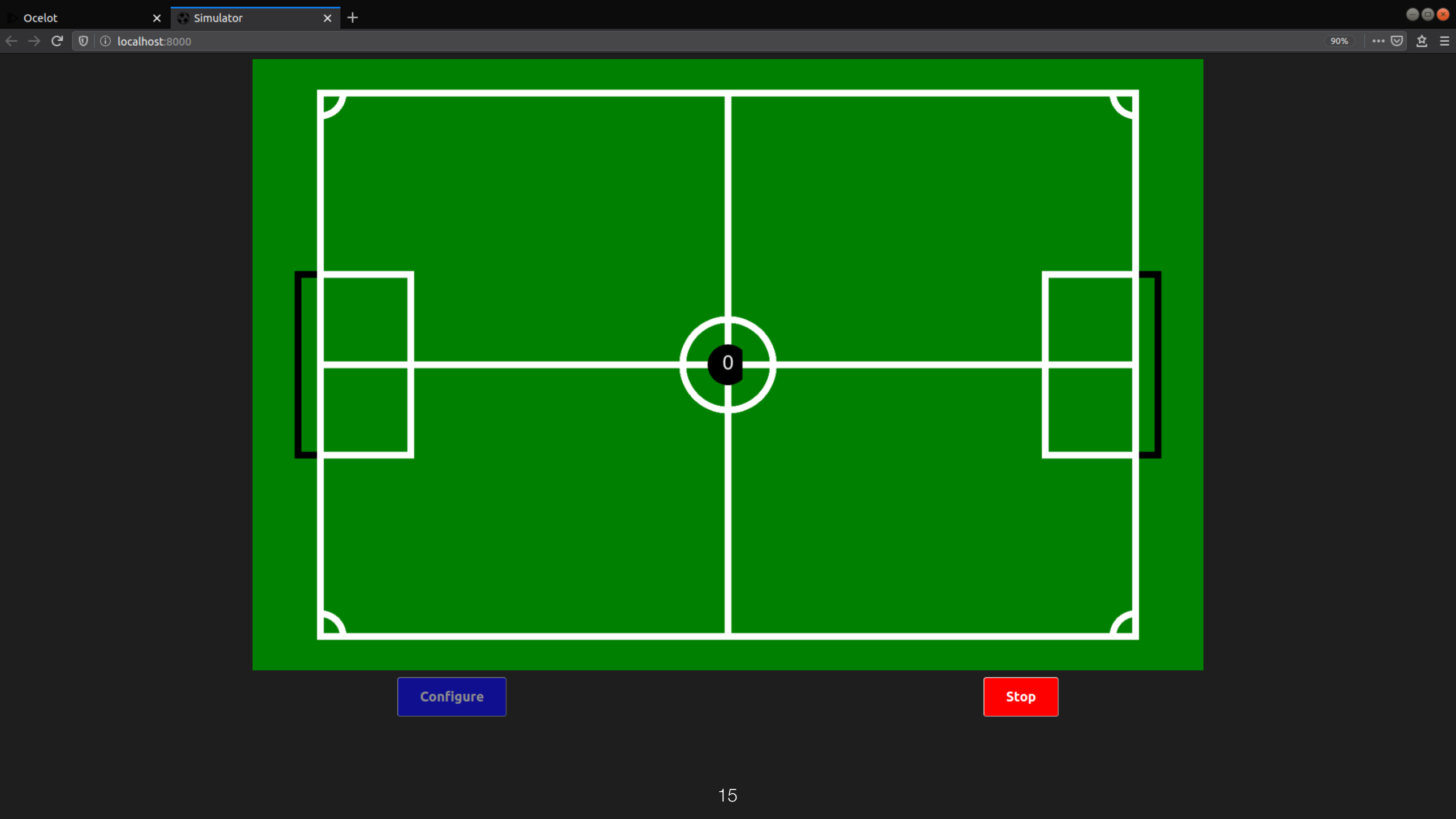
VIDEO: https://www.instagram.com/p/BzrYJI9hTBI/

Robot Safety

- **Command Timeouts:** A sliding 5s window to complete an action
- **Field Boundaries:** Robots do not leave view of vision system

Extending SSL Software for Ease-of-Use

- Publish the system state for external consumption


Ocelot
Simulator
localhost:8000
90%
0
Configure
Stop

Extending SSL Software for Ease-of-Use

- Publish the system state for external consumption
- Have system receive external commands via the network:
  - Alleviates installation and configuration hassles

Extending SSL Software for Ease-of-Use

- Publish the system state for external consumption
- Have system to receive external commands via the network:
  - Alleviates installation and configuration hassles
  - Provide a simpler development environment (*Exclusively* in the browser!)

