How I Got Back my Coding Mojo!

Mark West

Safe Harbour

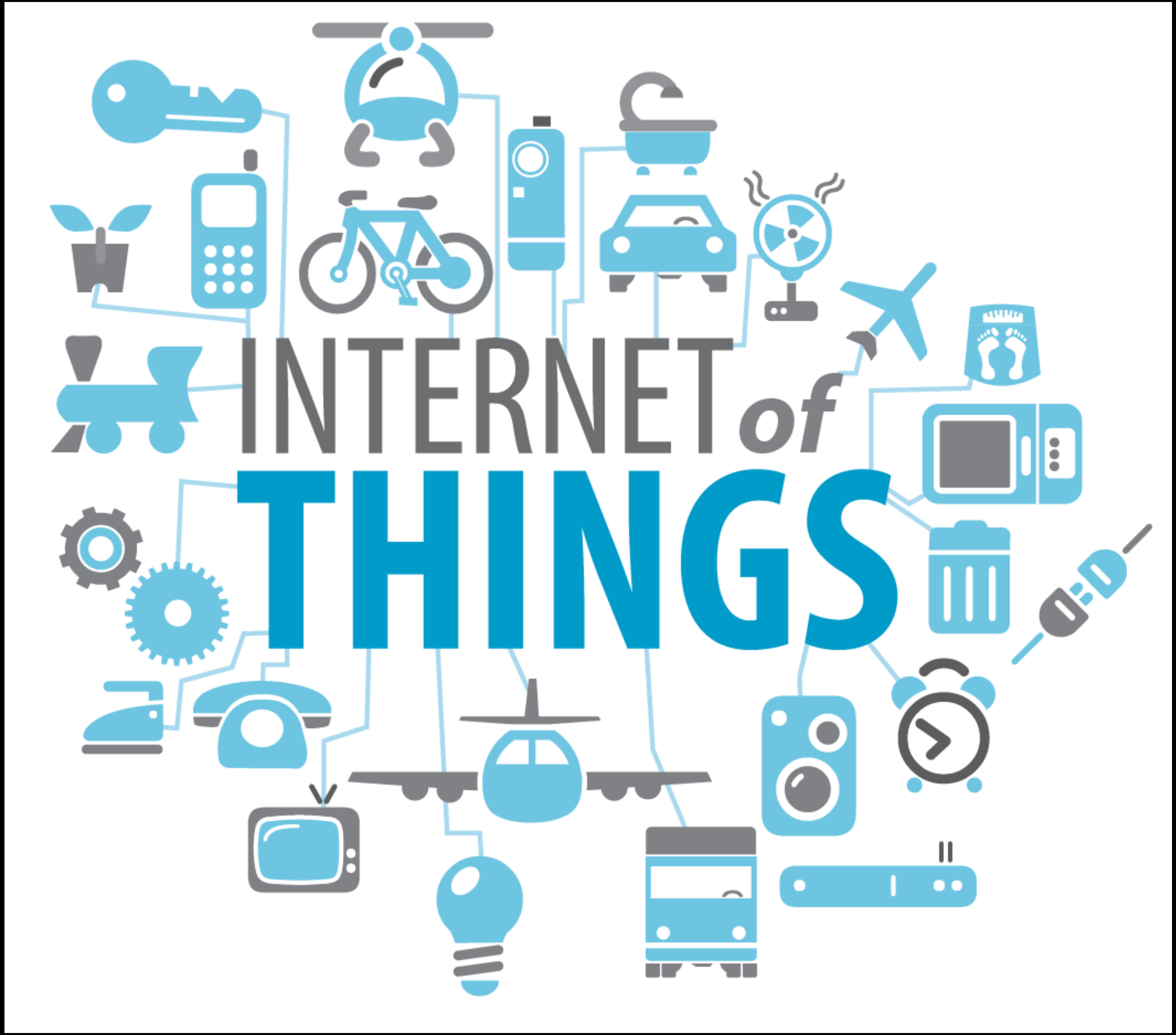
What is Mojo?

OH NO!
I'VE LOST MY MOJO!
imgflip.com

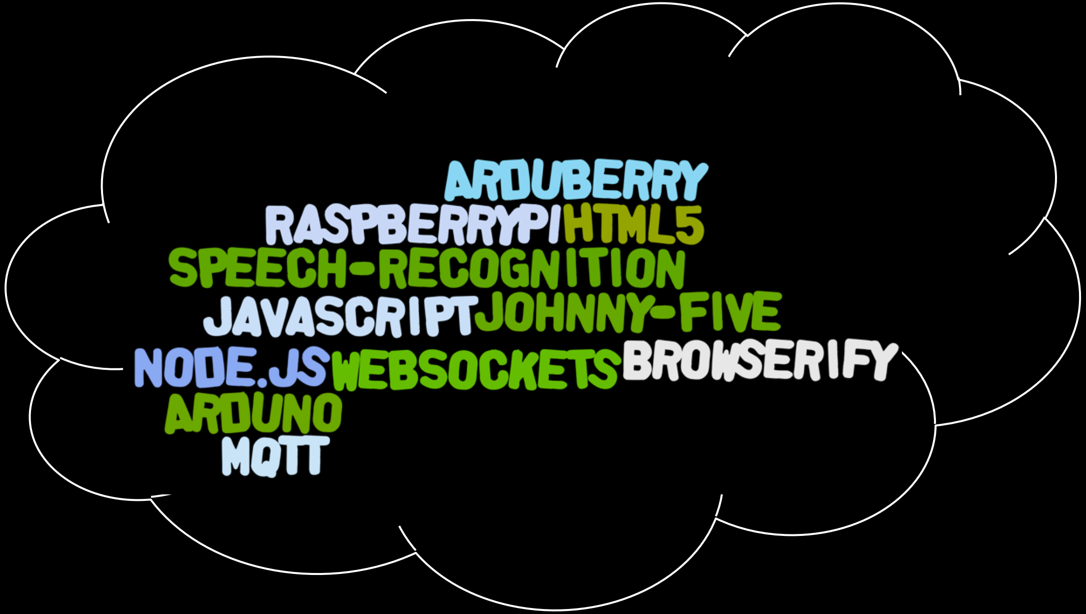
“Any application that can be written in JavaScript, will eventually be written in JavaScript”

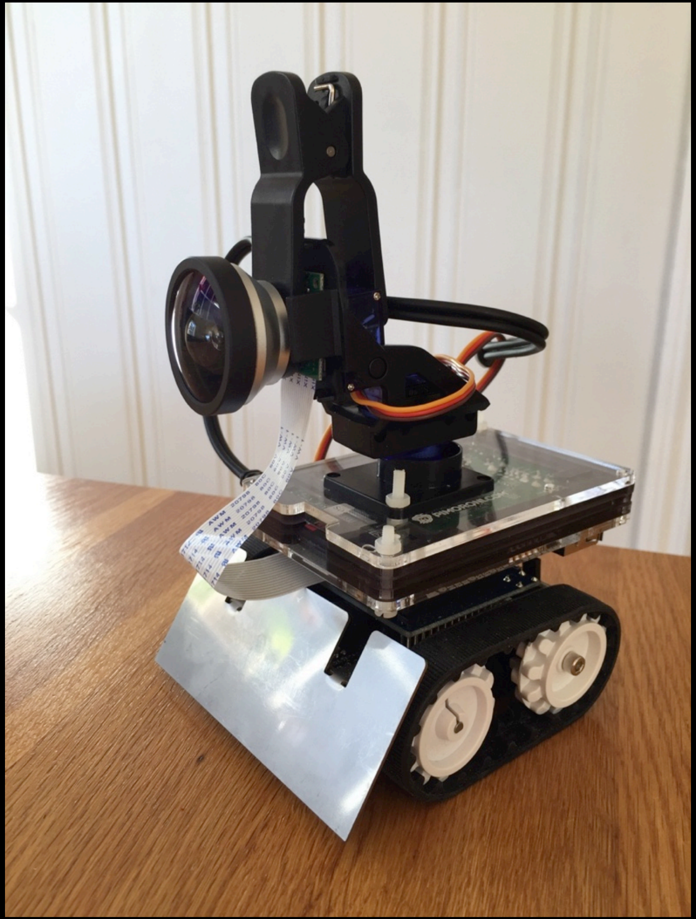
James Atwood (founder, stackoverflow.com)

Maker
Faire®


INTERNET of
THINGS

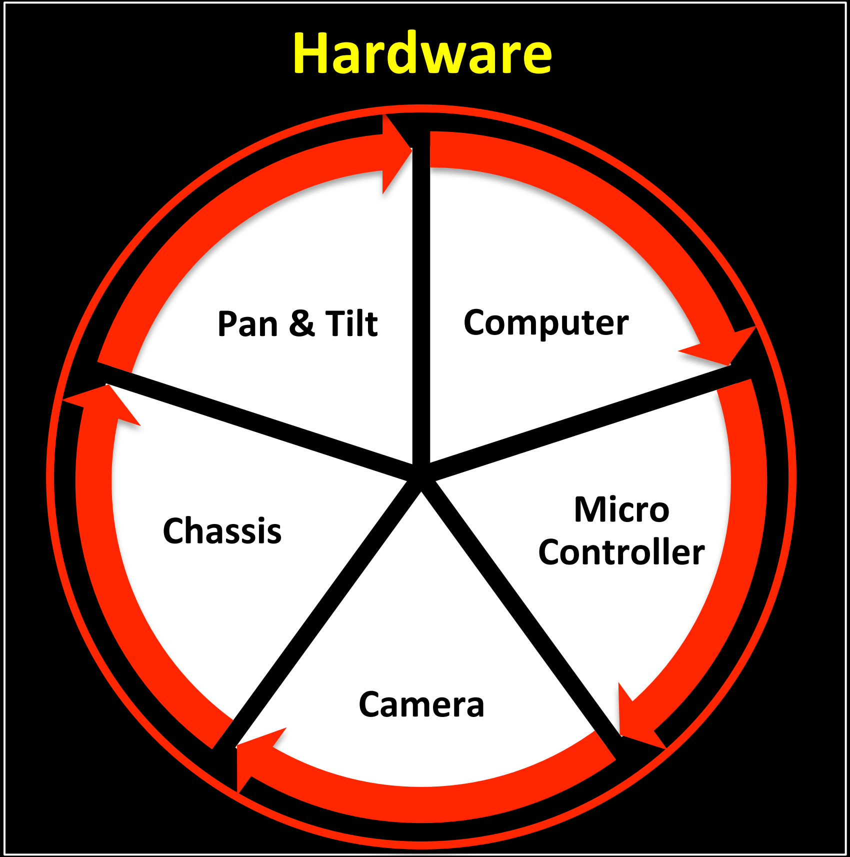
NodeBot Rover

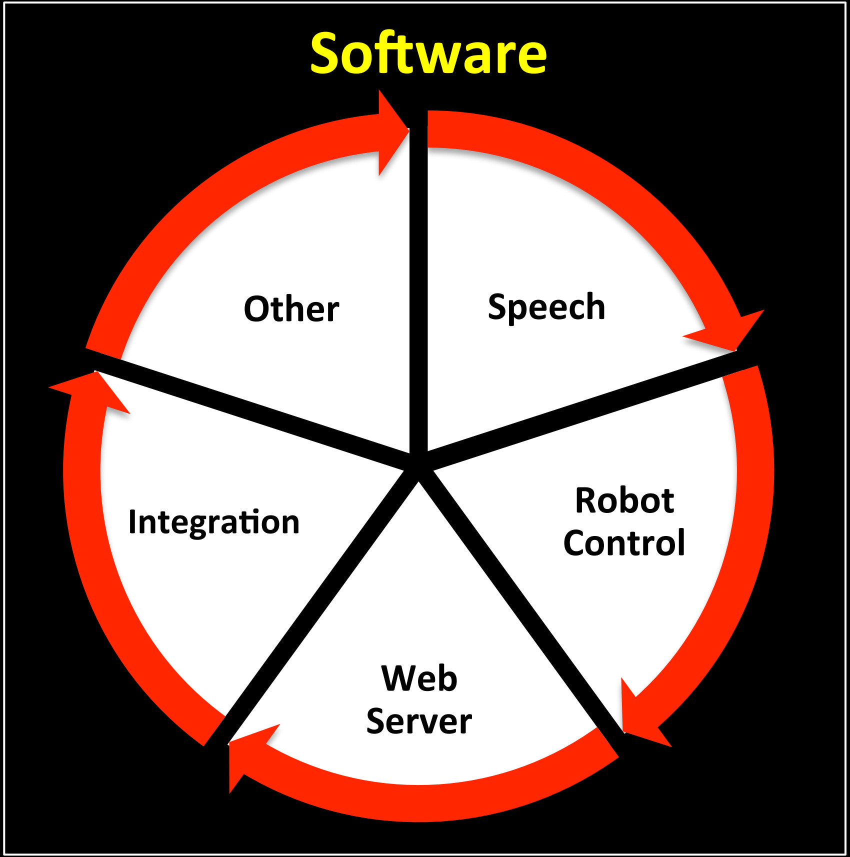




NodeBot Rover Demo

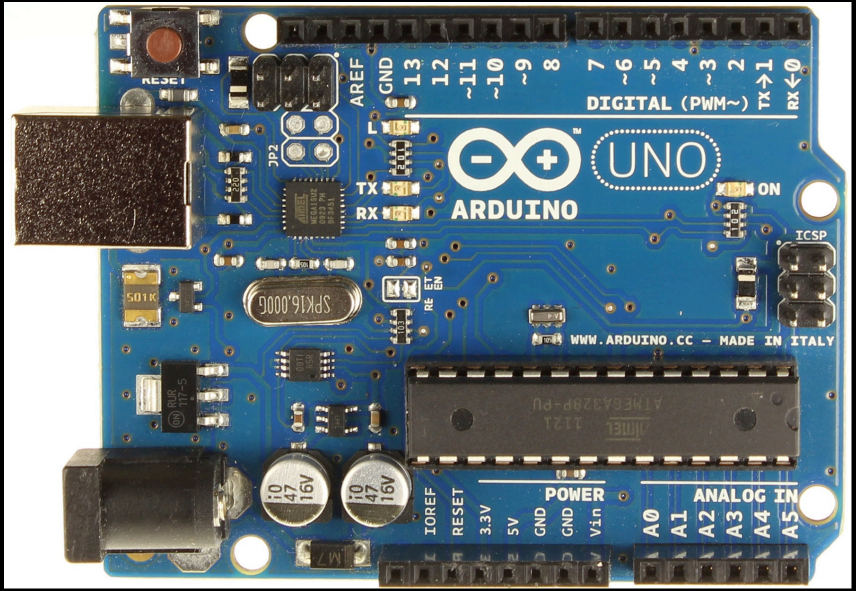
NodeBot Rover Component Overview





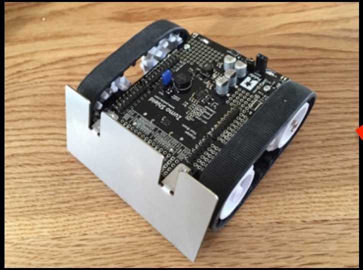
The Arduino Ecosystem

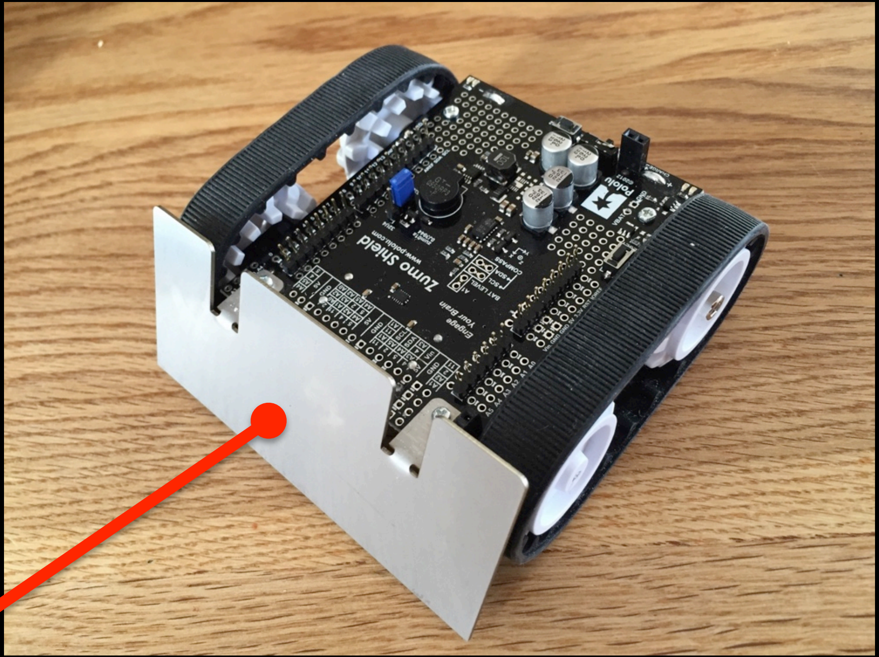
- MicroController platform.
- Many different Arduino models.
- Open Source design.
- MicroControllers extensible via “Shields”.

