> Why not Tapster?

> “A robot on every desk and in every home”

Jason Huggins

Selenium + Sauce Labs + Appium + Tapster Robotics, Inc.

Picture: Tapster Robotics, Inc.

> Pierre-Yves Lapersonne

> software developer

> pylapp.github.io

Picture: Freepik, Flaticon Basic License

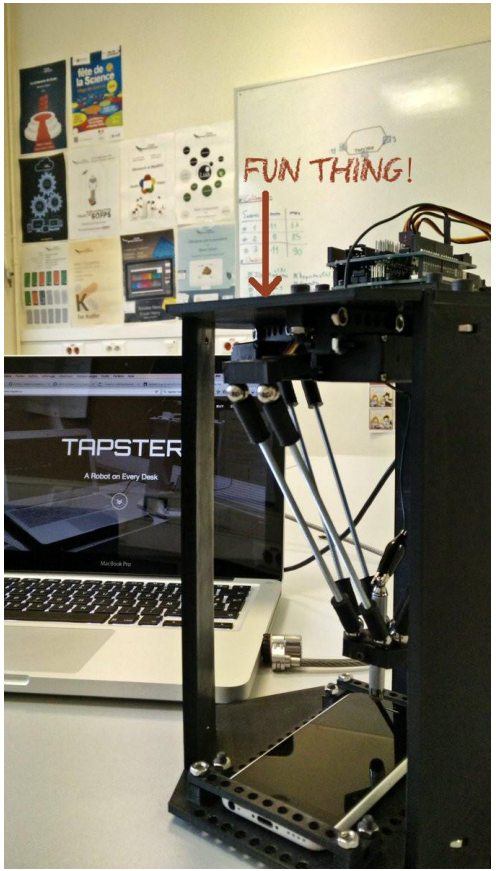
> Plan

- why a bot?
- open hardware!
- open source!
- use case #1: basic
- use case #2: automation

> Why a bot?

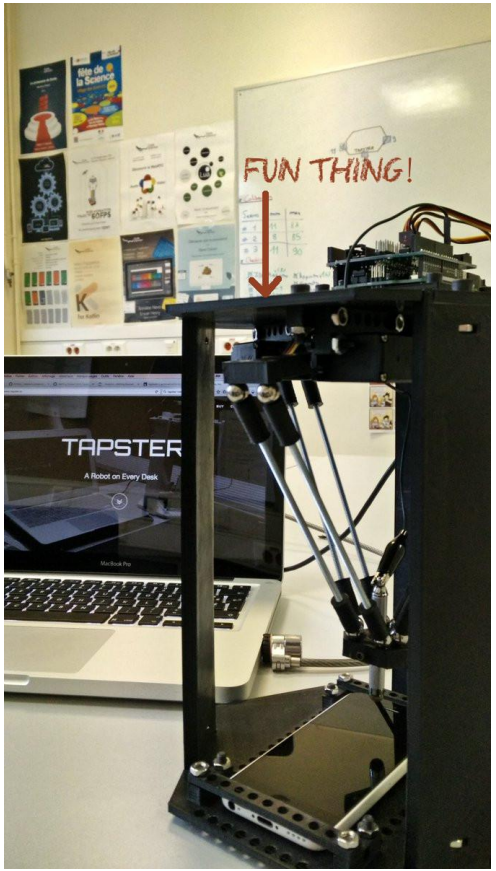
> Why a bot?

- boring tasks
  - I'm so excited to tap on screen...
- need to automate tests on devices
  - they want a finger... but not one of mine
- need to make repetitive tasks
  - I'm not a monkey
- need to repeat tasks day & night
  - bots are not yet syndicated
- need to have an agnostic tool
  - all OS and devices must be targeted
- need fun!



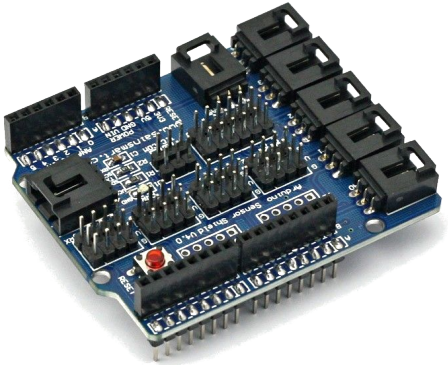
> For what?