Challenges with JS

- No function arity checks

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, 2); // 3
```

Challenges with JS

- No function arity checks

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, 2); // 3
add(1, 2, 3);
```

Challenges with JS

- No function arity checks

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, 2); // 3
add(1, 2, 3); // 3
```

Challenges with JS

- No function arity checks

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, 2); // 3
add(1, 2, 3); // 3
add(1);
```

Challenges with JS

- No function arity checks

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, 2); // 3
add(1, 2, 3); // 3
add(1); // NaN
```

Challenges with JS

- No function arity checks
- Implicit type conversions

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, ‘2’);
```

Challenges with JS

- No function arity checks
- Implicit type conversions

```
function add(a1, a2) {
  return a1 + a2;
}

add(1, ‘2’); // ‘12’
```

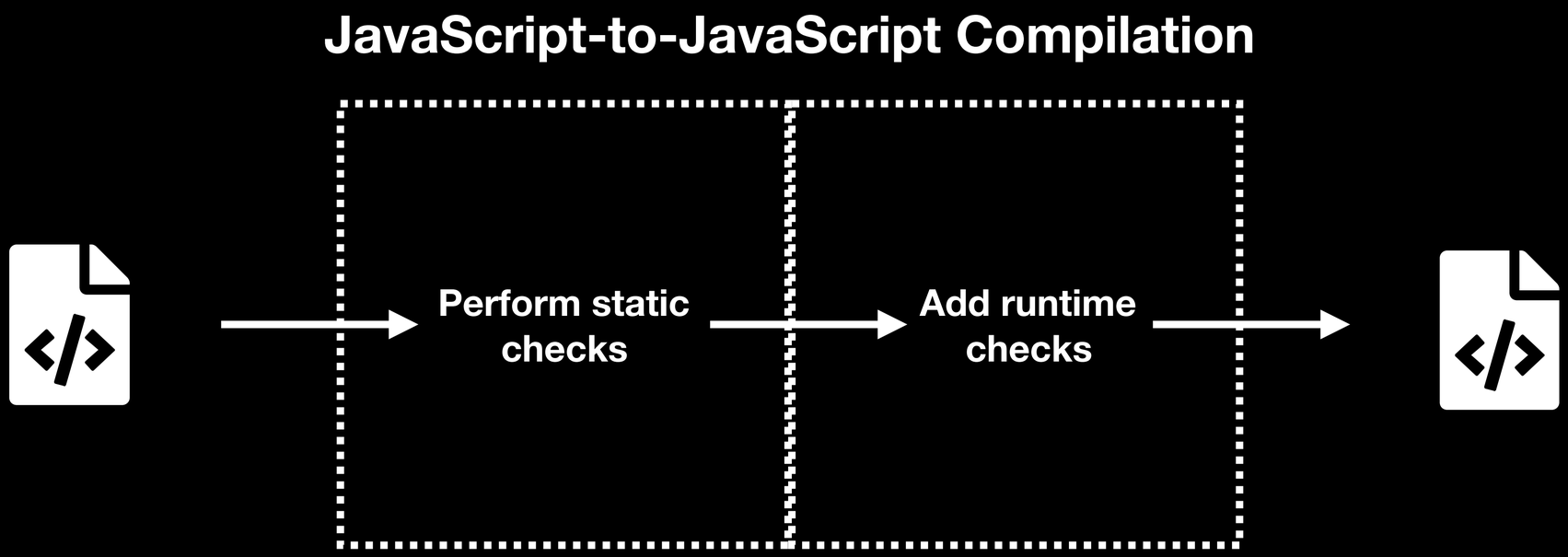
Challenges with JS

- No function arity
- Implicit type conversions
- ...

