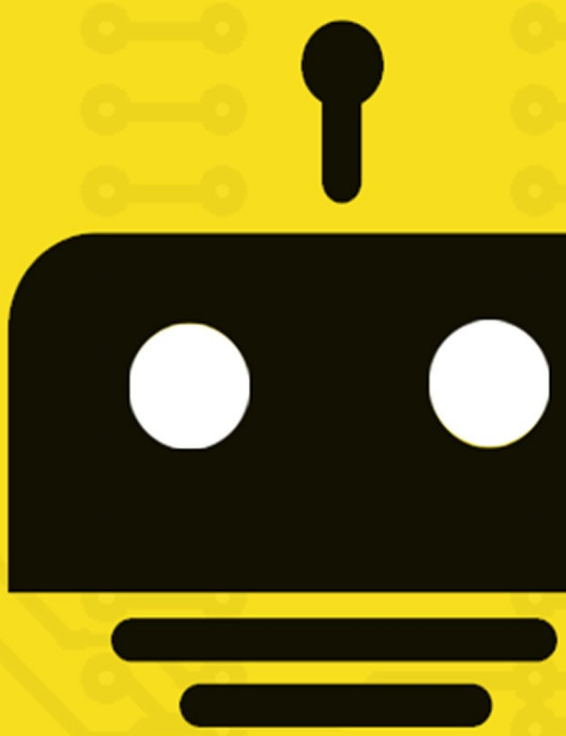
International

NodeBots Day

July 25 / Charleston, SC



A one day gathering for people to learn, create, and discover NodeBots

nodebotschs.github.io

Table of Contents

1. Introduction
2. Robots are fun?
3. Schedule of Activities
4. Particle Core Development Setup
    i. Install Node.js and Particle CLI
    ii. Particle Core Setup
5. Sumobot Instructions
    i. Hello World
    ii. Assembly
    iii. Connecting the brain (Particle Core)
6. Coding Up Your New Robot!
    i. Johnny Five is Alive
    ii. Running a script
    iii. Code Samples
7. Resources/Links for Making Robots
8. What Now? What to do when Robot comes home.
9. Special Thanks and Sponsors

Welcome to International NodeBots Day 2015

A one day gathering for people to learn, create, and discover NodeBots.

Come experience a full day of assembling, coding, and challenging your fellow attendees to a friendly battle or race of robots.

Mentors

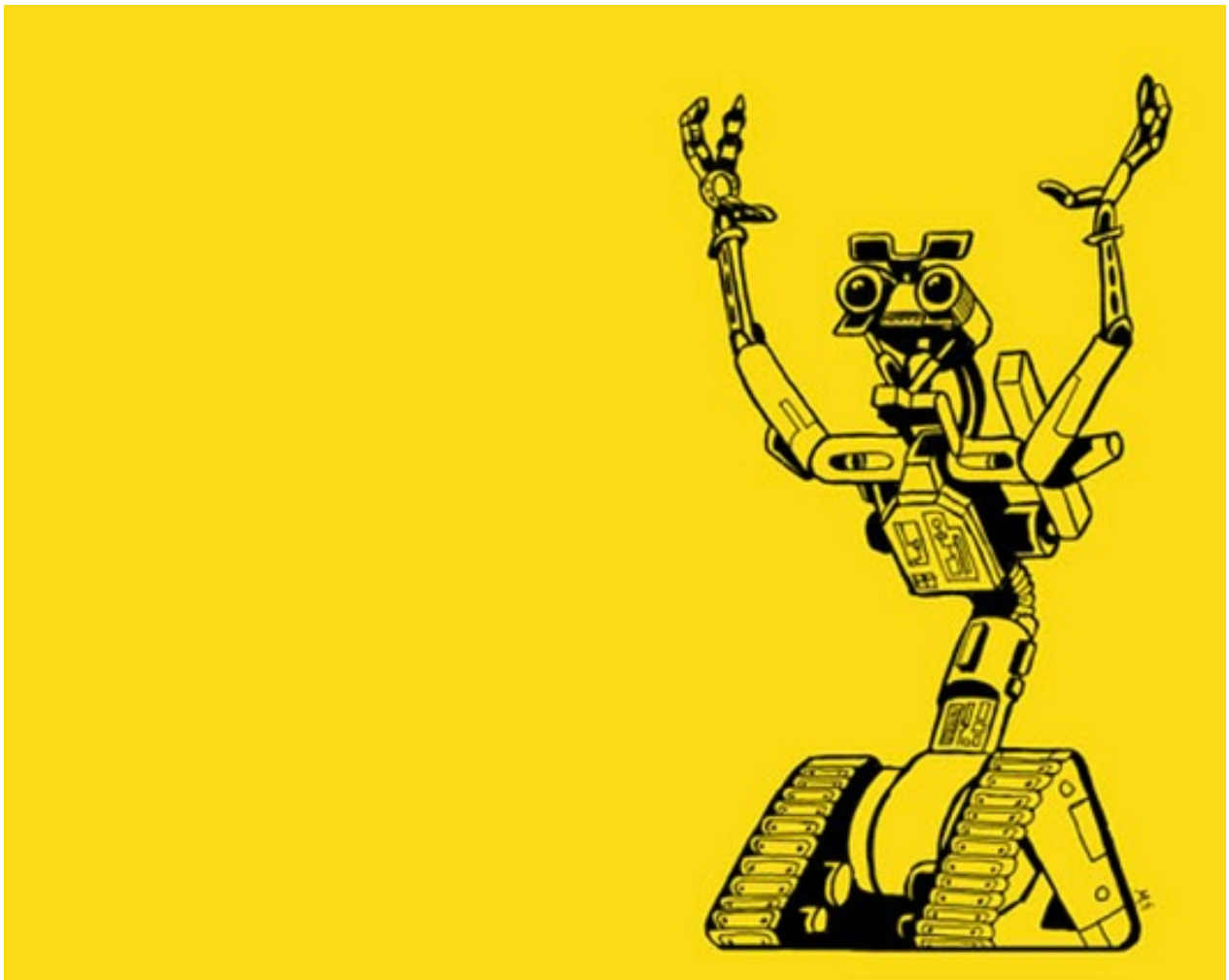
- Calvin Webster
- Amanda Hodges
- Jeremy Martin
- Mark Funk
- Jason Frockenstein

Robots are fun?

Now, more than ever, making robots has become super accessible! With a little bit of code, some electronics and materials, we can make little robots that obey our commands, help us in the house or delight us with thier behavior!

Thanks for coming to International NodeBots Day!

> What is this? He is delicate equipment! A completely unique, self-mobile, micro-computer robotics system! -- Ben Jahrvi



Schedule of Activities

Schedule of Activities

- 8:30 AM - 9:00 AM - Check In
- 9:00 AM - 9:10 AM - Introduction, Logistics and Sponsors
- 9:10 AM - 10:00 AM - NodeBots Workshop 101
  - What is in your kit
  - Setup and Claim your particle Core
  - Flash your core with voodoospark
  - Blink a LED with Johnny Five
- 10:00 AM - 12:00 AM - Build your SumoBot Jr
- 12:00 AM - 1:00 PM - Box Lunches
- 1:00 PM - 3:30 PM - Hack your Sumo Bot JR
- 3:30 PM - 4:00 PM - NodeBot ART Demos
- 4:00 PM - 4:30 PM - NodeBot Exhibition Demos
- 4:30 PM - 5:00 PM - NodeBot BattleBot Tournament

Particle Core Development Setup

- Install Node.js and Particle CLI
- Particle Core Setup

Install Node.js and Particle CLI

Node.js is an incredible platform that allows you to build things in javascript, which we will need in order to download and use the Particle CLI (Command Line Interface) which is the way you will be interacting with your hardware (Particle Core) to program your robot.

Depending on the operating system, you will need to follow specific directions, but generally you will:

1. Install NodeJS (Click Install)

   Once Node.js is installed, you will also have available the Node Package Manager which is how you will install libraries and utilities that help you build and interact with your hardware/software.

2. Then install the Particle CLI

The Particle Command Line Interface is a tool written by the folks who also made your device. This tool allows you to connect, claim, and use your device, so you'll need to install it:

```
git clone git@github.com:spark/particle-cli.git
cd particle-cli
npm i . -g
```

1. To make sure that the installation worked, go ahead and try to login to your particle account by typing `particle login` (this will prompt you for your username and password.) This will actually create an account on particle.io which will allow you to manage your devices. see Particle Setup

For Windows Users

> For windows users you may want to follow these instructions:
> https://community.particle.io/t/tutorial-particle-cli-on-windows-07-jun-2015/3112

For Ubuntu Users

> For ubuntu users you may want to follow these instructions: https://community.particle.io/t/how-to-install-the-spark-toolchain-in-ubuntu-14-04/4139

Particle Core Setup

> We want to setup the particle core via usb.

1. Plug in your usb cable to your computer and the particle core.
2. Run particle-cli setup

```
particle setup
```

1. Enter your email address and password, this will create an account on particle.io.
2. Detect your wifi, you should have a handout that contains the wifi info, use it to send your particle core the wifi information.

You should see the core light go from blinking blue to breathing cyan:

http://docs.particle.io/core/connect/

If you get any errors please let one of the mentors know.

1. Flash the core

> Once your particle core is connected via wifi, we need to flash the core with the voodoospark cpp file

1. Checkout voodoospark

```
git clone  https://github.com/voodootikigod/voodoospark.git
cd voodoospark
particle flash --usb firmware/voodoospark.cpp
```

1. Get your Device ID and Access Token

> To get your device id simply Run

```
particle list
```

> To get your access token

In you home directory there should be a folder called `.particle` and in that folder a file called `particle.config.json` - in that file should be your access token.

