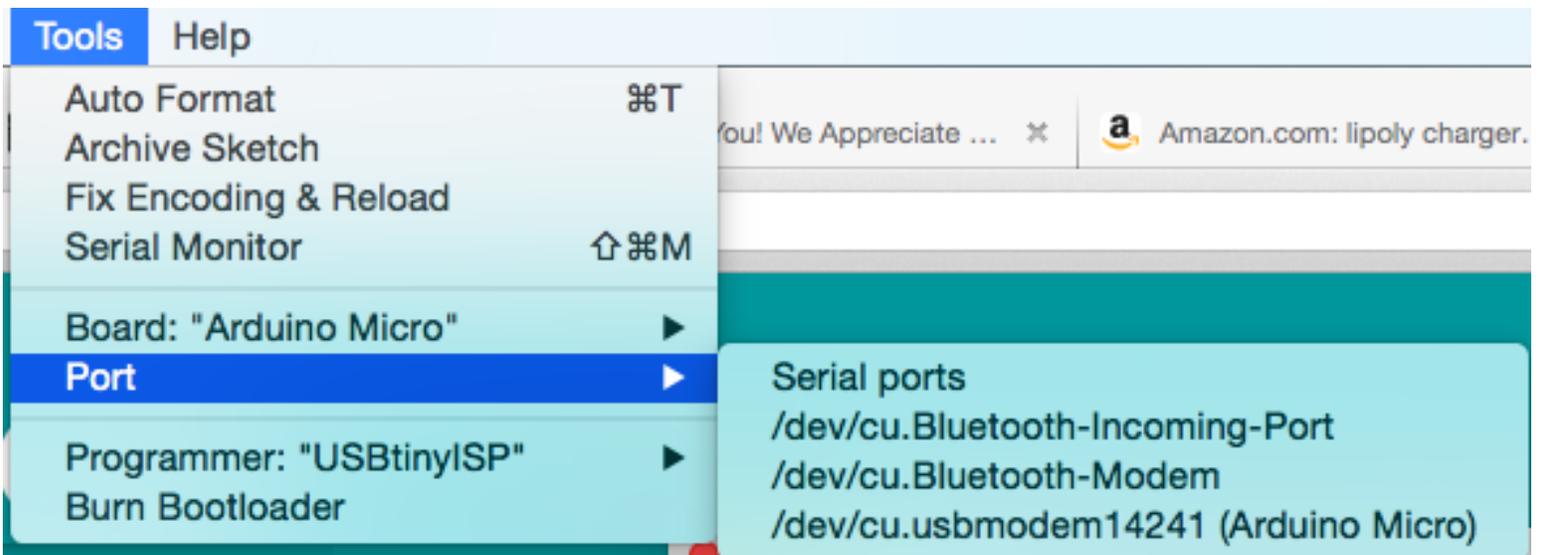


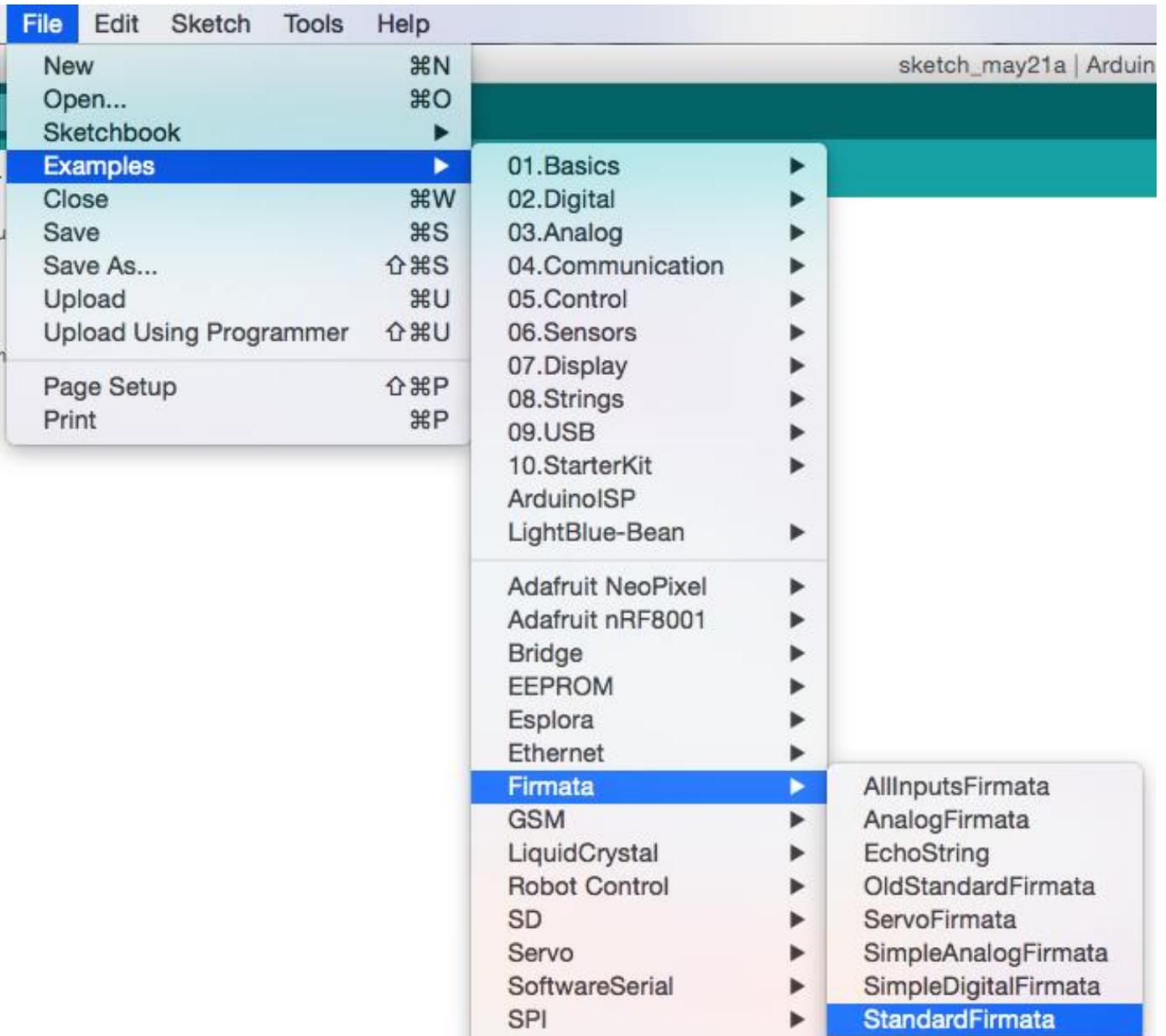
# 1

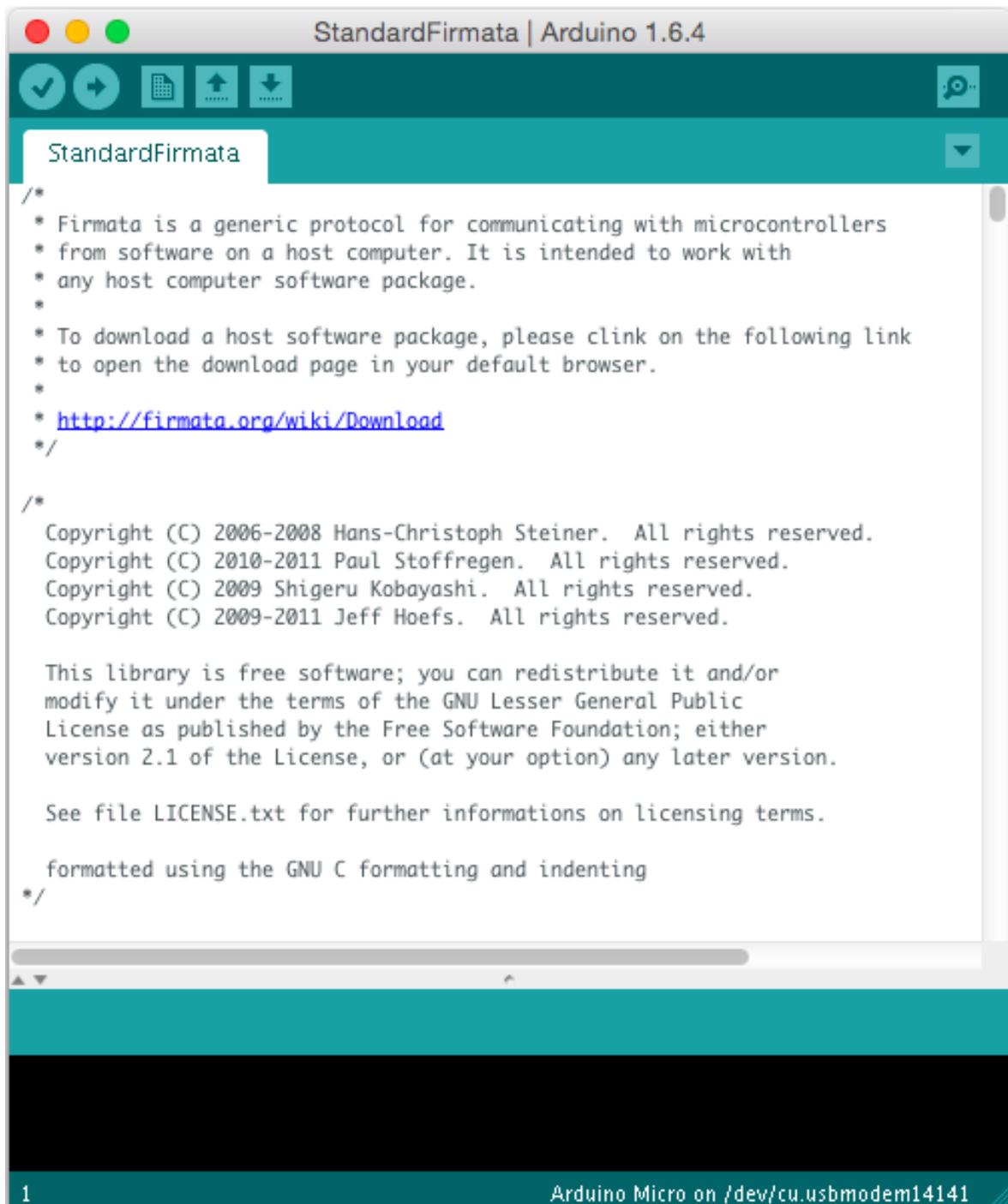
## Getting Started with JS Robotics

Setting up your development environment

Connecting your Microcontroller and installing Firmata





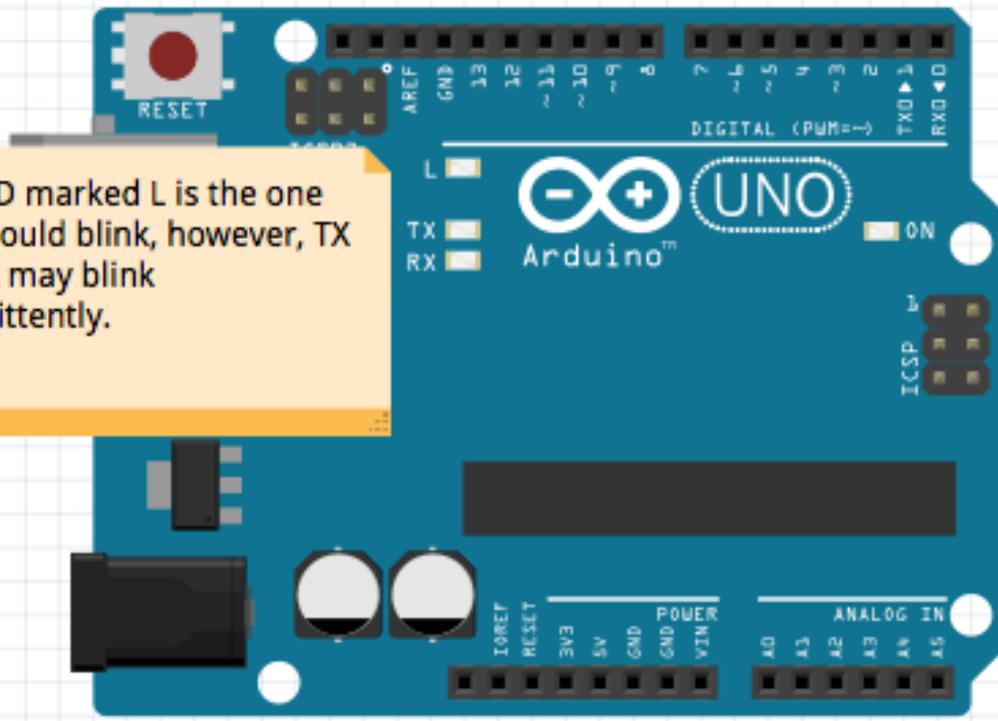


## Hello, World! – Blinking an onboard LED

### Running the script

```
nodebotanist@Kassandras-MacBook-Air ~/Writing/packt-j5-book/chapter1/code node hello-world.js
1432242187981 Looking for connected device
1432242226377 Device(s) /dev/cu.usbmodem14131
1432242226406 Connected /dev/cu.usbmodem14131
1432242229711 Repl Initialized
>>
```

The LED marked L is the one that should blink, however, TX and RX may blink intermittently.