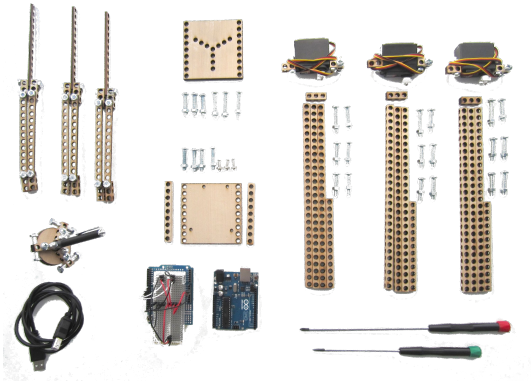
- instrumented tests
  - Appium
- stress tests
  - monkey
- “all things a human finger can do”
  - swipe, scroll, tap, press, release



> Tapster - The hardware

> Tapster::hardware





Pictures: Arduino, SainSmart, Bitbeam

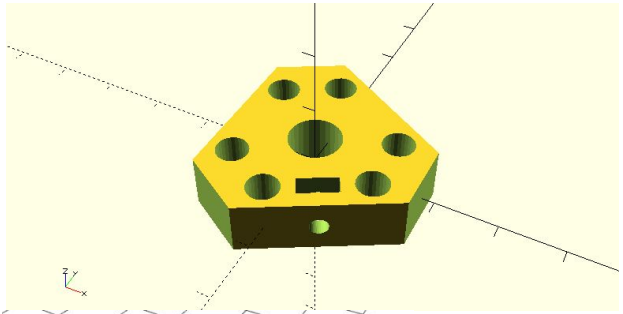
> Tapster::hardware

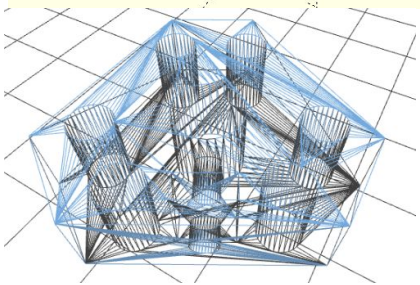
- Arduino UNO
- Sensor Shield V4
- 3D-printed components
  - made with Bitbeam
  - OpenSCAD
- Servos, arms, ties, ...
- open hardware
  - BSD License

Picture: Tapster Robotics, Inc.

> Tapster::hardware::openscad

- for creating solid 3D CAD objects
- under GPL v2 License

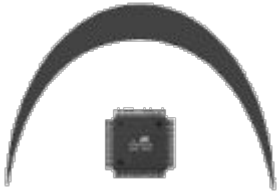


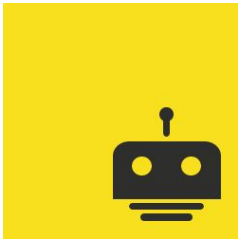


```
beam_width    = 8;
hole_diameter = 5.25;
hole_radius   = hole_diameter / 2;

clipping_distance = 26;

difference(){
  cylinder(r=56/2, h=9, $fn=3, center=true);

  translate([0,0,-10])
  cylinder(r=4.3, h=20, $fn=30);


  rotate([60,90,0])
  translate([0,0,0])
  cylinder(r=1.5, h=20, $fn=30);

  rotate(-60)
  translate([10,0,0])
  cube([3,6.5,30], center=true);

  translate([clipping_distance,0,0])
  cube([16,16,10], center=true);

  rotate(120)
  translate([clipping_distance,0,0])
  cube([16,16,10], center=true);

  rotate(240)
  translate([clipping_distance,0,0])
  cube([16,16,10], center=true);

  rotate(90)
  translate([-12,6,0])
  beam(3);


  rotate(30)
  translate([-12,-14,0])
  beam(3);

  rotate(150)
  translate([-12,-14,0])
  beam(3);

}


module beam(number_of_holes) {
    beam_length = number_of_holes * beam_width;
                for (x=[4 : 16 : beam_length]) {
                        //rotate([90,0,0])
                        translate([x,beam_width/2,-10])
                        cylinder(r=hole_radius, h=20, $fn=30);
        }
}
```