Sumobot Instructions

- Hello World
- Assembly
- Connecting the brain (Particle Core)

Hello World

The hello world application for NodeBots is the customary blink of a LED.

Requirements

1. NodeJS
2. Particle Core Device Id
3. Particle Core User Access Token

Getting started

On your particle core bread board take a led red or green and place the long end into the `D7` and the short end into the `GND` slot of the breadboard.

Then create a folder on your computer called `jf-helloworld` and cd into that folder.

Enter the following commands:

```
# init as an npm project - default through the prompts
npm init -y
npm install johnny-five --save
npm install spark-io --save
```

Create a file called `index.js`

```
var Spark = require('spark-io')
var five = require('johnny-five')

var board = new five.Board({
  io: new Spark({
    token: process.env.SPARK_TOKEN,
    deviceId: process.env.SPARK_DEVICE_ID
  })
})

board.on('ready', function() {
  var led = new five.Led("D7")
```

```
    led.blink()
  })
```

Finally setup your deviceId and access token as `env` variables:

```
export SPARK_TOKEN=[access token]
export SPARK_DEVICE_ID=[device id]
```

Now you are ready to run you app:

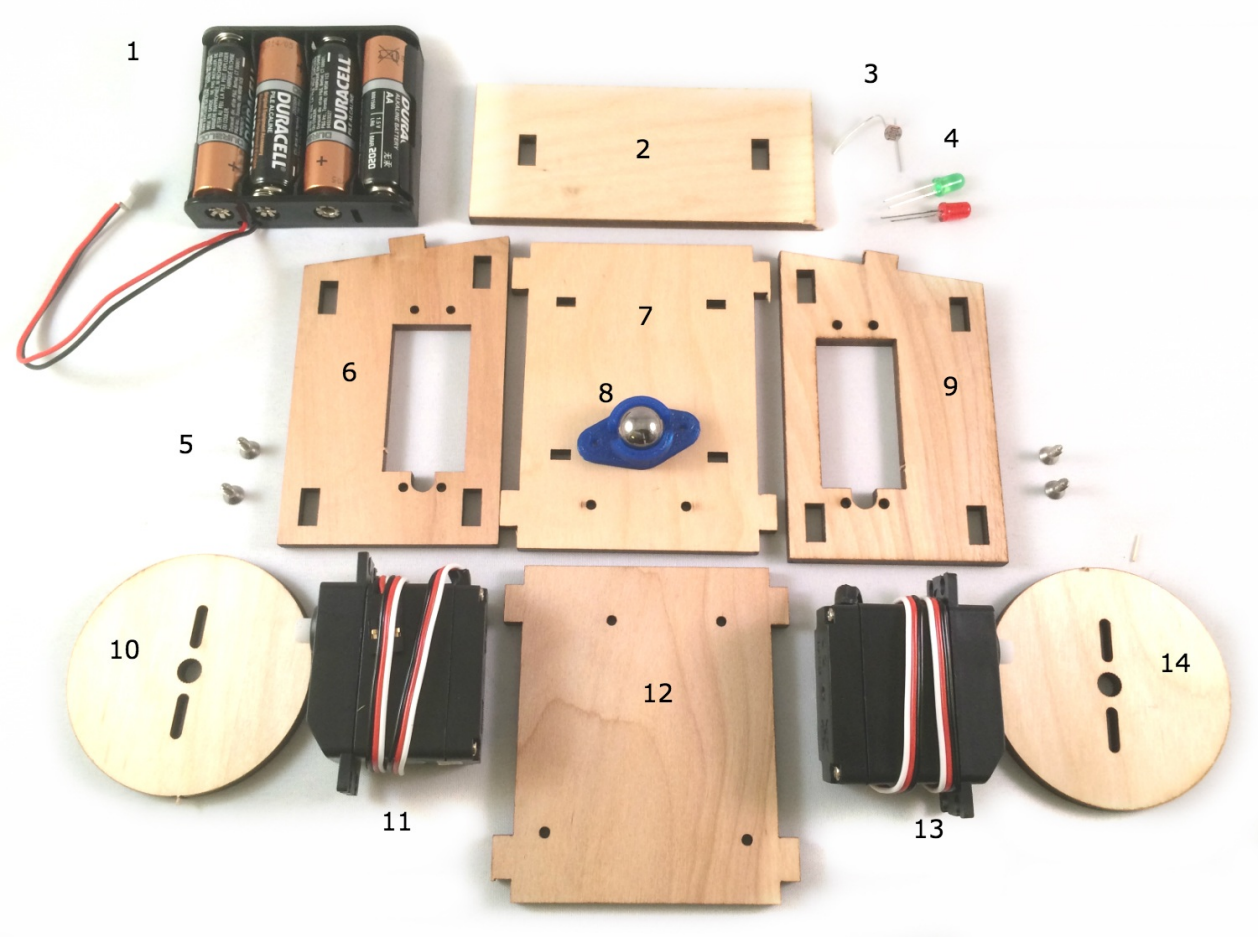
```
node index.js
```

!!! Congrats you should see a LED Blink !!!

[ctrl-c] will close the app.

Assembly

Inventory



1. Battery box with 4 AA batteries. A single AA battery provides a 1.5V source, delivered at 700mA; when connected in series, 4 AA batteries provide 6V, delivered at 700mA. All three of the boards can operate with 6V; the Electric Imp and Spark Core will regulate to 3.3V and the Arduino Uno will regulate to 5V. The servos included in the kit operate at 6V.
2. Front panel
3. Photoresistor, or light sensor. This can be sourced from the ARDX Kit and will be used in a later customization of the bot.
4. LEDs: one green and one red. These are sourced from the ARDX Kit and will be used in a later customization of the bot.
5. Screws; the kit provides 6 and the build will use 2 on each side to secure the servo to the body. The remaining 2 will be used in the Arduino Uno build to fasten the

board to the solderless breadboard for that build.
6. Left panel
7. Bottom panel
8. Ball caster; used in this build as the “back wheel” to support the rear of the bot.
9. Right panel
10. Left wheel
11. Left servo
12. Top panel
13. Right servo
14. Right wheel

Preparation

Before proceeding to the assembly step, there are three major preparatory tasks that must be completed. Consider the project blocked until these tasks are complete.

1. Start a project directory: `mkdir <directory name>`

2. Install Node.js modules from npm.

```
npm init (complete the initialization)
npm install johnny-five
npm install spark-io, npm install -g particle-cli
```

1. Setup and connection of controller board. Particle Core (requires Internet connected Wi-Fi)

    - `particle setup` : this process will set up an account and claim the Particle Core. (More commands and information available here: particle-cli)

    - `particle flash REPLACE_WITH_DEVICE_ID voodoo` : this will flash the VoodooSpark firmware to the Spark Core with the provided device id. The built-in LED will flash magenta during the upload.

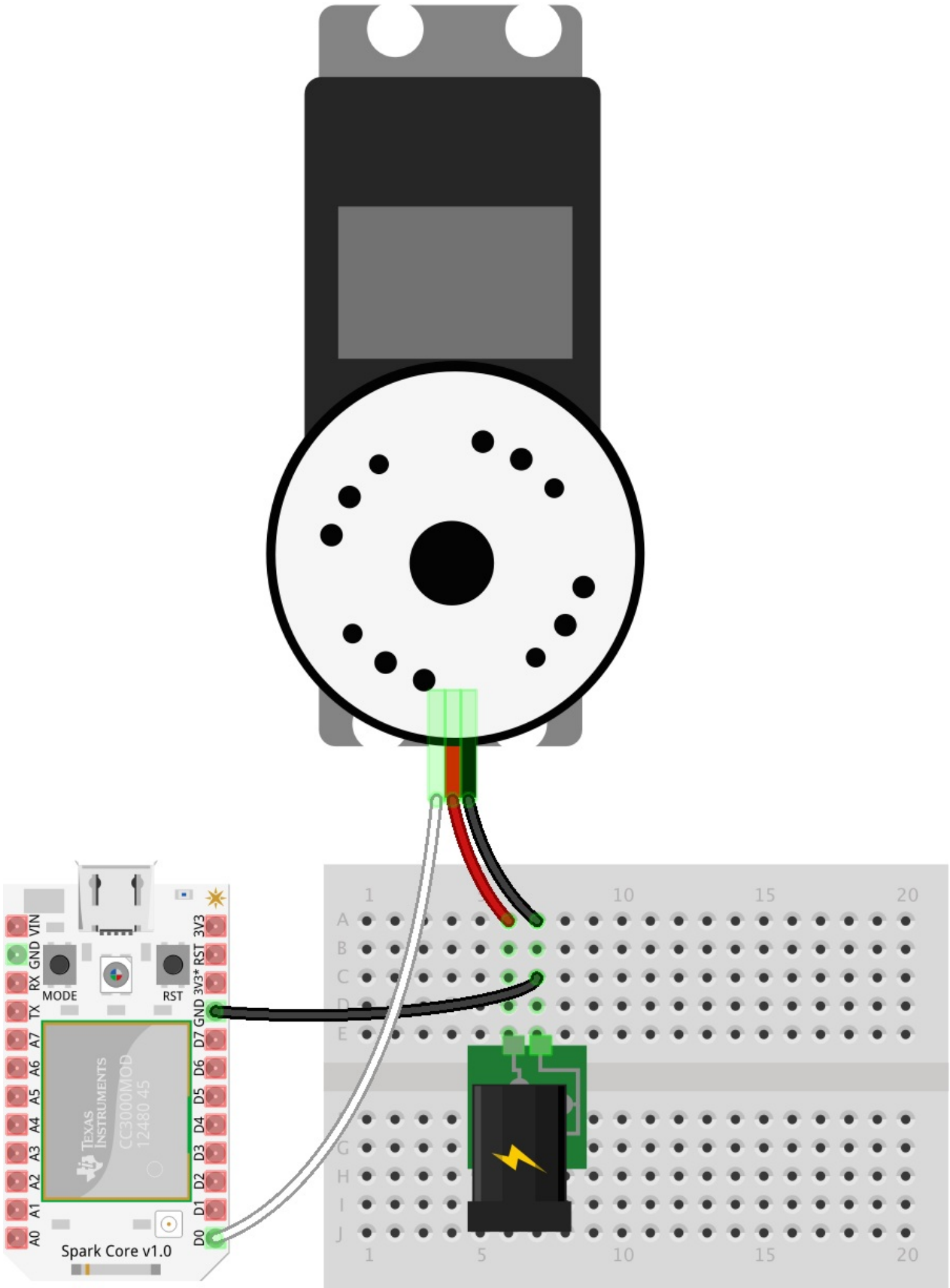
    - Once flashing is complete (as indicated by the built-in LED pulsing cyan), this terminal can closed.