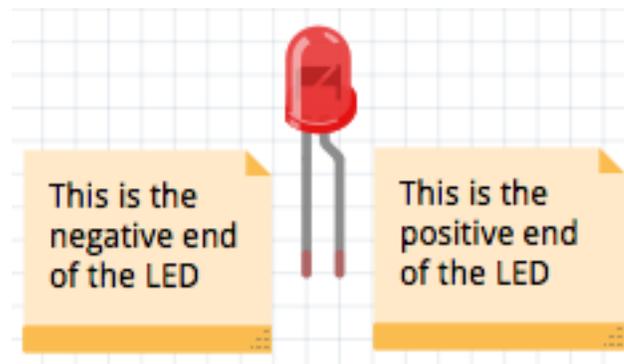


# 2

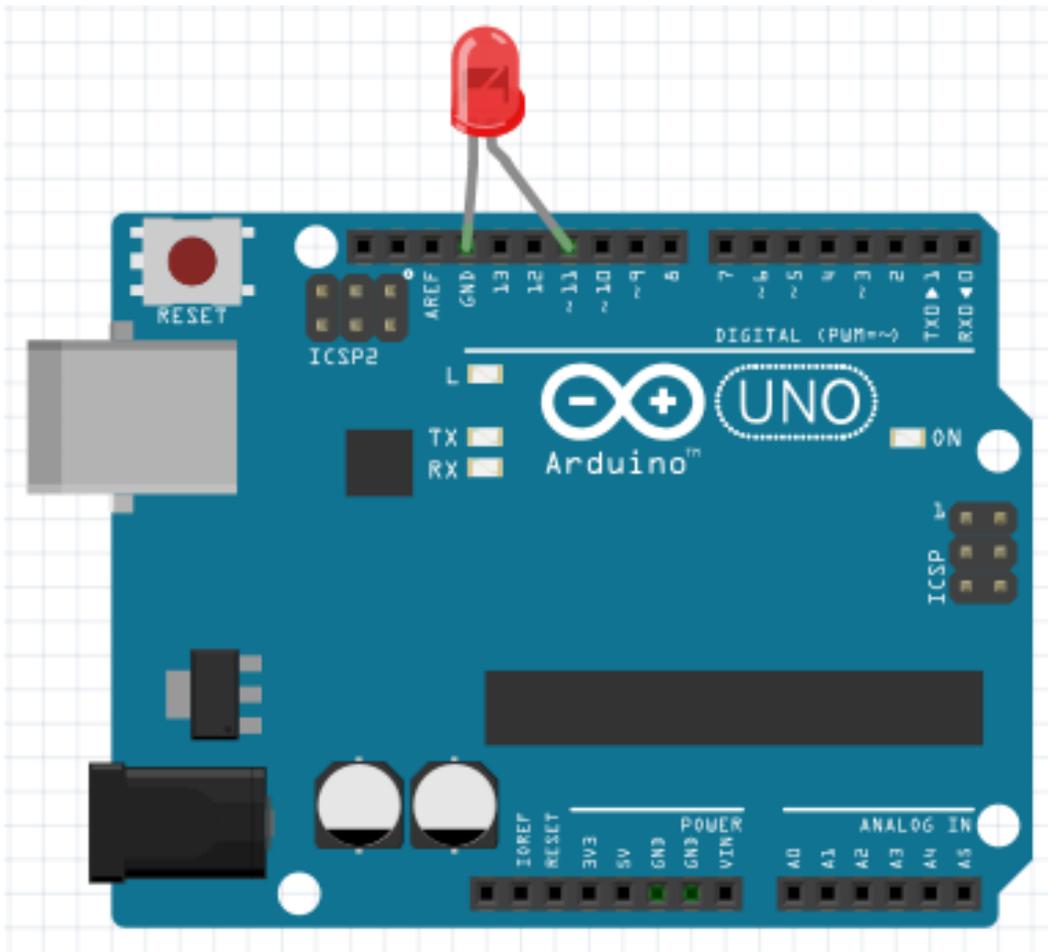
## Working with Johnny-Five

### Wiring up an external LED

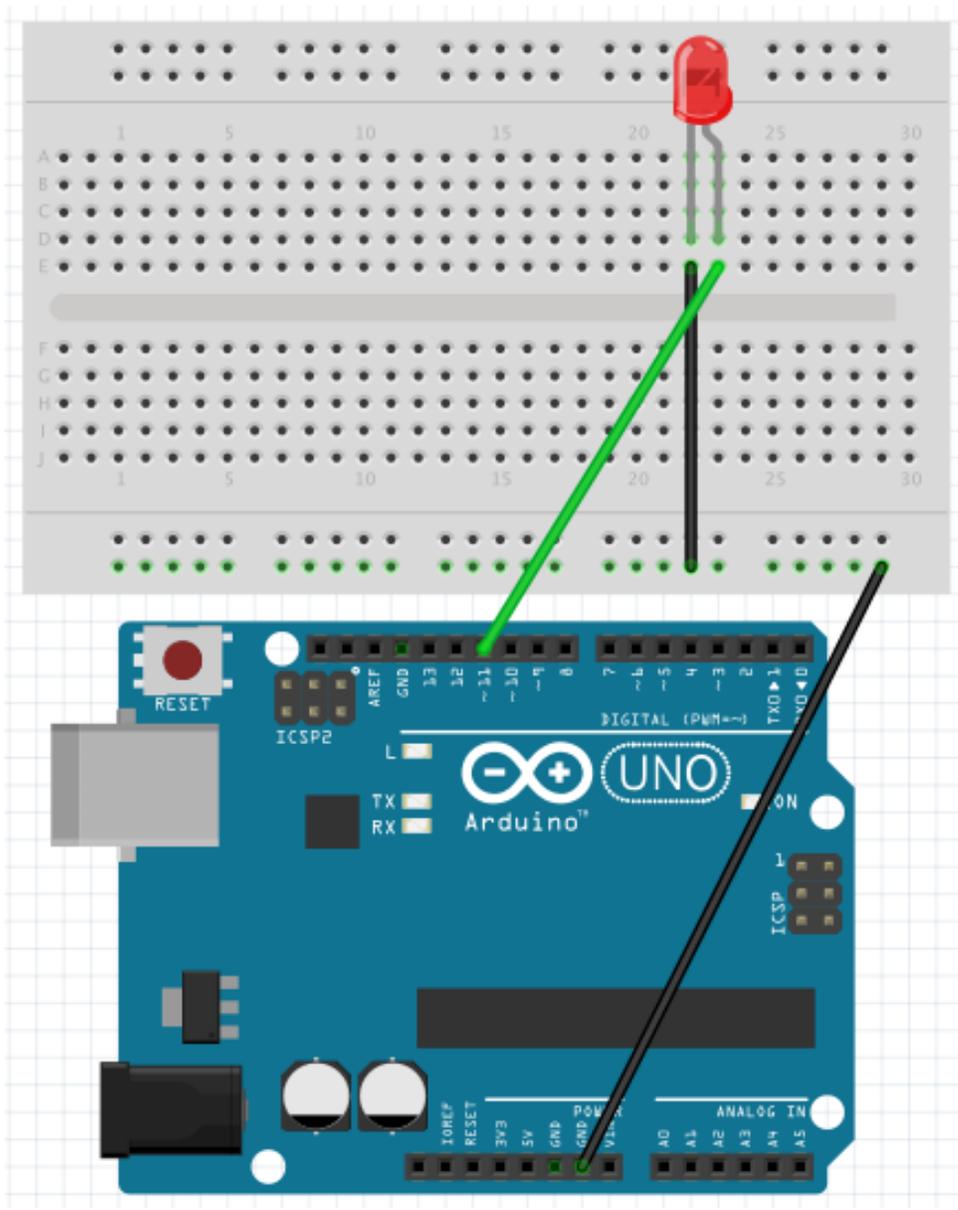
#### Setting up the hardware



Determining the positive and negative ends of an LED



Wiring up our LED



Wiring an LED with a breadboard

## Using the Read-Eval-Print-Loop (REPL)

### Using the REPL

```
nodebotanist@Kassandras-MacBook-Air ~/Writing/packt-j5-book/chapter1/code node hello-world.js
1432242187981 Looking for connected device
1432242226377 Device(s) /dev/cu.usbmodem14131
1432242226406 Connected /dev/cu.usbmodem14131
1432242229711 Repl Initialized
>>
```

Terminal setup for Johnny-Five REPL prompt

```
>> myLed
{ board:
  { timer:
    { '0': null,
      _idleTimeout: -1,
      _idlePrev: null,
      _idleNext: null,
      _idleStart: 567293514,
      _onTimeout: null,
      _repeat: false },
    isConnected: true,
    isReady: true,
    io:
      { domain: null,
        _events: [Object],
        _maxListeners: undefined,
        isReady: true,
        MODES: [Object],
        I2C_MODES: [Object],
        STEPPER: [Object],
        HIGH: 1,
        LOW: 0,
        pins: [Object],
        analogPins: [Object],
        version: [Object],
        firmware: [Object],
        currentBuffer: [],
```

The output of your myLed object in the REPL

```
>> myLed.stop()
{ board:
  { timer:
    { '0': null,
      _idleTimeout: -1,
      _idlePrev: null,
      _idleNext: null,
      _idleStart: 567293514,
      _onTimeout: null,
      _repeat: false },
    isConnected: true,
    isReady: true,
    io:
      { domain: null,
        _events: [Object],
        _maxListeners: undefined,
        isReady: true,
        MODES: [Object],
        I2C_MODES: [Object],
        STEPPER: [Object],
```

The output from myLed.stop();

```
>> myLed.stop().off()
{ board:
  { timer:
    { '0': null,
      _idleTimeout: -1,
      _idlePrev: null,
      _idleNext: null,
      _idleStart: 567293514,
      _onTimeout: null,
      _repeat: false },
    isConnected: true,
    isReady: true,
    io:
      { domain: null,
        _events: [Object],
        _maxListeners: undefined,
        isReady: true,
        MODES: [Object],
        I2C_MODES: [Object],
        STEPPER: [Object],
```

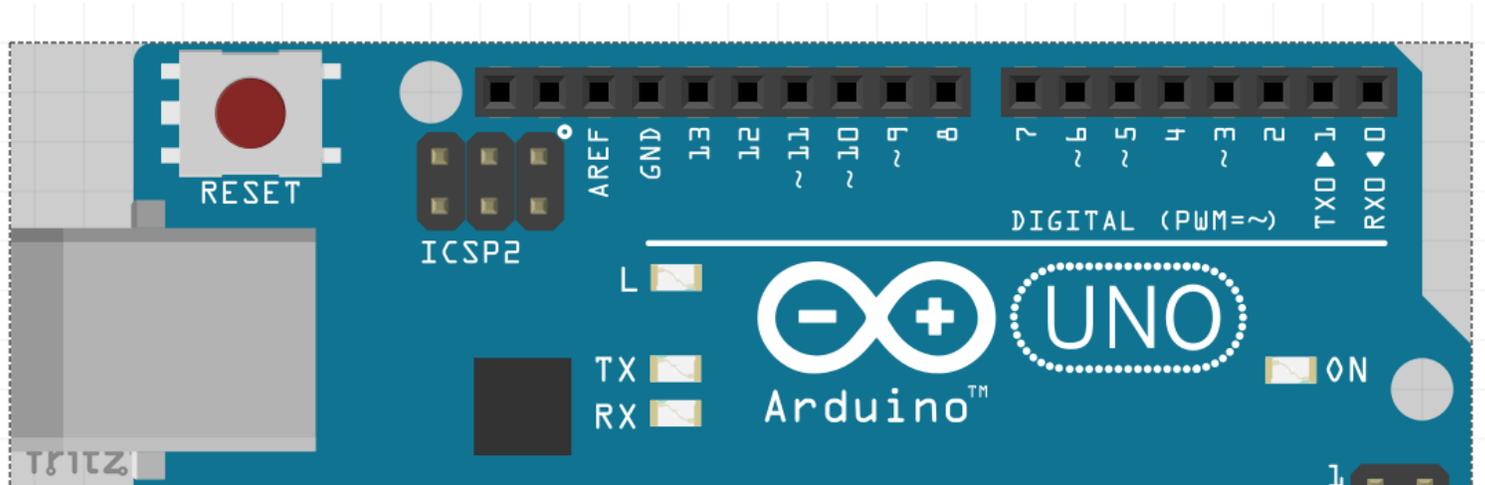
Using chainable function calls in the REPL

# 3

## Using Digital and PWM Output Pins

How GPIO pins work

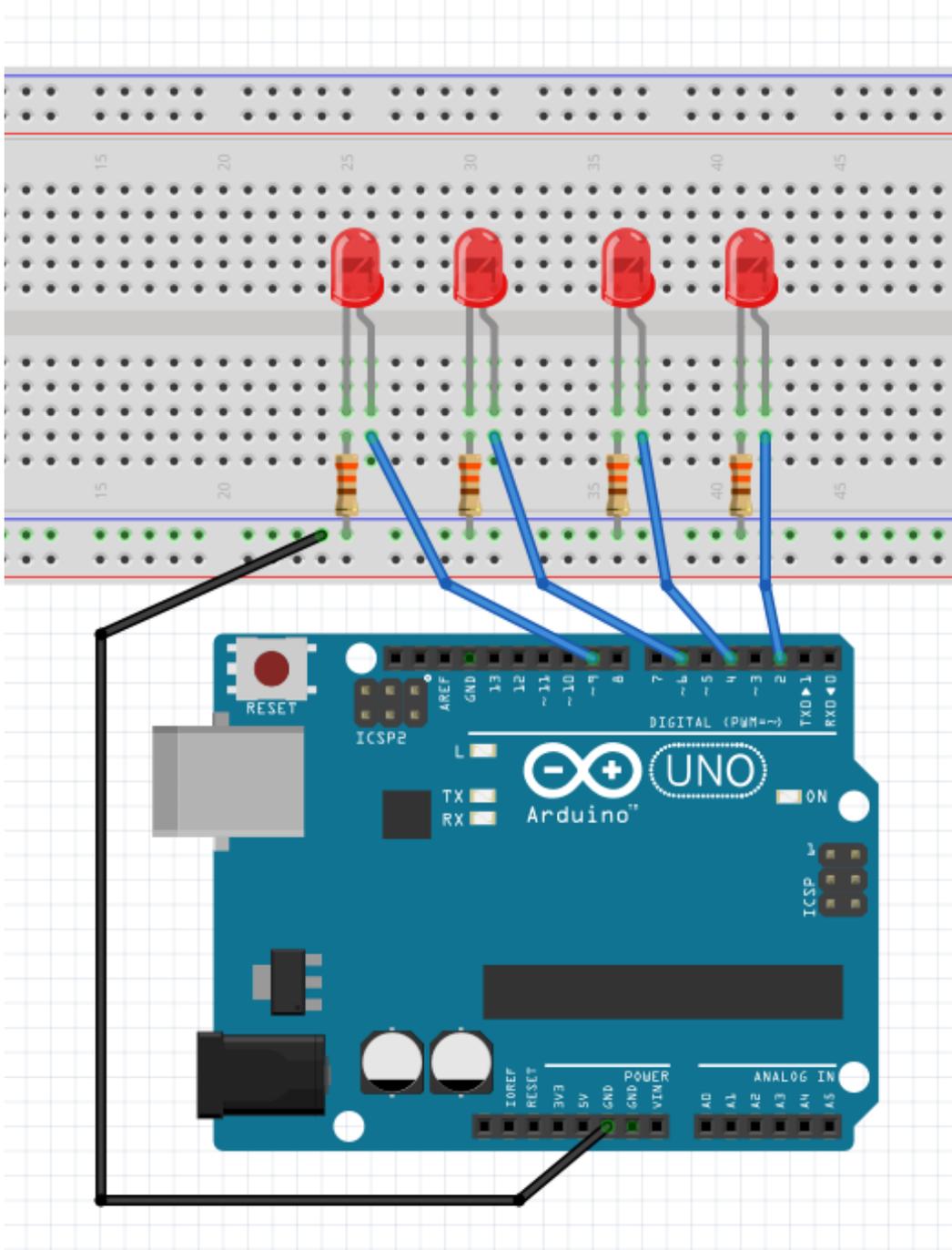
How to tell the difference between Digital and PWM pins



Determining the PWM pins on an Arduino Uno

# Multiple pins with several LEDs

## Setting up the hardware for the project



Wiring for our multiple LEDs project

## Exploring more about Led objects in Johnny-Five

```
_idlePrev: null,  
_idleNext: null,  
_idleStart: 629324856,  
_onTimeout: null,  
_repeat: null },  
defaultLed: 13,  
port: '/dev/cu.usbmodem1421' },  
id: null,  
pin: 2,  
interval: null }  
>> /Users/nodebotanist/node_modules/johnny-five/lib/board.pins.js:67  
throw new Error(  
  ^  
Error: Pin Error: 2 is not a valid PWM pin (Led)  
    at Function.Pins.Error (/Users/nodebotanist/node_modules/johnny-five/lib/board.pins.js:67:9)  
    at Led.Controllers.DEFAULT.write.value (/Users/nodebotanist/node_modules/johnny-five/lib/led/led.js:106:22)  
    at Led.(anonymous function) [as @@render] (/Users/nodebotanist/node_modules/johnny-five/lib/led/led.js:333:15)  
    at Animation.<anonymous> (/Users/nodebotanist/node_modules/johnny-five/lib/animation.js:250:34)  
    at Immediate.processQueue (/Users/nodebotanist/node_modules/johnny-five/node_modules/temporal/lib/temporal.js:197:20)  
    at processImmediate [as _immediateCallback] (timers.js:368:17)
```

Error when using a PWM method on a digital pin

```
>> myLed2.on().isOn  
true
```

