

ElectroTutor:

Test-Driven Physical Computing Tutorials

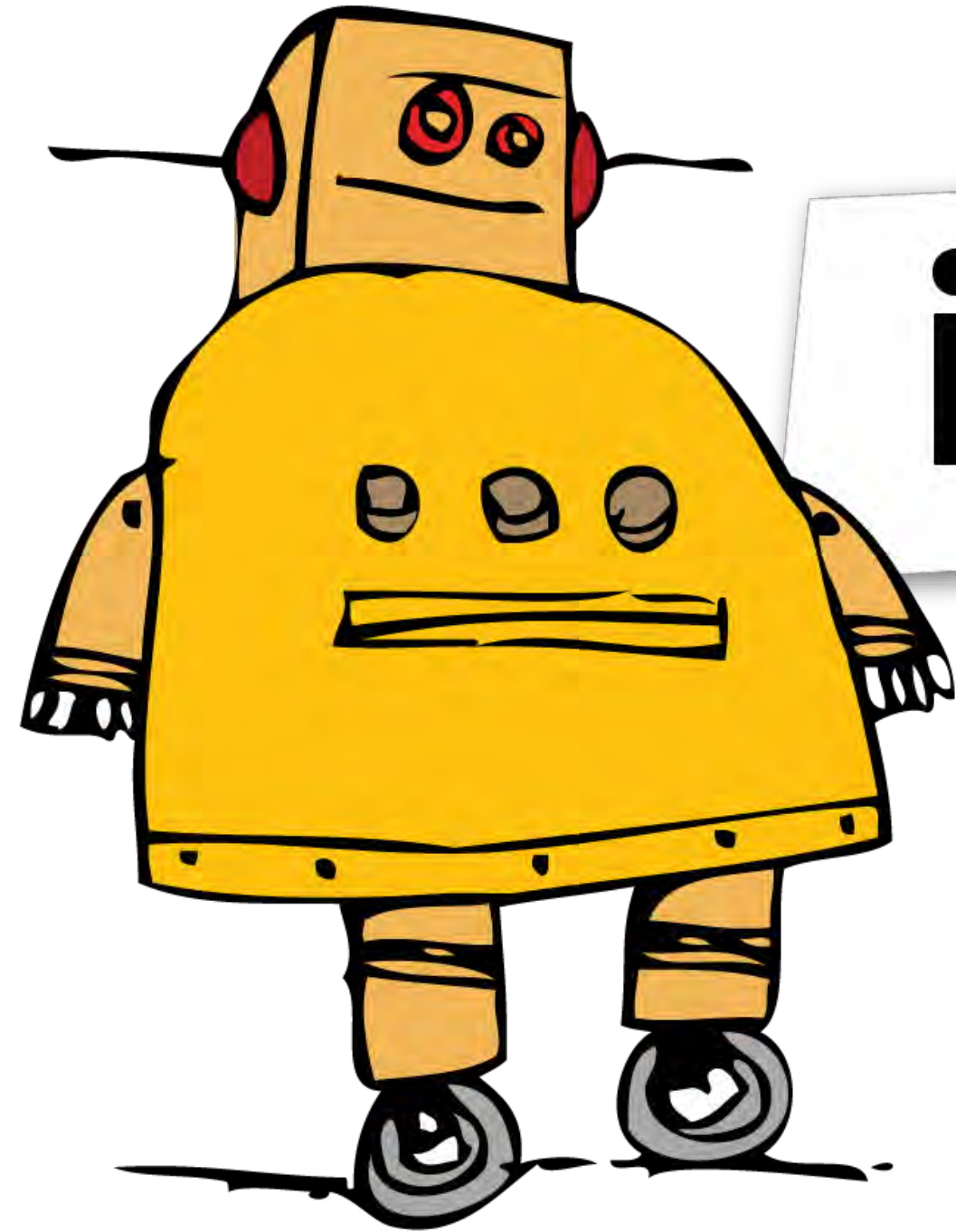
Jeremy Warner, Ben Lafreniere,
George Fitzmaurice, Tovi Grossman



A close-up photograph of a person's hands working on an electronics project. The person is wearing a silver ring on their left hand and a grey wristband. They are using a soldering iron to connect wires to a breadboard. The breadboard contains a microcontroller board with a green LED that is lit. Several colored wires (yellow, red, black) are connected to the breadboard. In the background, there is a computer keyboard and a monitor on a desk. The text "electronics tutorials" is overlaid in the center of the image.