> Tapster - The software

> Tapster::software



Pictures: Firmata, Johnny-Five, NodeBots, Node.js

> Tapster::software

- Firmata protocol
  - for communicating with microcontrollers from software on a host computer
- Johnny-Five
  - JavaScript Robotics & IoT Platform release by Bocoup
- NodeBots
  - project of bots running with Node.js
  - embed Johnny-Five and make easier to work with JS on Robotics & IoT things
- Node.js
  - to run the server managing the bot and to send commands

> Tapster::software::firmata

```h
#ifndef Firmata_h
#define Firmata_h

#include "Boards.h"  /* Hardware Abstraction Layer + Wiring/Arduino */
// ...

#define FIRMATA_STRING          0x71 // same as STRING_DATA
#define SYSEX_I2C_REQUEST       0x76 // same as I2C_REQUEST
#define SYSEX_I2C_REPLY         0x77 // same as I2C_REPLY
#define SYSEX_SAMPLING_INTERVAL 0x7A // same as SAMPLING_INTERVAL
//...

namespace firmata {

class FirmataClass
{
  public:
    typedef void (*callbackFunction)(uint8_t, int);
    typedef void (*systemCallbackFunction)(void);
    typedef void (*stringCallbackFunction)(char *);
    typedef void (*sysexCallbackFunction)(uint8_t command, uint8_t argc, uint8_t *argv);

    FirmataClass();

    /* Arduino constructors */
    void begin();
    void begin(long);
    void begin(Stream &s);

    //...

    /* serial send handling */
    void sendAnalog(byte pin, int value);
    void sendDigital(byte pin, int value);
```

C++

```js
/**
 * @class The Board object represents an arduino board.
 * @param {String} port This is the serial port the arduino is connected to.
 * @param {function} function A function to be called when the arduino is ready to communicate.
 * // ...
 * @property pins An array of pin object literals.
 * @property analogPins An array of analog pins and their corresponding indexes in the pins array.
 * // ...
 * @property {SerialPort} sp The serial port object used to communicate with the arduino.
 */
function Board(port, options, callback) {
  if (typeof options === "function" || typeof options === "undefined") {
    callback = options;
    options = {};
  }
  if (!(this instanceof Board)) {
    return new Board(port, options, callback);
  }

  Emitter.call(this);

  var board = this;
  var defaults = {
    reportVersionTimeout: 5000,
    samplingInterval: 19,
    serialport: {
      baudRate: 57600,
      bufferSize: 256,
    },
  };

  var settings = Object.assign({}, defaults, options);
```

JavaScript

Sources: https://github.com/firmata/arduino/blob/master/Firmata.h https://github.com/firmata/firmata.js/blob/master/lib/firmata.js

> Tapster::software::j5

```
module.exports = {
  Accelerometer: require("./accelerometer"),
  Altimeter: require("./altimeter"),
  Barometer: require("./barometer"),
  Board: require("./board"),
  Button: require("./button"),
  Compass: require("./compass"),
  LCD: require("./lcd"),
  Led: require("./led"),
  Light: require("./light"),
  Joystick: require("./joystick"),
  Pin: require("./pin"),
  Wii: require("./wii"),
  // ...
};

module.exports.Board.Virtual = // ...
module.exports.Multi = module.exports.IMU;
module.exports.Analog = // ...
module.exports.Digital = function(opts) {
  var pin;

  if (typeof opts === "number" || typeof opts === "string") {
    pin = opts;
    opts = {
      type: "digital",
      pin: pin
    };
  } else {
    opts.type = opts.type || "digital";
  }

  return new module.exports.Sensor(opts);
};
```

JavaScript

Source: https://github.com/rwaldron/johnny-five/blob/master/lib/johnny-five.js

> Tapster::software::nodebots

```
var five = require("johnny-five");
var Controller = require("../lib/kb_controller.js");

var opts = {};
opts.port = process.argv[2] || "";

var board = new five.Board(opts);

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

    console.log("Control the bot with the arrow keys, and SPACE to stop.")

    var controller = new Controller({
        left: new five.Servo.Continuous(9),
        right: new five.Servo.Continuous(8),
        lstop: 90, // use these to set the stop value of the servo
        rstop: 90,
    });
});

board.on("error", function(err) {
    console.log(err.message);
    process.exit();
});
```