2. Calibrate the Continuous Servos.

Missing or skipping this step is a direct path to failure.

When a continuous servo's idle pulse is not calibrated to the available PWM range of the controller board, its behavior will be erratic and uncontrollable. Calibration is simple: send the known idle pulse to the servo and adjust the built-in potentiometer until the servo horn comes to a complete stop. If the servo is not moving when the idle pulse is written to the servo, then it's safe to assume that the servo is already calibrated. To confirm, adjust the built-in potentiometer in clock-wise and counter clock-wise directions; the horn should respond by turning in the corresponding direction.

To calibrate the servos that came with the SumoBot kit, create a new file called calibrate.js containing the program for the controller board you're setting up below. The purpose of the program is to connect to the board, initialize a continuous servo instance with the Servo.Continuous class and immediately call the instance's stop method (which will send the idle pulse to the servo)




```
// calibrate.js
var five = require("johnny-five");
var Spark = require("spark-io");
```

```
var board = new five.Board({
  io: new Spark({
    token: "token",
    deviceId: "device id"
  })
});
board.on("ready", function() {
  new five.Servo.Continuous("D0").stop();
});
```

Run the program with the following:

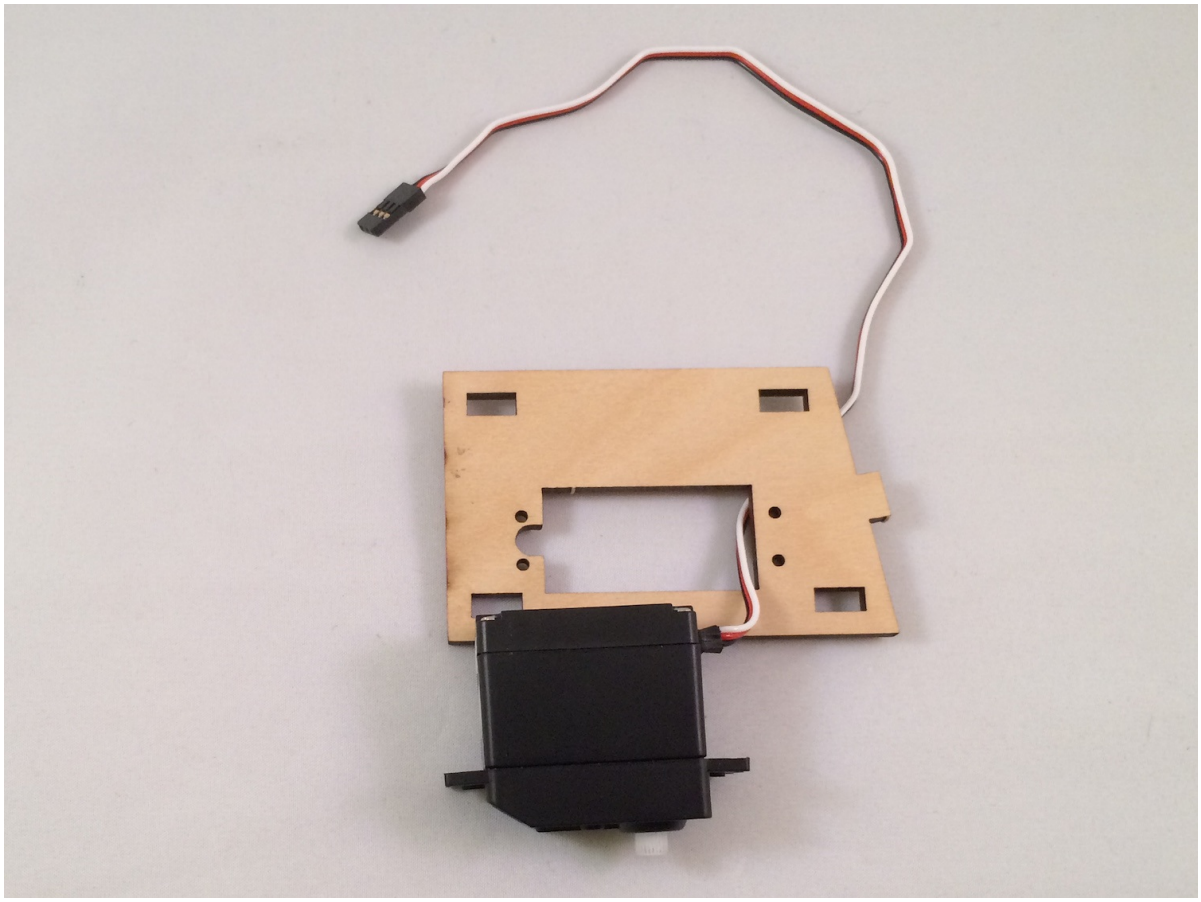
```
node calibrate.js
```

While running, adjust the servo's potentiometer until it comes to a complete stop. This video shows the operation in action:

Assembly

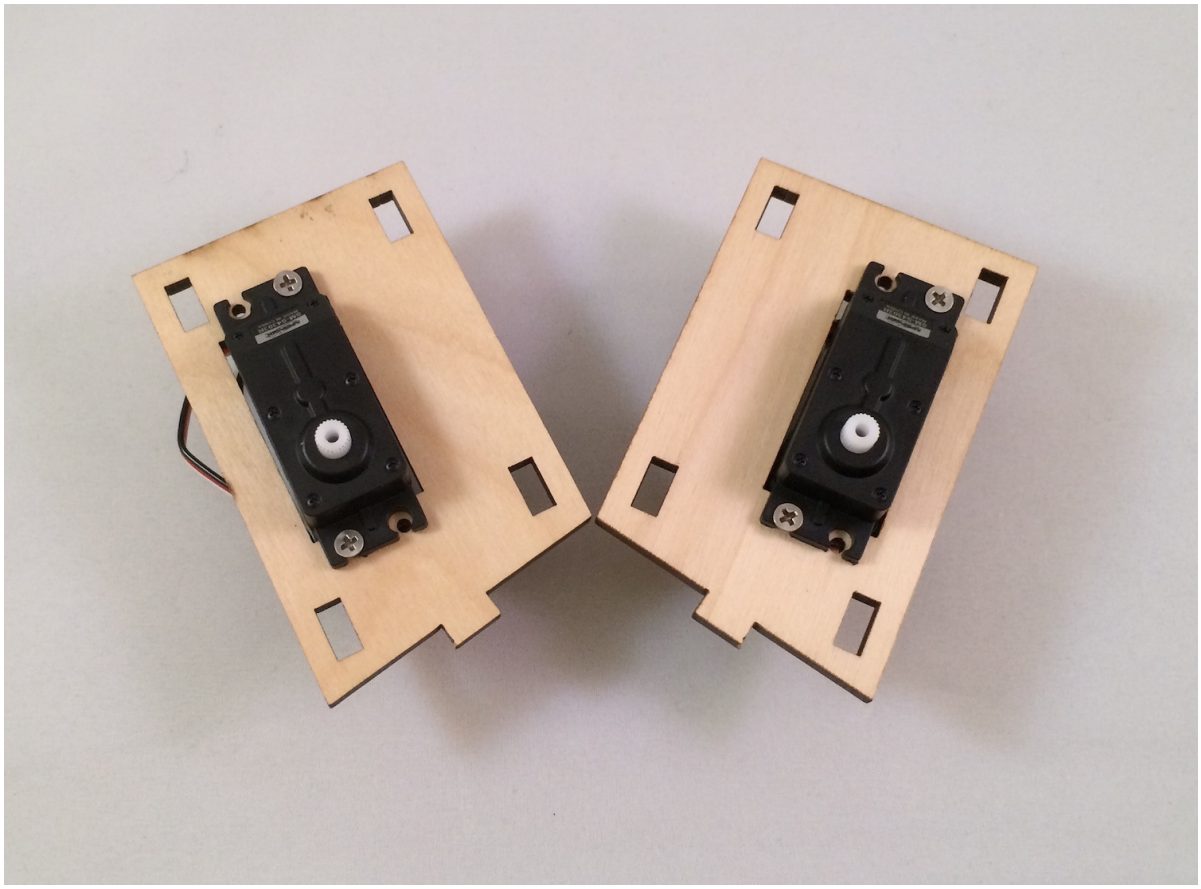
The SumoBot Kit includes an insert that provides a numbered and illustrated assembly guide; while the guide is adequate, this build is different enough to warrant its own set of instructions.