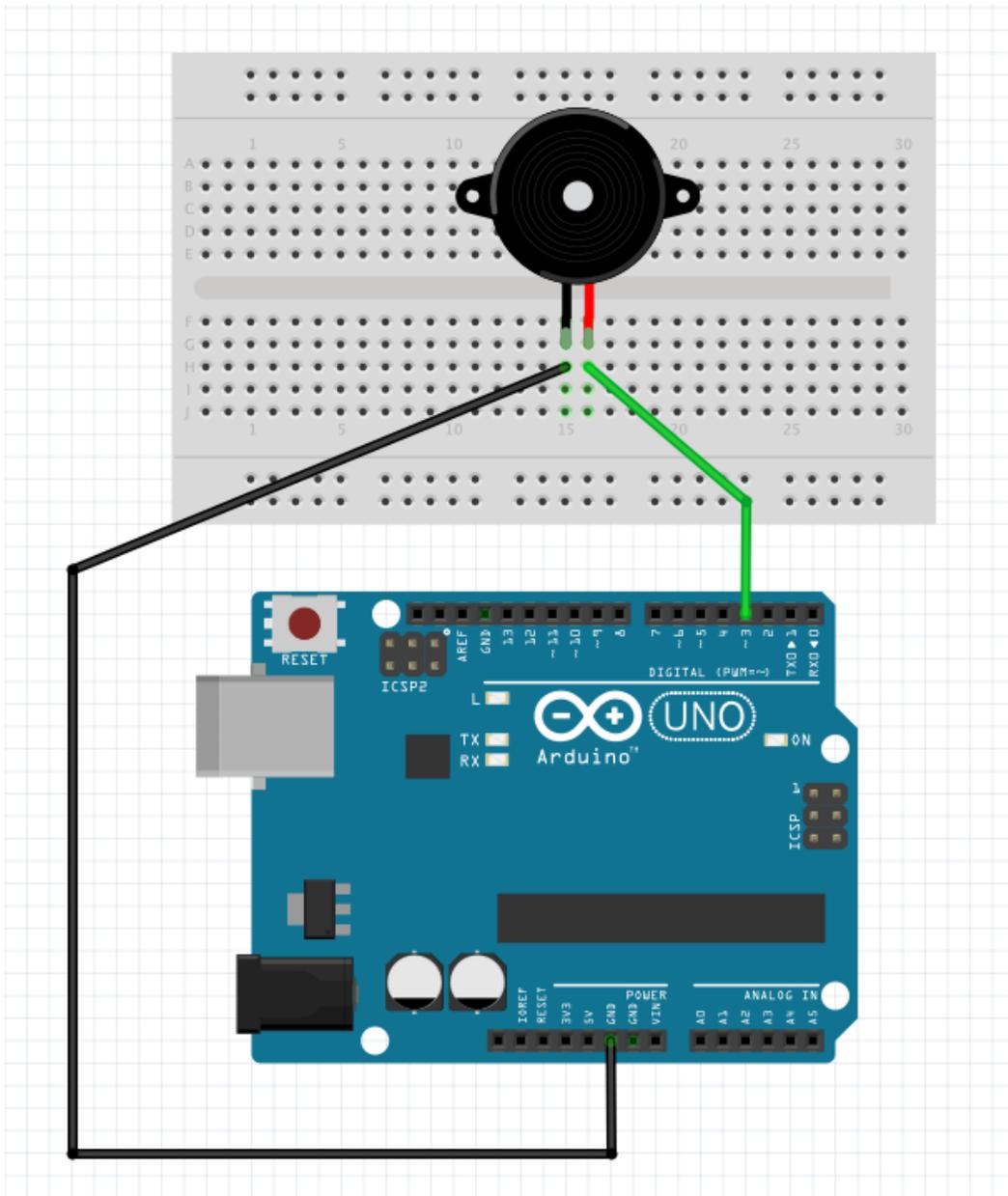
The isOn attribute

```
>> myLed2.value  
1  
>> myLed2.isOn  
true  
>>
```

The other Led attributes

# Using PWM pins with the Piezo element

## Setting up the hardware

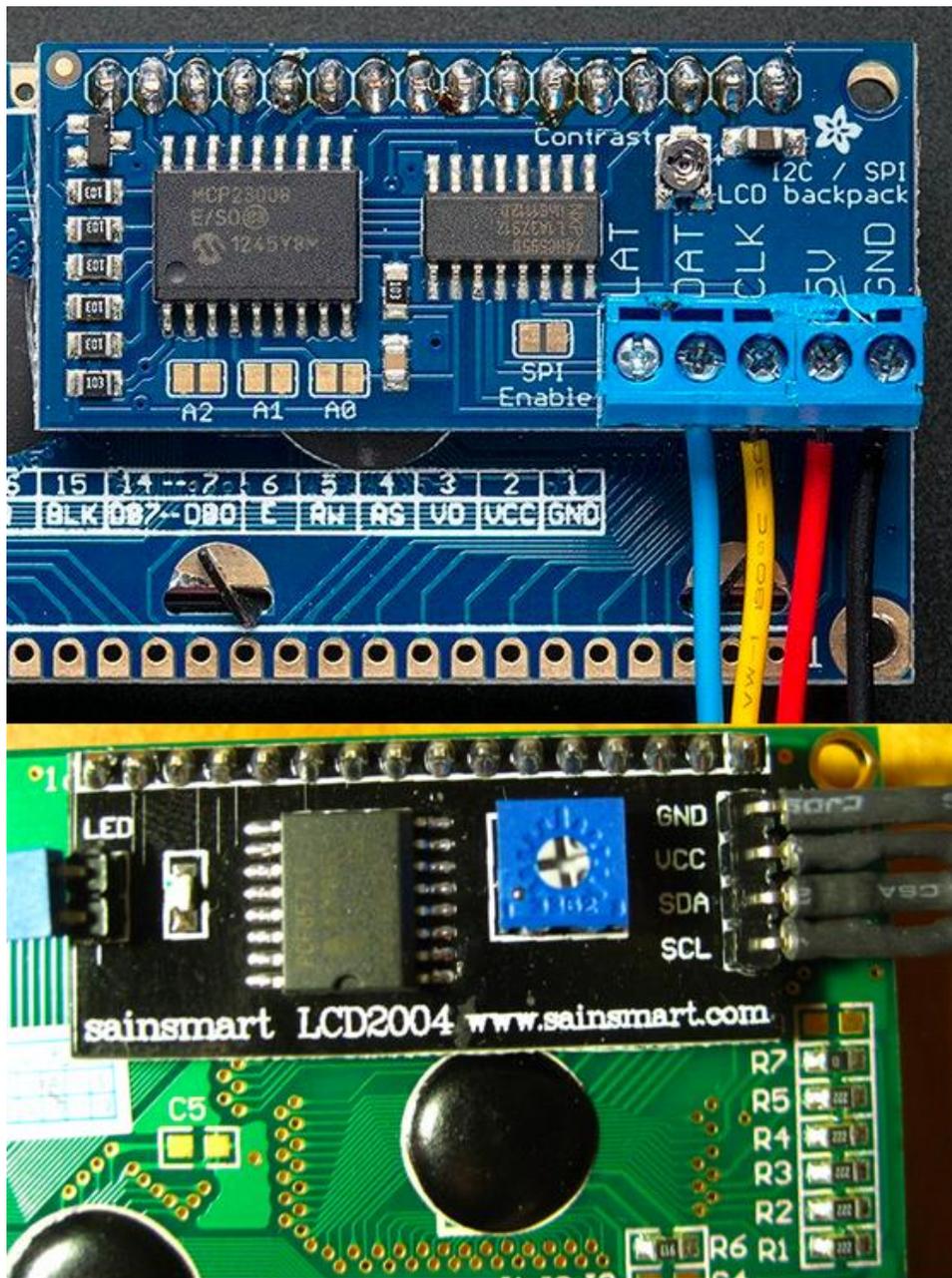


Piezo wiring diagram

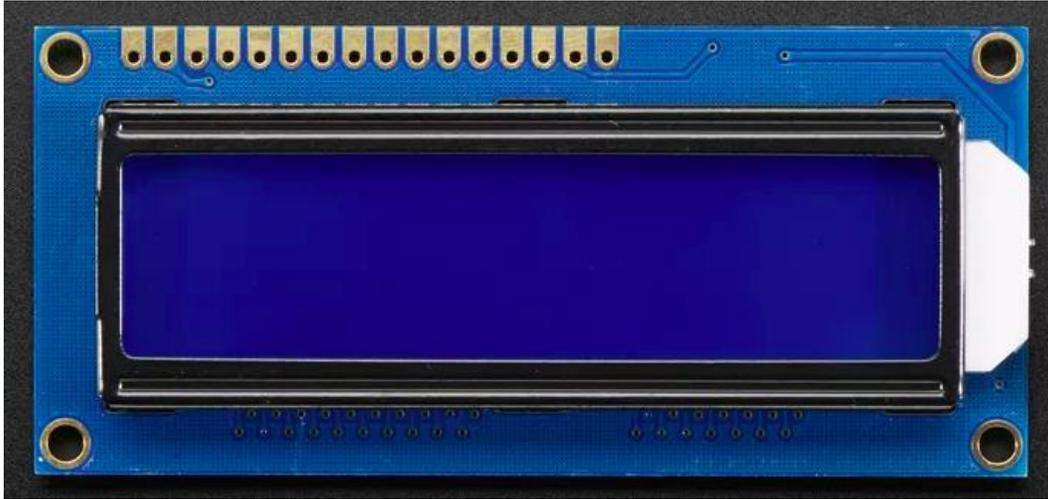
# 4

## Using Specialized Output Devices

What you'll need for this chapter

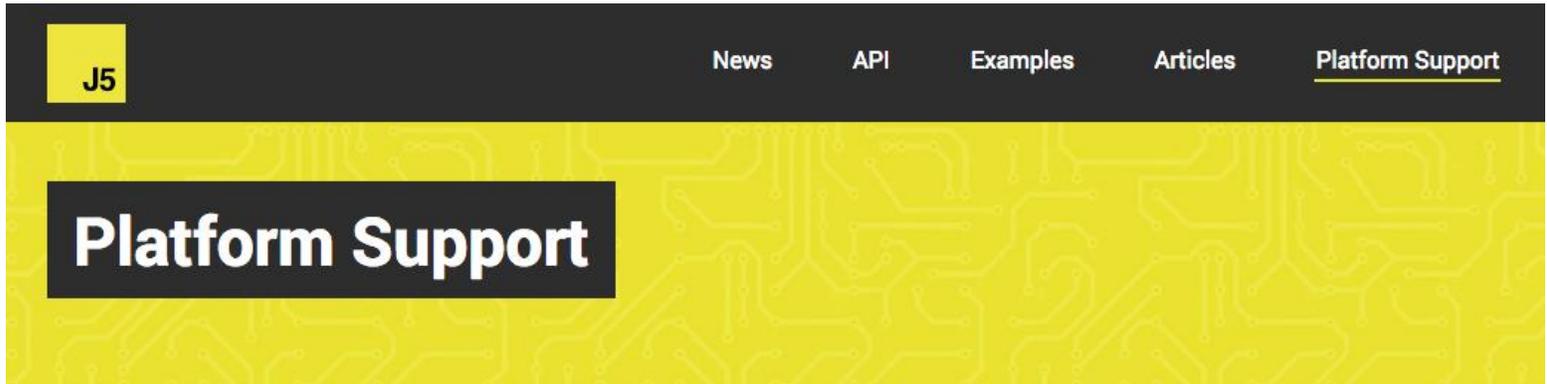


Examples of i2c backpacks on character LCDs



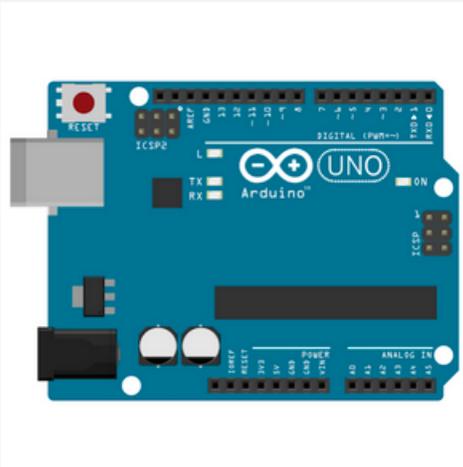
A non-i2c character LCD

## Checking compatibility with Johnny-Five



The johnny-five.io header

# Arduino Uno



- Firmware/Runtime: [StandardFirmata](#) (additional instructions)
- The JavaScript program is executed on a **host** machine that runs node.js/io.js. The program transmits basic IO instructions to the board via **usb serial**, which acts as a **thin client**. Requires tethering.

Analog Read	yes
Digital Read	yes
Digital Write	yes
PWM	yes
Servo	yes
I2C	yes
One Wire	yes
Stepper	yes

The Platform Support page entry for Arduino Uno

## Obtaining documentation, wiring diagrams, and so on

### API Documentation

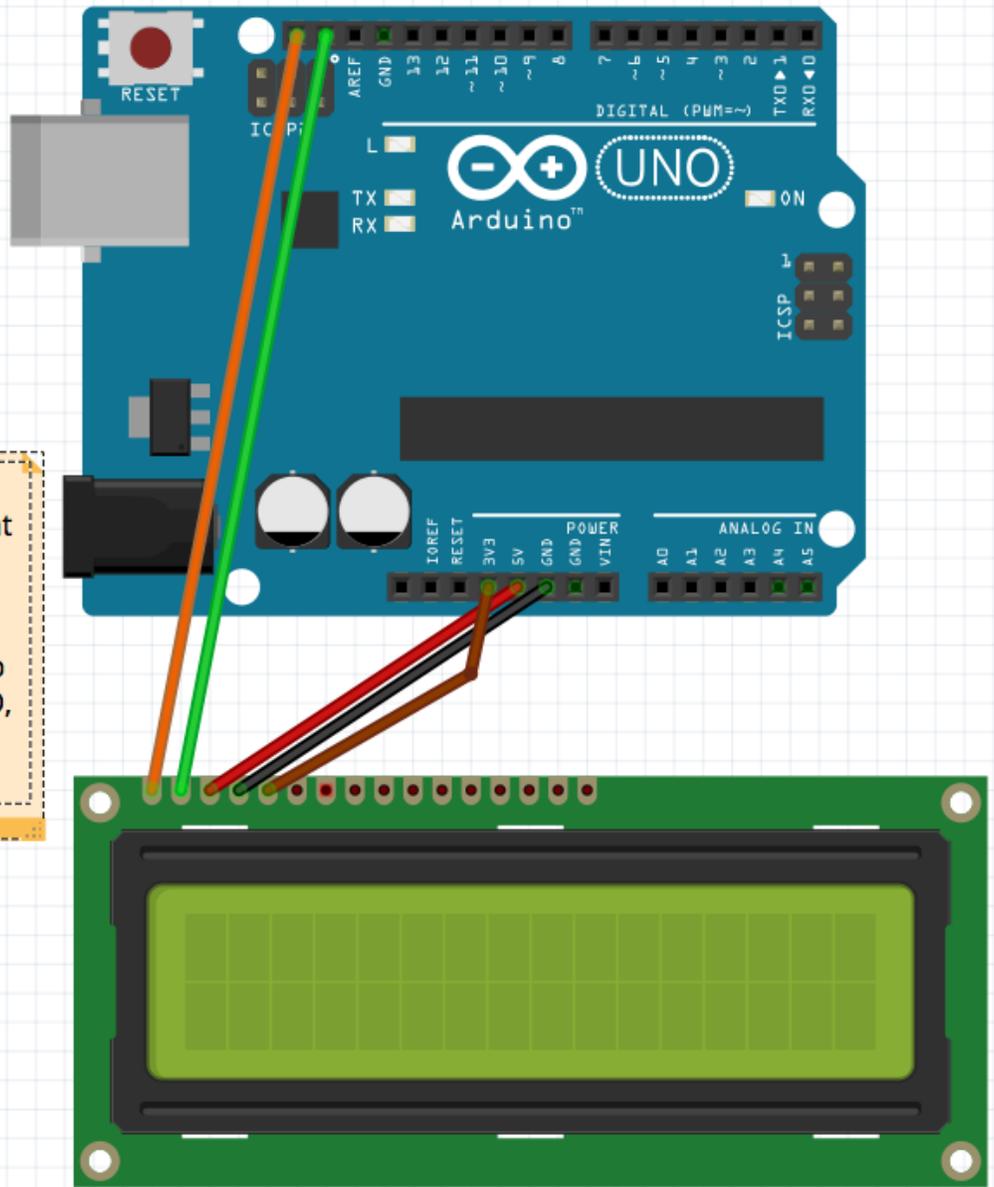
Component Classes	Guides
<a href="#">Accelerometer</a>	<ul style="list-style-type: none"><li>• <a href="#">Getting Started</a> Get Johnny-Five alive on your machine. (es_ES, nL_NL, pt_BR, fr_FR)</li><li>• <a href="#">Prerequisites</a> Prerequisites for Linux, OSX, and Windows.</li><li>• <a href="#">IO Plugins</a> Create IO Plugins for any platform!</li><li>• <a href="#">Control System Hierarchy Overview</a> A terminology primer for component class and IO Plugin authors.</li></ul>
<a href="#">Animation</a>	
<a href="#">Barometer</a>	
<a href="#">Board</a>	