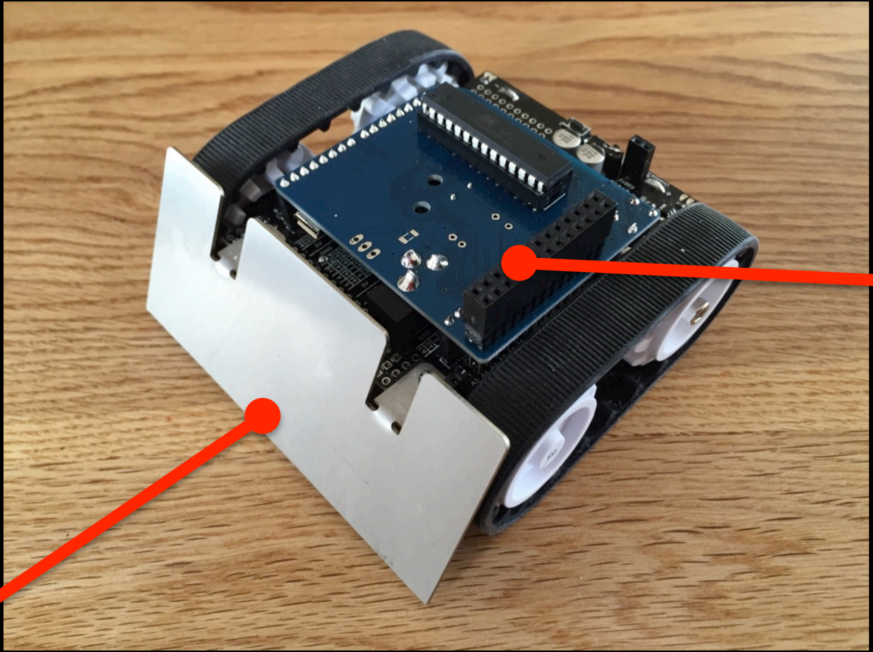


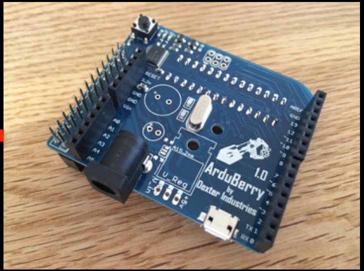
Putting the Hardware Together

Arduino Chassis



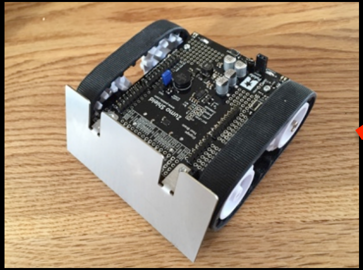




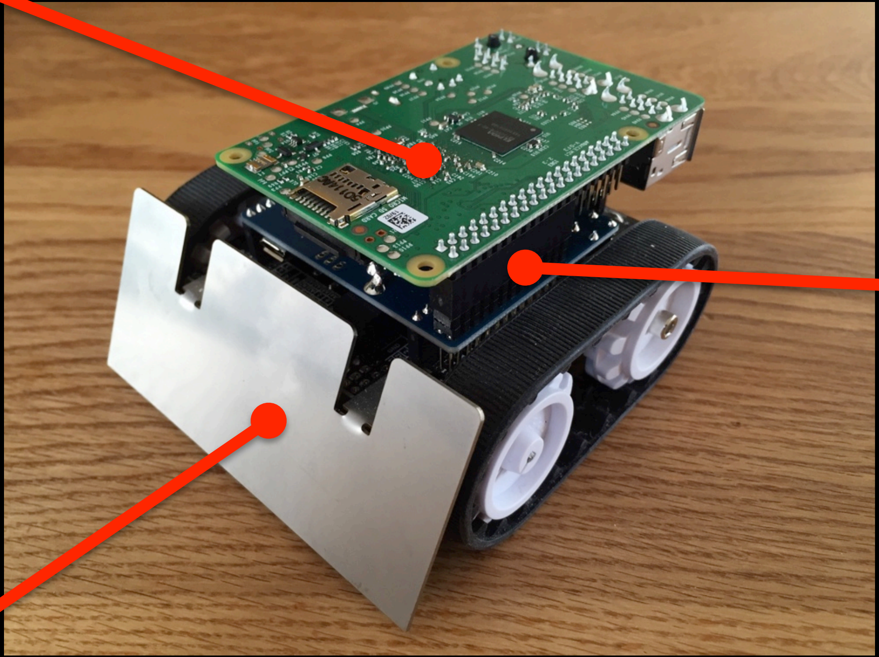


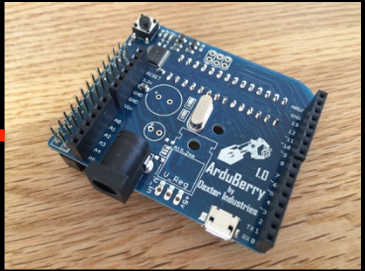
Arduberry MicroController

Arduino Chassis



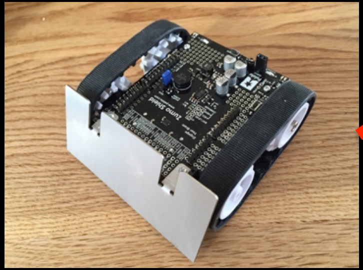
Raspberry PI 2

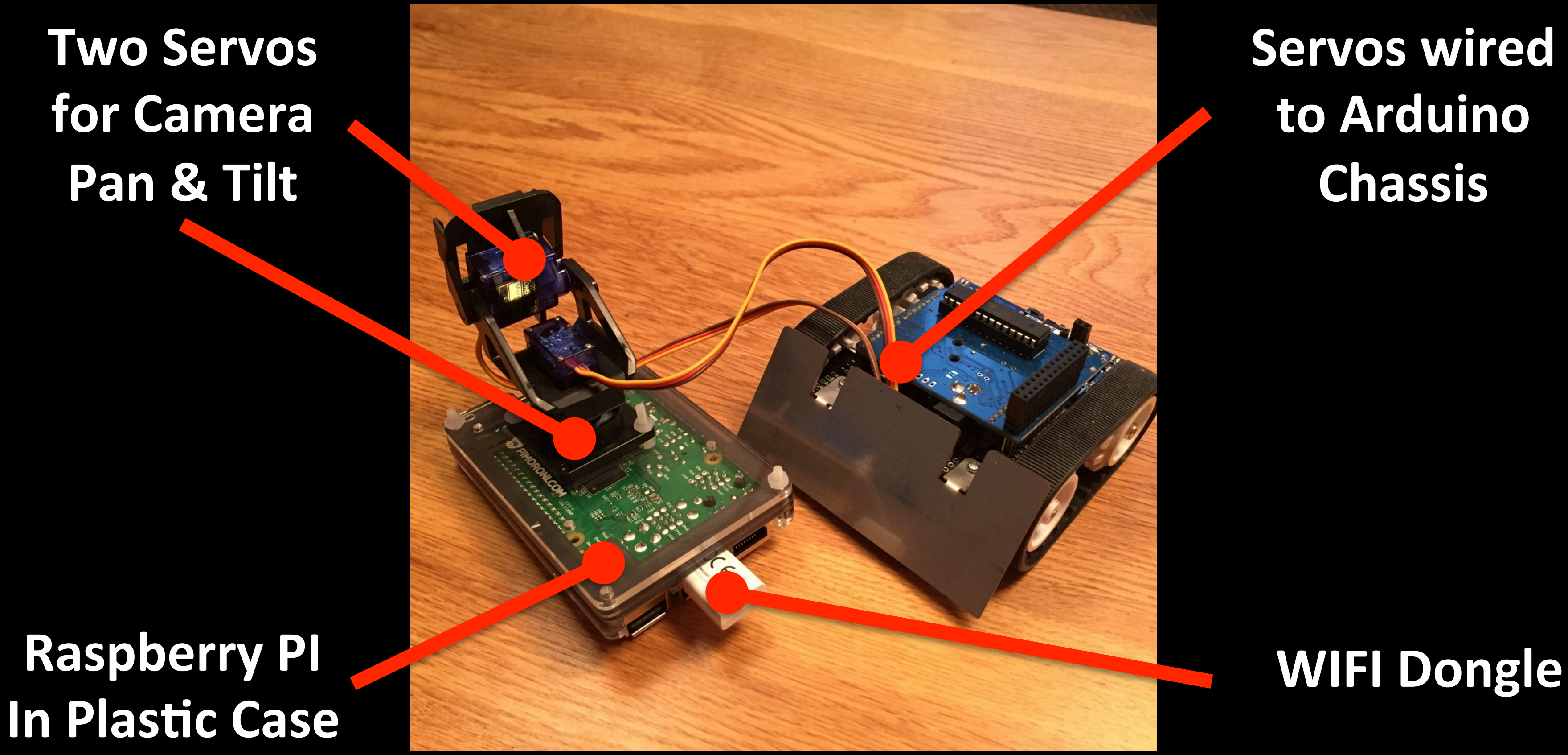




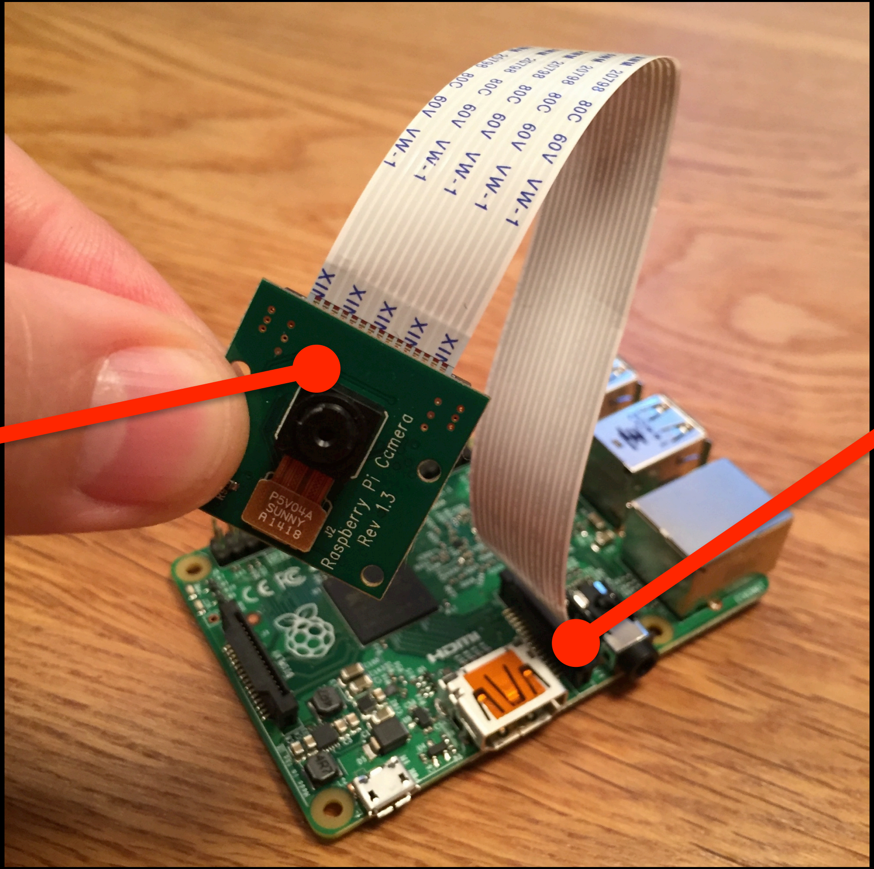
Arduberry MicroController

Arduino Chassis




Two Servos for Camera Pan & Tilt
Servos wired to Arduino Chassis
Raspberry PI In Plastic Case
WIFI Dongle

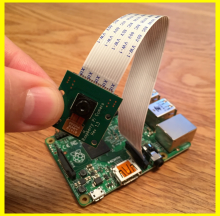
Raspberry PI Camera



Wired directly to the Raspberry PI

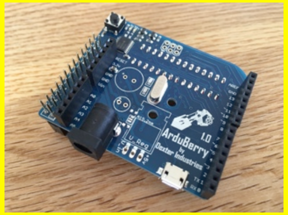
NodeBot Rover Hardware

Sensors

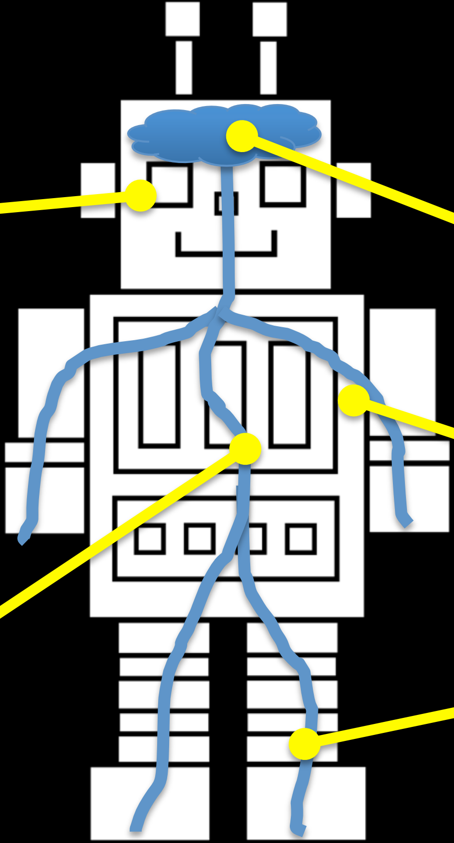


Raspberry PI Cam

Nervous System



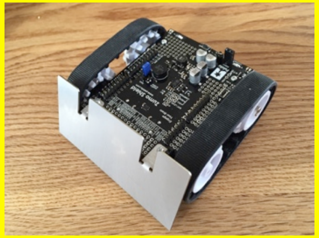
Arduberry Microcontroller



Brain

Raspberry PI 2

Actuators



Arduino Chassis

Servo

Servo

Combining the Raspberry PI & Arduino

Raspberry PI

- Linux PC.
- Supports USB peripherals.
- Programming.

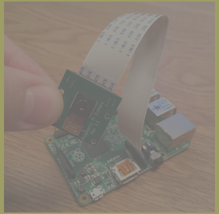
Arduino Platform

- MicroController.
- Robust.
- Flexibility (input/output).

The whole is greater than the sum of parts!