1. For each side panel, line up a servo so that the horn is closest to the front of that panel. The end of the servo with the wiring harness must be opposite of the cut out notch in the panel that would logically be used to thread the wiring through (which is facing to the back of the bot), but this build ignores that logic ;)

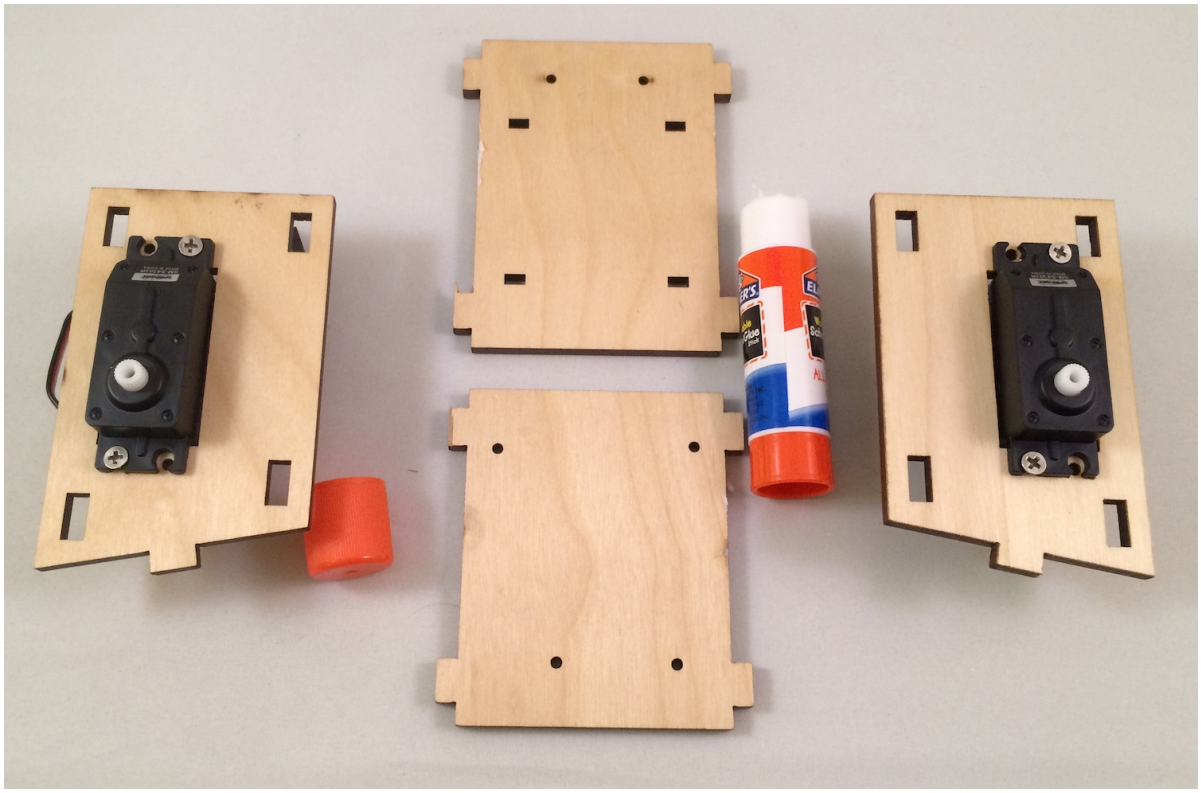


2. Push the servo through the mounting cut-out and line up its mounting tabs with the through holes in the side panel.

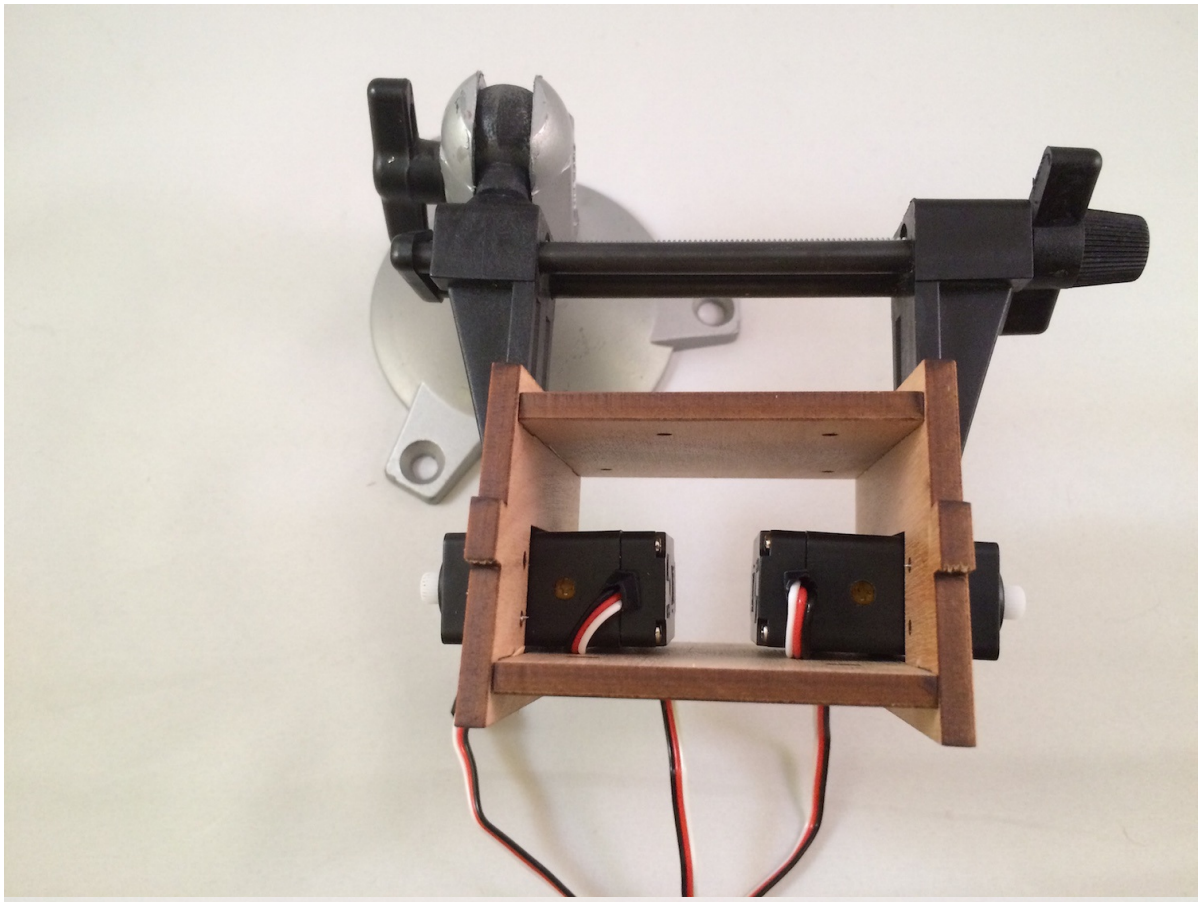
3. This is what correctly assembled right and left panels look like:

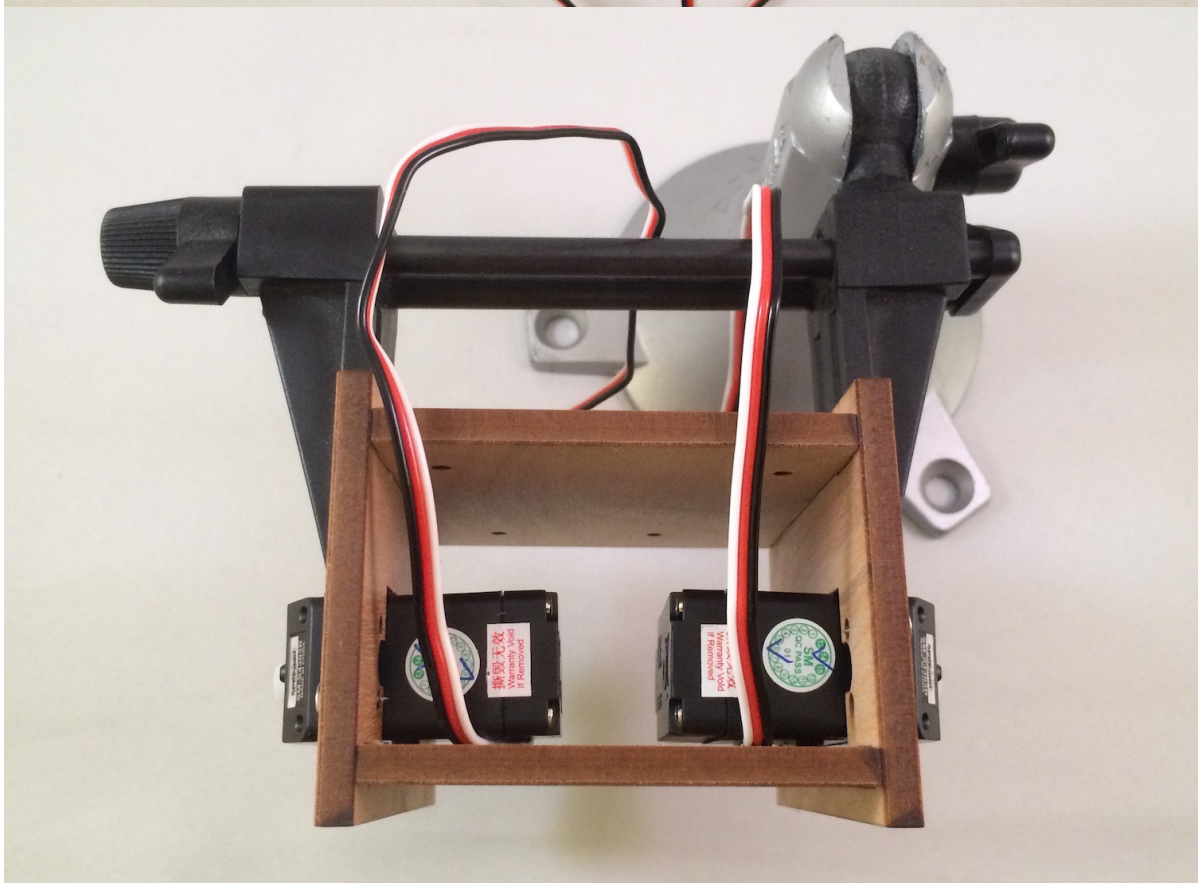


4. Lay out the (clockwise) bottom, right, top and left panels as shown here:

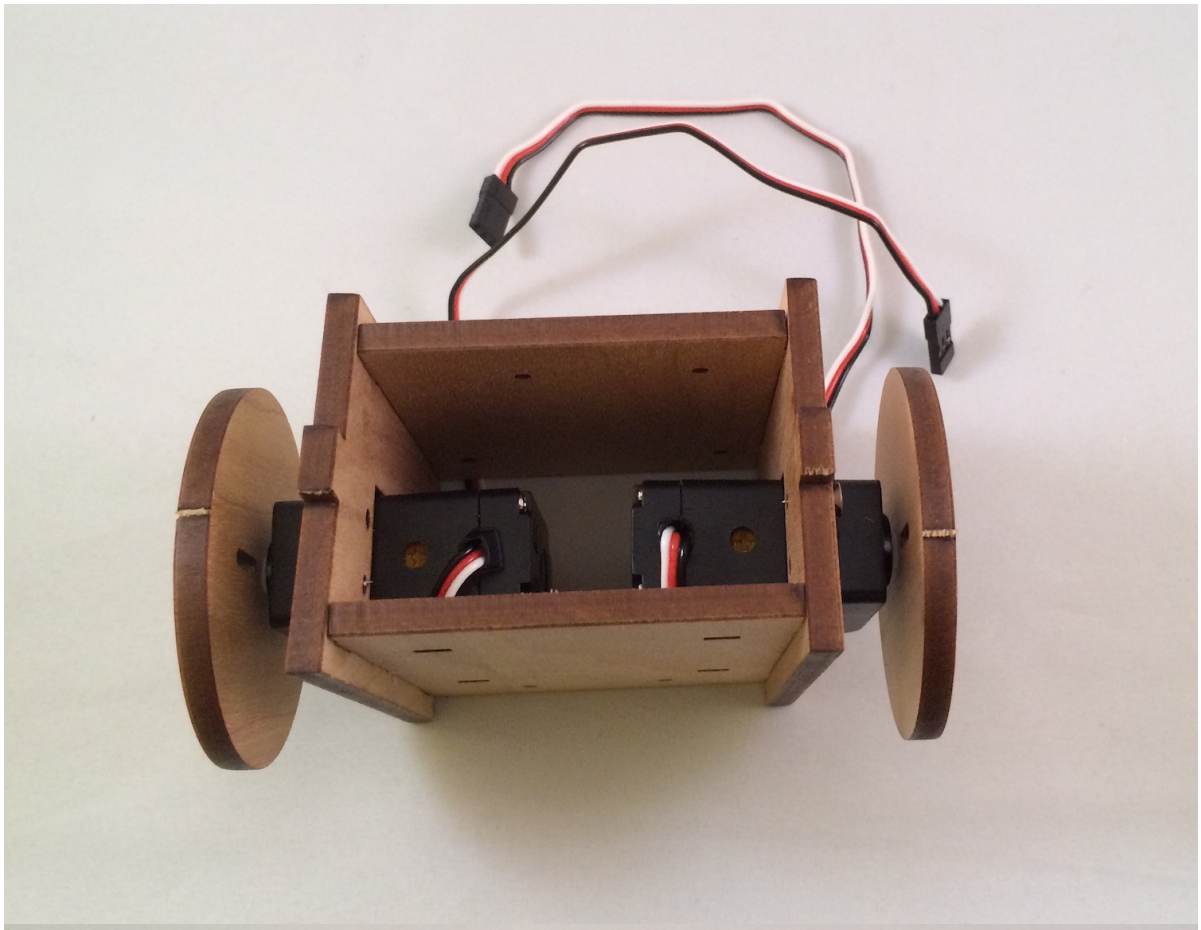


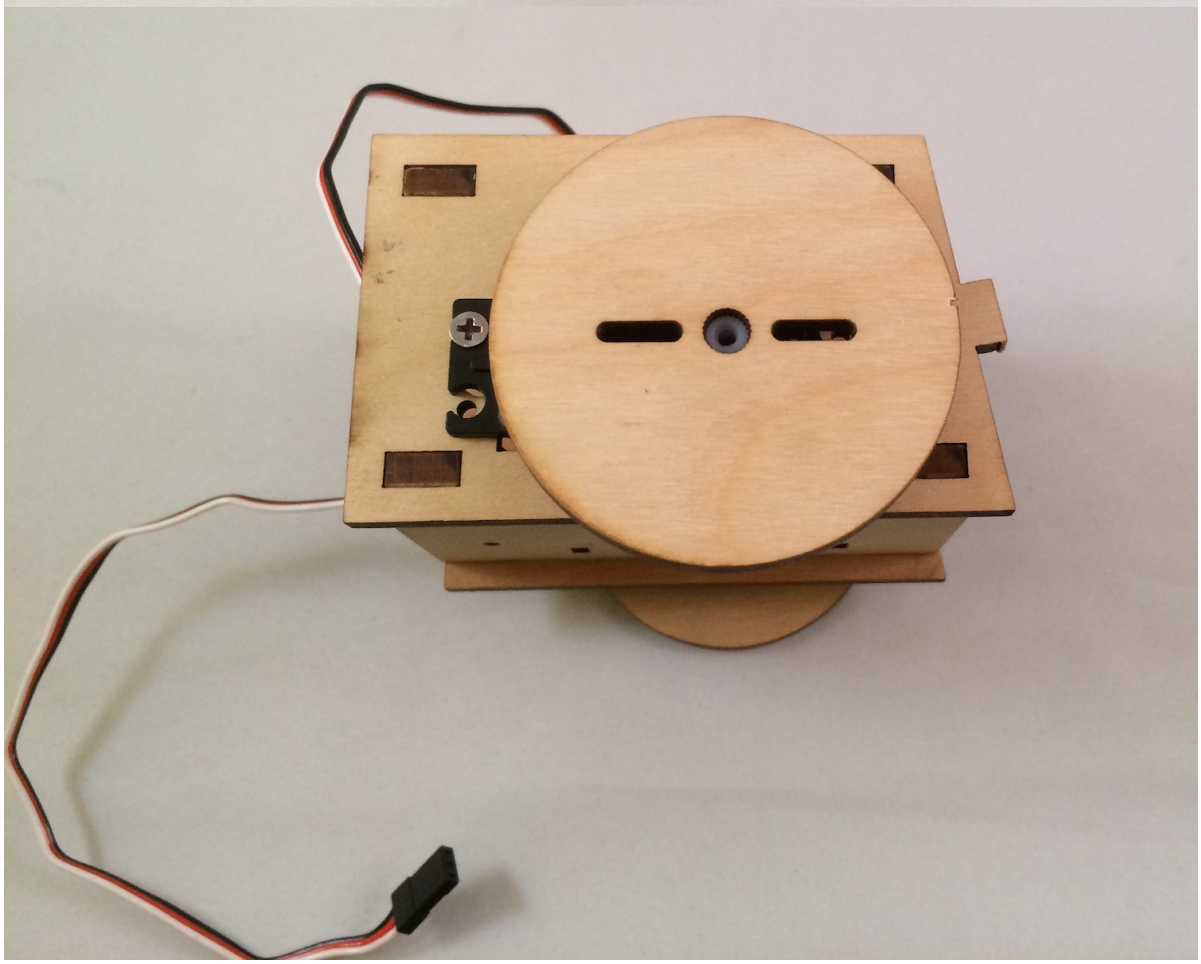
5. Using a bit of glue at each seam, combine the panels so that they match the following. (Note that the ball caster screws should be in the rear of the assembled bot.)