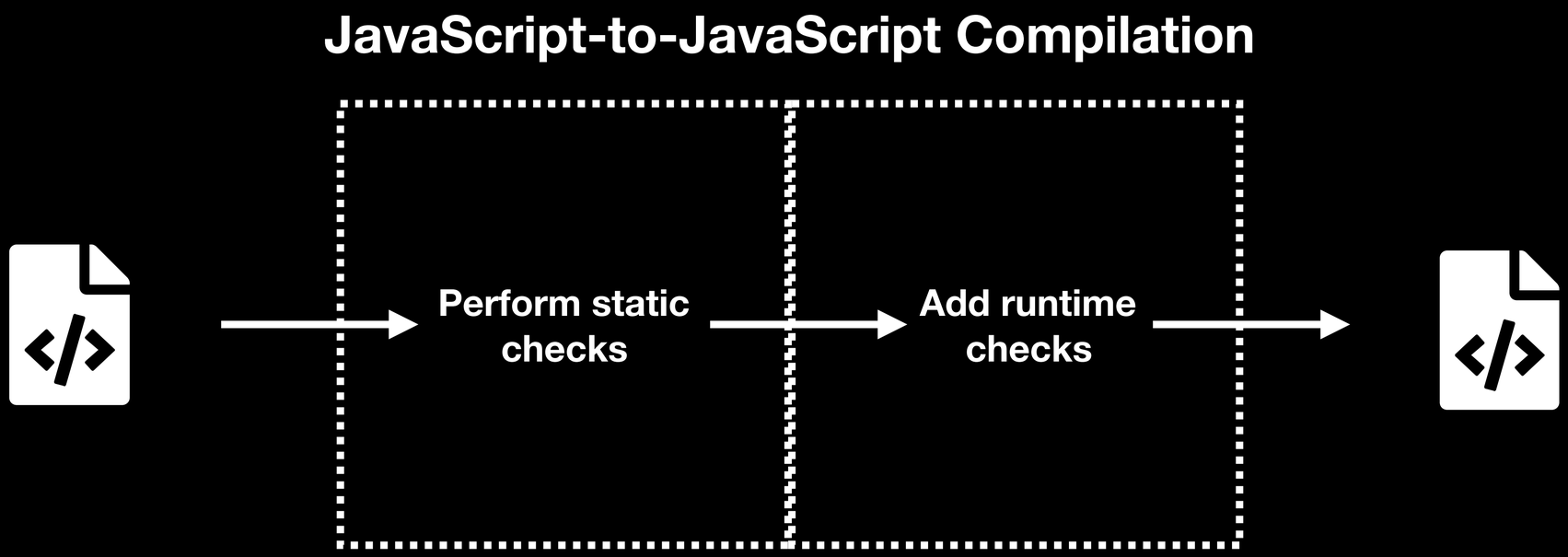
RoboJS

A compiler implementing a “nice” subset of JavaScript, with a robotics library.

RoboJS


JavaScript-to-JavaScript Compilation
Perform static checks
Add runtime checks

RoboJS



Transparent to the user

RoboJS

Exemplary *runtime* check:

Function Arity

```
function add(a1, a2) {
  return a1 + a2;
}
```

RoboJS

Exemplary *runtime* check:

Function Arity

```
function add(a1, a2) {
  console.assert(arguments.length === 2);
  return a1 + a2;
}
```

RoboJS

Exemplary *runtime* check:

Function Arity

```
function add(a1, a2) {
  console.assert(arguments.length === 2);
  return a1 + a2;
}

add(1, 2); // 3
```

RoboJS

Exemplary *runtime* check:

Function Arity

```
function add(a1, a2) {
  console.assert(arguments.length === 2);
  return a1 + a2;
}

add(1, 2); // 3
add(1, 2, 3); // 3
```