NodeBot Rover Hardware

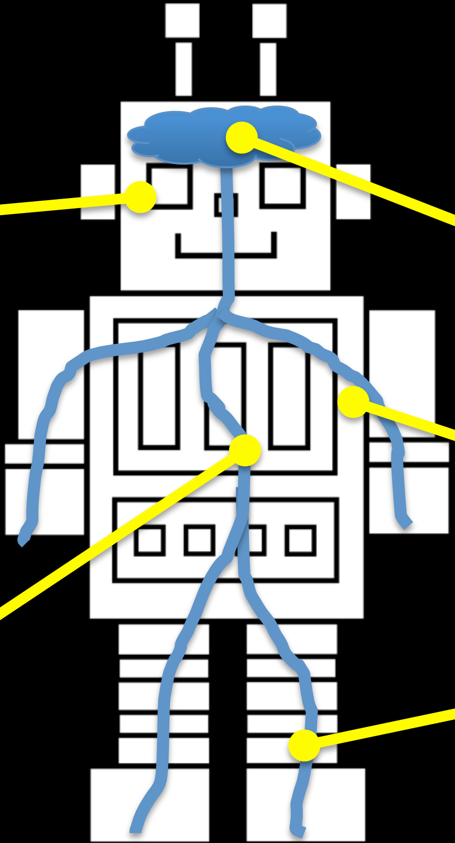
Sensors



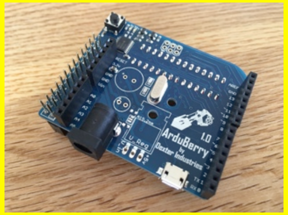
Raspberry PI Cam

Brain

Raspberry PI 2

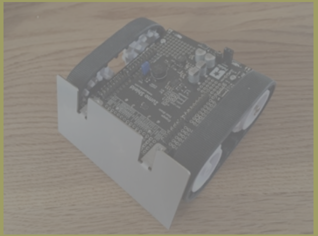


Nervous System



Arduberry Microcontroller

Actuators

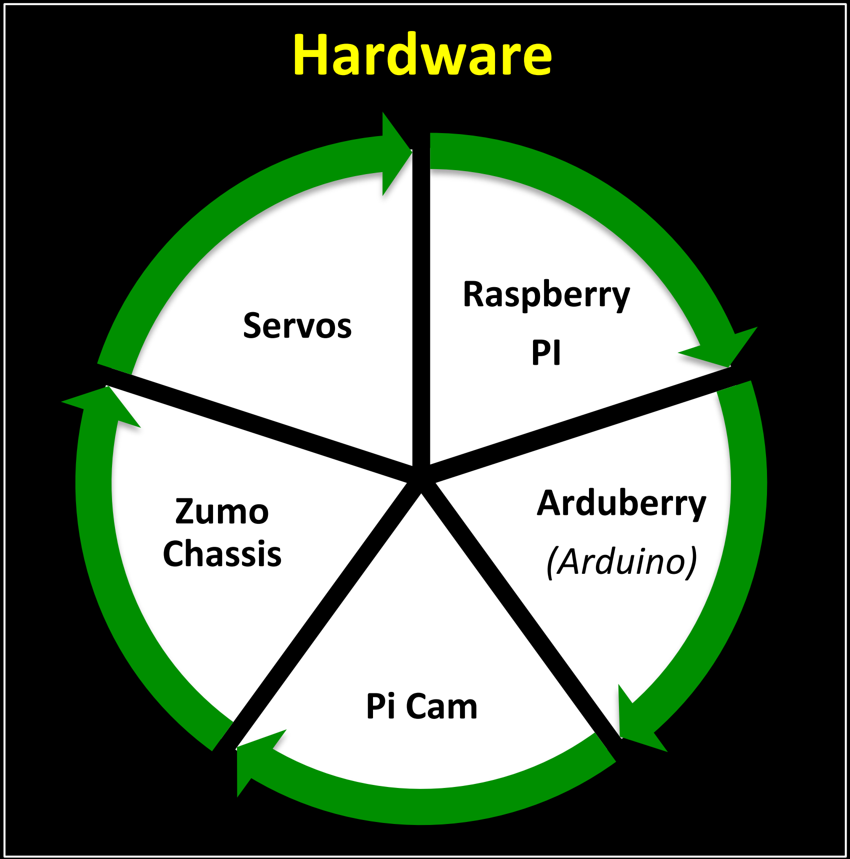


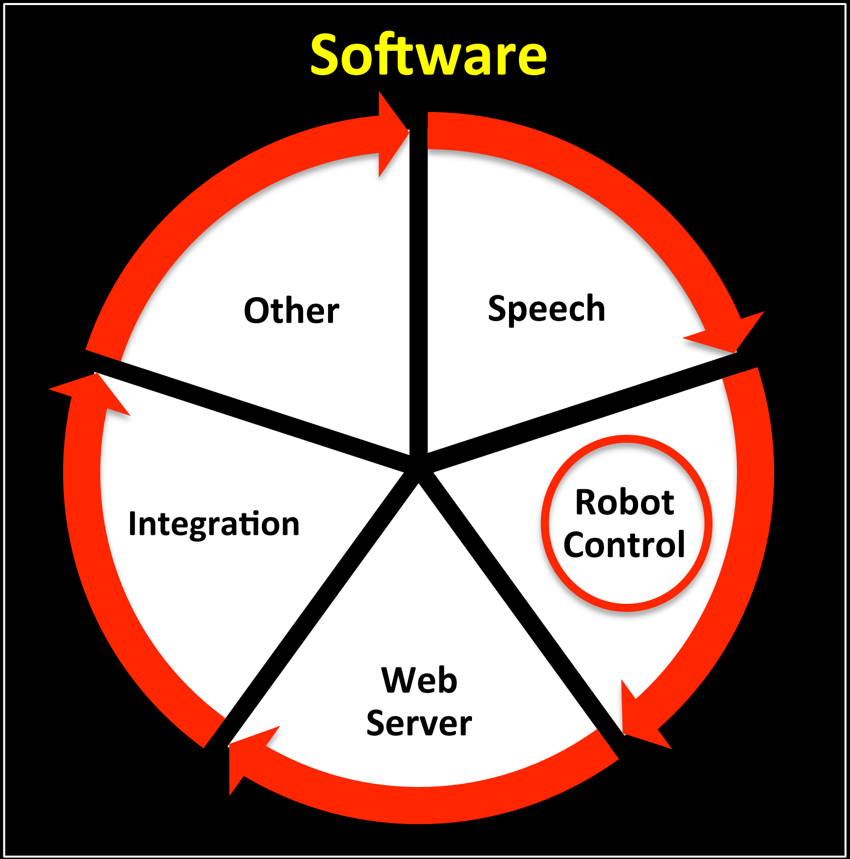
Arduino Chassis

Servo

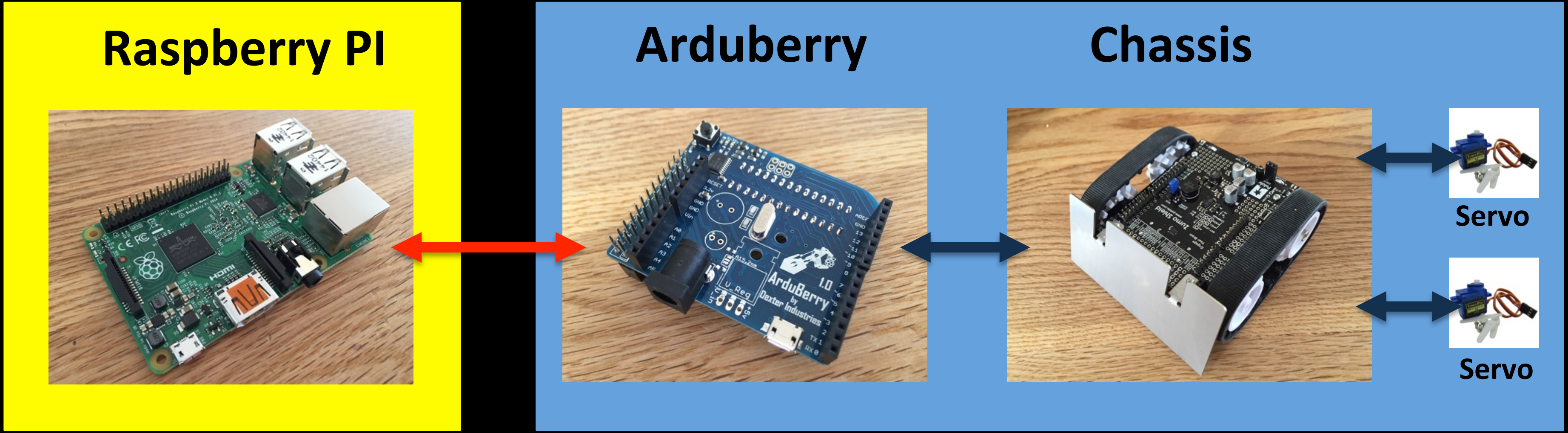
Servo

NodeBot Rover Component Overview



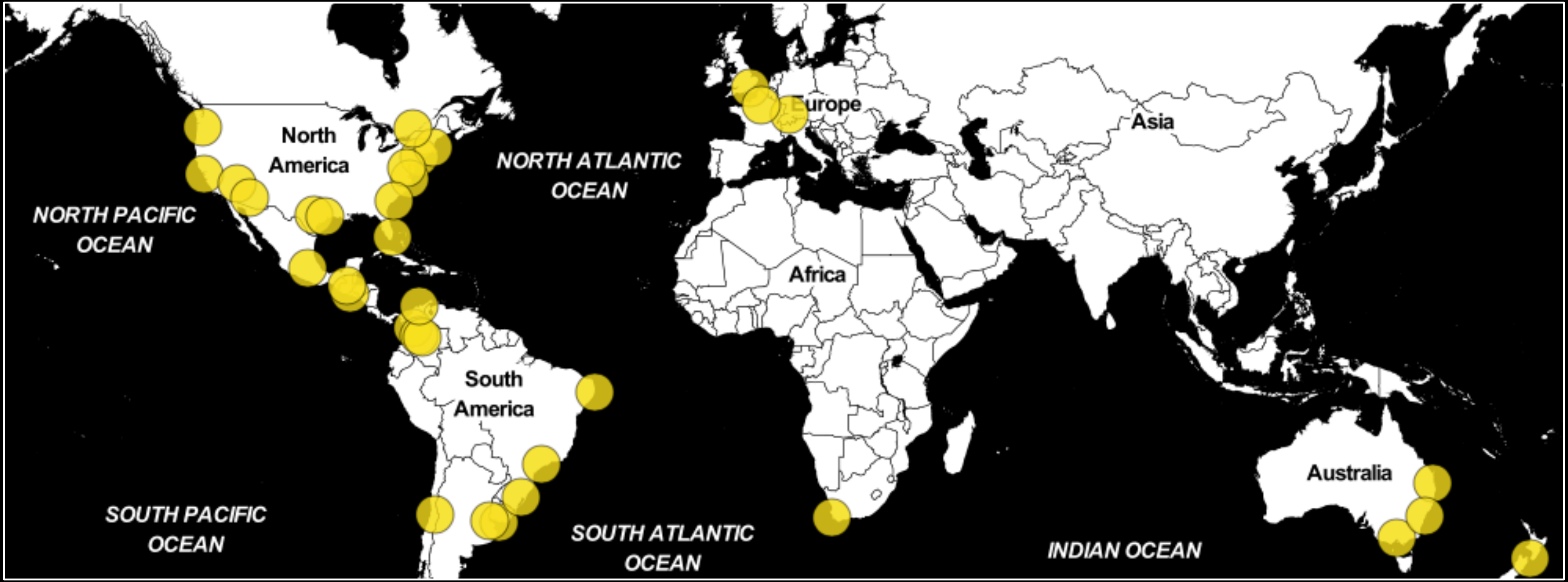


Software Communication across Hardware Layers

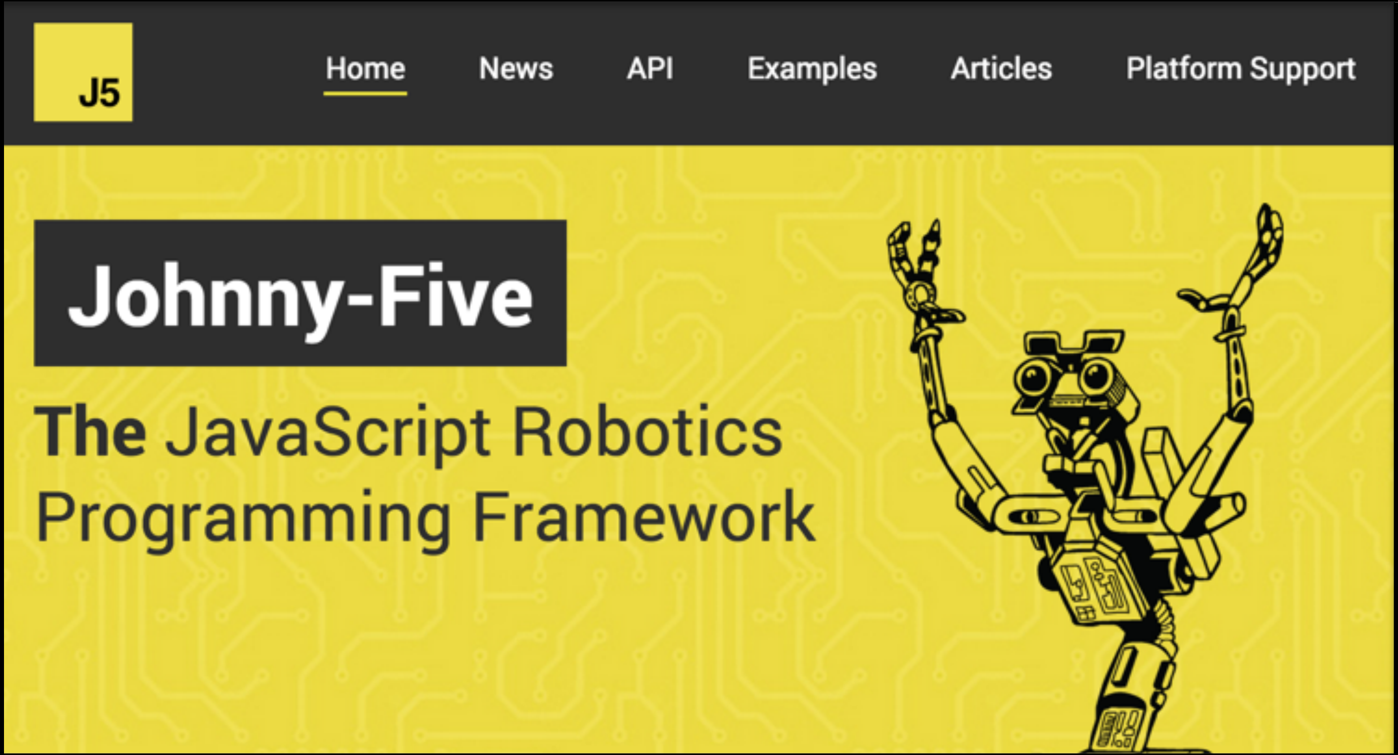


{ JavaScript } { Binary (Compiled C / C++) }

The NodeBots Movement

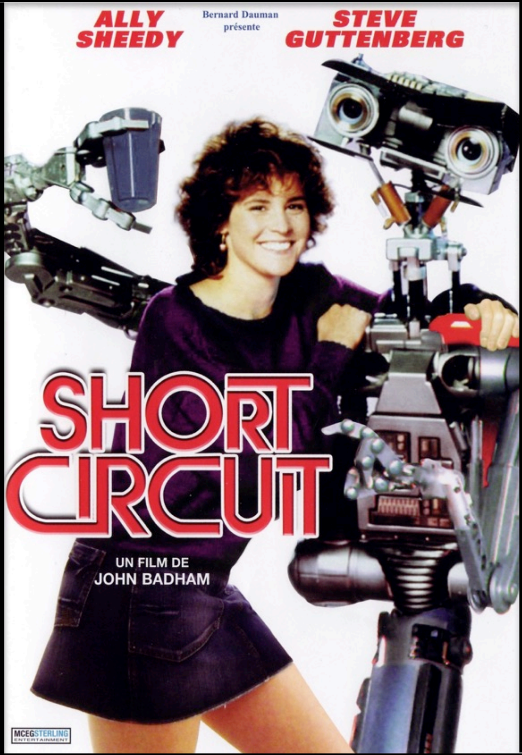


Source : nodebots.io



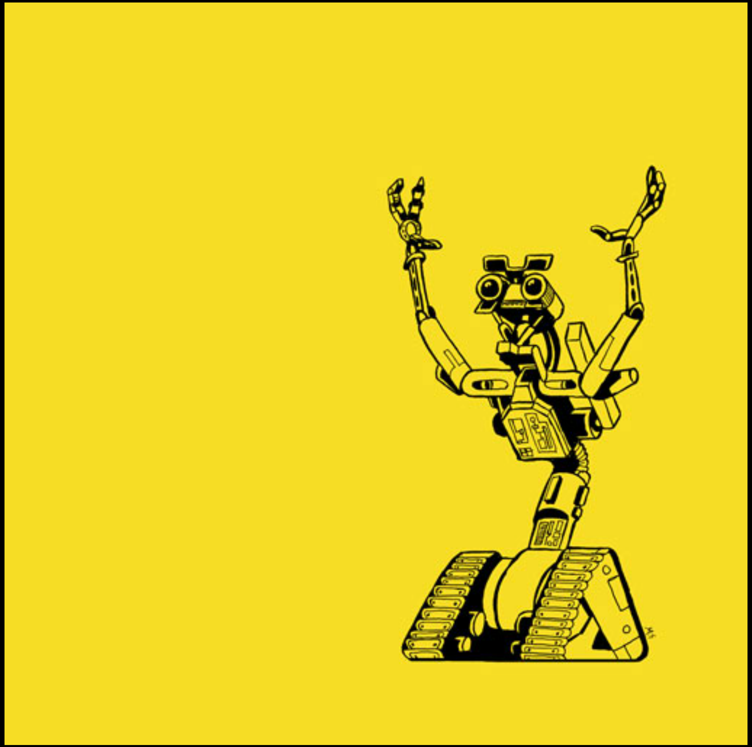
Johnny-Five

The JavaScript Robotics Programming Framework



Benefits of using Johnny-Five

1. Maturity
2. Community
3. DSL
4. Portability
5. Open Source
6. Node.js ecosystem
7. REPL



Johnny-Five Code Example

```
var five = require("johnny-five");

var myBoard = new five.Board();

myBoard.on("ready", function() {

        var myLed = new five.Led(13);

        myLed.blink(500);

        this.repl.inject({
                replLed: myLed
        });
});
```