JavaScript

Source: https://github.com/nodebotsau/simplebot/blob/master/examples/servos.js

> Tapster::software::nodejs

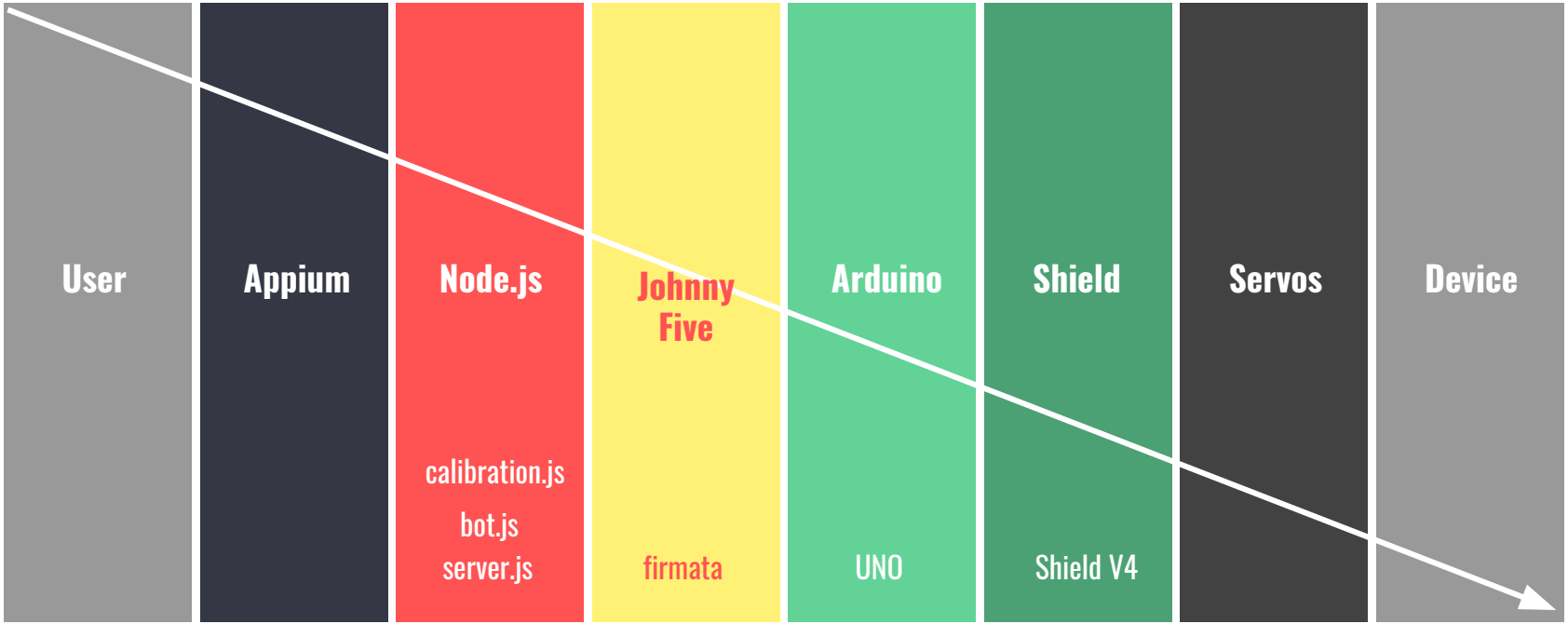
```
five = require("johnny-five");
ik = require("./ik");
board = new five.Board({ debug: false });

board.on("ready", function() {
    servo1 = five.Servo({
        pin: 9,
        range: [0,90]
    });
    servo2 = five.Servo({
        pin: 10,
        range: [0,90]
    });
    servo3 = five.Servo({
        pin: 11,
        range: [0, 90]
    });
    // ...
    var max = 15;
    var min = 5;
    servo1.to(min);
    servo2.to(min);
    servo3.to(min);
});

go = function(x, y, z) {
  reflected = reflect(x,y);
  rotated = rotate(reflected[0],reflected[1]);
  angles = ik.inverse(rotated[0], rotated[1], z);
  servo1.to((angles[1]).map( 0 , 90 , 8 , 90 ));
  servo2.to((angles[2]).map( 0 , 90 , 8 , 90 ));
  servo3.to((angles[3]).map( 0 , 90 , 8 , 90 ));
}
```