The API documentation page

# Project – Character LCD

## Wiring up – i2c LCDs

No official part exists for the i2c backpack yet, so NOTE that this will look a tad different (see photo). The orange wire should go to the pin labeled SCL on the backpack, green to SDA, red to VCC, black to GND, and brown to LED (if that exists for your LCD)



tzina

A diagram of i2c LCD hookup

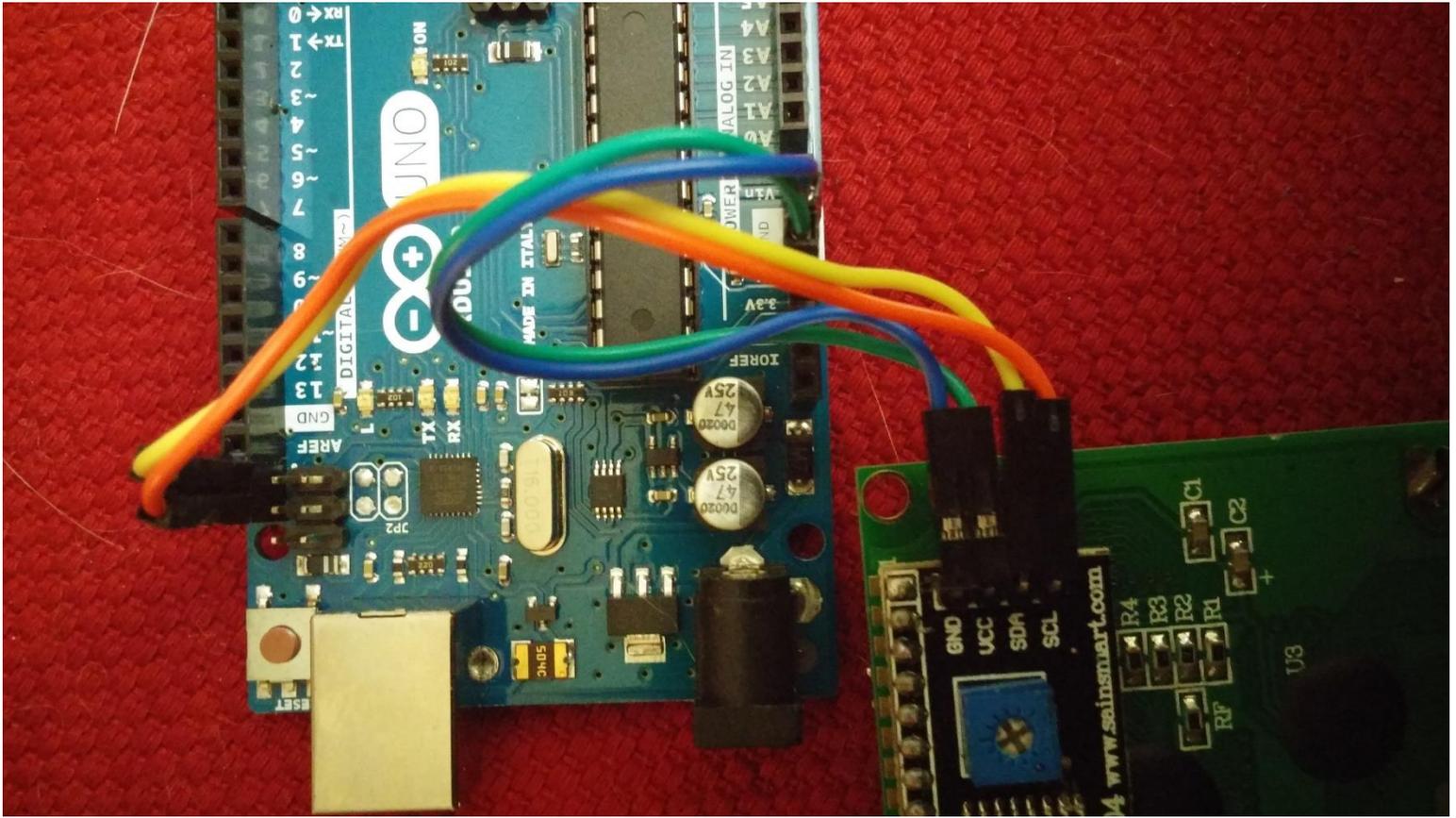


Photo of i2c backpack wiring. Wiring up regular LCDs



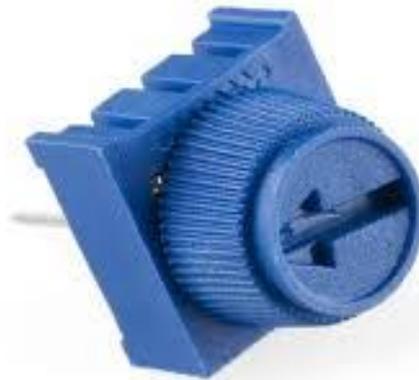
# 5

## Using Input Devices and Sensors

**What you'll need for this chapter**



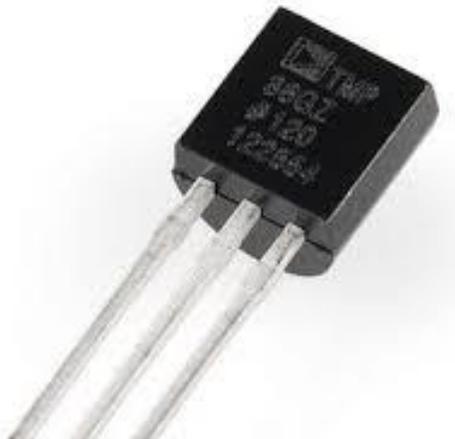
A common push button for robotics projects



A basic rotating potentiometer



A light sensor diode



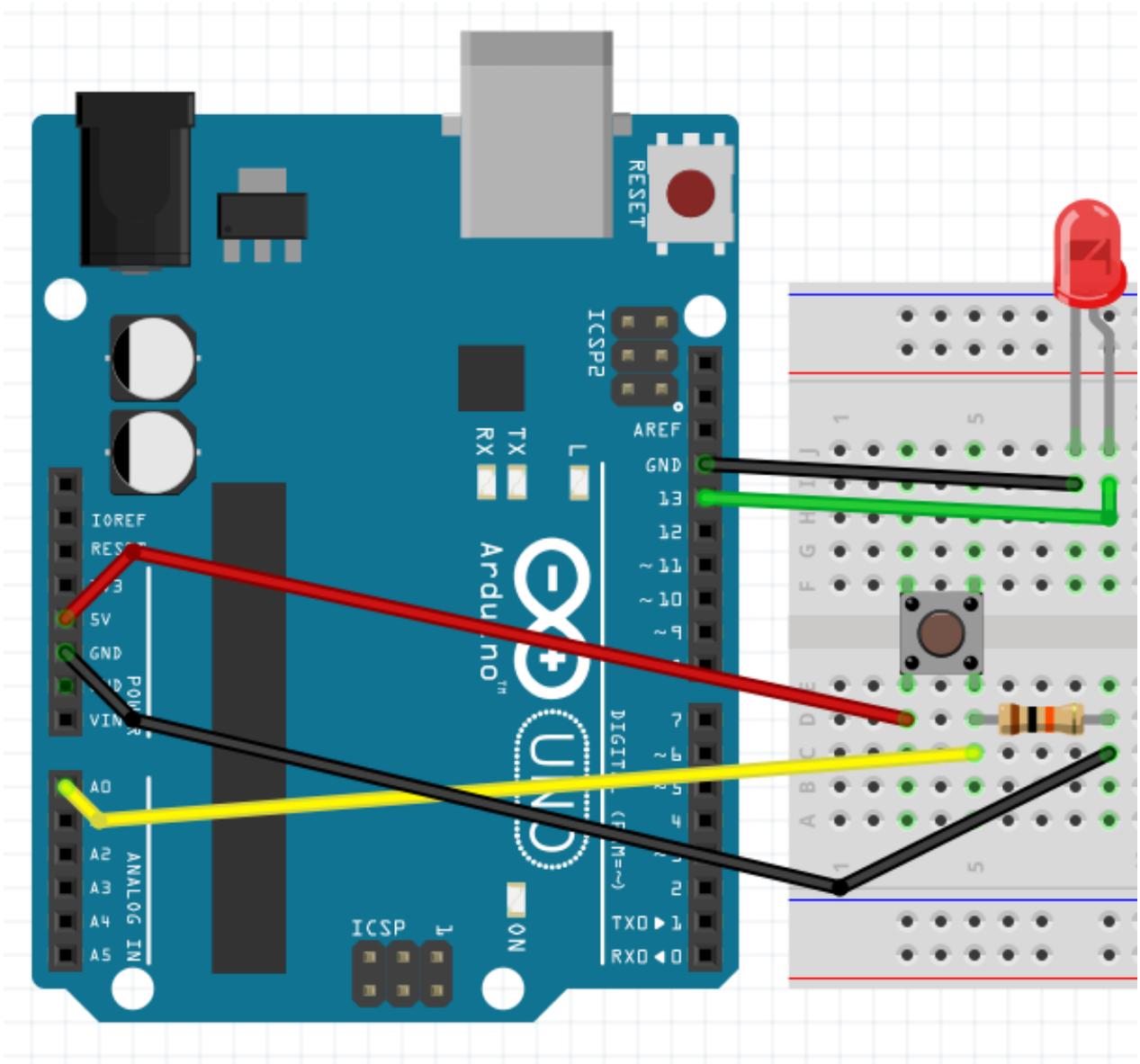
A temperature sensor



A 10k ohm resistor

# Using basic inputs – buttons and potentiometers

## Wiring up our button and LED



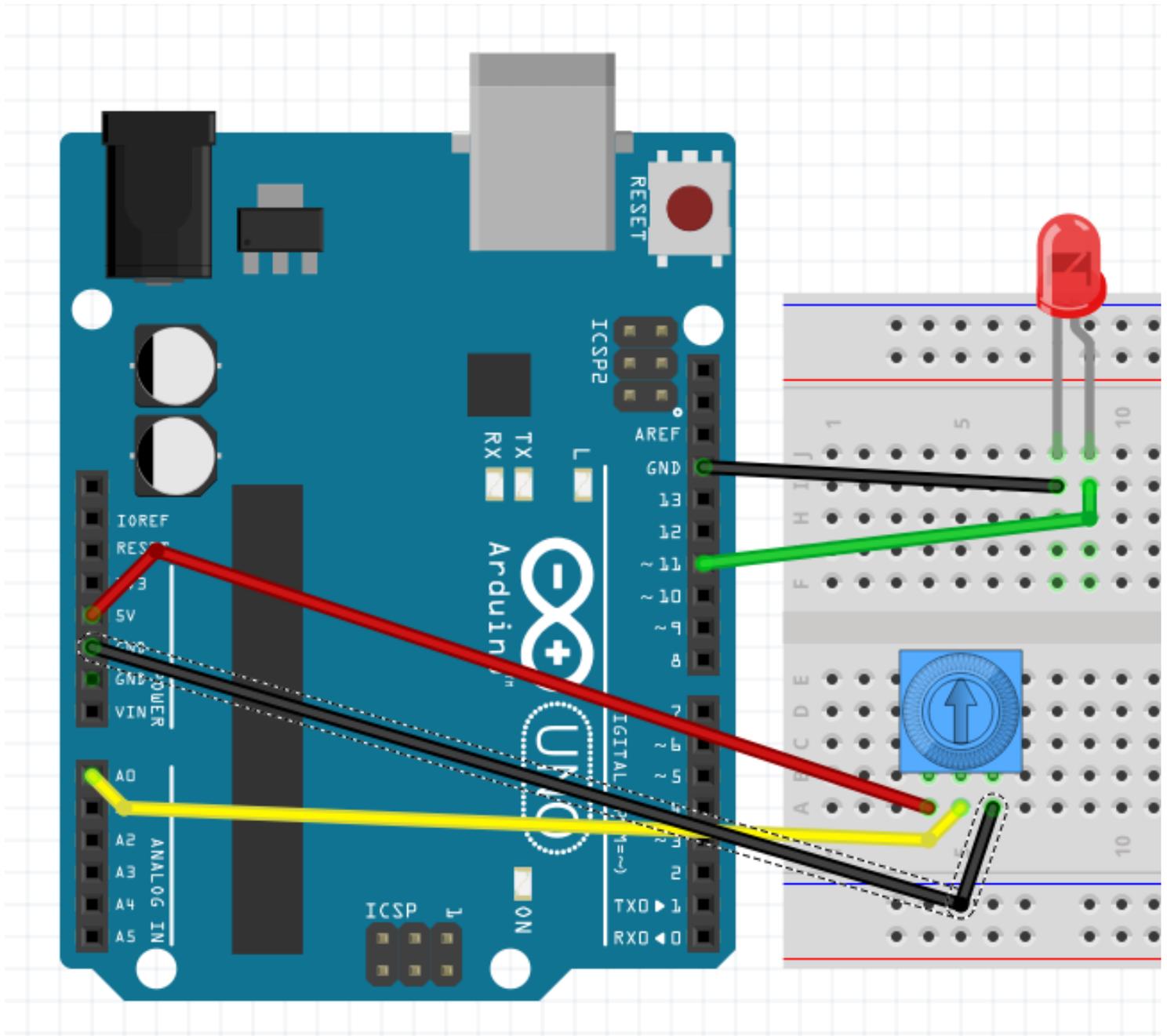
Wiring diagram for a button and an LED

## Coding button-led.js

```
The button has been pressed!  
The button has been released!  
The button has been pressed!  
The button has been released!  
The button has been pressed!  
The button has been released!
```

The output from led-button.js

## Wiring the potentiometer and the LED



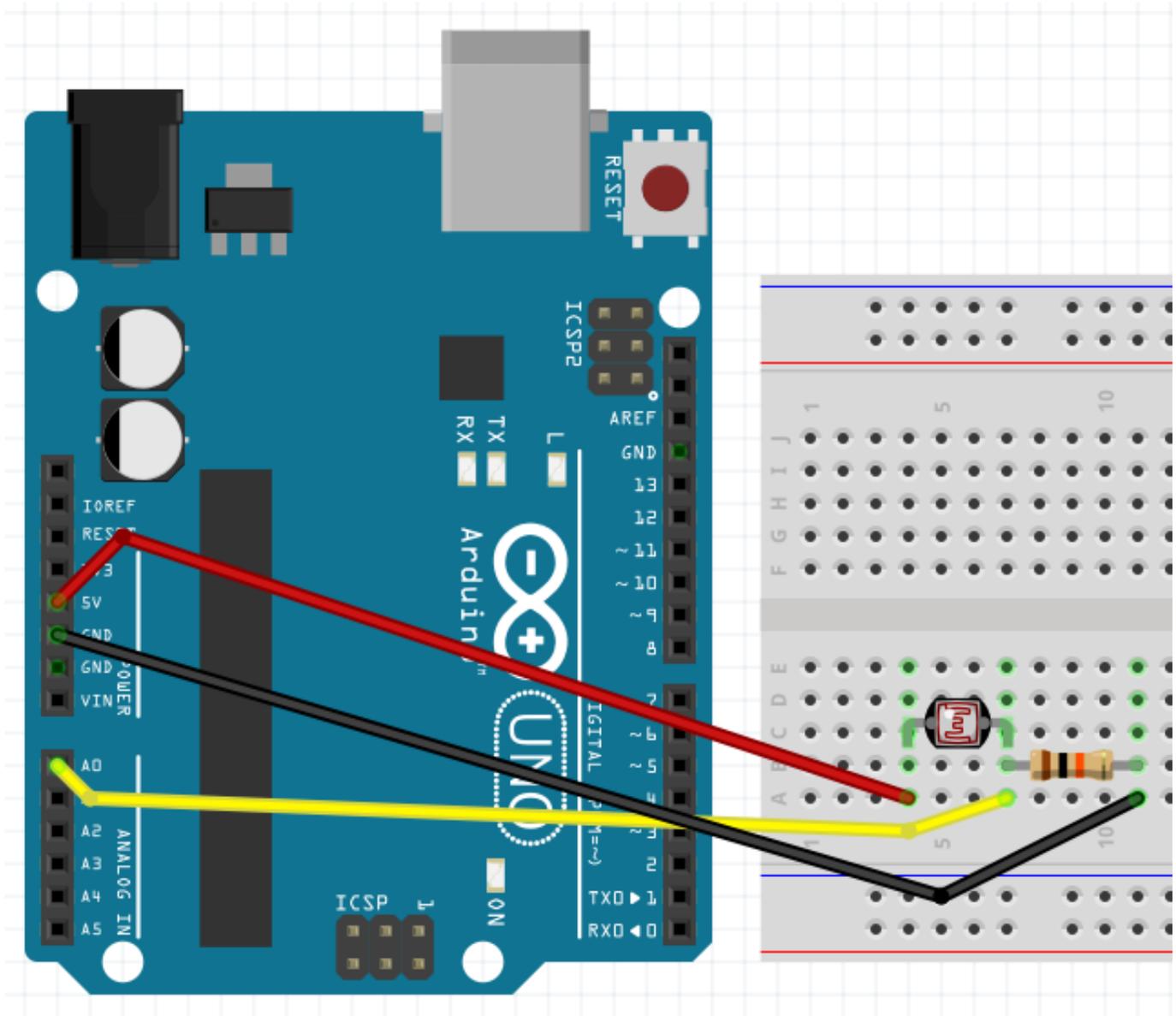
Wiring diagram for a dimmer switch  
Coding our dimmer switch

```
The scaled potentiometer value is: 74.37396244634874
The raw potentiometer value is: 301.6830960523803
The scaled potentiometer value is: 74.62085480079986
The raw potentiometer value is: 302.8240005893167
The scaled potentiometer value is: 74.09539178735577
The raw potentiometer value is: 300.57111365231685
The scaled potentiometer value is: 74.63298110873438
The raw potentiometer value is: 301.508853206411
The scaled potentiometer value is: 74.48949674842879
The raw potentiometer value is: 301.591794724809
The scaled potentiometer value is: 74.6783123346977
The raw potentiometer value is: 302.1139590013772
```