Imports J5 Dependancy

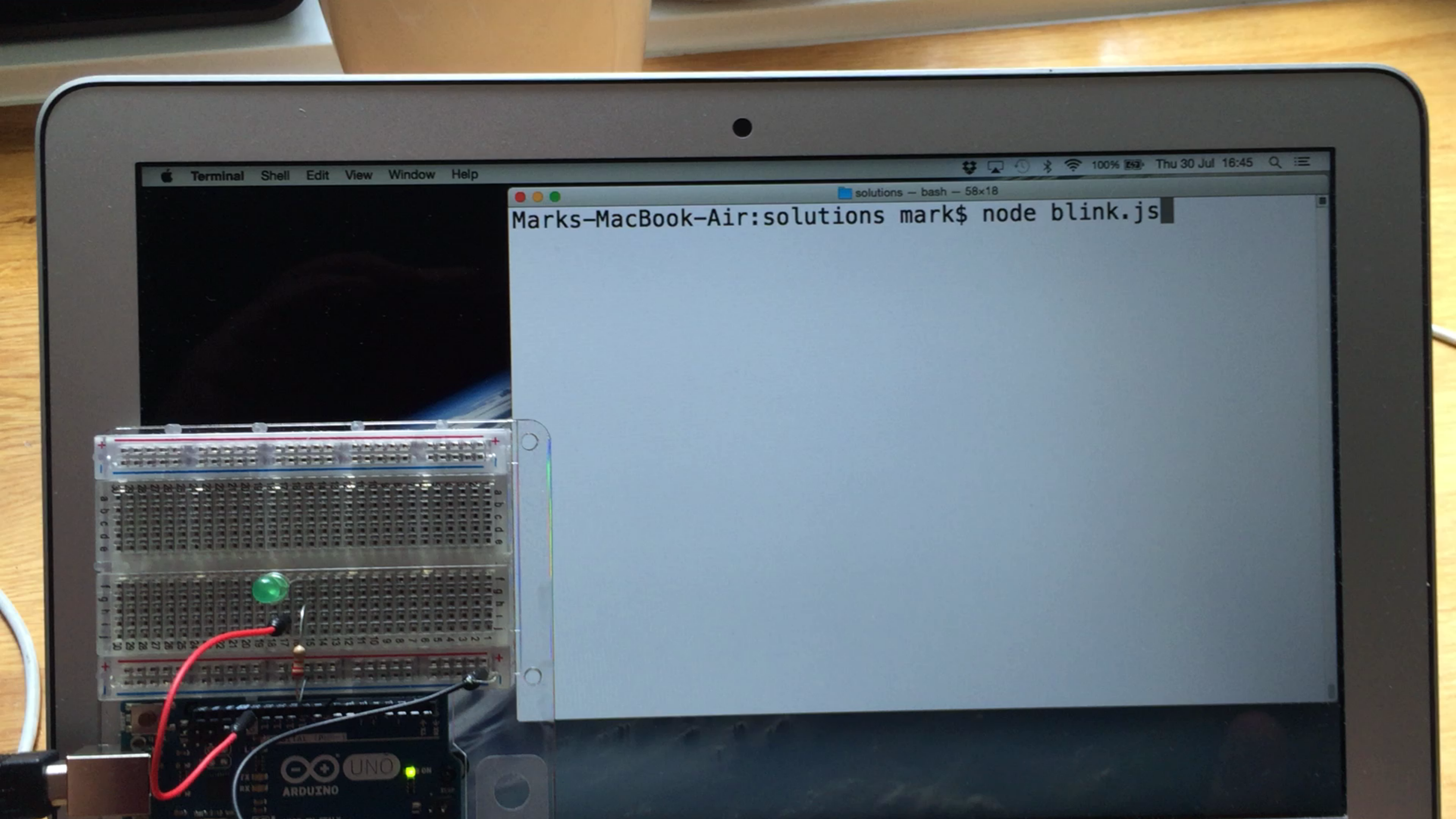
Initialises UNO

Code block triggered by UNO ”Ready” Event

Declares LED as connected to UNO Pin 13

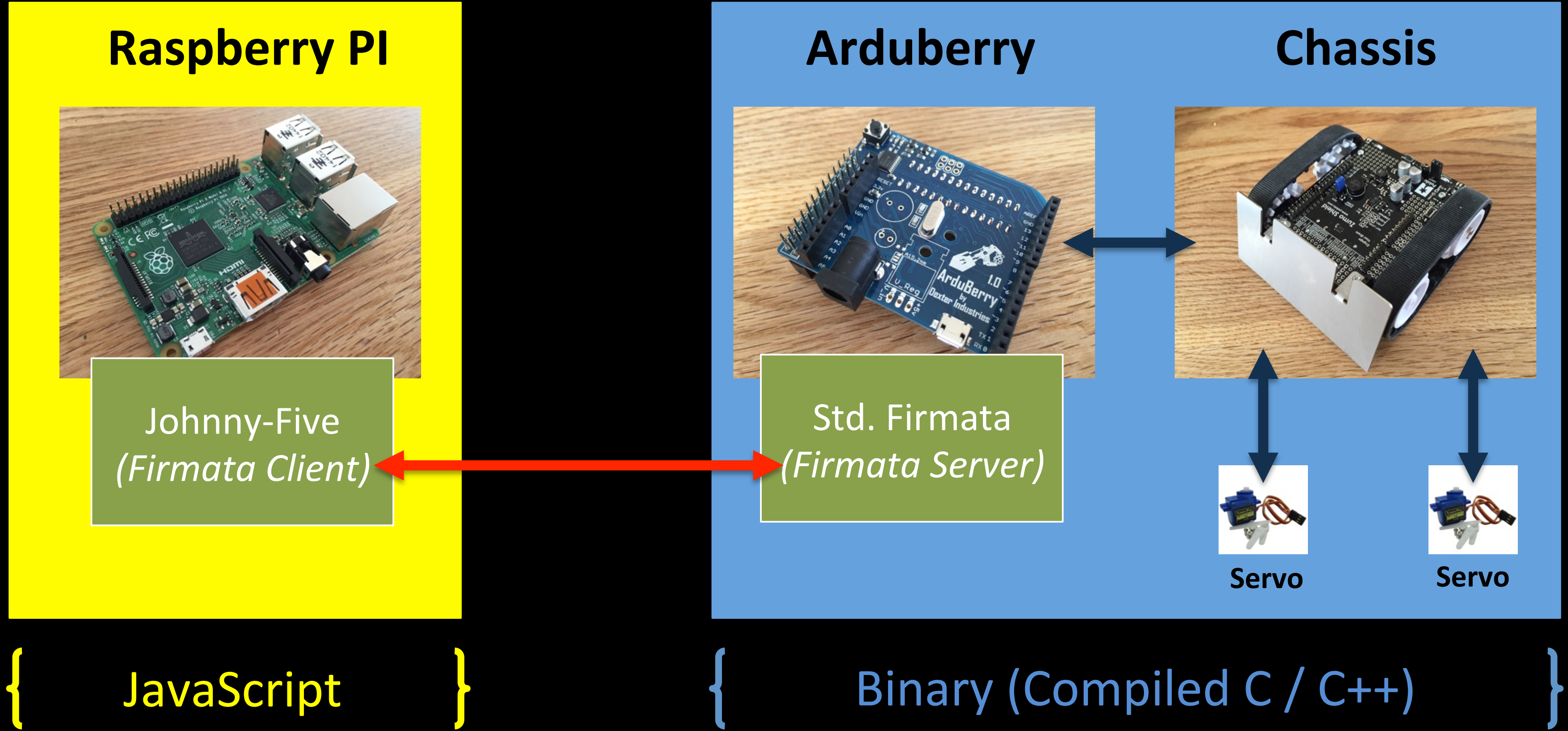
Blinks LED every 500 milliseconds

Adds LED instance to REPL



```
Marks-MacBook-Air:solutions mark$ node blink.js
```

Bridging the gap with Firmata



node JS™

npm

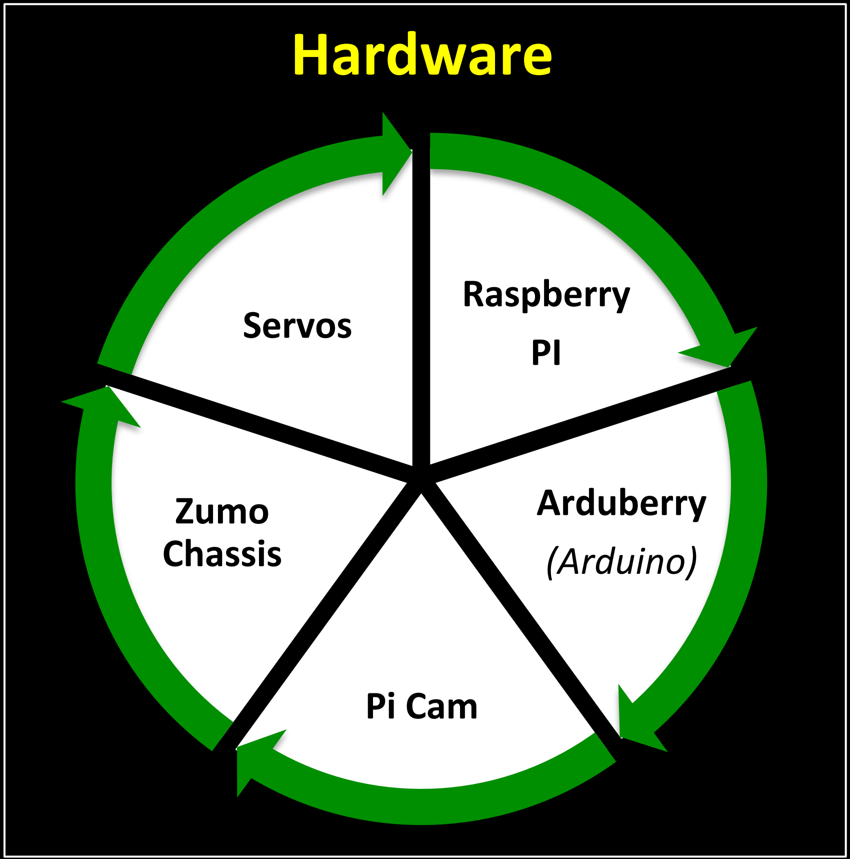
Getting Started with Johnny-Five

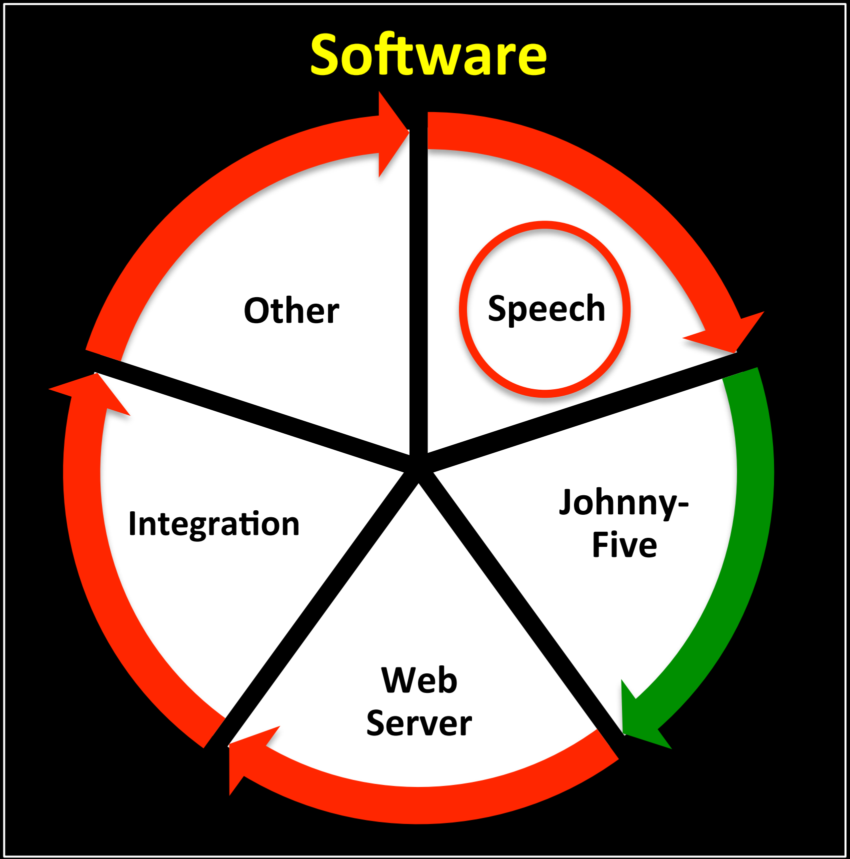
1. Buy an **Arduino Experimenters Kit.**

2. Follow the tutorials at **http://node-ardx.org.**

3. Visit **http://johnny-five.io** for more information and inspiration.

NodeBot Rover Component Overview





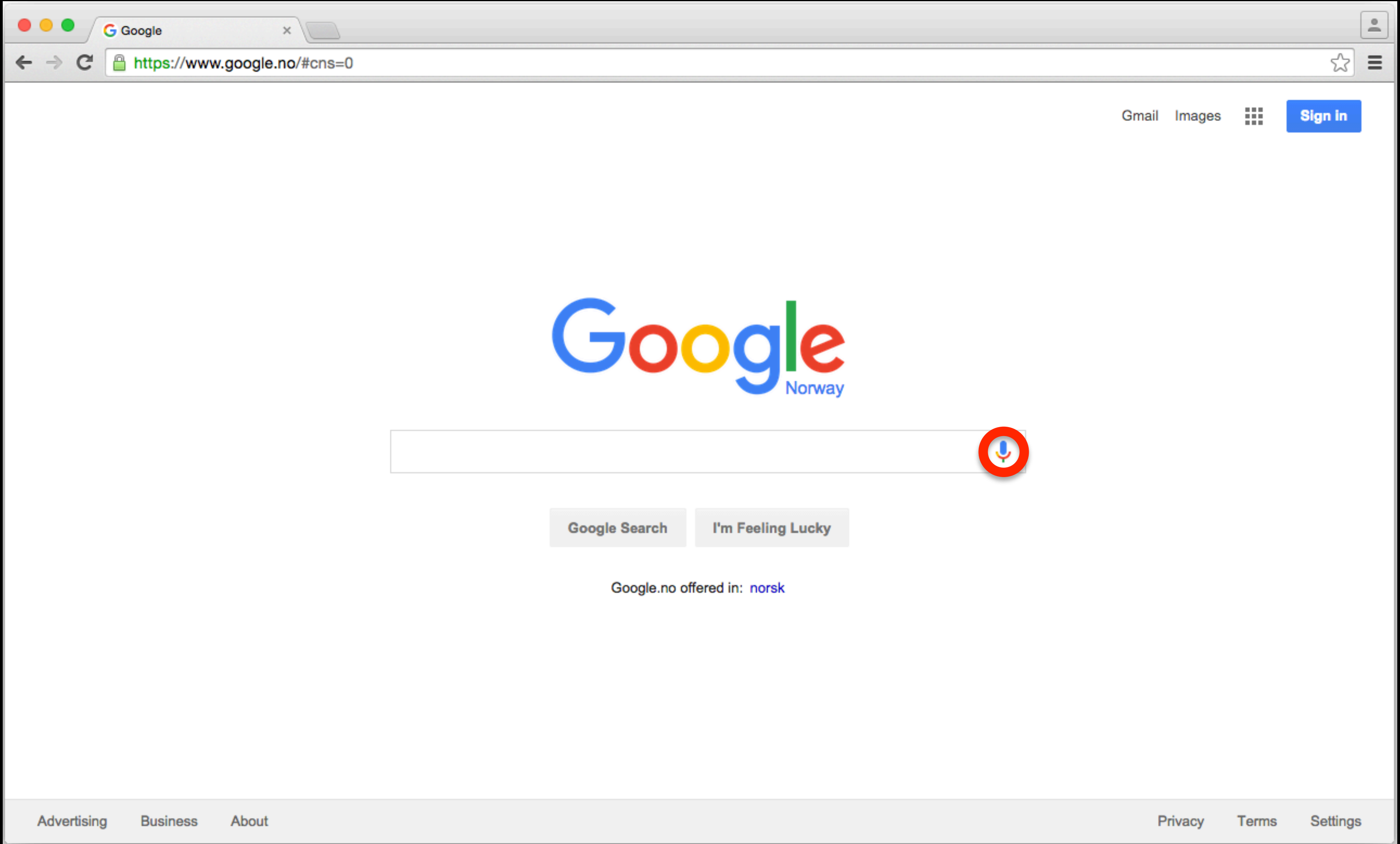
Speech Recognition Requirements

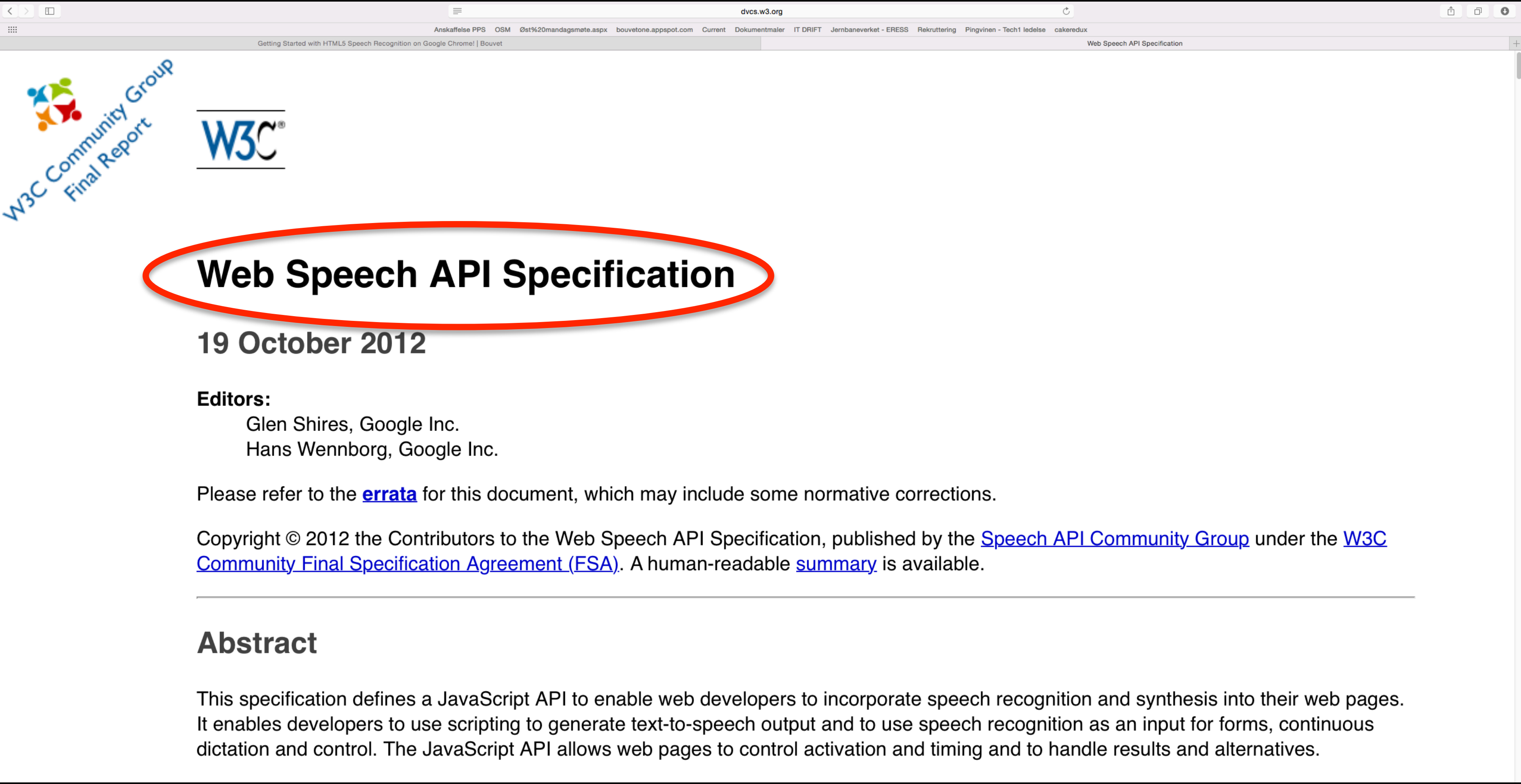
Need to have

- Quality and speed of speech recognition.
- Free, no restrictions.
- Software based.
- JavaScript.