JavaScript

Source: https://github.com/hugs/tapsterbot/blob/master/software/src/bot.js

> Tapster - How to use it

> Tapster::case#1::layers



> Tapster::case#1::commands

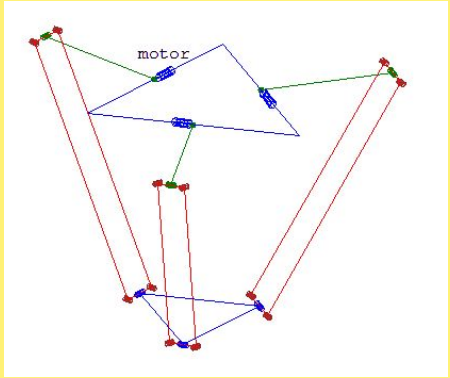
```
$ node demo/hi.js

> hi()
```

```
$ node bot.js

> draw.test()
...

> draw.drawSquare(50)
...

> draw.drawStar()
...
```

```
$ node bot.js

> go(0, 0, -140)
...

> svg.drawSVG(“example/tutorial.svg”)
...

> svg.drawSVG(“hello/helloFrF.svg”)
...
```

> Tapster - Make more automation!

> Tapster::case#2::automation::why-calibration

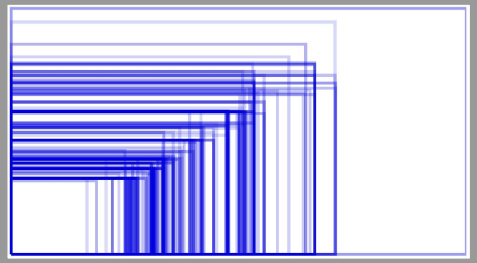


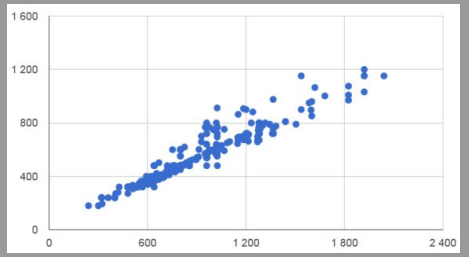
3D coordinates

(X, Y, Z)

angles for servomotors of delta-robot

(α1, α2, α3)





2D coordinates for screens of handsets, with numerous sizes

(x , y)

Credits: OpenSignal, Wikipedia, Samsung, Apple

> Tapster::case#2::automation::calibrating-iOS

 # terminal (Appium side)

$ appium --pre-launch -U my_device_udid --app Appium.RobotCalibration
--platform-name iOS --platform-version my_ios_version &
> ...

 # other terminal (Tapster side)

$ node server.js &
> ...
$ node calibrate.js -o my_calibration_file.json
> ...

Source: https://medium.com/devs-foodit/iphone-automation-with-a-one-fingered-robot-a2936c840285