The output from dimmer-switch.js

# Using sensors – Light and Temperature

## Wiring up our photocell



The wiring diagram for the photocell

## Coding our photocell example

barcli



```
324.4568812660873
27.077454514801502
35.51635257899761
86.42502119764686
356.6767932847142
275.8839665912092
34.698570612818
122.3692636936903
353.91647312790155
274.5486353524029
120.49119230359793
181.22242279350758
257.2020875290036
193.85588942095637
361.46779088303447
256.5282253548503
375.8651088923216
172.19067979604006
64.6804491057992
326.5608750283718
383.0669150687754
196.57427808269858
27.206077240407467
341.97616344317794
96.23732138425112
181.62367893382907
11.388890445232391
53.237830847501755
363.83112063631415
290.77909123152494
280.3087676875293
75.24486519396305
214.3381329253316
203.87549018487334
```

An output in the days before barcli

photocell: |

| 563

A barcli graph in the console

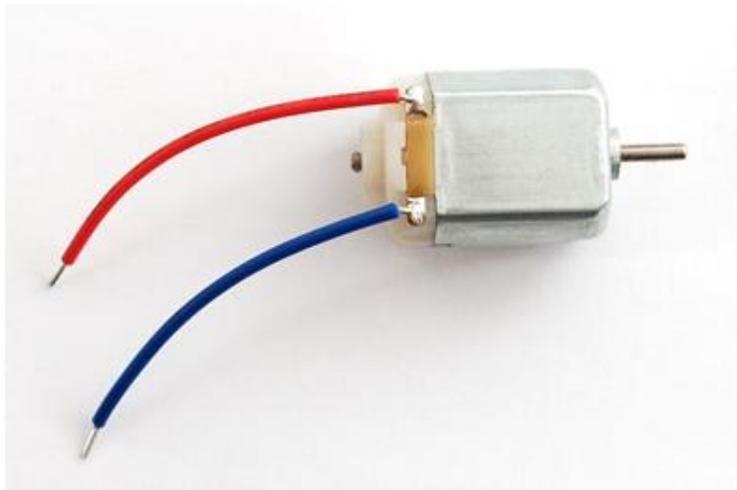


# 6

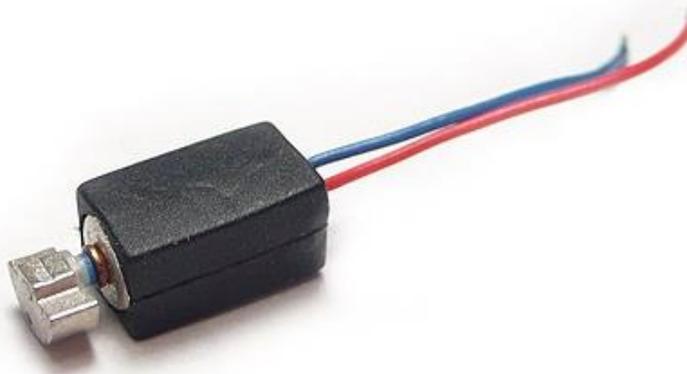
## Moving Your Bot

### The different kinds of servos and motors

#### Types of motors



A standard DC hobby motor