Nice to have

- Speech to text.
- One stop service.
- Battle tested.


Google
https://www.google.no/#cns=0
Gmail
Images
Sign in
Google
Norway
Google Search
I'm Feeling Lucky
Google.no offered in: norsk
Advertising
Business
About
Privacy
Terms
Settings


dvcs.w3.org
Anskaffelse PPS OSM Øst%20mandagsmøte.aspx bouvetone.appspot.com Current Dokumentmaler IT DRIFT Jernbaneverket - ERESS Rekruttering Pingvinen - Tech1 ledelse cakeredux
Getting Started with HTML5 Speech Recognition on Google Chrome! | Bouvet
Web Speech API Specification
W3C Community Group Final Report

Web Speech API Specification

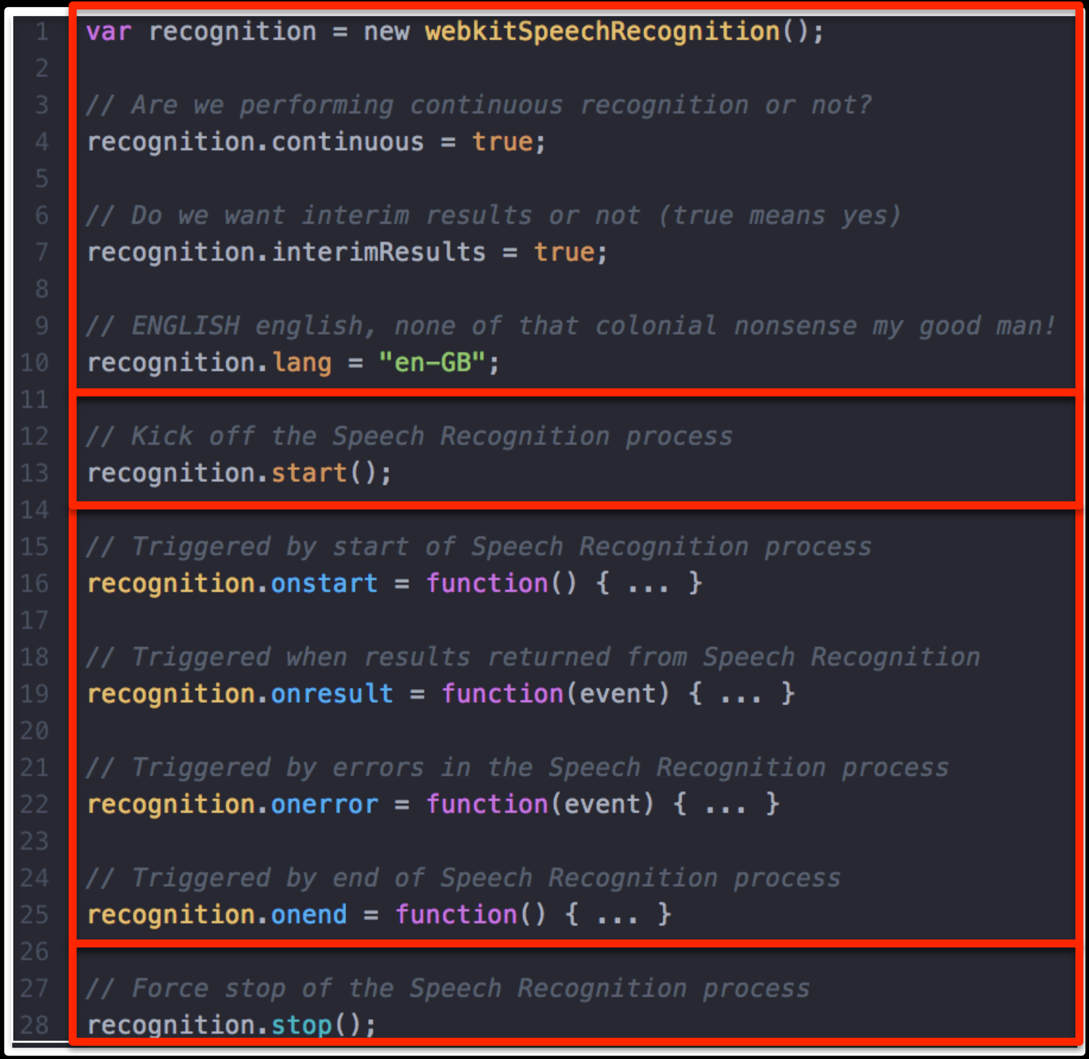
19 October 2012

Editors:
Glen Shires, Google Inc.
Hans Wennborg, Google Inc.

Please refer to the **errata** for this document, which may include some normative corrections.

Abstract

This specification defines a JavaScript API to enable web developers to incorporate speech recognition and synthesis into their web pages. It enables developers to use scripting to generate text-to-speech output and to use speech recognition as an input for forms, continuous dictation and control. The JavaScript API allows web pages to control activation and timing and to handle results and alternatives.

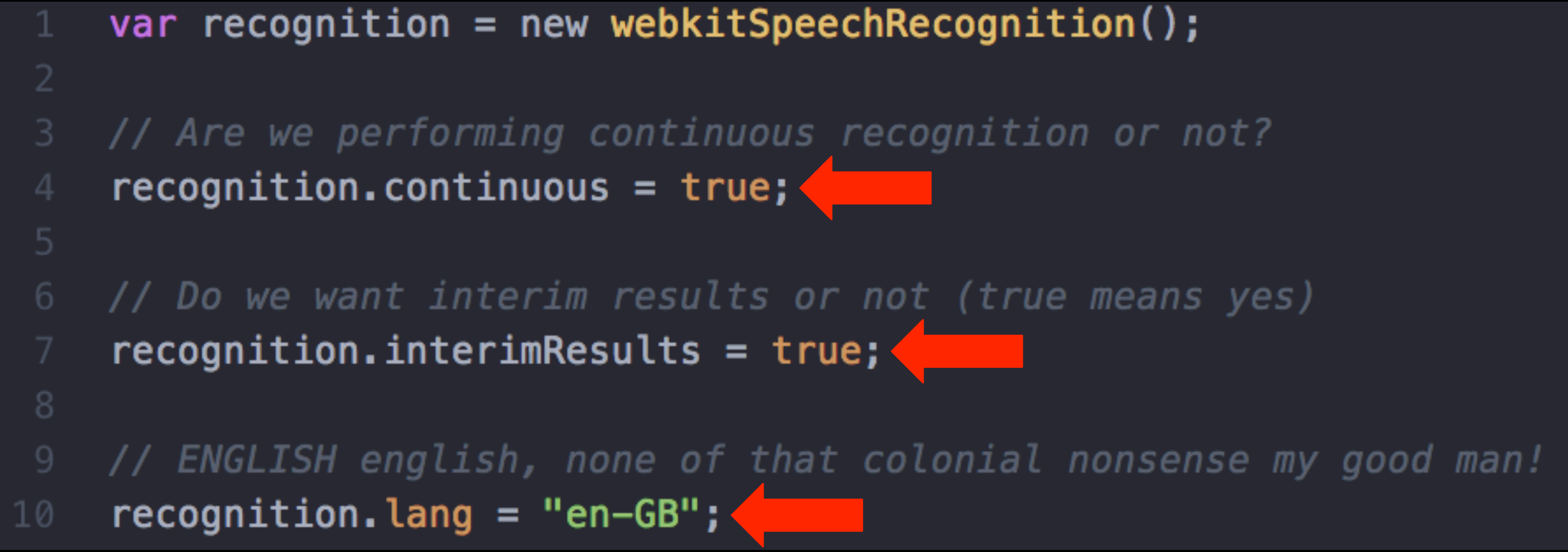


```
var recognition = new webkitSpeechRecognition();

// Are we performing continuous recognition or not?
recognition.continuous = true;

// Do we want interim results or not (true means yes)
recognition.interimResults = true;

// ENGLISH english, none of that colonial nonsense my good man!
recognition.lang = "en-GB";

// Kick off the Speech Recognition process
recognition.start();

// Triggered by start of Speech Recognition process
recognition.onstart = function() { ... }

// Triggered when results returned from Speech Recognition
recognition.onresult = function(event) { ... }

// Triggered by errors in the Speech Recognition process
recognition.onerror = function(event) { ... }

// Triggered by end of Speech Recognition process
recognition.onend = function() { ... }

// Force stop of the Speech Recognition process
recognition.stop();
```

Web Speech API

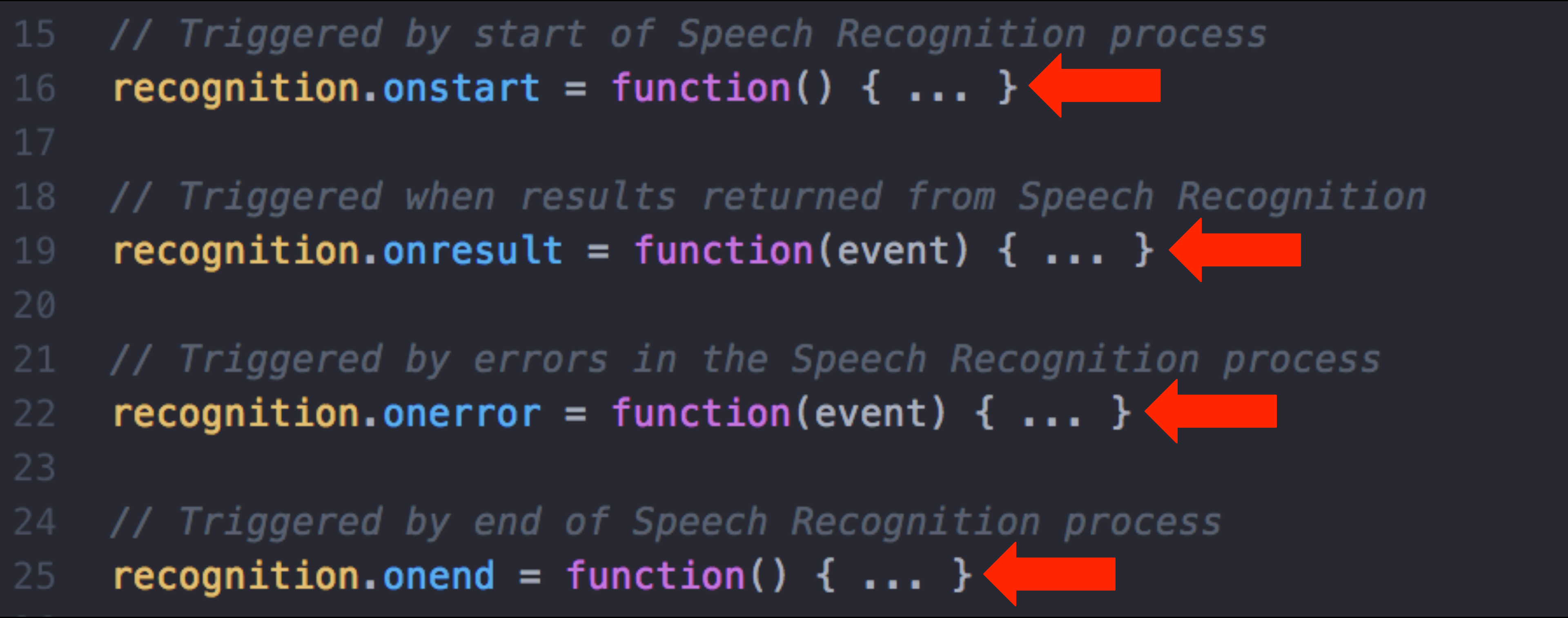
(limited to Google Chrome)

Web Speech API : Configuration



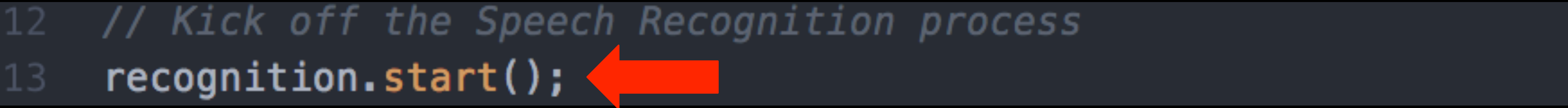
```
var recognition = new webkitSpeechRecognition();

// Are we performing continuous recognition or not?
recognition.continuous = true;

// Do we want interim results or not (true means yes)
recognition.interimResults = true;

// ENGLISH english, none of that colonial nonsense my good man!
recognition.lang = "en-GB";
```

Web Speech API : Events

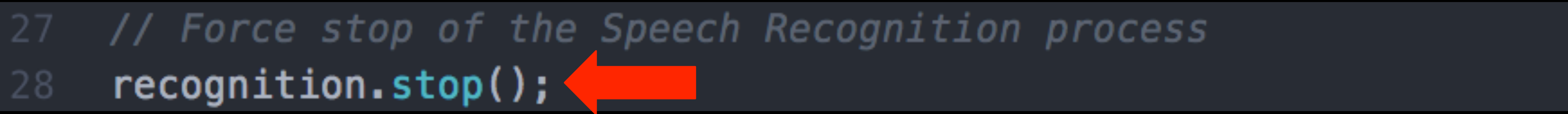


```
// Triggered by start of Speech Recognition process
recognition.onstart = function() { ... }

// Triggered when results returned from Speech Recognition
recognition.onresult = function(event) { ... }

// Triggered by errors in the Speech Recognition process
recognition.onerror = function(event) { ... }

// Triggered by end of Speech Recognition process
recognition.onend = function() { ... }
```

Web Speech API : Control

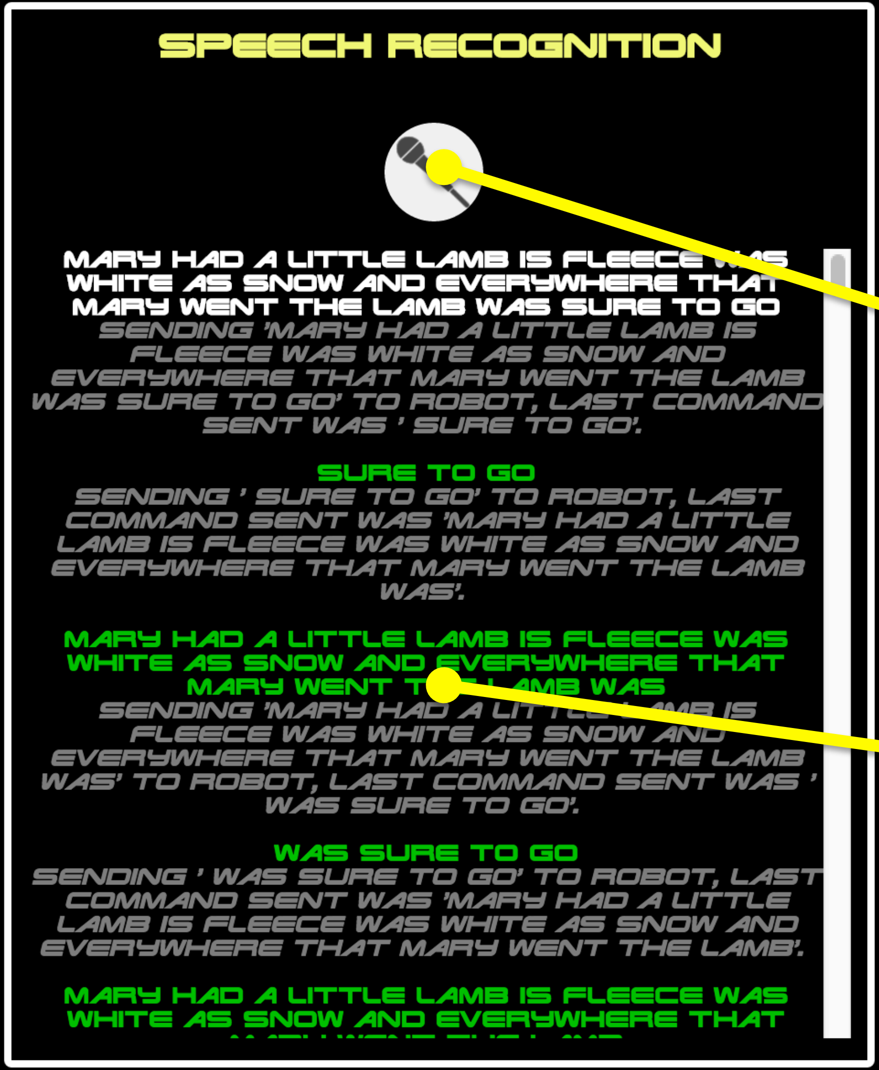


```
// Kick off the Speech Recognition process
recognition.start();
```



```
// Force stop of the Speech Recognition process
recognition.stop();
```

Speech Demo


SPEECH RECOGNITION

MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB WAS SURE TO GO

SENDING 'MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB WAS SURE TO GO' TO ROBOT, LAST COMMAND SENT WAS ' SURE TO GO'.

SURE TO GO

SENDING ' SURE TO GO' TO ROBOT, LAST COMMAND SENT WAS 'MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB WAS'.

MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB WAS

SENDING 'MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB WAS' TO ROBOT, LAST COMMAND SENT WAS ' WAS SURE TO GO'.

WAS SURE TO GO

SENDING ' WAS SURE TO GO' TO ROBOT, LAST COMMAND SENT WAS 'MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT MARY WENT THE LAMB'.