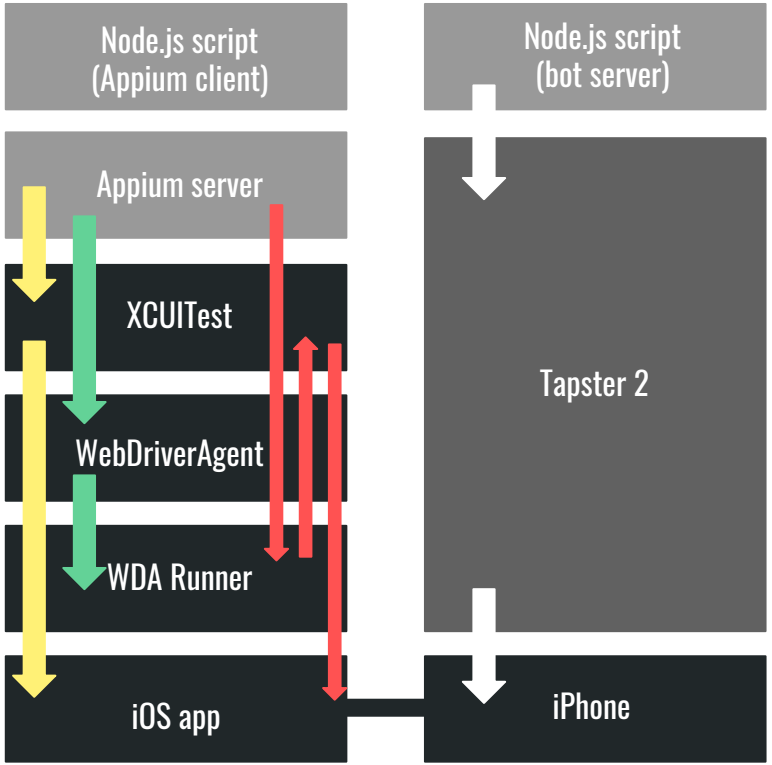
Credits: Appium, Node.js, Apple

> Tapster::case#2::automation::calibrating-iOS

1 - download app

2 - download runner

3 - process commands



> Tapster::case#2::automation::calibrating-Android

 # terminal (Appium side)

$ appium --pre-launch --default-capabilities ‘{“udid”: “XXX”, “app”: “YYY”, “platformName”: “Android”, “deviceName”: “ZZZ”, “platformVersion”: “OOO”, “autoLaunch”: “true”}’
> ...

 # other terminal (Tapster side)

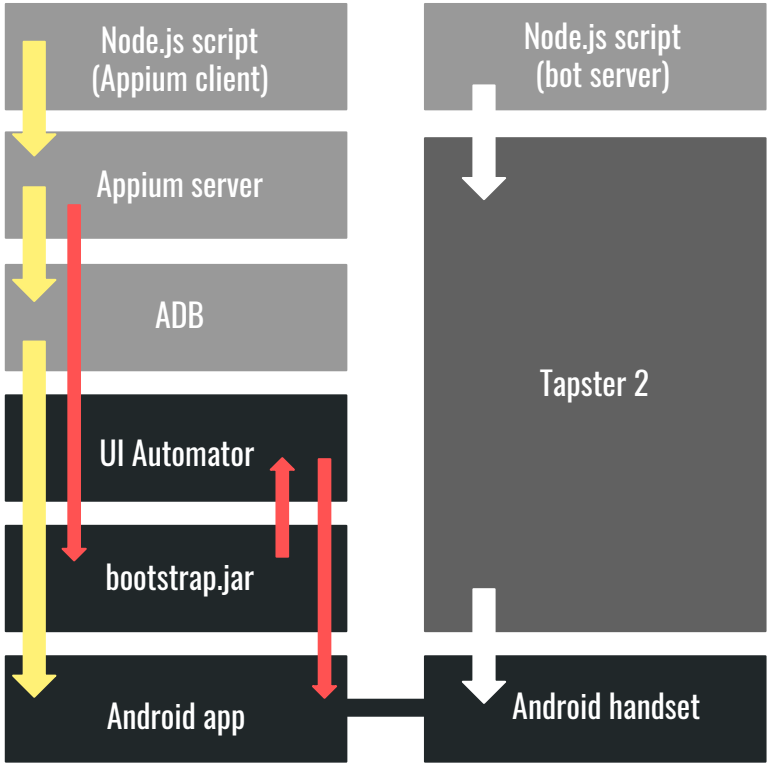
$ node server.js &
> ...
$ node calibrate.js -o my_calibration_file.json
> ...