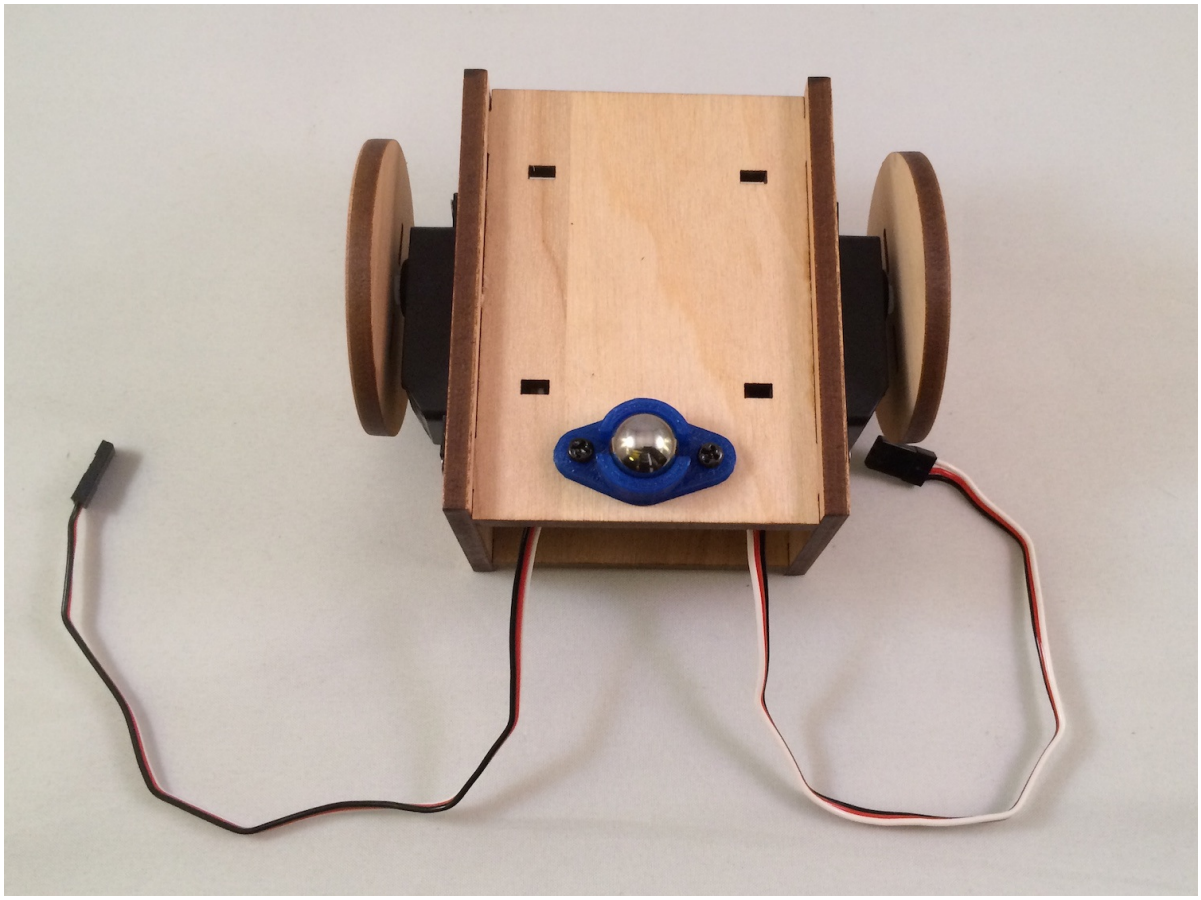


6. Using adequate force, push the wheels onto the servo horns. Notice that the wheels are notched to match the servo horn.

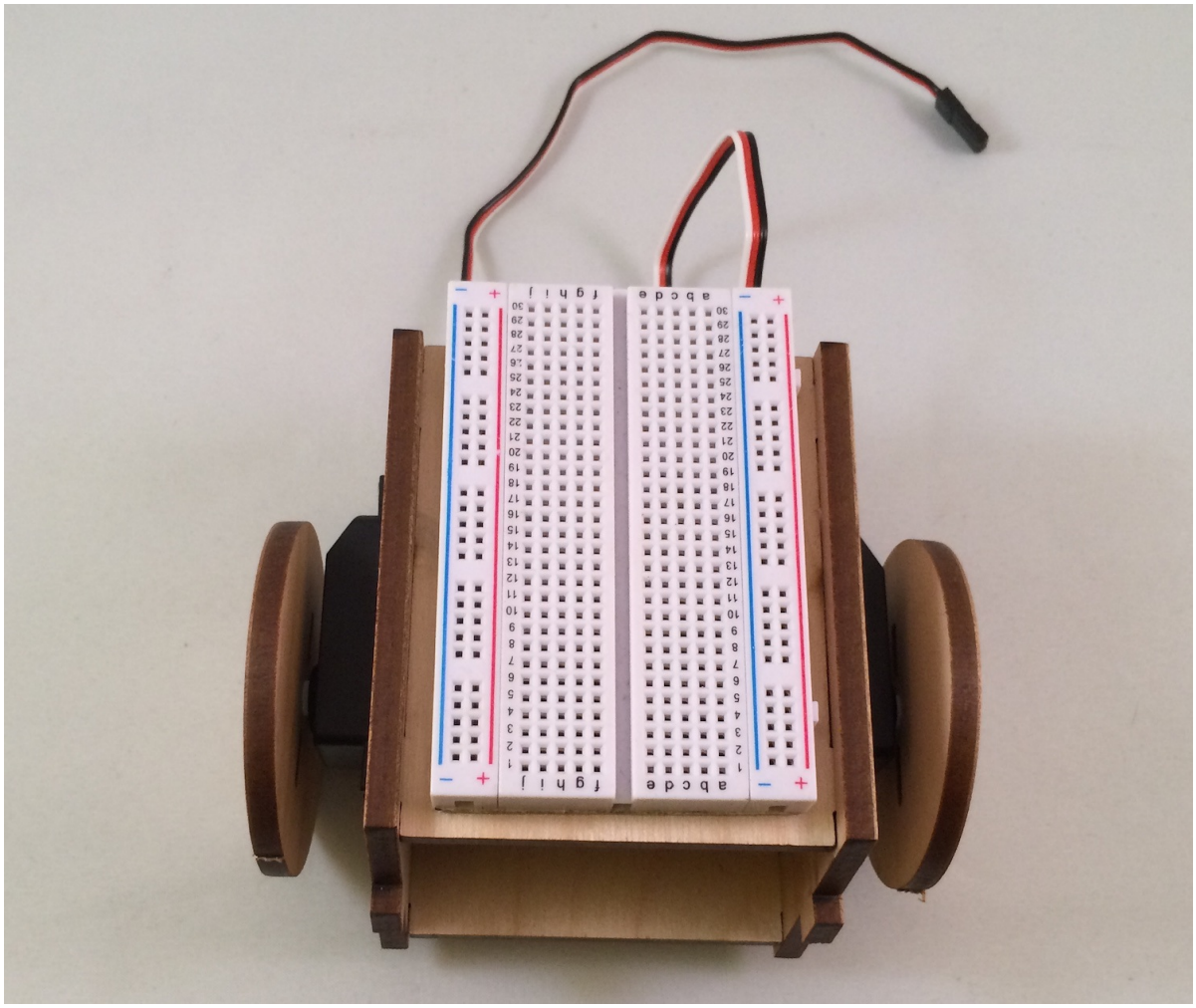




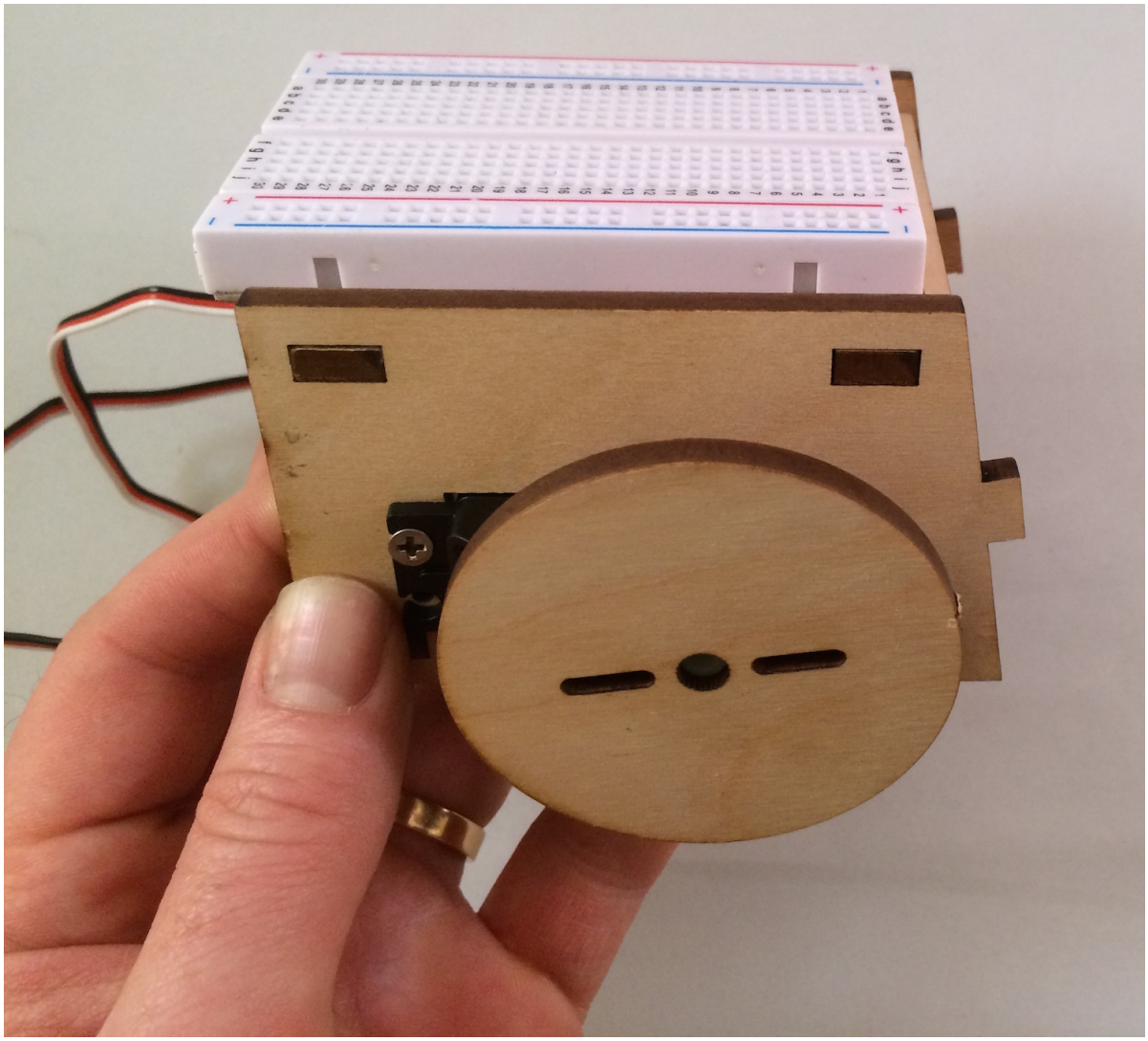
7. With two screws from one of the servo accessory bags, screw the ballcaster into the bottom panel:



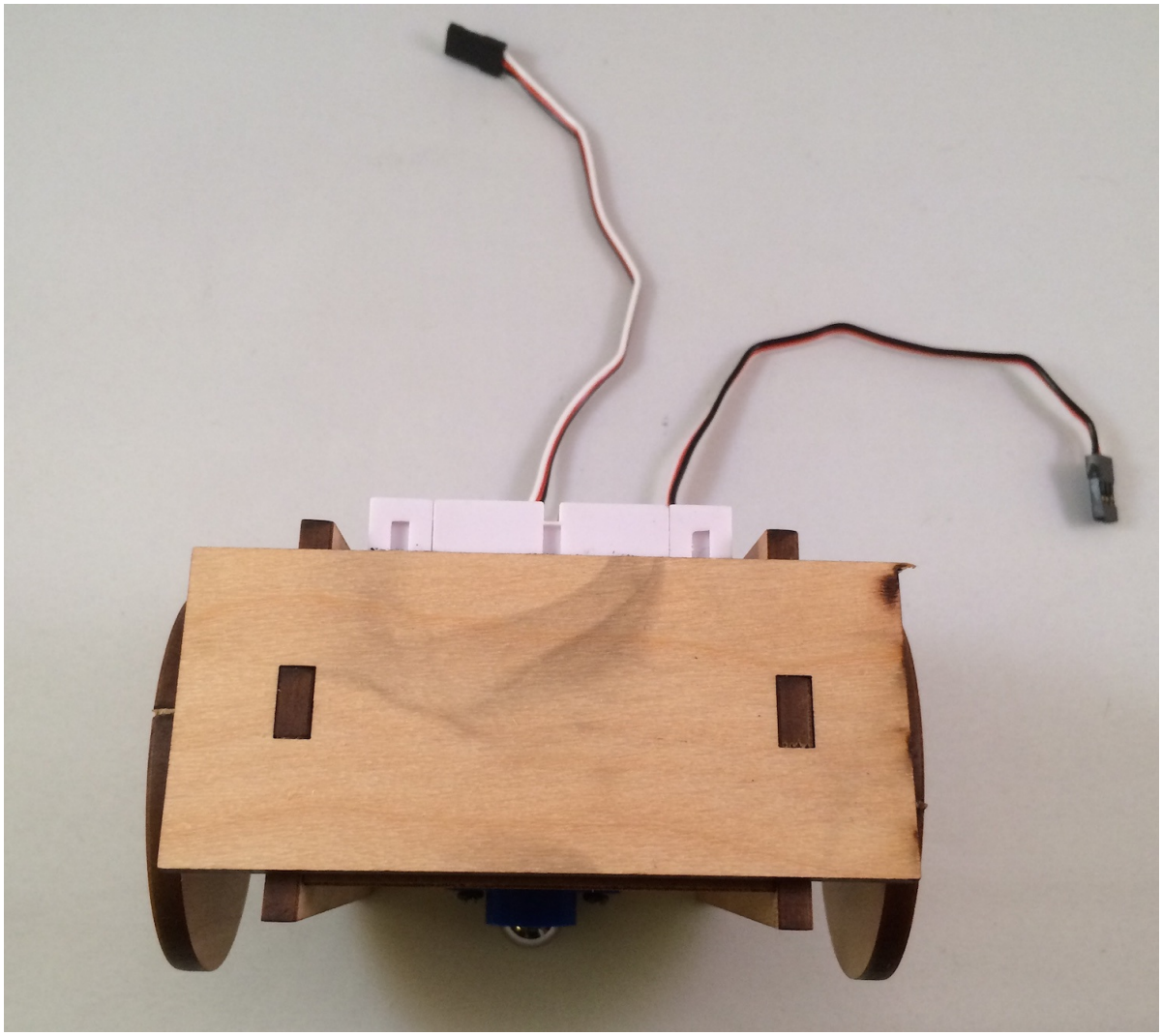
8. Remove the adhesive backing from the solderless breadboard and attach it directly to the top of the bot:



9. Attach the front panel to the bot (glue is optional, as you may want to customize this later).



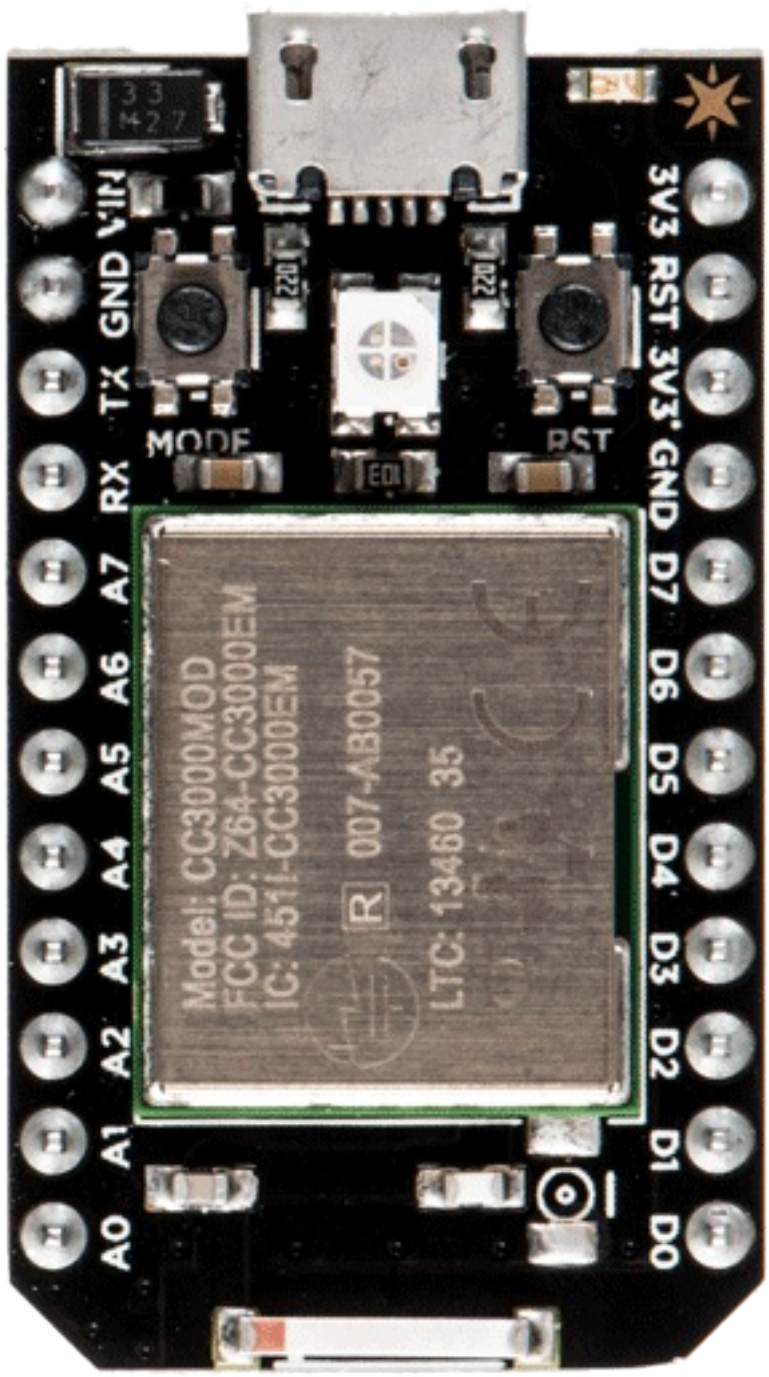
10. Enjoy your assembled robot!