RoboJS

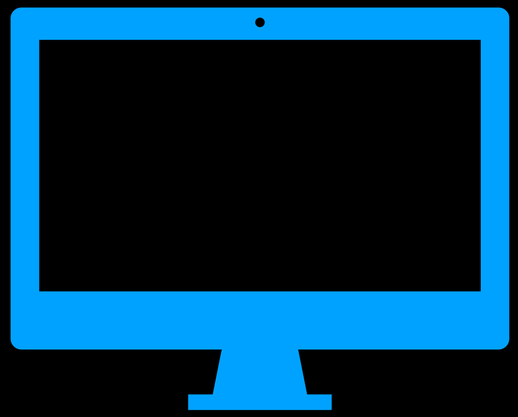
Exemplary *runtime* check:

Function Arity

```
function add(a1, a2) {
  console.assert(arguments.length === 2);
  return a1 + a2;
}

add(1, 2); // 3
add(1); // NaN
```

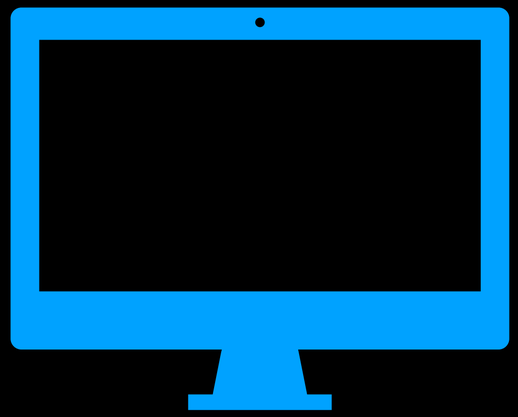
Programming Robots



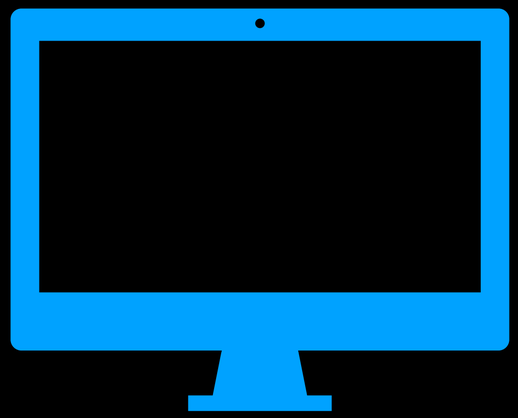
Bidirectional Communication Channel

Programming Robots

```
robot.moveForward();
robot.turnLeft();
```



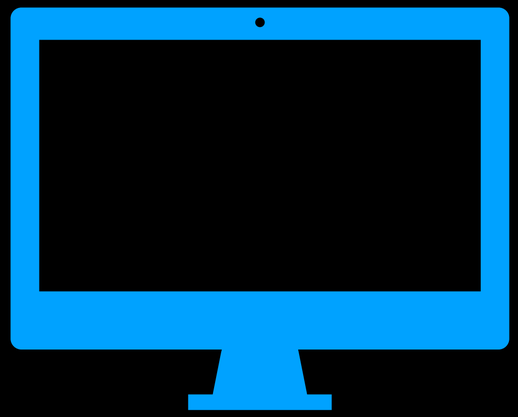
Programming Robots



```
robot.moveForward();
```

Done.

Programming Robots



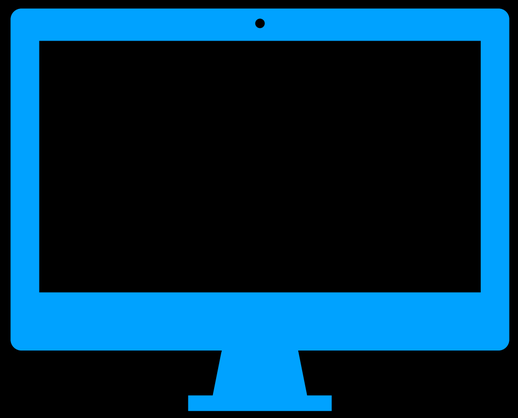
```
robot.moveForward();
```

Done.

```
robot.turnLeft();
```

Done.