A vibration motor



A stepper motor

## Types of servos

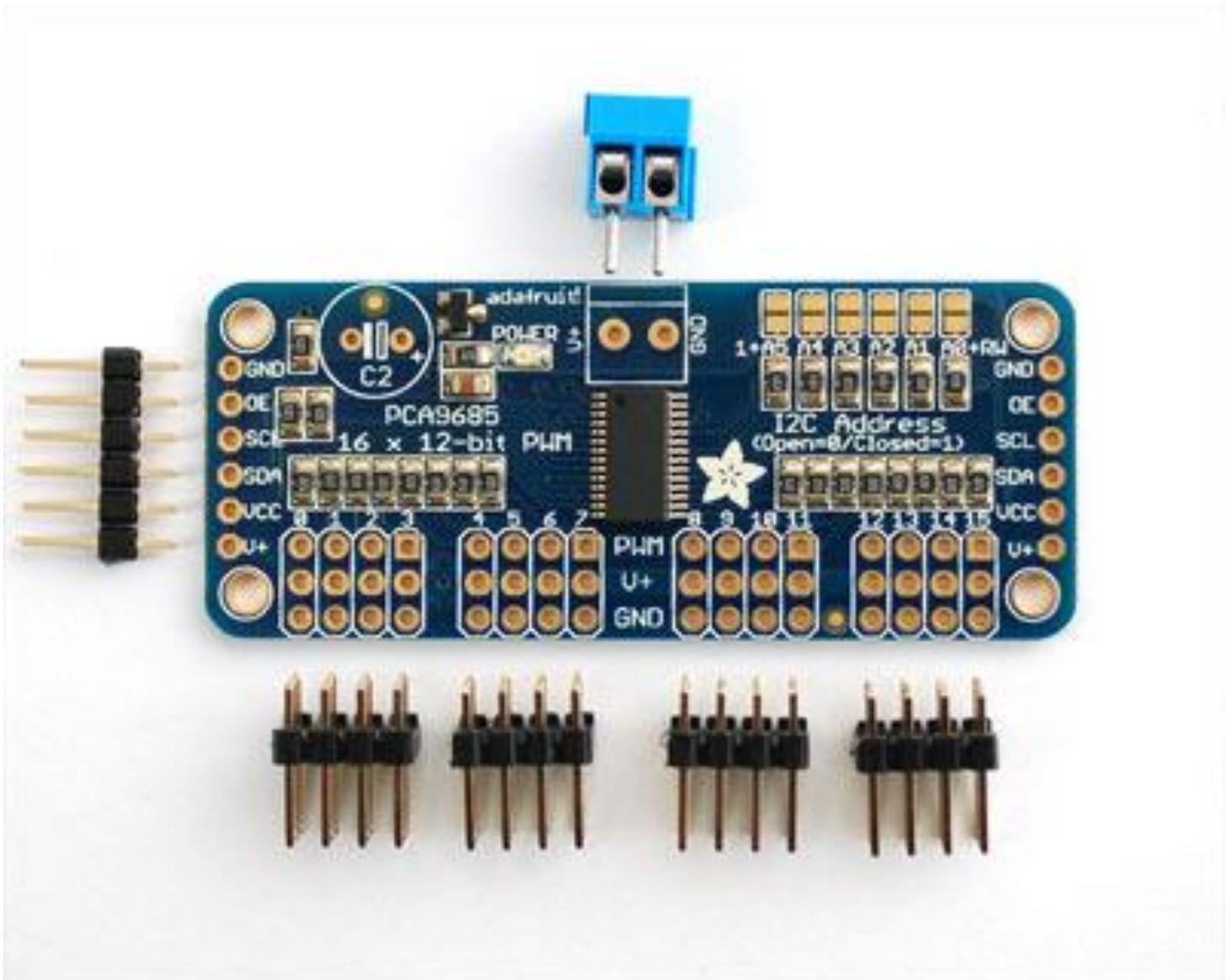


A standard servo



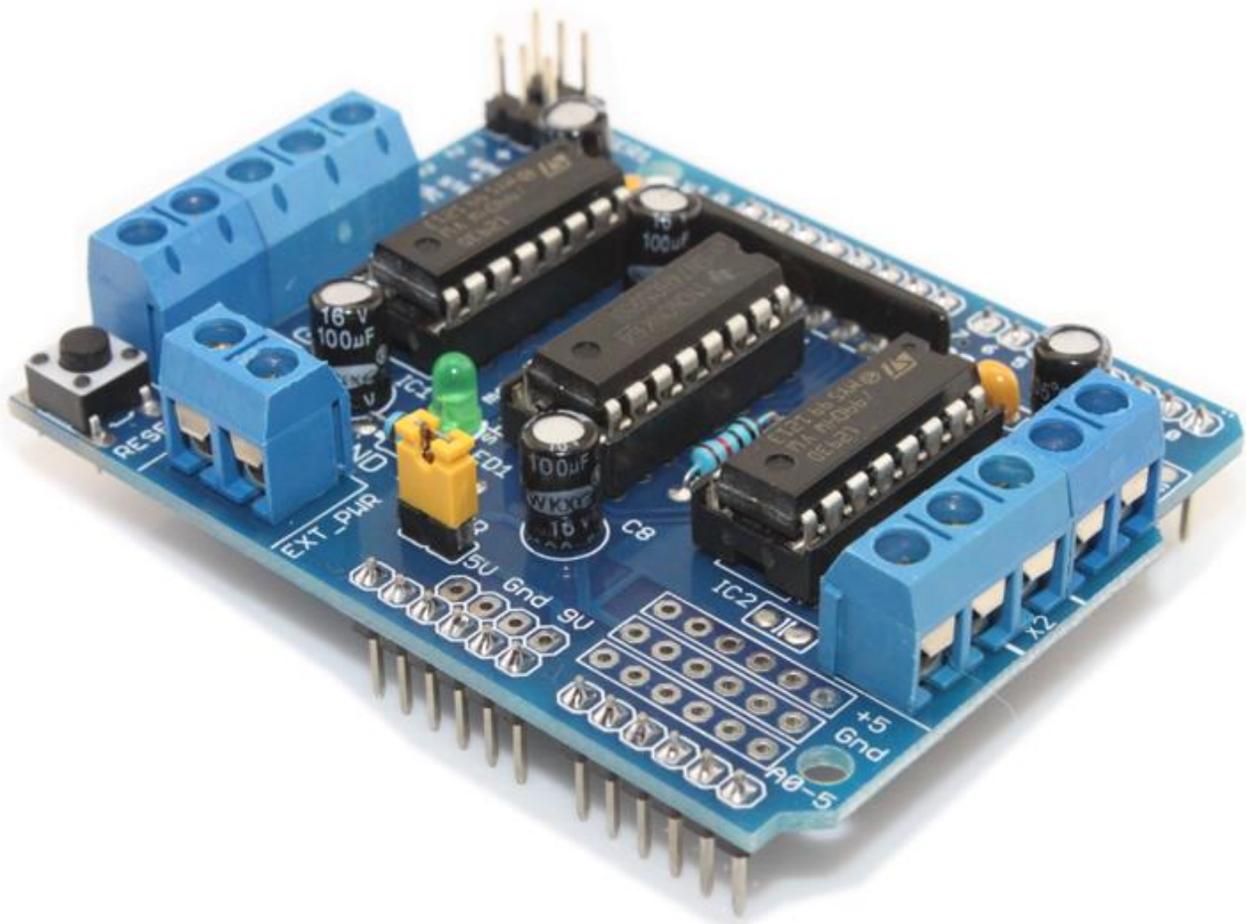
A continuous servo

## Servo and motor controllers



An example of a servo controller with an I2C interface

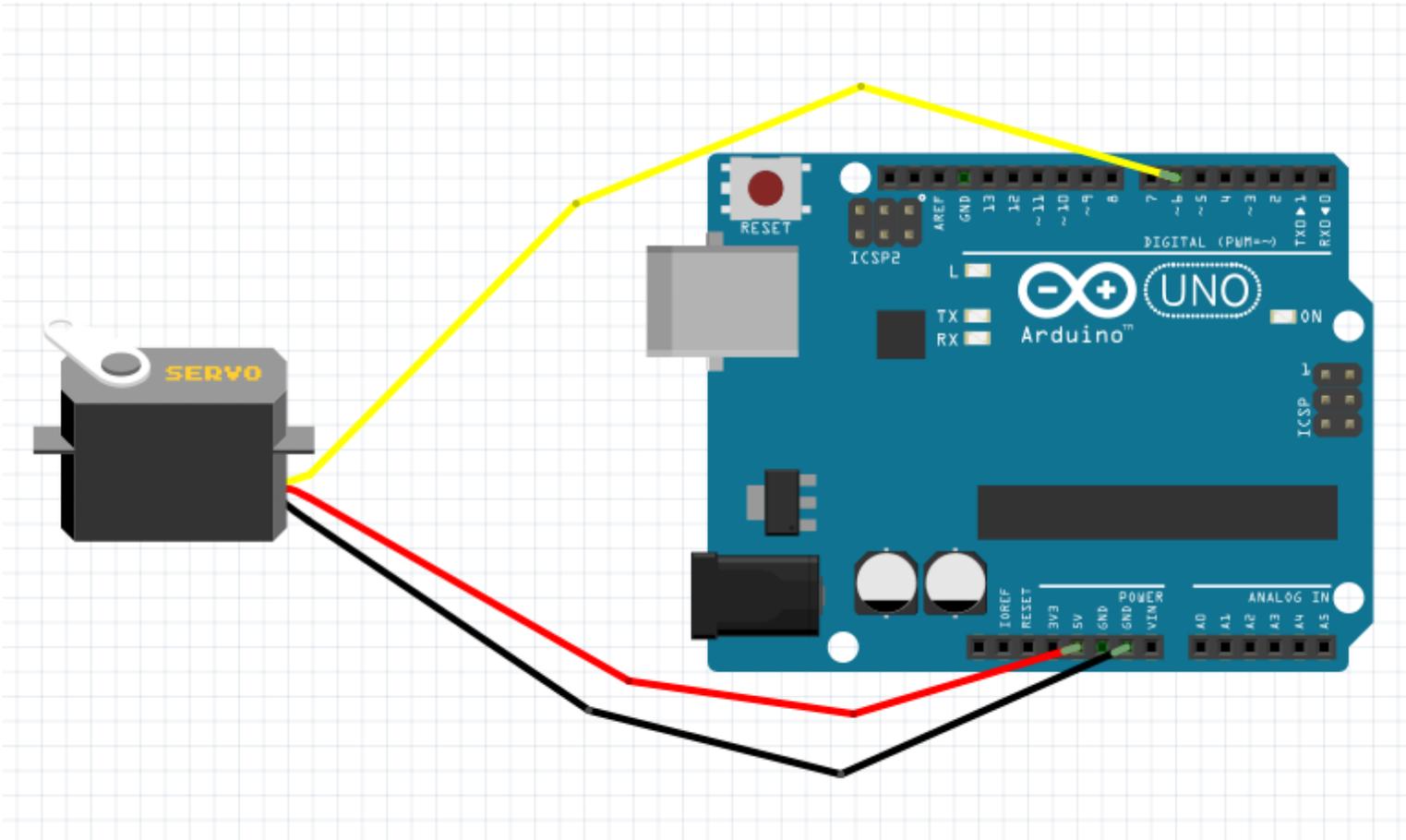
## Motor and servo shields



An example of a motor shield

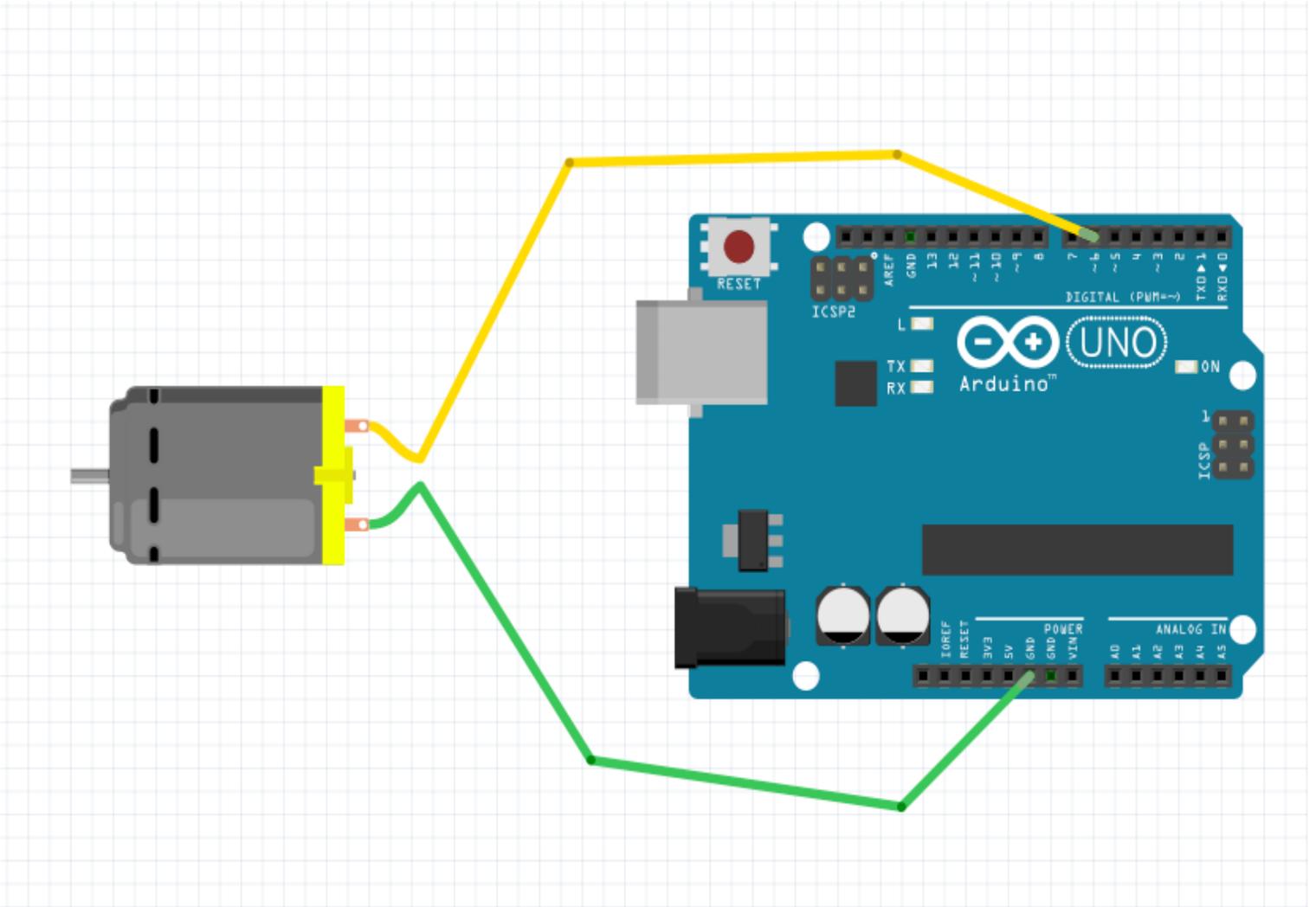
# Wiring up servos and motors

## Wiring up servos



A servo wiring diagram

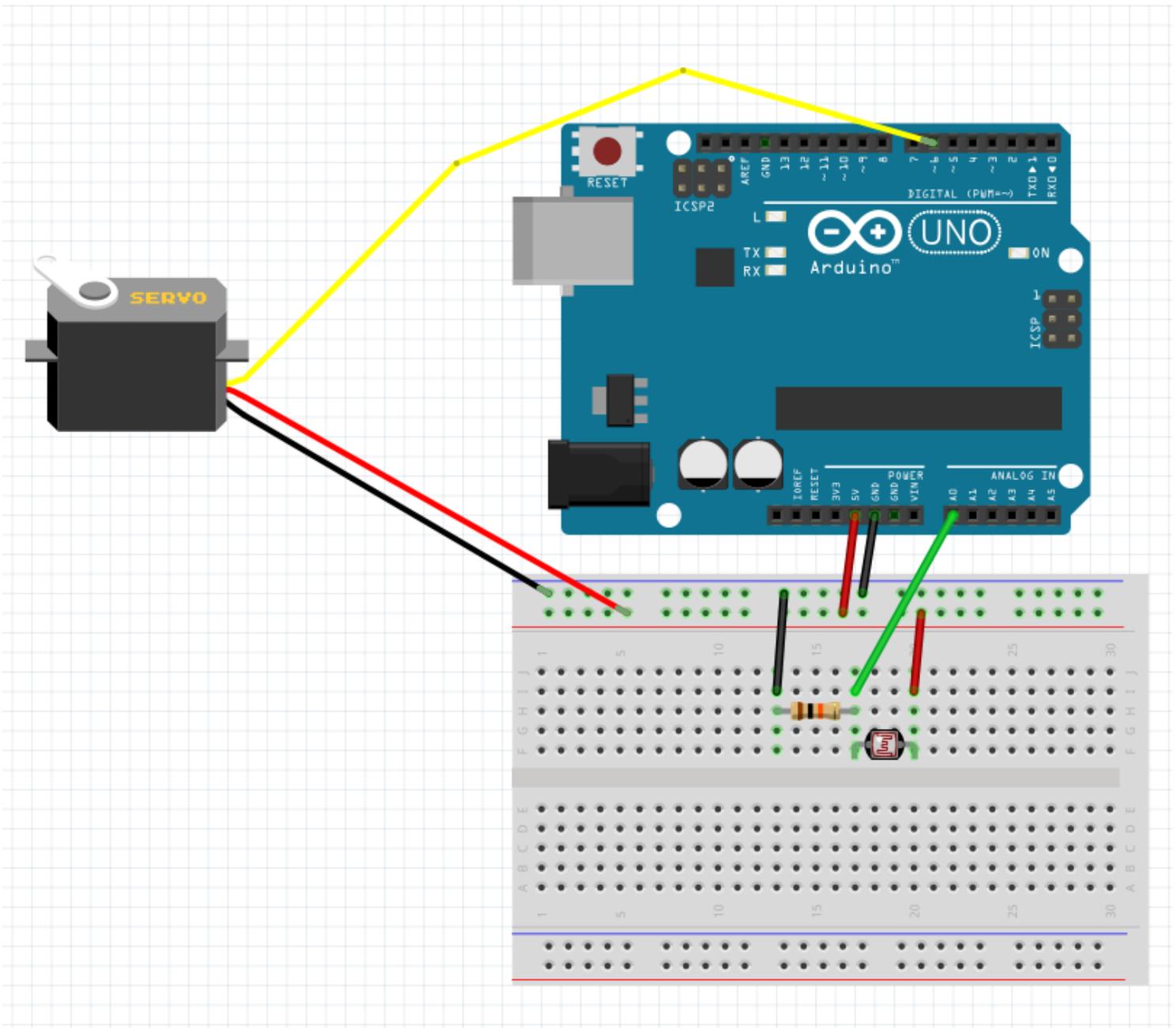
## Wiring up motors



A motor wiring diagram

# Creating a project with a servo and a sensor

## Exploring the servo API with the REPL



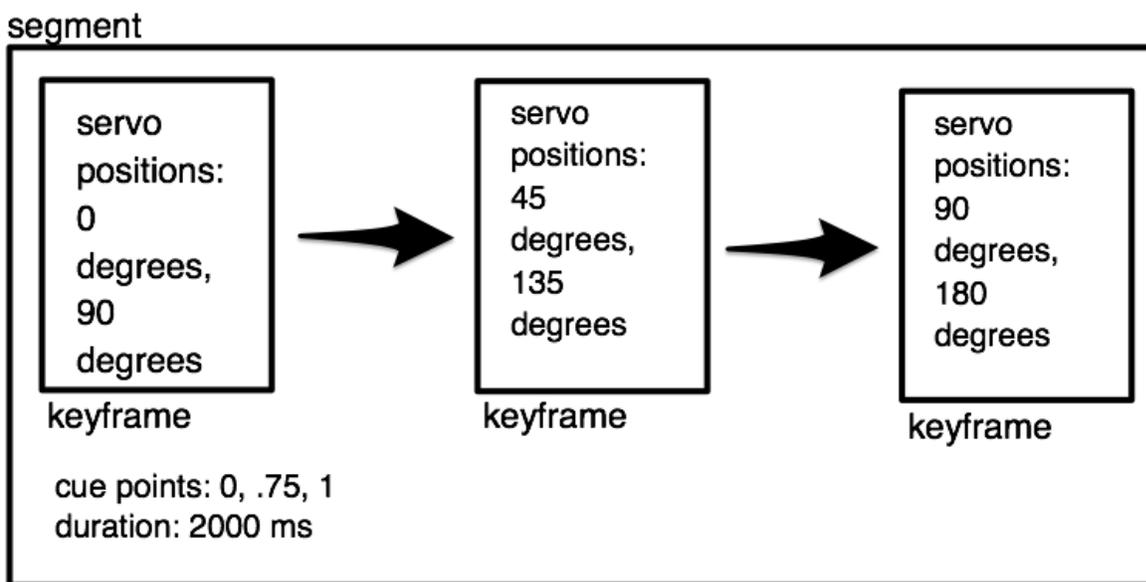
A servo and photoresistor wiring diagram

# 7

## Advanced Movement with Animation Library

### Looking at the Animation API

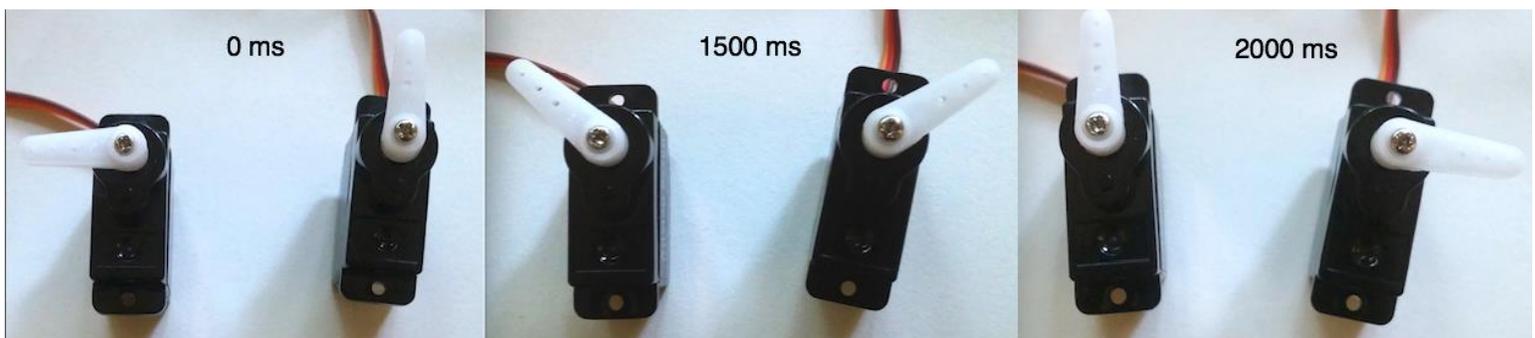
### Learning the terminology



A graphical representation of an Animation segment

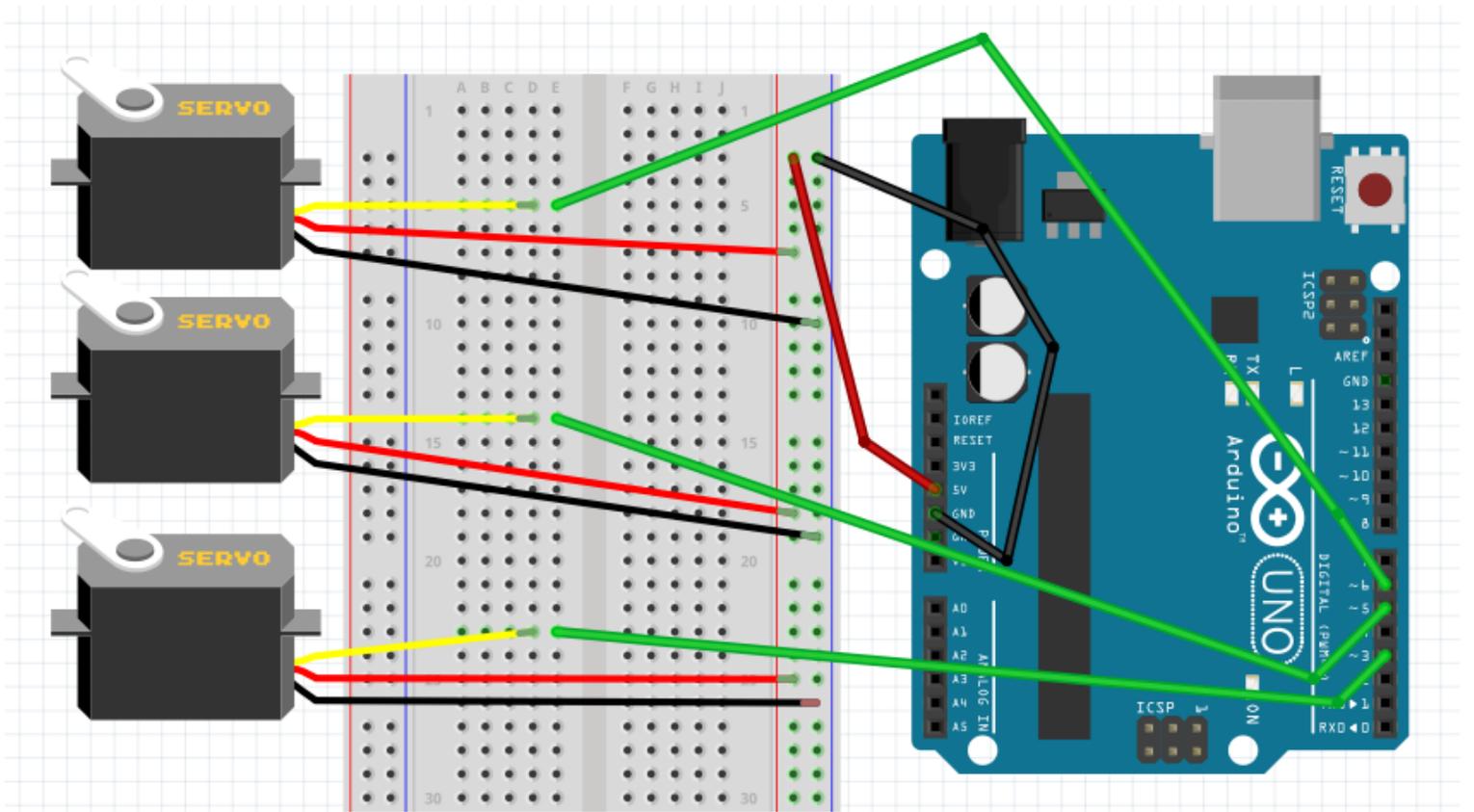
$$\text{time of cue point} = \text{cue point value} * \text{duration}$$