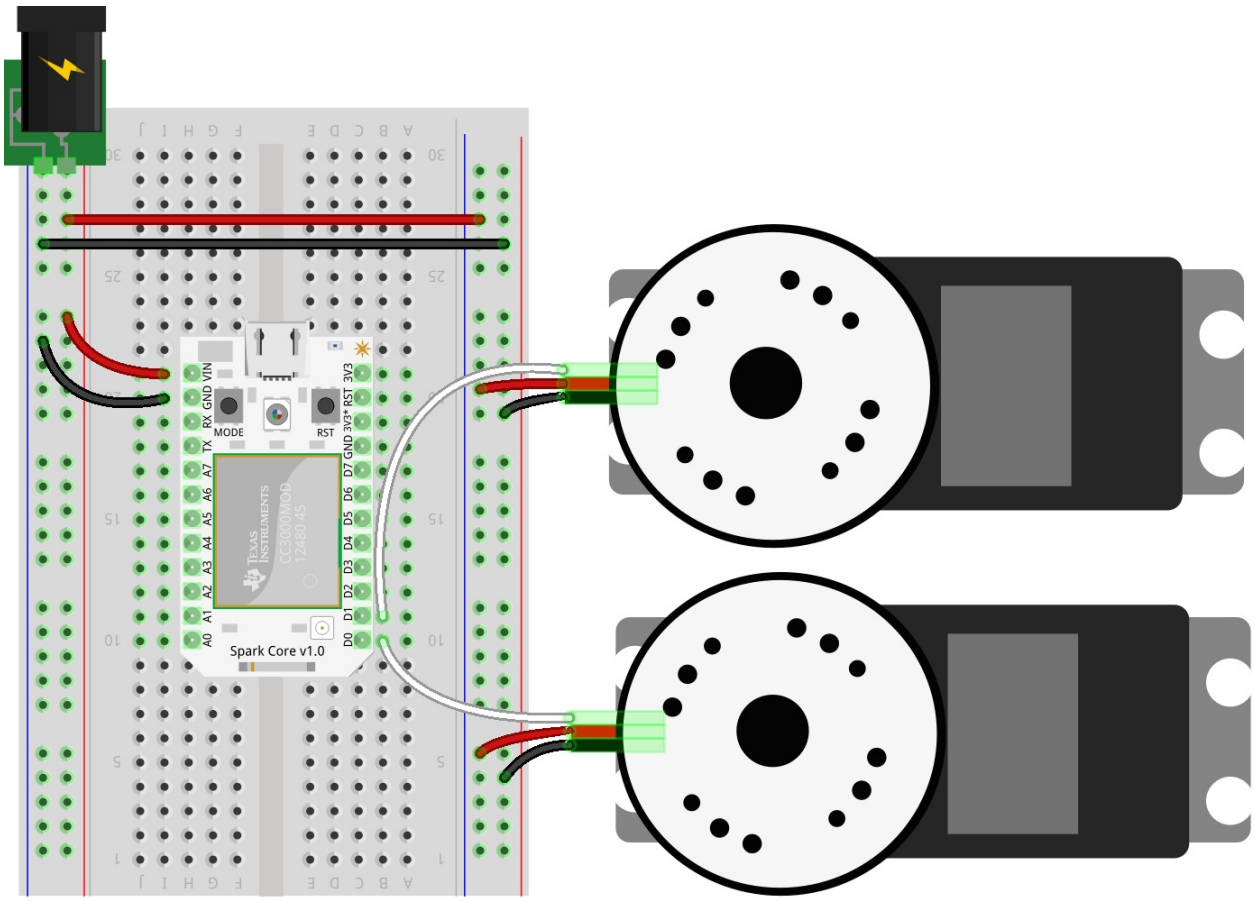
These instructions are adapted from Assembling and Preparing the RobotsConf Sumobot with Johnny-Five by Rick Waldron, creator of Johnny Five

Connecting the brain (Particle Core)

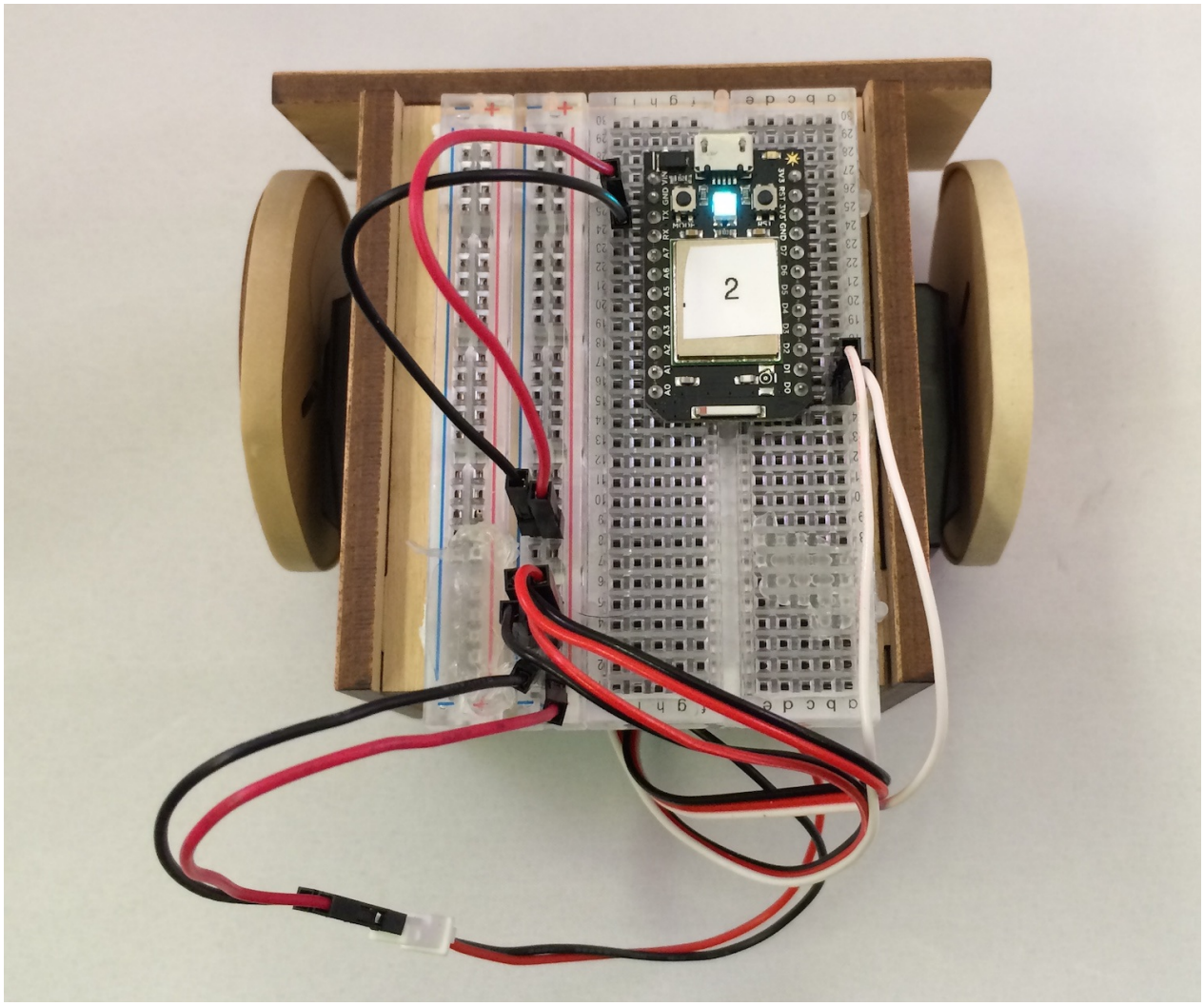
Particle Core



Sumobot Circuit



Fully assembled circuit attached to the Sumobot body



Controlling The SumoBot

The controller program has been adapted from the original program provided in the SumboBot Jr. repo and is available via npm. The controls library, `keypress`, must also be installed.

```
npm install keypress sumobot
```

The keypress module will turn your laptop into a remote control device for the SumoBot, using the following keys -> command mappings:

Key	Command
Arrow Up	Forward
Arrow Down	Reverse

Arrow Left	Turn Left
Arrow Right	Turn Right
Space	Stop
Q	Quit

```
// keyControl.js
var keypress = require("keypress");
var Spark = require("spark-io");
var five = require("johnny-five");
var Sumobot = require("sumobot")(five);

keypress(process.stdin);

var board = new five.Board({
  io: new Spark({
    token: process.env.SPARK_TOKEN,
    deviceId: process.env.SPARK_DEVICE_2
  })
});

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

  console.log("Welcome to Sumobot Jr!");

  // Initialize a new Sumobot.
  // - Left Servo is attached to pin D0
  // - Right Servo is attached to pin D1
  // - Speed set to 0.50 (half of max speed)
  //
  var bot = new Sumobot({
    left: "D0",
    right: "D1",
    speed: 0.50
  });

  // Maps key names to bot methods
  var actions = {
    up: "fwd",
    down: "rev",
    left: "left",
    right: "right",
    space: "stop"
  };

  // Ensure the bot is stopped
  bot.stop();

  // A bit of keypress ceremony ;)
  process.stdin.resume();
  process.stdin.setEncoding("utf8");
  process.stdin.setRawMode(true);
```

```
  process.stdin.on("keypress", function(ch, key) {
    var action;

    if (!key) {
      return;
    }

    action = actions[key.name] || key.name;

    if (action == "q") {
      console.log("Quitting");
      bot.stop();
      setTimeout(process.exit, 500);
    }

    if (bot[action]) {
      bot[action]();
    }
  });
});
```

These instructions are adapted from Controlling the RobotsConf Sumobot with Spark Core & Johnny-Five by Rick Waldron, creator of Johnny Five

Coding Up Your New Robot!

Johnny Five is Alive

Running a script

Code Samples

Resources/Links for Making Robots

What Now? What to do when Robot comes home.

Special Thanks and Sponsors