Programming Robots



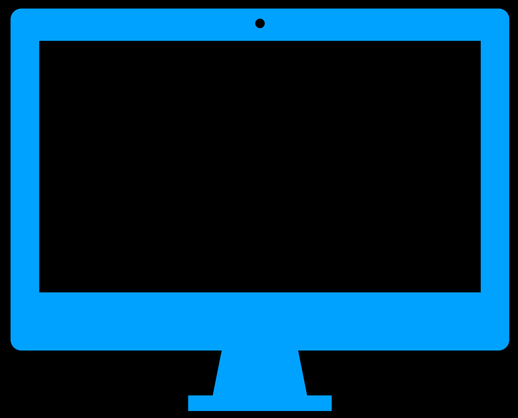
```
robot.moveForward();
```

Done.

```
robot.turnLeft();
```

Done.

Programming Robots on the Web



```
robot.moveForward();
```

Done.

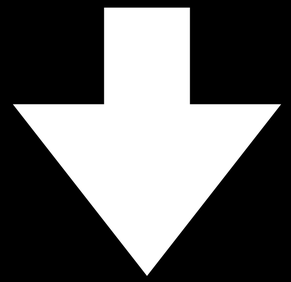
```
robot.turnLeft();
```

Done.

Not possible in JavaScript

Programming Robots on the Web

```
robot.moveForward();
robot.turnLeft();
```

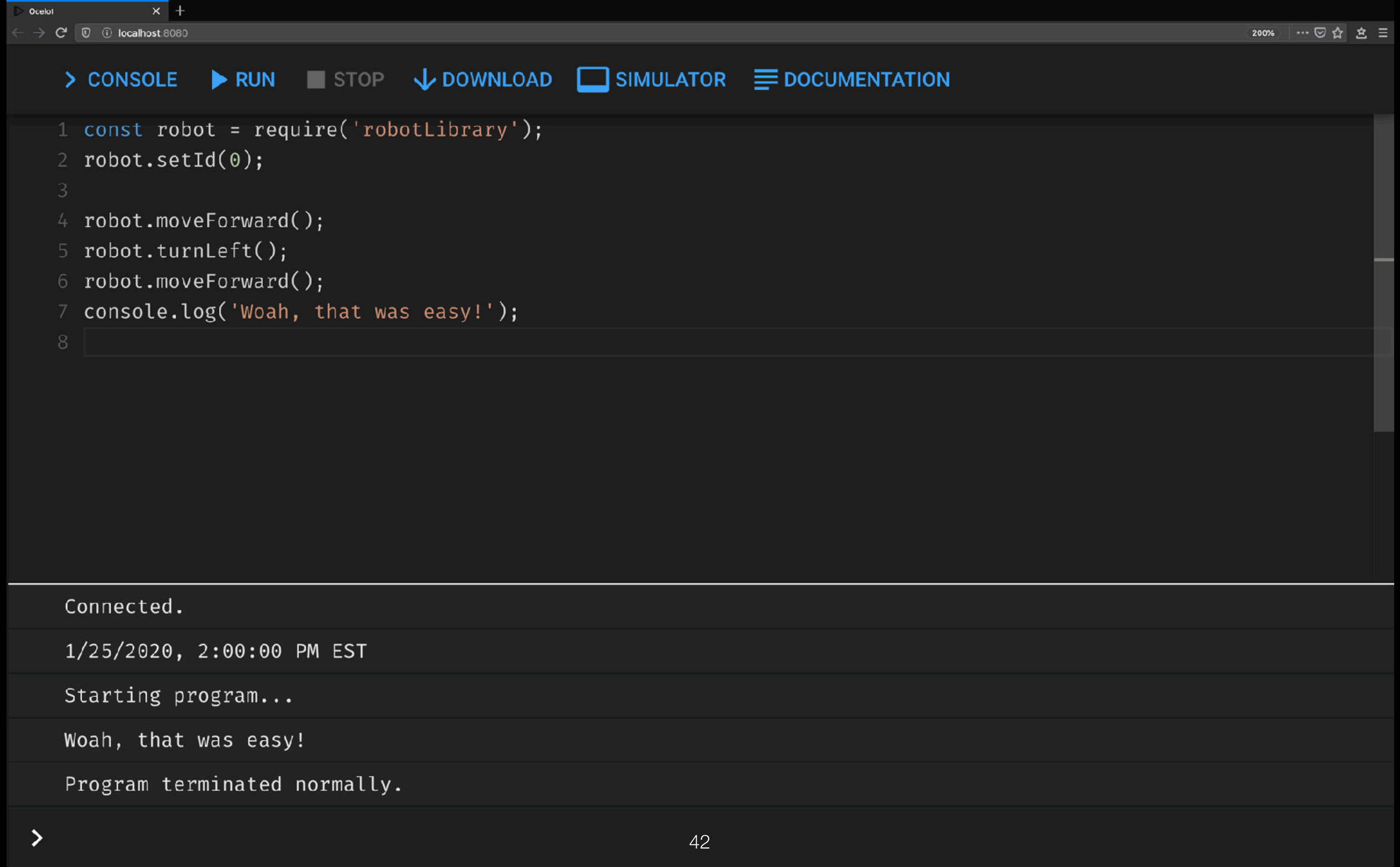


```
robot.moveForward(function() {
  robot.turnLeft();
});
```