Credits: Appium, Node.js, Google

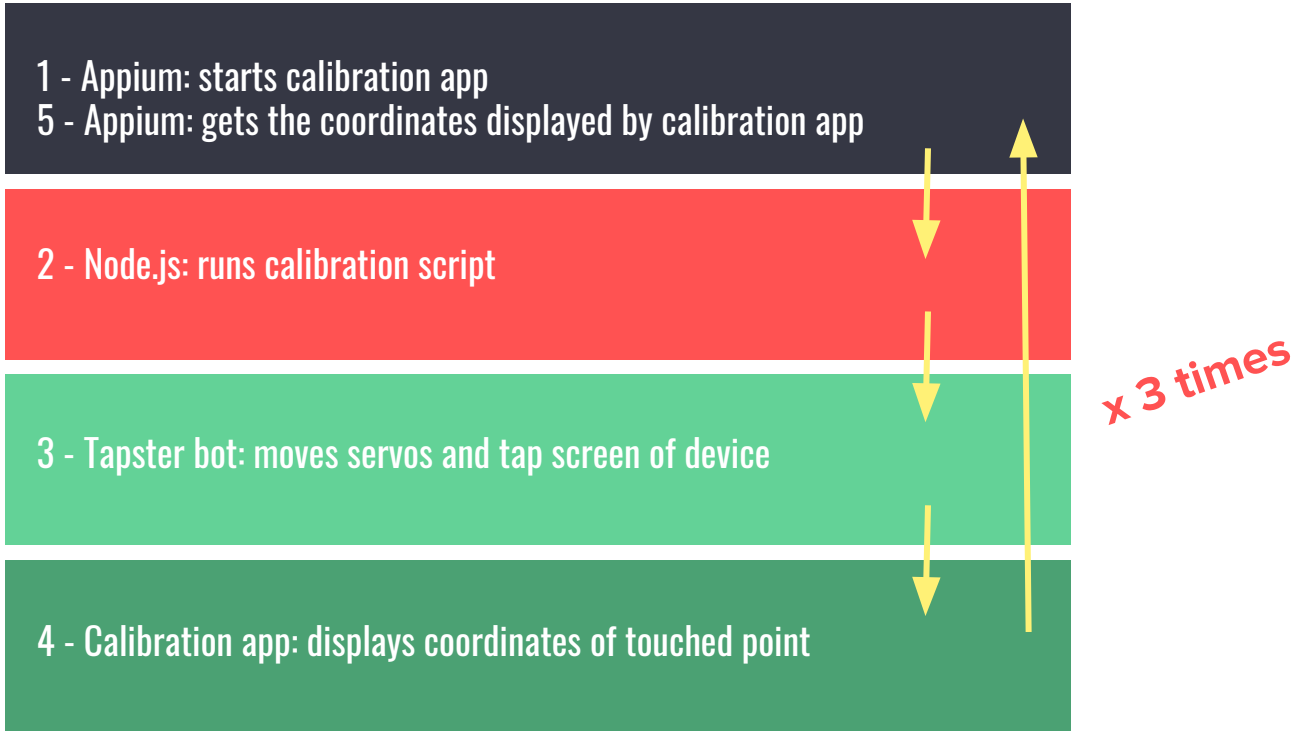
> Tapster::case#2::automation::calibrating-Android

1 - download app

2 - process commands



> Tapster::case#2::automation::calibration-workflow



> Tapster::case#2::automation::calibrated

 # terminal (Tapster side)

$ node server.js -c my_calibration_file.json
> ...

 # other terminal (Appium side)

$ appium --robot-address 127.0.0.1 --robot-port:4242
> ...

Source: https://medium.com/devs-foodit/iphone-automation-with-a-one-fingered-robot-a2936c840285 Credits: Appium, Node.js, Tapster

> Conclusion

> Conclusion

- Tapster bot can be built from scratch
- Tapster bot can be homemade
- The hardware & the software are pleasant & elegant
- The project is open source & open hardware

- Don’t be afraid of having to look deeper in the code
- Bugs might appear between Tapster and Appium
- The team & community behind Tapster need support

- Bots are here to automate boring tasks… so move on!

Picture: Freepik, Flaticon Basic License

> Resources

> Here be people & orgs.

- Appium
  http://appium.io/
  https://twitter.com/appiumdevs
  https://github.com/appium

- Dan Cuellar
  https://twitter.com/thedancuellar
  https://github.com/penguinho

- Jason Huggins
  https://twitter.com/hugs
  https://github.com/hugs

- Tapster Robotics, Inc.
  http://www.tapster.io/
  https://twitter.com/tapsterbot
  https://github.com/tapsterbot

- Bitbeam
  https://bitbeam.org/
  https://twitter.com/bitbeam

- Sauce Labs
  https://saucelabs.com/
  https://twitter.com/saucelabs
  https://github.com/saucelabs

- Selenium
  http://www.seleniumhq.org/
  https://twitter.com/seleniumhq?lang=fr
  https://github.com/SeleniumHQ