MARY HAD A LITTLE LAMB IS FLEECE WAS WHITE AS SNOW AND EVERYWHERE THAT

Web Speech API : Robot

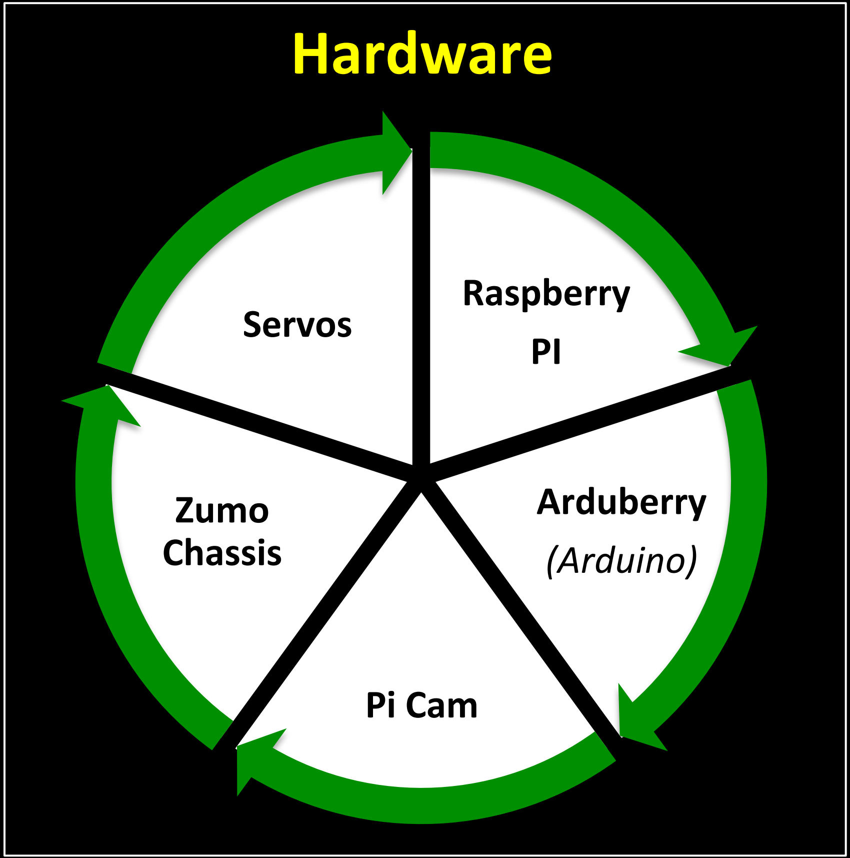
Button press and release connected to Web Speech API *start* and *stop* Control methods.

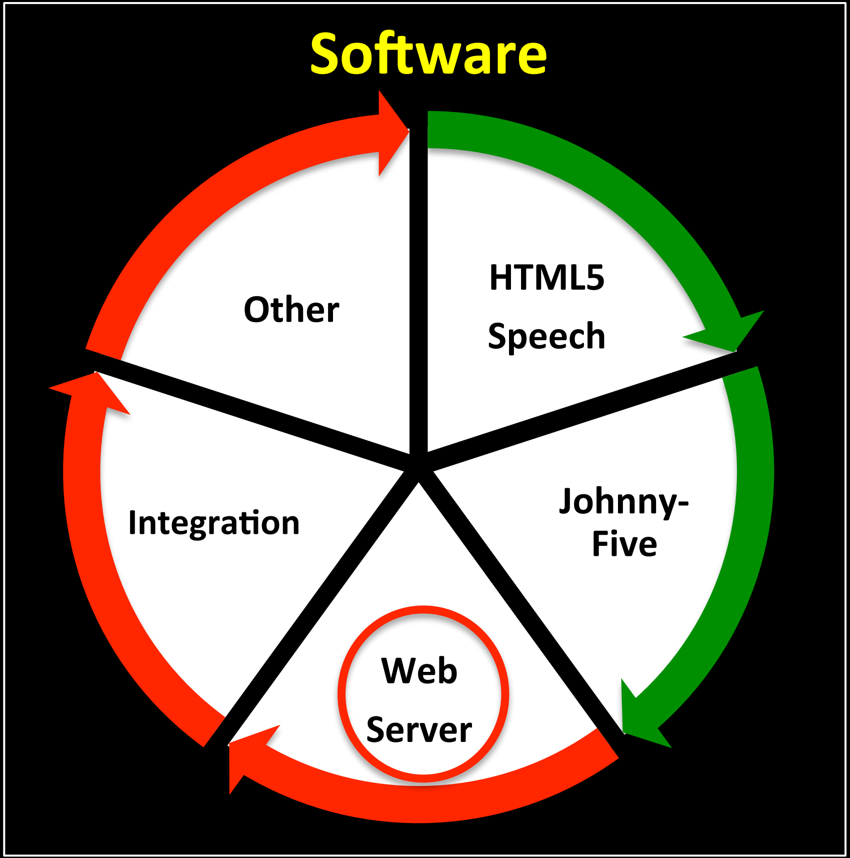
Continuous dictation switched on, to avoid cutting commands short.

Interim results switched on – shown in green text (final results in white).

Each set of results checked for uniqueness to avoid sending duplicate commands to the Robot.

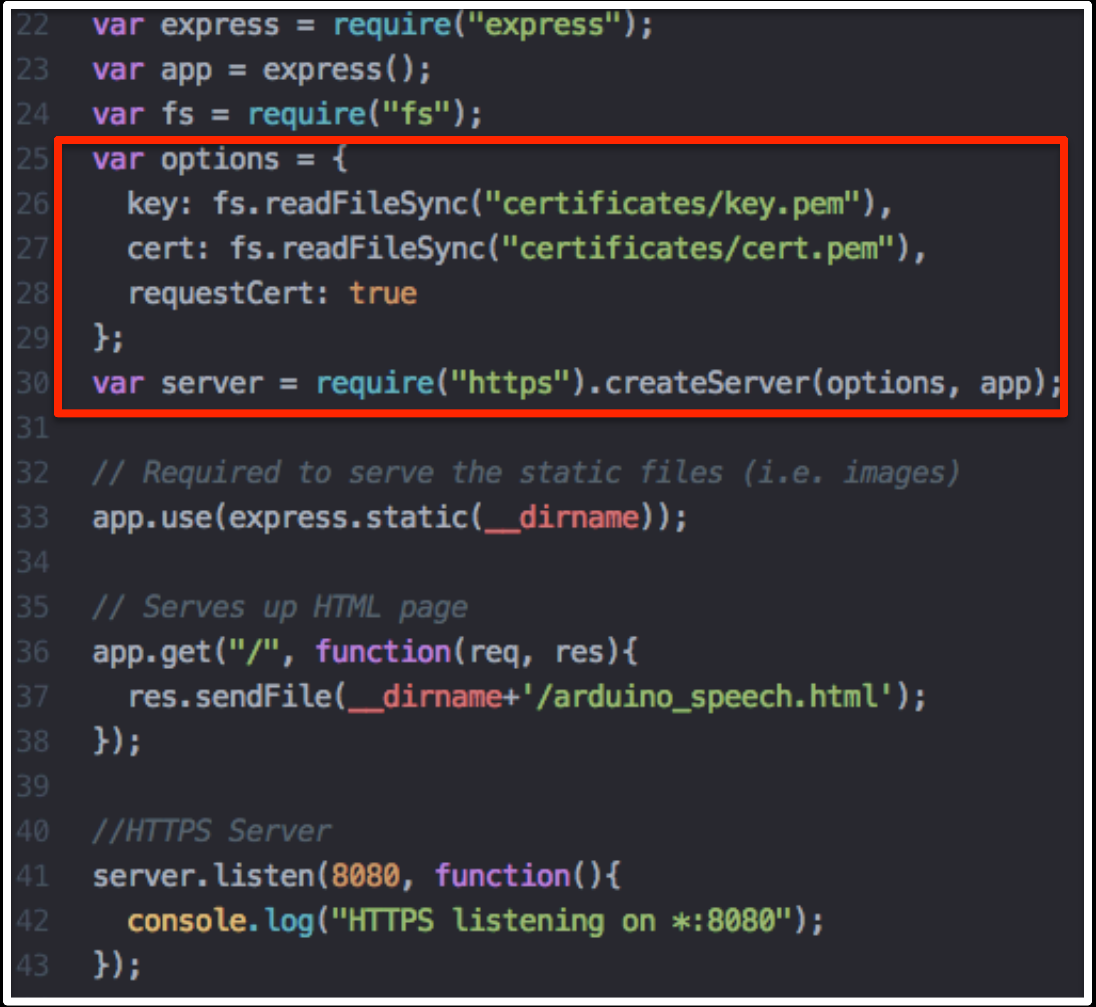
NodeBot Rover Component Overview





node JS ™

express

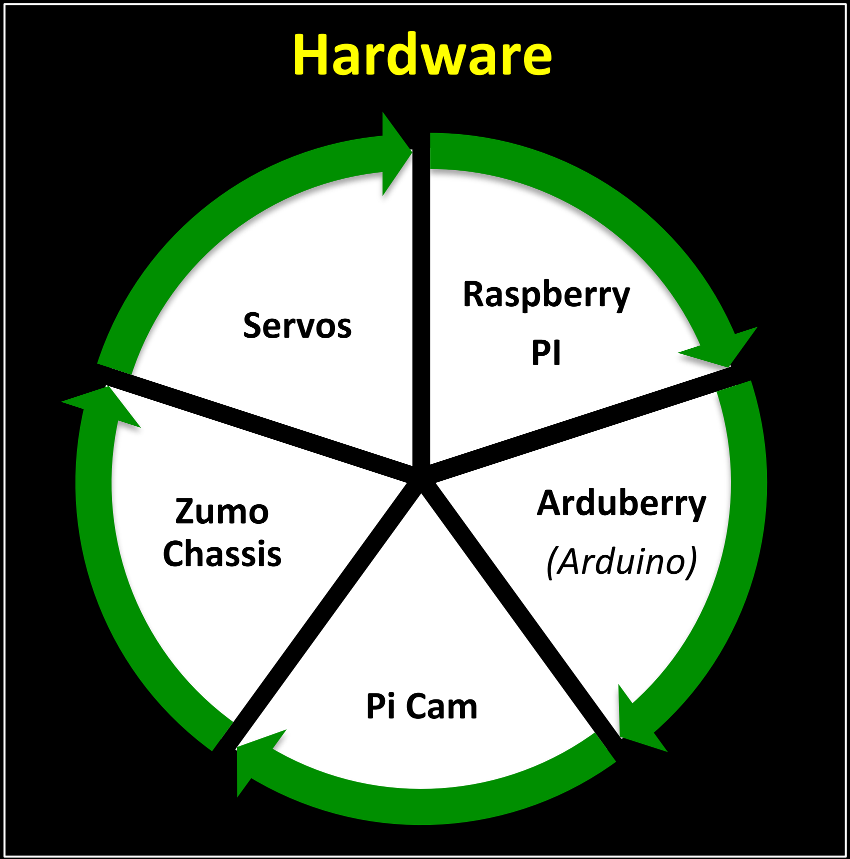


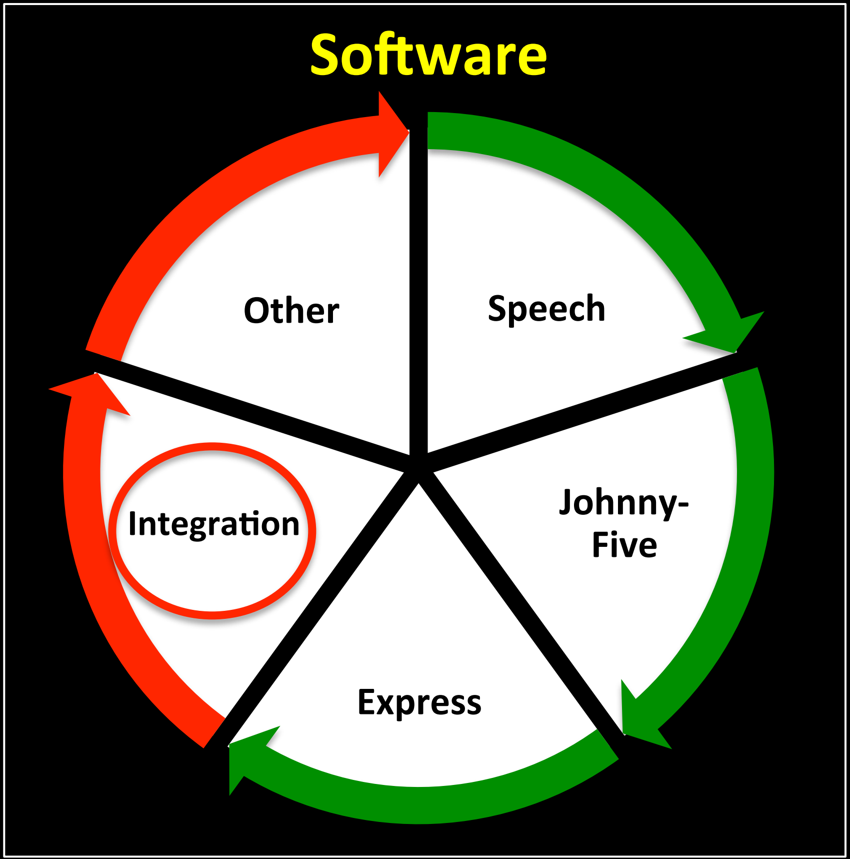
```
var express = require("express");
var app = express();
var fs = require("fs");
var options = {
  key: fs.readFileSync("certificates/key.pem"),
  cert: fs.readFileSync("certificates/cert.pem"),
  requestCert: true
};
var server = require("https").createServer(options, app);

// Required to serve the static files (i.e. images)
app.use(express.static(__dirname));

// Serves up HTML page
app.get("/", function(req, res){
  res.sendFile(__dirname+'/arduino_speech.html');
});

//HTTPS Server
server.listen(8080, function(){
  console.log("HTTPS listening on *:8080");
});
```