The formula for a cue point time



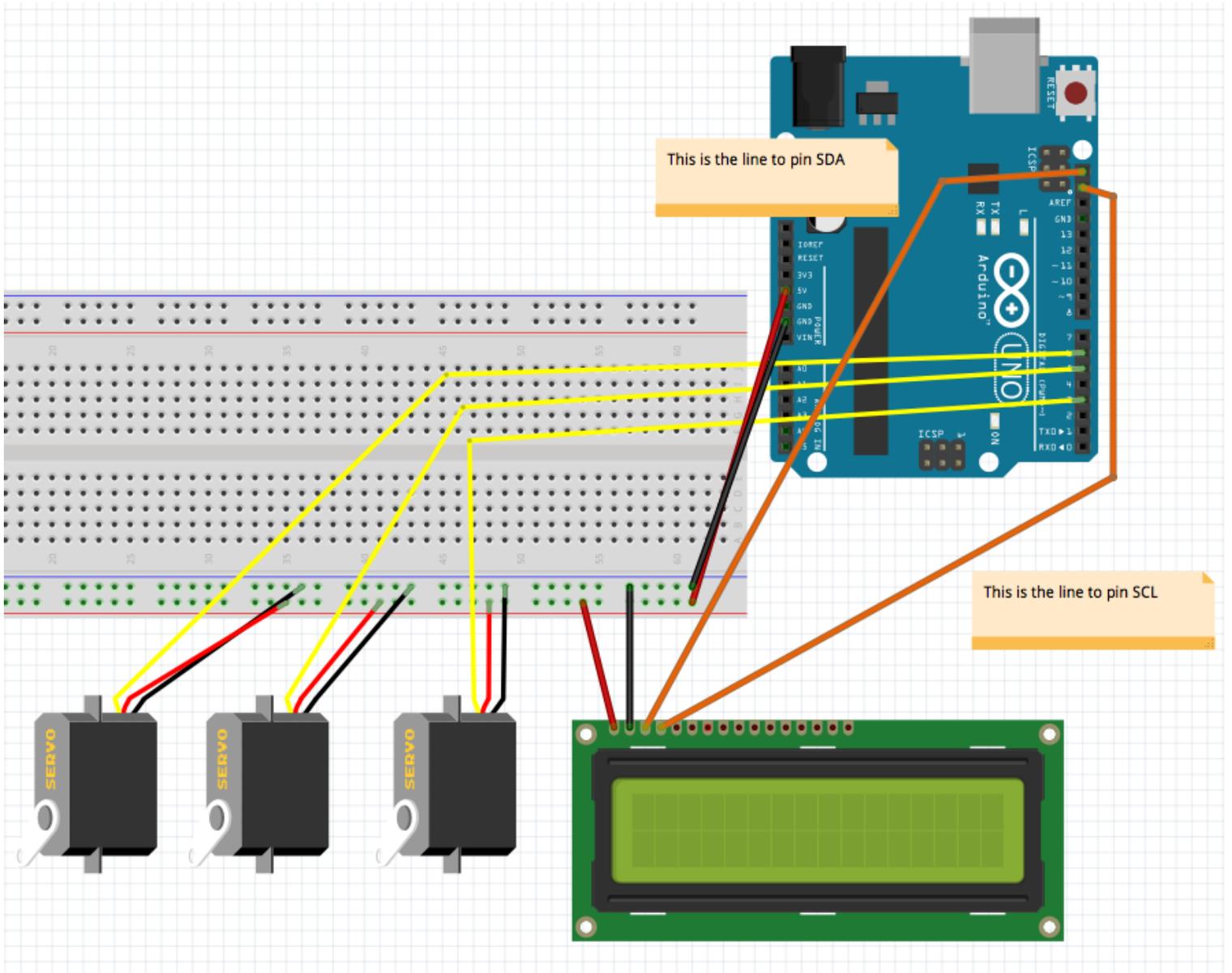
The pictures of the servo movement from the segment

## Project – wiring up three servos and creating an array

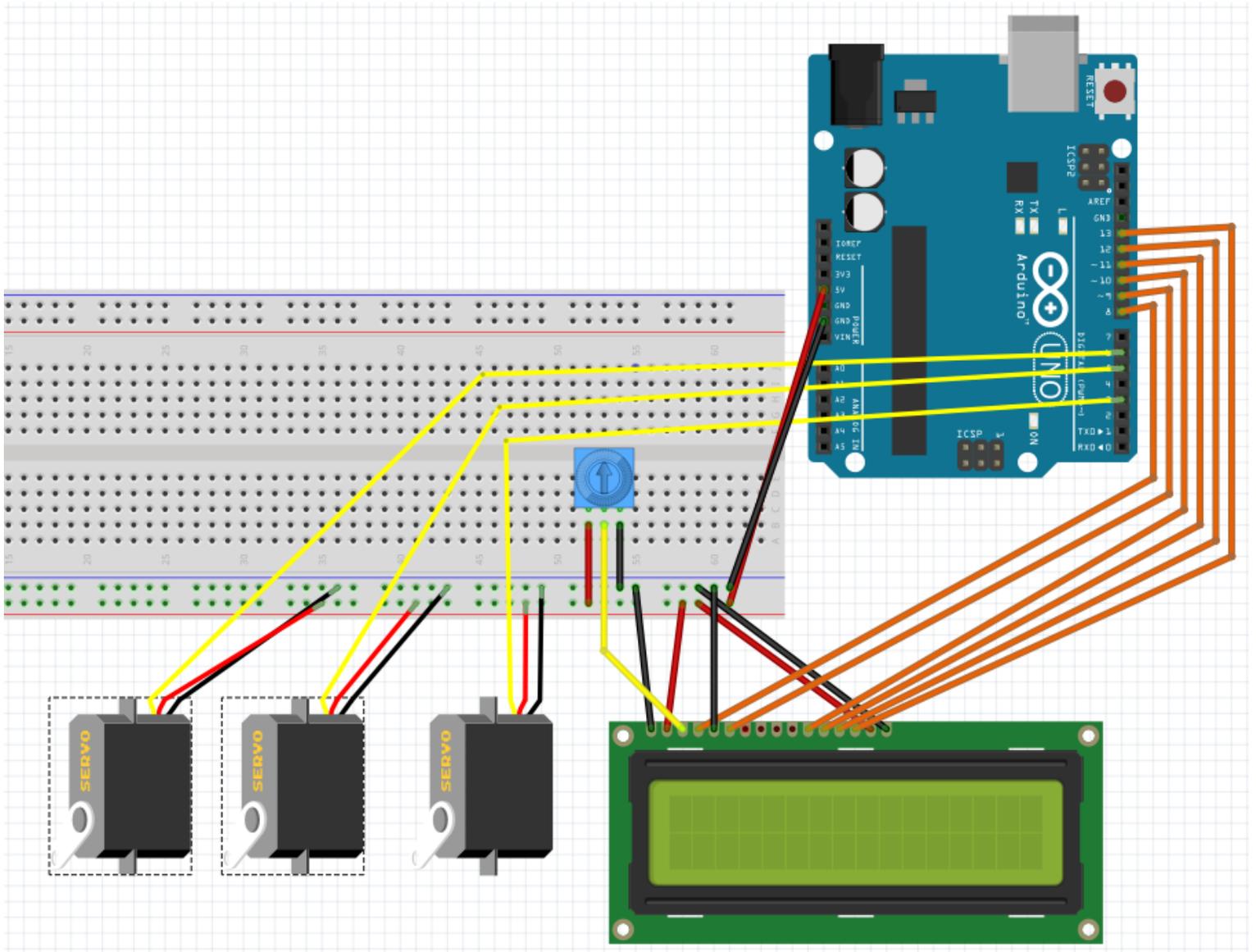


# Animation events

## Building a servo array with informative LCD readout



Events project wiring diagram—i2c LCD



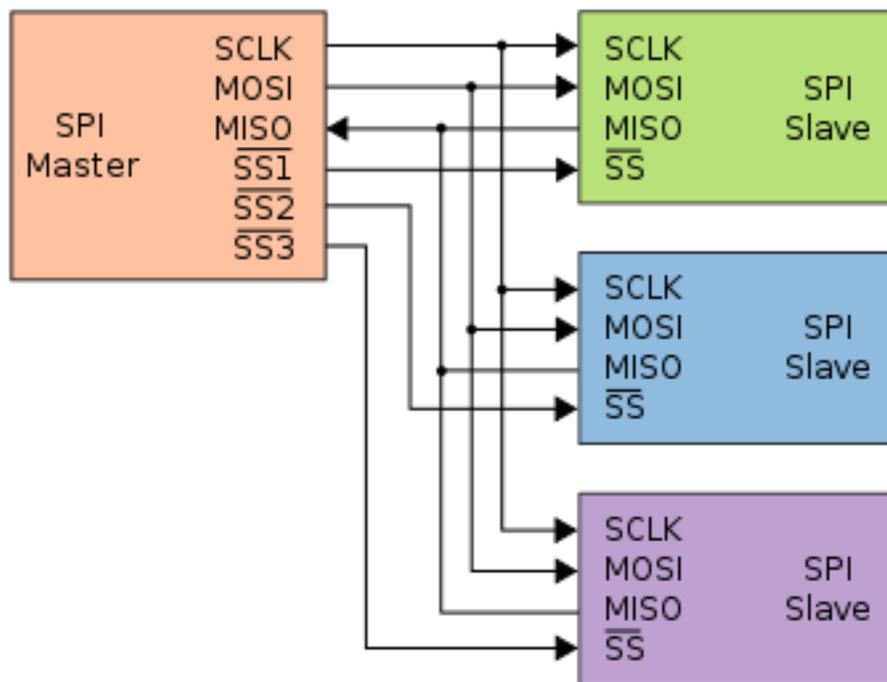
Events project wiring diagram—Standard LCD

# 8

## Advanced Components –SPI, I2C, and Other Devices

### Exploring SPI (Serial Peripheral Interface) devices

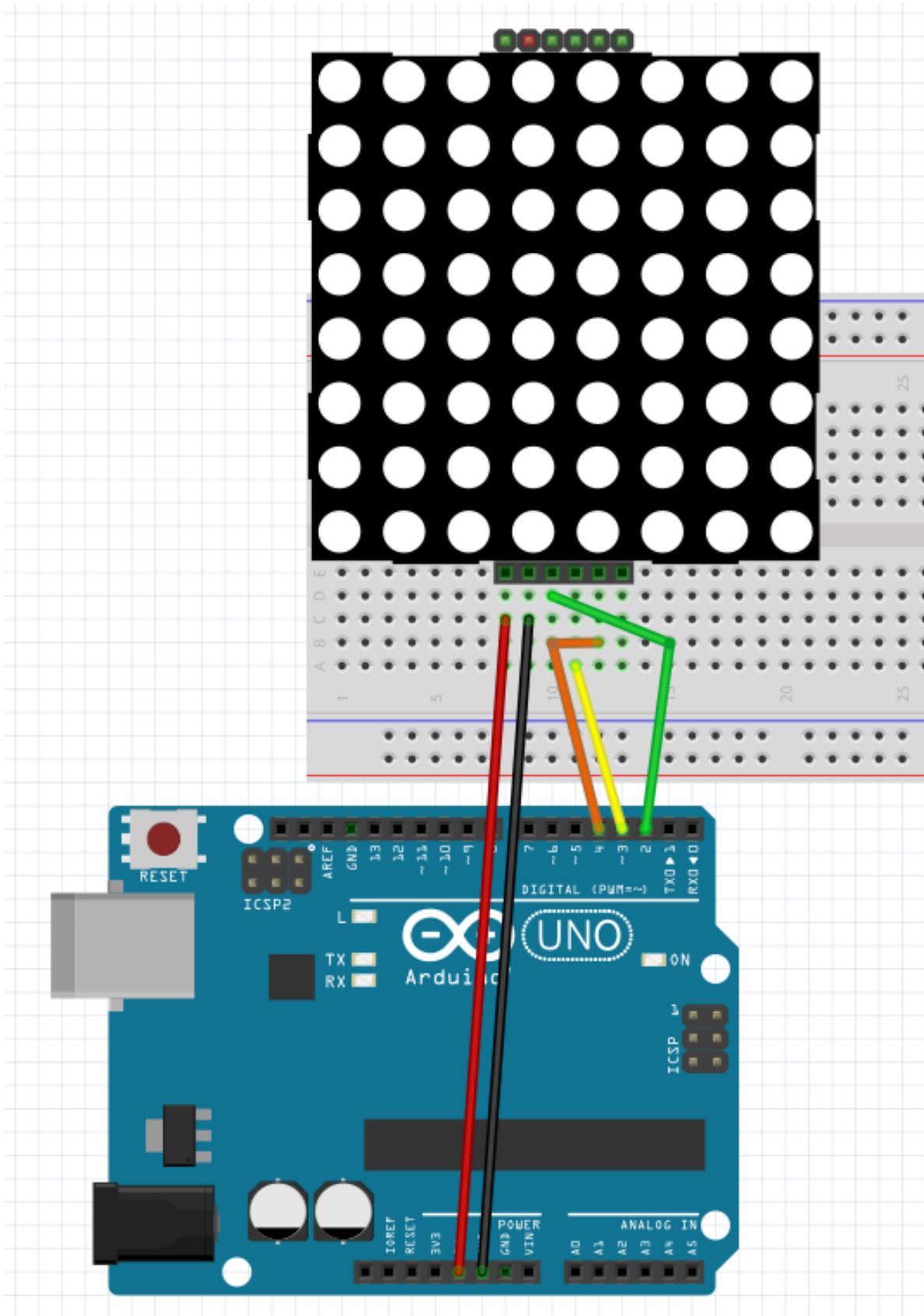
#### How SPI works



The SPI explained—Image credit [https://en.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface\\_Bus](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus)

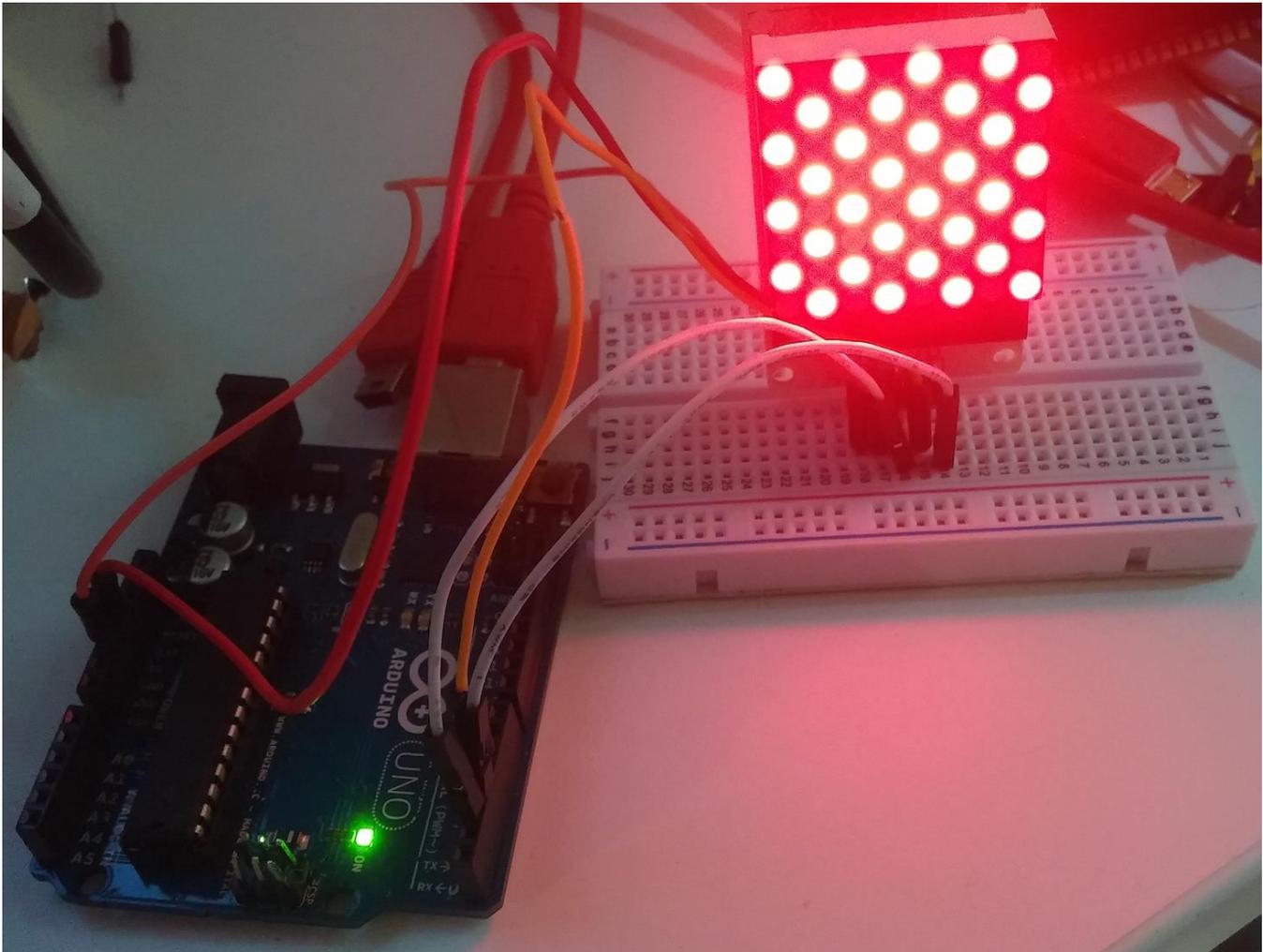
# Building with an SPI device – an LED matrix