- Taspter bot
  https://github.com/hugs/tapsterbot
  https://www.tindie.com/products/hugs/tapster/

> Here be videos & repos

- Video: 2016: A Robot Odyssey
  https://www.youtube.com/watch?v=gt5Kj2xNKcA
- Video: Dancing Deltabot! Breakdance or Jig?
  https://www.youtube.com/watch?v=lbjilf5cz88
- Video: Deltabots!
  https://www.youtube.com/watch?v=upE2eo7pg9k
- Video: Ohai, Tapster!
  https://www.youtube.com/watch?v=in2av9LtCfE
- Video: Tapster 2 - Gesture Demo
  https://www.youtube.com/watch?v=JQ-_l8UrPgM
- Video: Tapster Calibration
  https://www.youtube.com/watch?v=d4YhHkPKidE
- Video: Tapster Duet
  https://www.youtube.com/watch?v=9F-Vb9EhdIc
- Video: Tapster plays Angry Birds
  https://www.youtube.com/watch?v=mZjD1B5rI4Q
- Video: Tapster Sidekick - Controller Demo
  https://www.youtube.com/watch?v=MAu_O9O79Sc
- Video: Tic tac toe agains Tapster robot
  https://www.youtube.com/watch?v=GoPzooYah3U

- GitHub: appium/robots
  https://github.com/appium/robots
- GitHub: firmata/protocol
  https://github.com/firmata/protocol
- GitHub: Getting Started With Tapster
  https://github.com/hugs/tapsterbot/wiki/Getting-Started-With-Tapster
- GitHubGist: taspter-appium-calibration.txt
  https://gist.github.com/hugs/6132879
- GitHub: tapsterbot/hardware
  https://github.com/hugs/tapsterbot/tree/master/hardware
- GitHub: tapsterbot/software
  https://github.com/hugs/tapsterbot/tree/master/software
- GitHub: tapsterbot/software/src/demo/angry-birds
  https://github.com/hugs/tapsterbot/tree/master/software/src/demo/angry-birds
- GitHub: tapsterbot/taspterbot
  https://github.com/tapsterbot/tapsterbot
- GitHub: Tapster Bill of Material
  https://github.com/hugs/tapsterbot/blob/master/hardware/tapster-2/BOM.md
- GitHub: forks of Tapster's project repo with demos
  https://github.com/jackskalitzky/tapsterbot
  https://github.com/penguinho/tapsterbot

> Here be resources

- Arduino - Arduino Board Uno
  https://www.arduino.cc/en/Main/ArduinoBoardUno
- Bitbeam - About
  https://bitbeam.org/about/
- Bitbeam - A Robot on Every Desk
  https://bitbeam.org/2013/05/02/a-robot-on-every-desk/
- Delta robot kinematic
  http://forums.trossenrobotics.com/tutorials/introduction-129/delta-robot-kinematics-3276/
- Firmata Library
  https://www.arduino.cc/en/Reference/Firmata
- FOODit - A Robot Should Be Running Your Appium Tests
  https://medium.com/devs-foodit/iphone-automation-with-a-one-fingered-robot-a2936c840285
- Robot with long finger wants to touch your iPhone apps
  https://www.wired.com/2013/08/tapster/
- SainSmart Sensor Shield V4
  https://www.sainsmart.com/sainsmart-sensor-shield-v4-module-for-arduino-duemilanove-uno-mega2560-atmel.html
- This Startup Taught a Robot to Play Angry Birds. Now It's testing Touchscreens for Car Makers
  http://chicagoinno.streetwise.co/2016/03/22/software-testing-robot-tapster-taps-touchscreens-in-cars/

> Here be websites

- Appium
  http://appium.io/
- Arduino
  https://www.arduino.cc/
- Bitbeam
  https://bitbeam.org/
- Bocoup
  https://bocoup.com/
- Firmata
  http://firmata.org/wiki/Main_Page
- Johnny-Five
  http://johnny-five.io/
- OpenSCAD
  http://www.openscad.org/
- Node.js
  https://nodejs.org/en/
- NodeBots.io
  http://nodebots.io/
- SainSmart
  https://www.sainsmart.com/

> Here be dragons

Picture: Tapster Robotics, Inc.