Programming Robots on the Web with RoboJS

```
robot.moveForward();
robot.turnLeft();
```

RoboJS custom runtime makes all I/O blocking!


Ocelot
localhost:8080
200%
CONSOLE
RUN
STOP
DOWNLOAD
SIMULATOR
DOCUMENTATION
1 const robot = require('robotLibrary');
2 robot.setId(0);
3
4 robot.moveForward();
5 robot.turnLeft();
6 robot.moveForward();
7 console.log('Woah, that was easy!');
8
Connected.
1/25/2020, 2:00:00 PM EST
Starting program...
Woah, that was easy!
Program terminated normally.

Workshop

Workshop Details

- 1 week; 6 hours/day
- 12 HS students; several had experience with Lego Mindstorms and block-based programming
- Lecture introduced a topic or activity, followed by student work time

Workshop Details

- Gradual introduction of more sophisticated concepts, supported by the layers of abstraction in the RoboJS API
- Programs were first written and tested in student's simulator before moving on to the real robots
- Later projects required more testing on real robots

Does RoboJS Help?

Data

- IDE saves a revision each time program is run (if changed)
- **3,230** student written revisions across **237** distinct files containing **106,168** lines
- ~10% of the revisions had JavaScript syntax errors
- Inspection of the remainder for RoboJS errors

Errors Caught by RoboJS

Pitfall

Count

Contrast

1. JS
2. Consequence
3. RoboJS

Errors Caught by RoboJS

Pitfall	Conditional assignment **Ex:** `if(x = 0)`
Count	28
Contrast **1. JS** **2. Consequence** **3. RoboJS**	1. Branches on value of RHS as a Boolean 2. Potential branching based on non-Boolean literals 3. Same branch behavior, but only allowed if RHS evaluates to `true/false`

Errors Caught by RoboJS

Pitfall	Conditional assignment **Ex:** `if(x = 0)`	Operator type mismatch **Ex:** `'x' * 2`
Count	28	42
Contrast **1. JS** **2. Consequence** **3. RoboJS**	1. Branches on value of RHS as a Boolean 2. Potential branching based on non-Boolean literals 3. Same branch behavior, but only allowed if RHS evaluates to `true/false`	1. `NaN` reference in the case of arithmetic/bit-wise operators and shorthand assignment 2. Potential propagation of `NaN` 3. Not allowed

Conclusion

- Extended the software stack of high-performance RoboCup Small-Size-League robots
- Ensured safety at various levels of abstraction
- Developed a rich robotics library in JS, targeted to newcomers — RoboJS
- Packaged RoboJS into a browser IDE capable of blocking I/O
- Effectively leveraged the platform during a weeklong outreach workshop

Acknowledgments

- Students of the UT-AMRL & UMASS-PLASMA research labs
- Andrew Pasquale & Holyoke Codes

The following software has been modified from its original version. It has been formatted for personal use and edited for content:

https://github.com/ut-amrl/robo-js