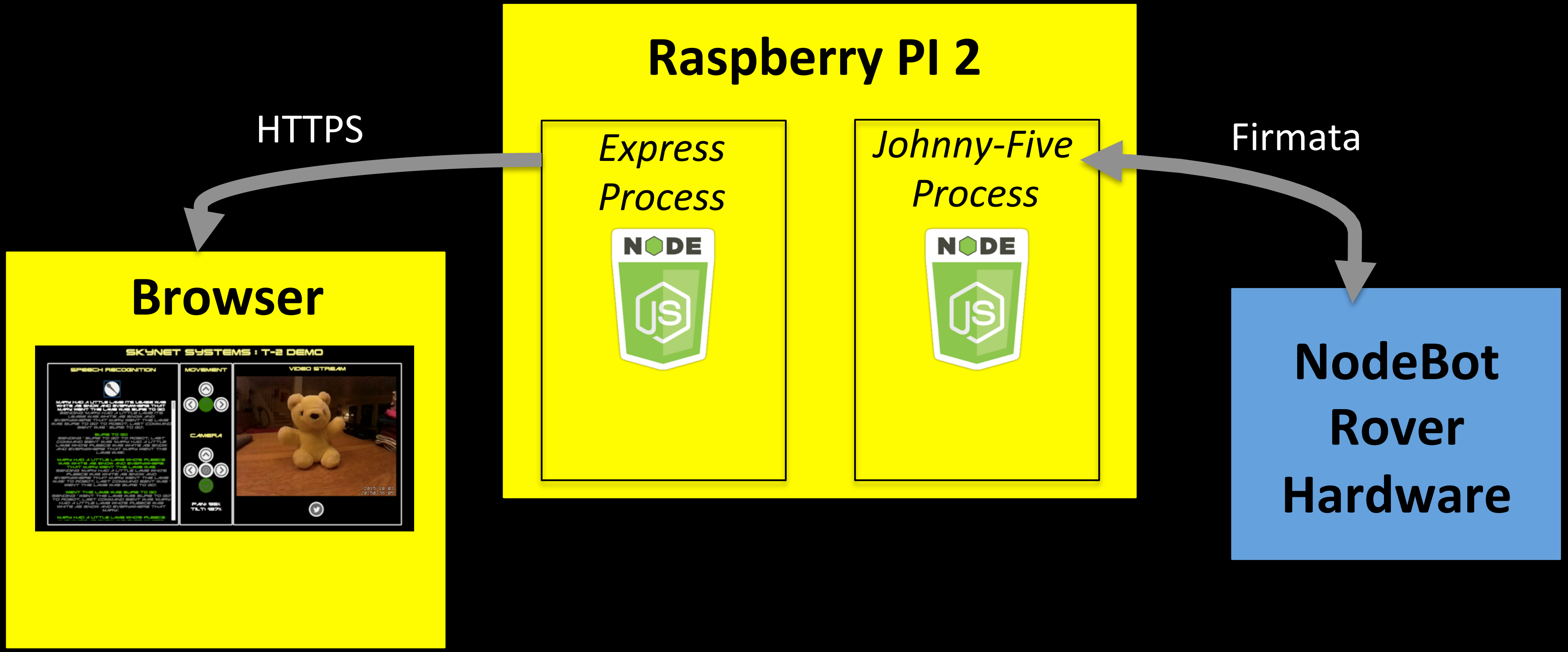
Node.js Express Web Server

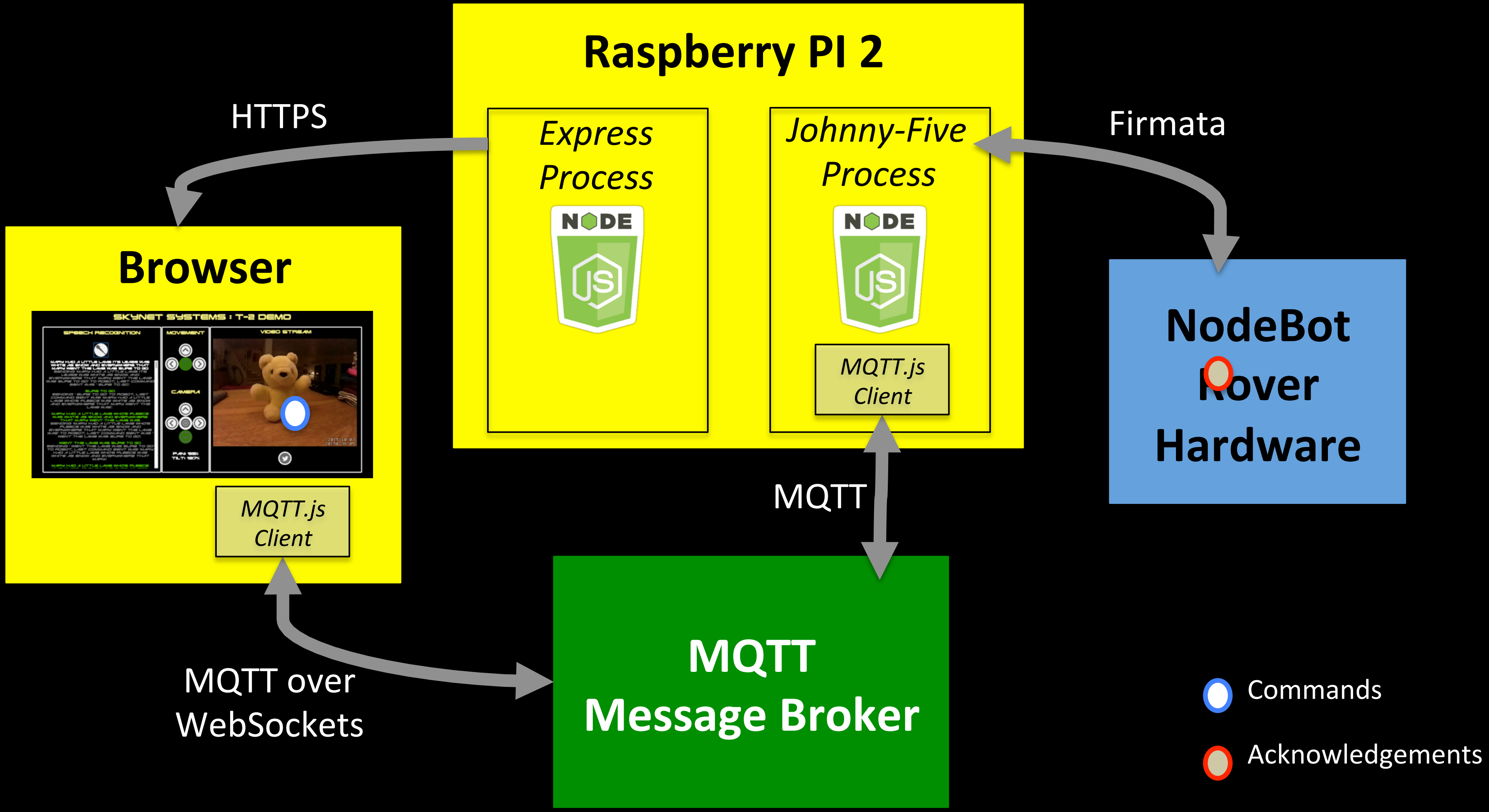
(17 lines of code)

NodeBot Rover Component Overview



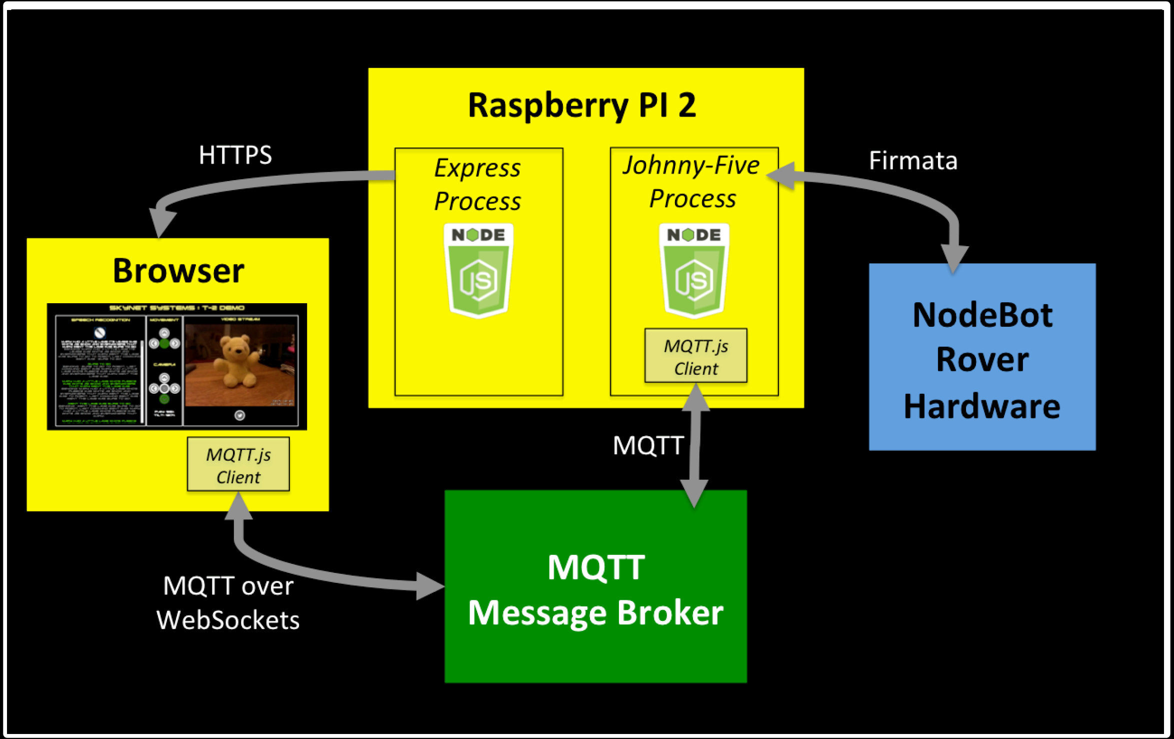



Raspberry PI 2
HTTPS
Express Process
Johnny-Five Process
Firmata
Browser
SKYNET SYSTEMS : T-2 DEMO
SPEECH RECOGNITION
MOVEMENT
CAMERA
VIDEO STREAM
NodeBot Rover Hardware


Raspberry PI 2
Express Process
Johnny-Five Process
MQTT.js Client
HTTPS
Firmata
Browser
SKYNET SYSTEMS : T-2 DEMO
SPEECH RECOGNITION
MOVEMENT
CAMERA
VIDEO STREAM
MQTT.js Client
NodeBot Rover Hardware
MQTT
MQTT over WebSockets
MQTT Message Broker
Commands
Acknowledgements

Why Add a Message Broker?

- Seperation of concerns.
- Isolate main components for easier testing.



MQTT – MQ Telemetry Transport

- Internet of Things connectivity protocol.
- Designed to be lightweight with a small footprint and little overhead.
- Is a protocol **and** a Pub-Sub Message Broker.
- Used by Facebook for pushing updates to mobile clients.

Adding MQTT to the NodeBot Rover

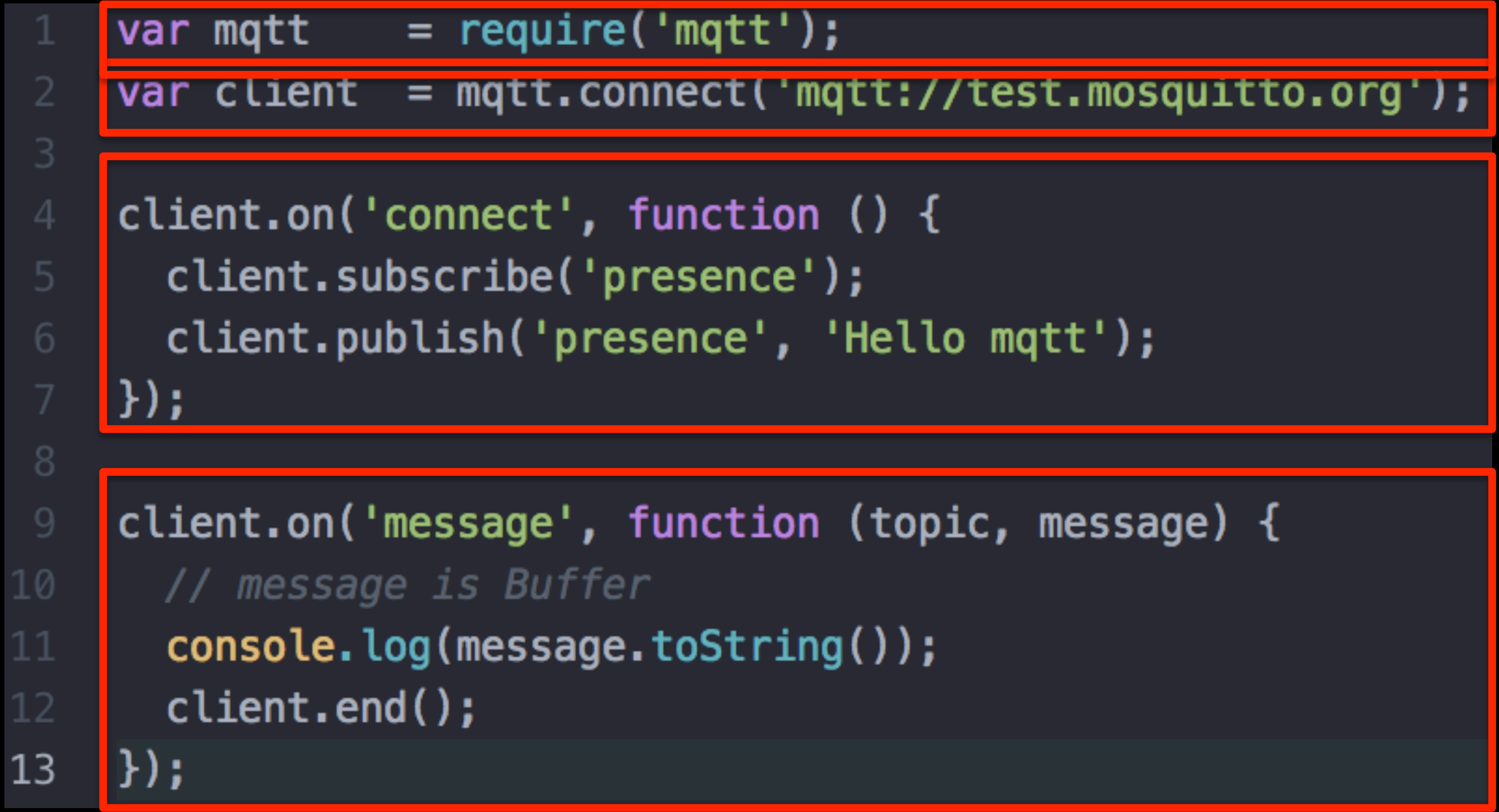
Broker

- Public MQTT Broker
  - Many Public Brokers exist.
  - One less process to run on Raspberry PI.

Client

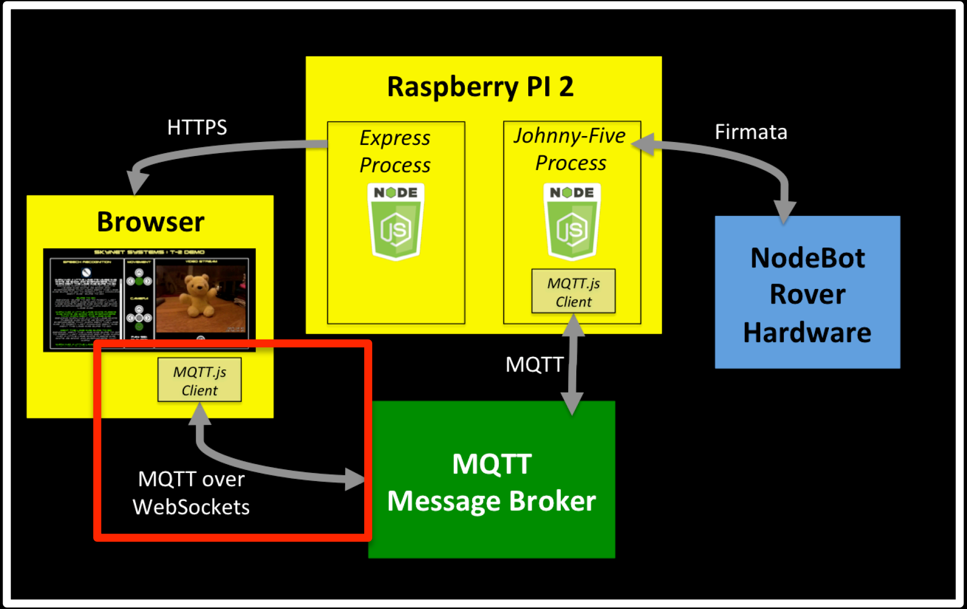
- MQTT.js
  - Provides an MQTT client library for Node.js.
  - Extremely simple to use.
  - Integrates seamlessly with Johnny-Five.

MQTT.js (Node.js) Client Example



```javascript
var mqtt    = require('mqtt');
var client  = mqtt.connect('mqtt://test.mosquitto.org');

client.on('connect', function () {
  client.subscribe('presence');
  client.publish('presence', 'Hello mqtt');
});

client.on('message', function (topic, message) {
  // message is Buffer
  console.log(message.toString());
  client.end();
});
```

MQTT Over WebSockets with MQTT.js

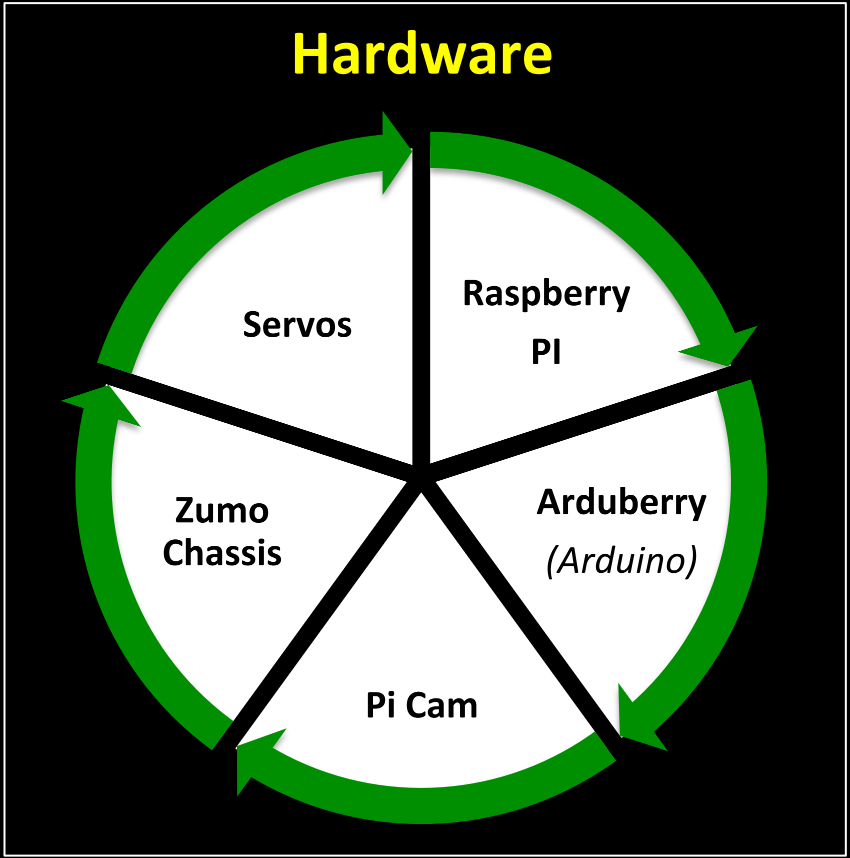


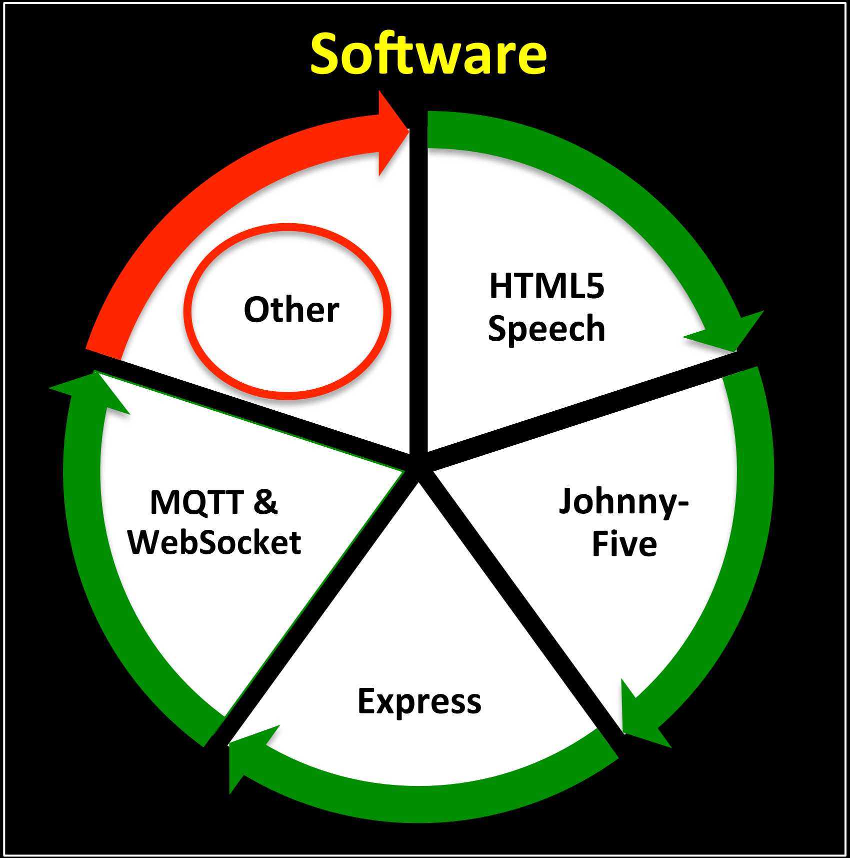
browserify
Browserify lets you require('modules') in the browser by bundling up all of your dependencies.