## The build



Wiring diagram for the LED matrix

The API  
The Code

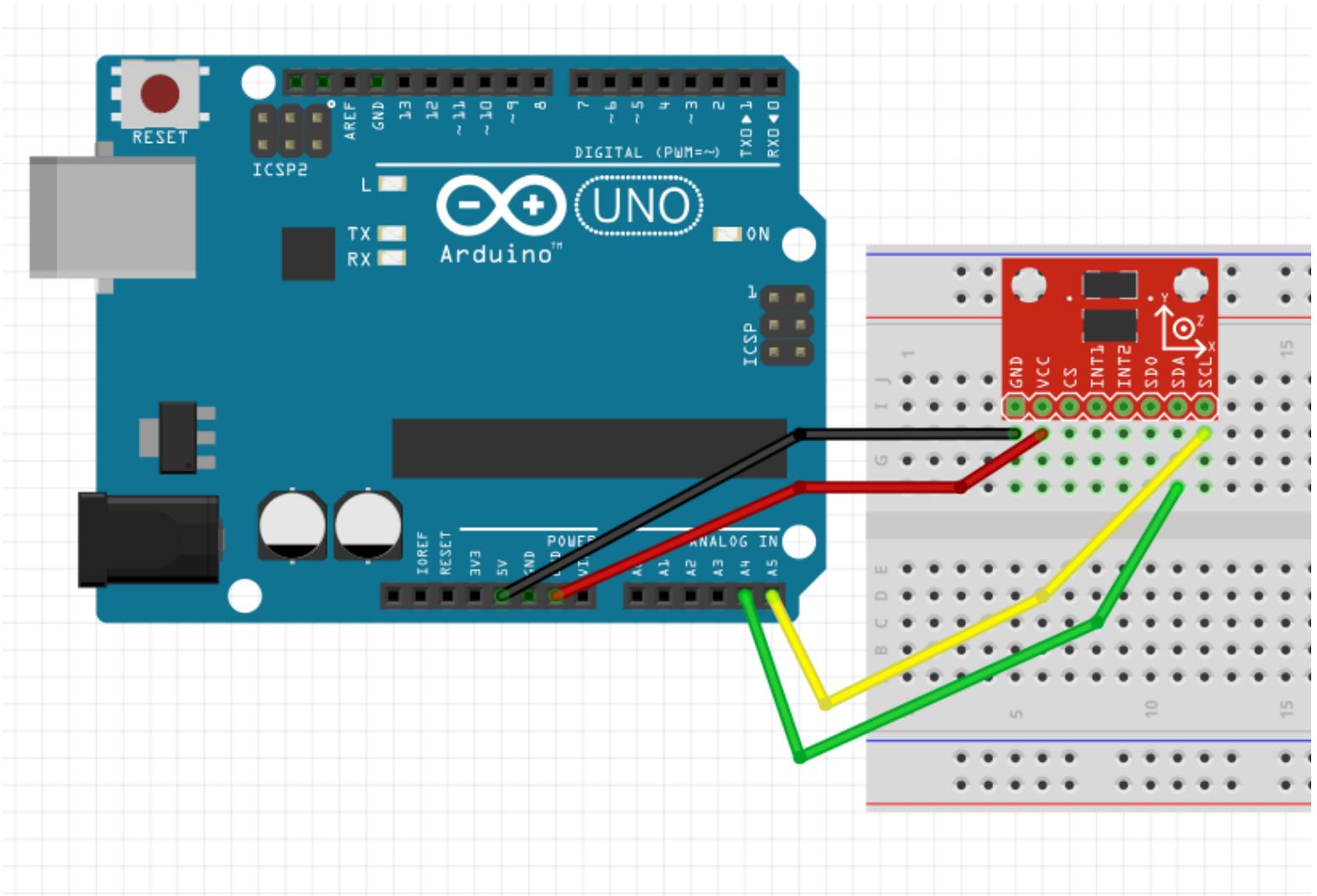


A matrix example with checkbox character

# Exploring I2C devices

## Building with an I2C device – Accelerometer

### Wiring up our accelerometer



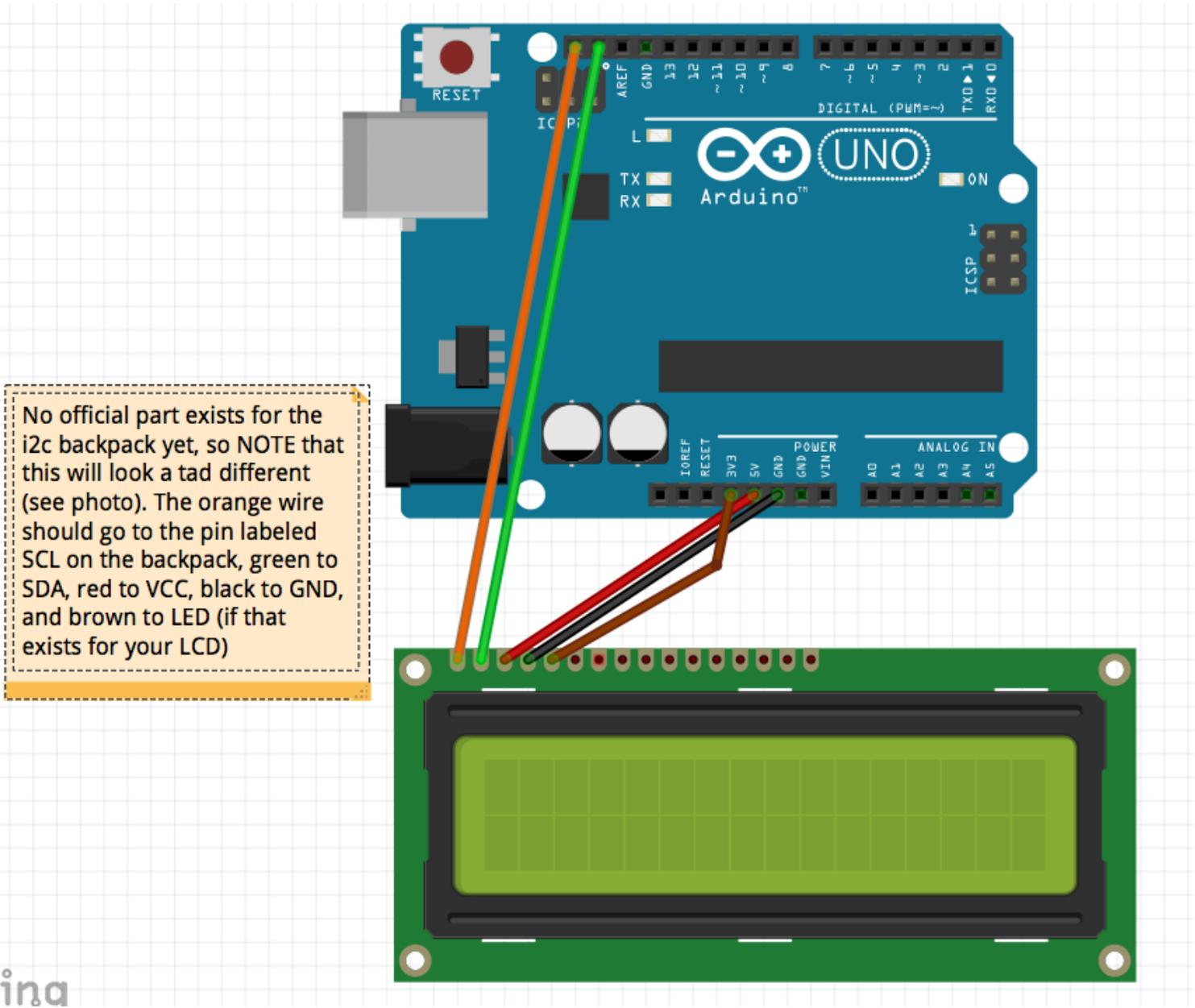
Accelerometer wiring for NON-R3 Arduino Uno



# External Devices

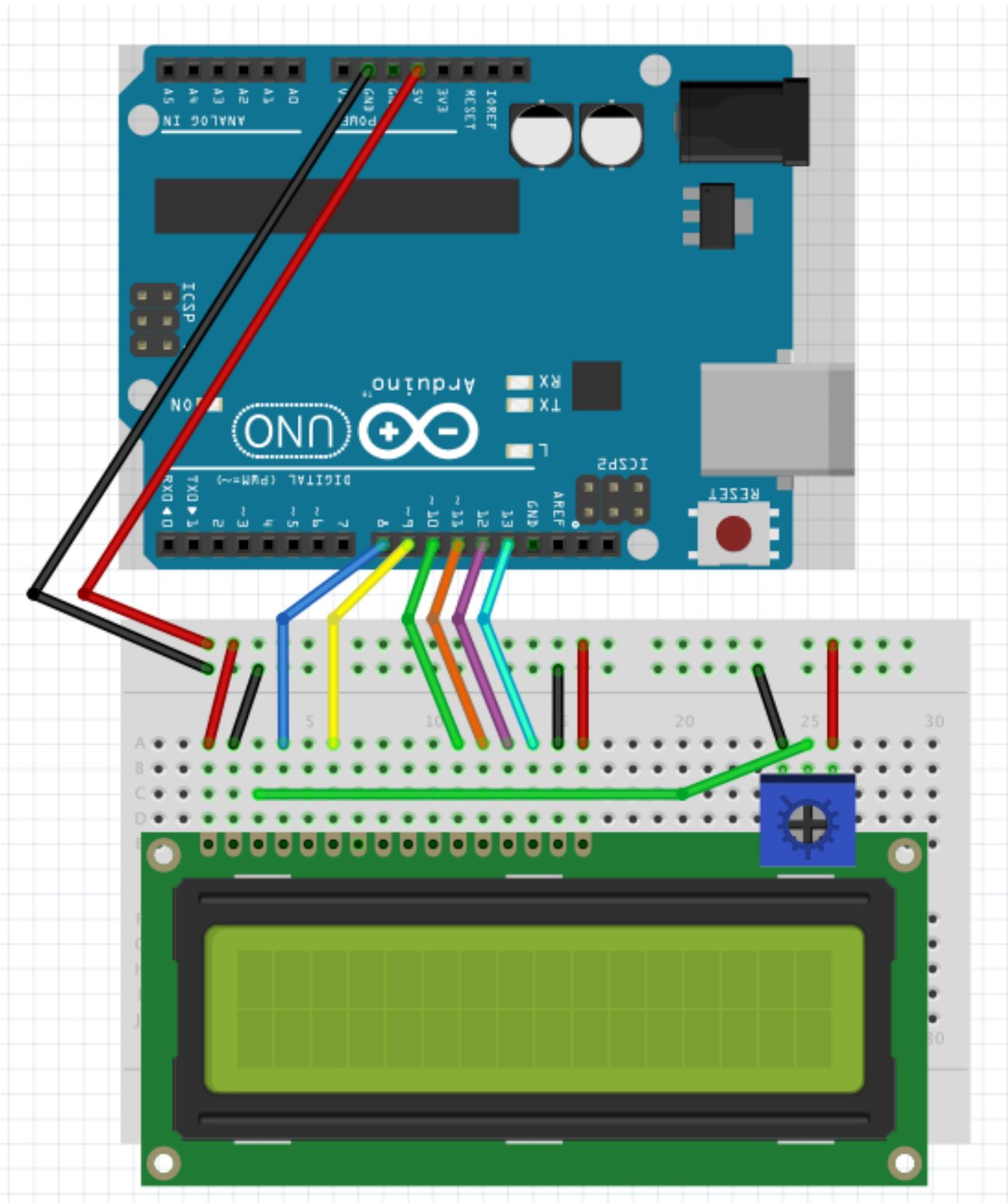
## Build – a USB gamepad

### The hardware



tzina

Wiring diagram—I2C LCD



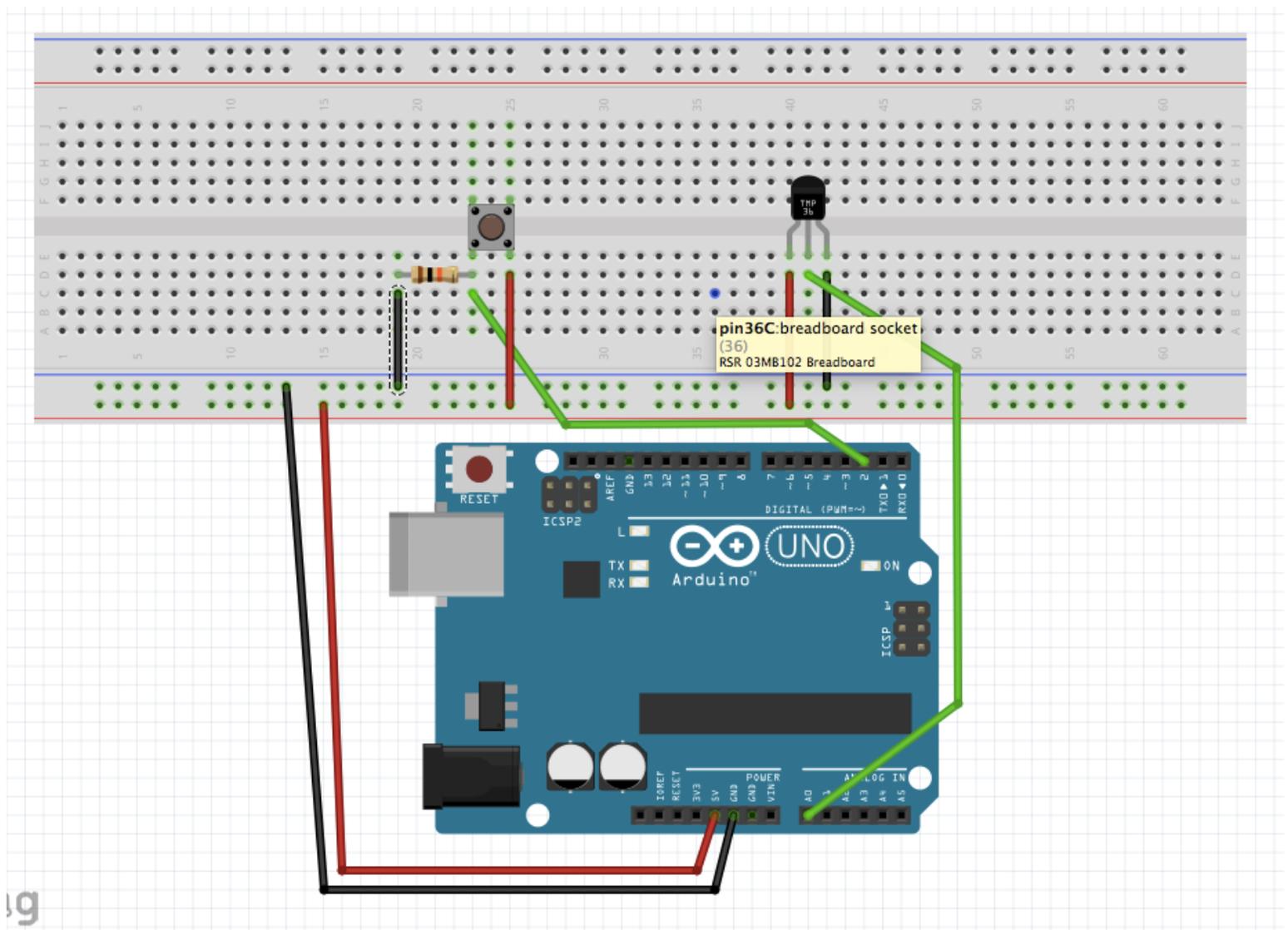
Wiring diagram—regular LCD

# 9

## Connecting NodeBots to the World, and Where to Go Next

### Connecting NodeBots to the Web

### Building the WeatherBot



The Arduino WeatherBot schematic