- MQTT.js is a Node.js package and is meant for use on the Server Side.
- To allow MQTT.js to run from the Browser, we first needed to "Browserify" the MQTT library.
- We could then access the "Browserified" MQTT client library from the Browser.
- This solution requires that your MQTT Broker has a WebSocket endpoint.

MQTT Over WebSockets Demo

NodeBot Rover Component Overview



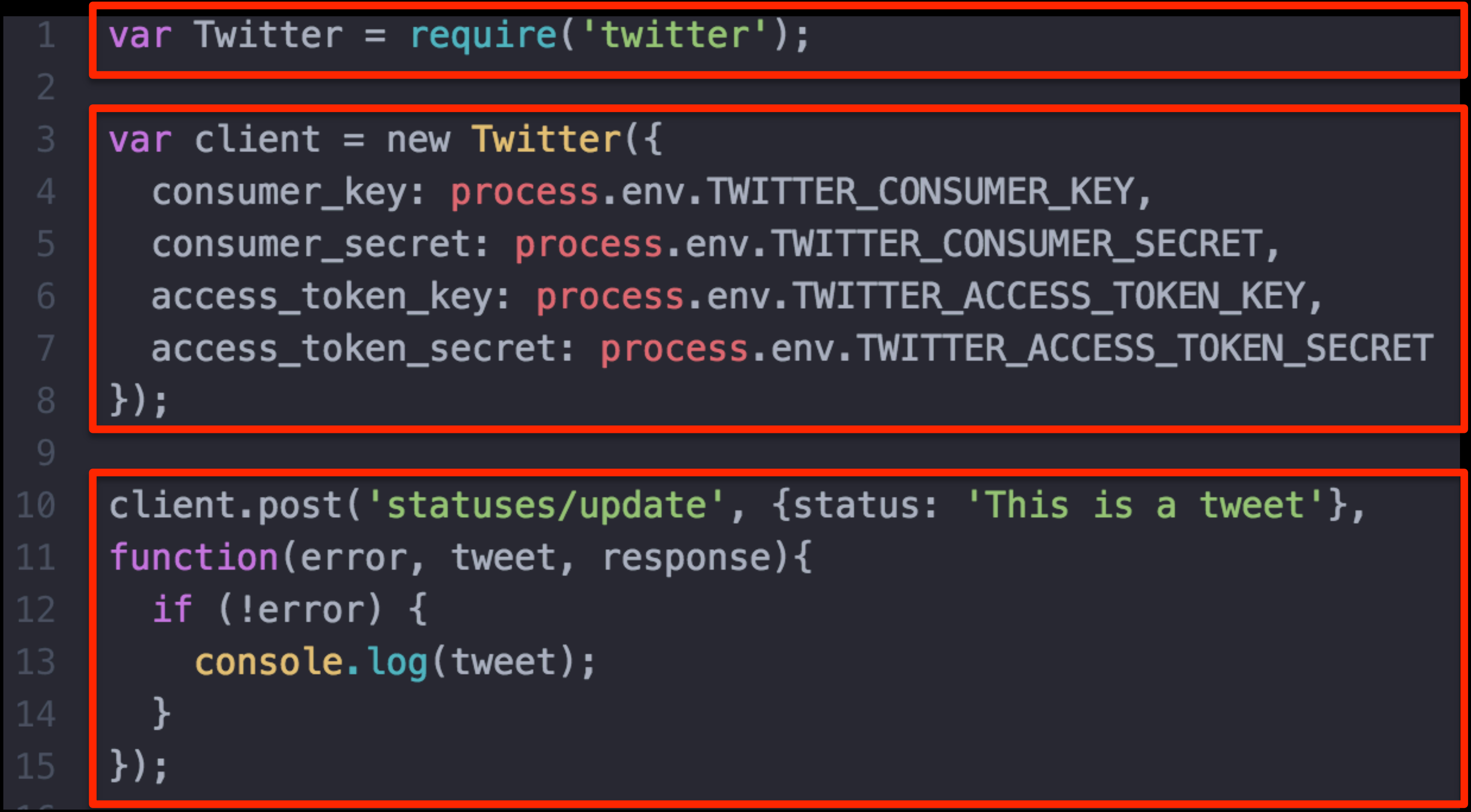


Node.js Twitter Client

- Fully fledged Twitter Client:
  - Asynchronous.
  - Supports *REST API* (write and write)
  - Supports *Streaming API* (events and tweets).

- Requires developer credentials from Twitter:
  - Trivial to get hold of.

Node.js Twitter Client REST Example



```
var Twitter = require('twitter');

var client = new Twitter({
  consumer_key: process.env.TWITTER_CONSUMER_KEY,
  consumer_secret: process.env.TWITTER_CONSUMER_SECRET,
  access_token_key: process.env.TWITTER_ACCESS_TOKEN_KEY,
  access_token_secret: process.env.TWITTER_ACCESS_TOKEN_SECRET
});

client.post('statuses/update', {status: 'This is a tweet'},
function(error, tweet, response){
  if (!error) {
    console.log(tweet);
  }
});
```

Video Streaming via Motion

- **Motion**: Software Motion Detector.

- Provides streaming video with possibility to create snapshots.

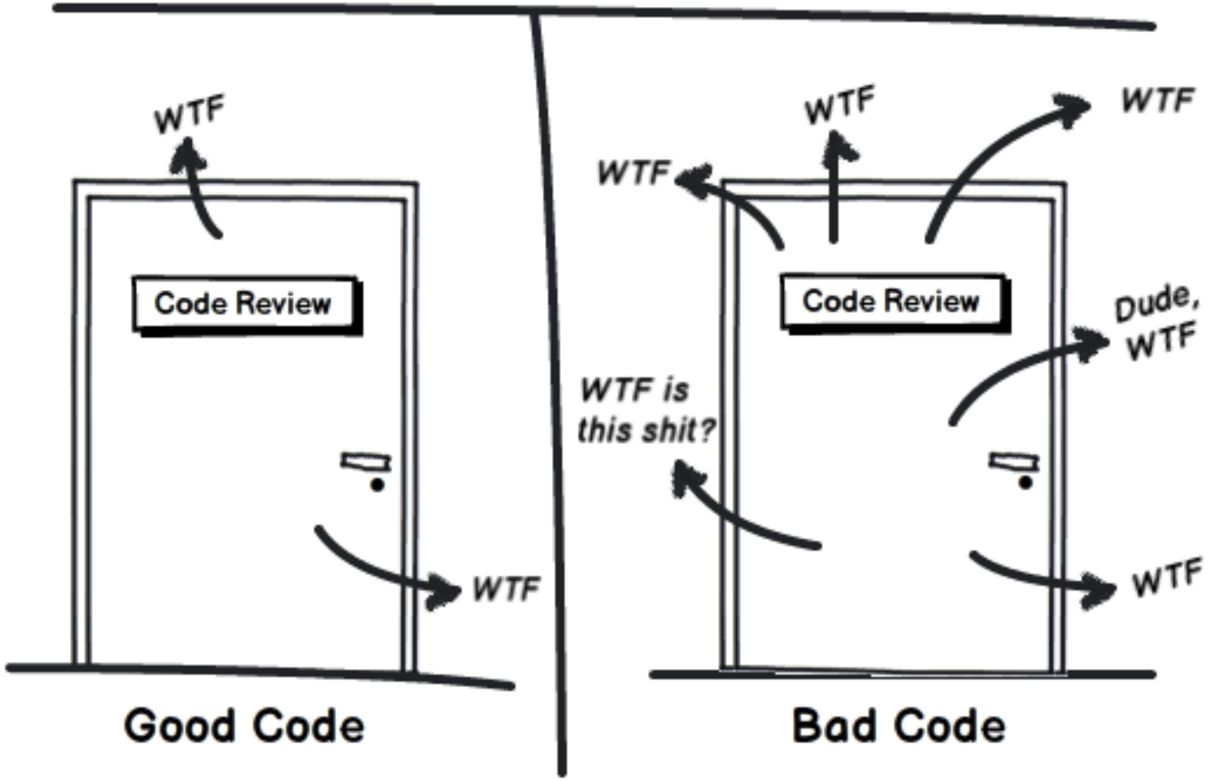
- Good performance on the Raspberry PI.

- **Potential side project**: get Robot to follow moving objects?

Other things I picked up

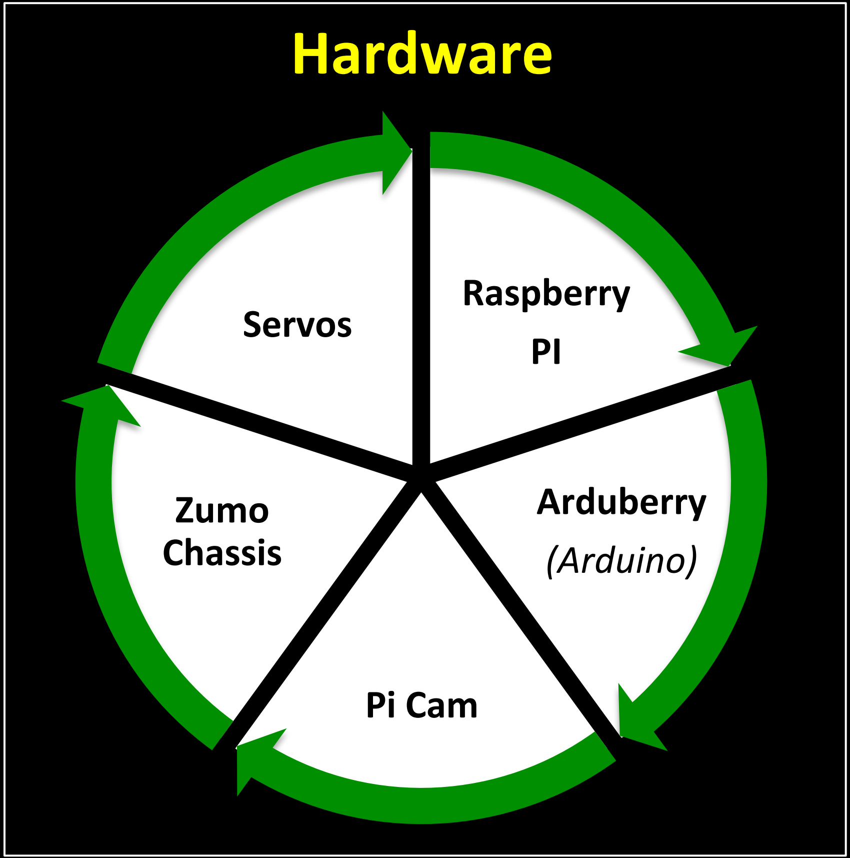
- Git / GitHub
- HTML5
- CSS
- Linux

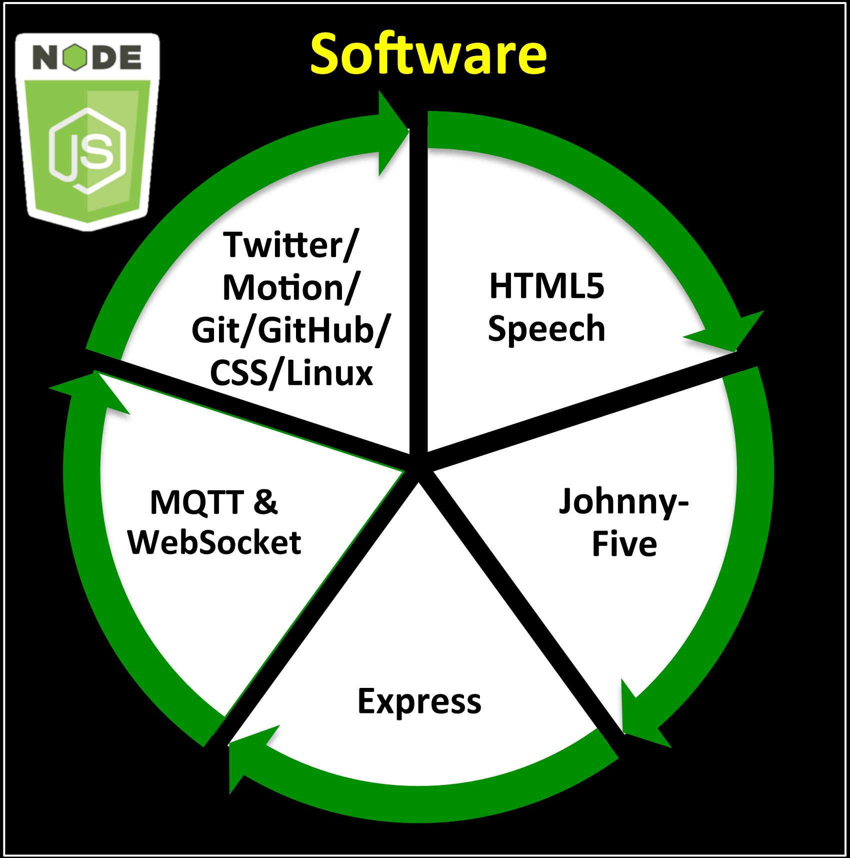
Code Quality Measurement: WTFs/Minute



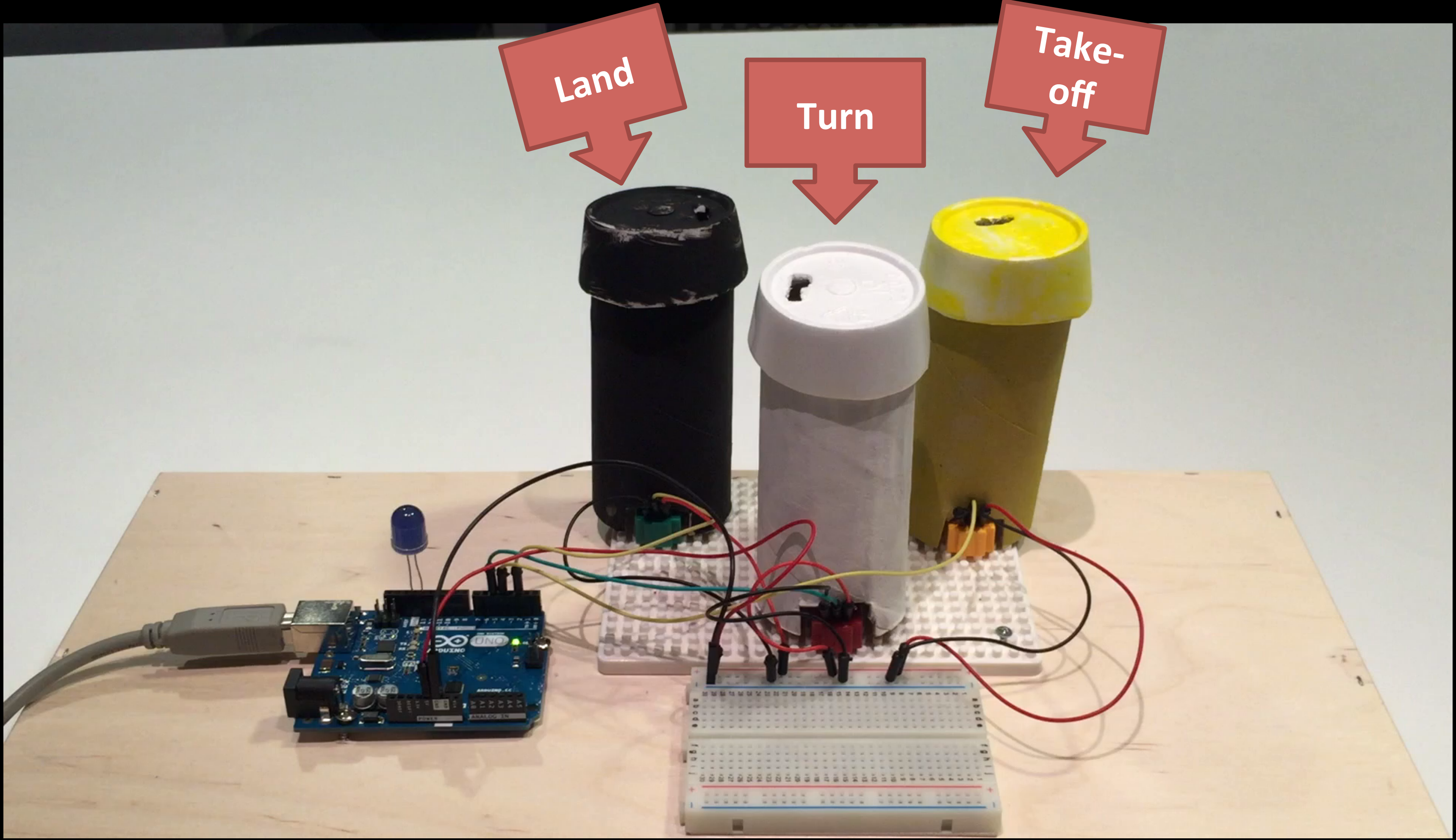
http://commadot.com

NodeBot Rover Component Overview





So Did I Get My Coding Mojo Back?


Land
Turn
Take-off

@markawest