**electronics
tutorials**

motivation



instructables

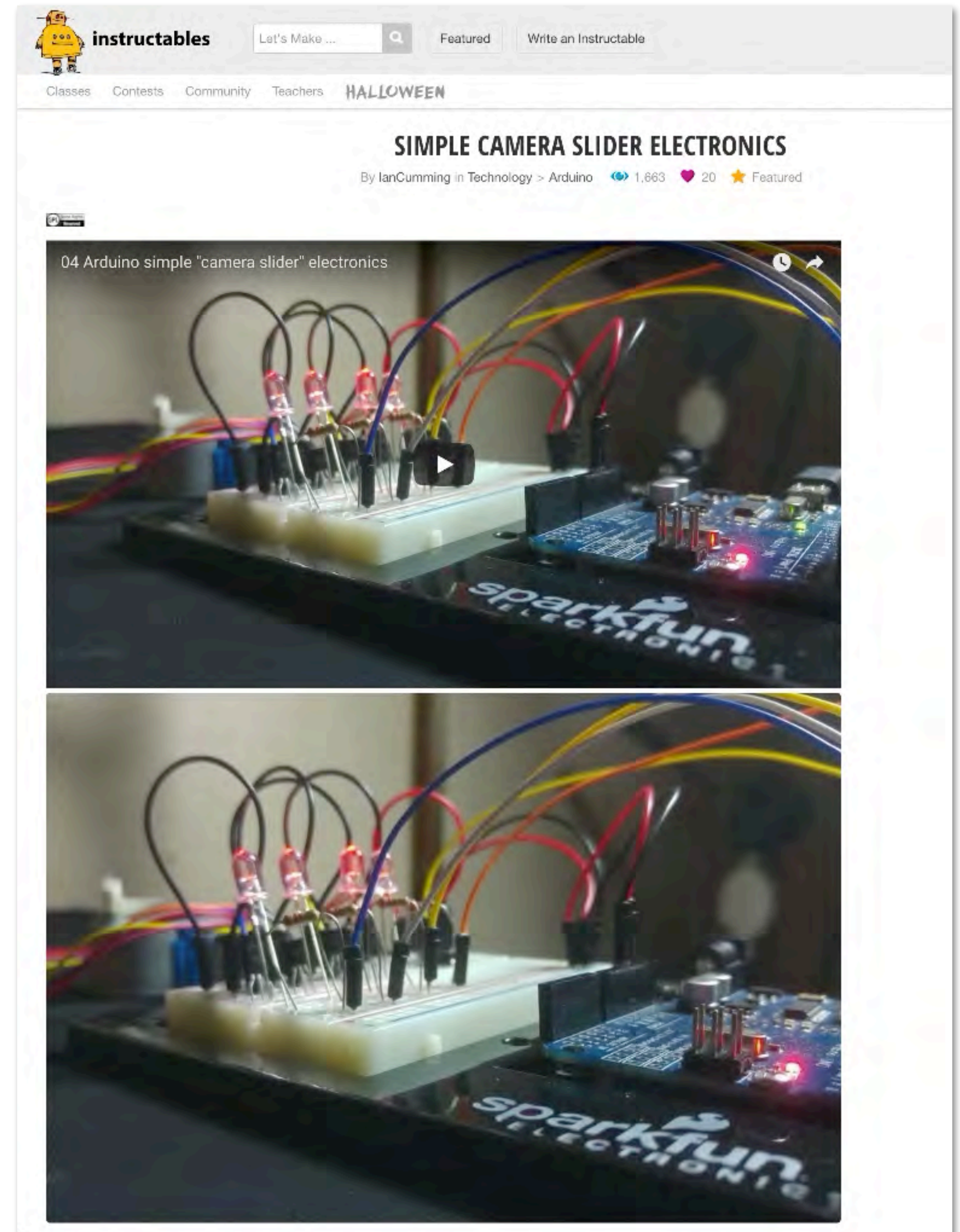


The screenshot shows the Arduino Project Hub interface. At the top, there is the Arduino logo and 'PROJECT HUB' text, along with 'ADD PROJECT' and 'SEARCH PROJECTS' buttons. Below this are filter menus for 'All products', 'All categories', 'Trending', 'Any difficulty', and 'Any type'. The main content area displays three project cards:

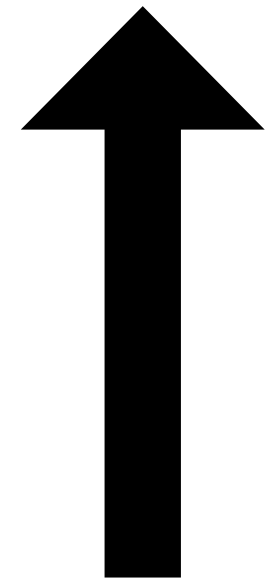
- Simplest Way for Voice Recognition Project Using...**
Project tutorial by Jalal_Mansoori
3,855 VIEWS 22 COMMENTS 32 RESPECTS
- Easy Motion and Gesture Detection by PIR Sensor &**
Project tutorial by ElectroPeak
11,704 VIEWS 4 COMMENTS 88 RESPECTS
- Jrobot Self Drive Powered by TensorFlow Lite**
Project showcase by joechen
5,516 VIEWS 11 COMMENTS 47 RESPECTS

motivation

What if we could embed checkpoints to improve the tutorial experience?



The image shows a screenshot of the Instructables website. At the top, the Instructables logo is visible on the left, and a search bar with the text "Let's Make ..." is in the center. To the right of the search bar are buttons for "Featured" and "Write an Instructable". Below the search bar, there are navigation links for "Classes", "Contests", "Community", "Teachers", and "HALLOWEEN". The main content area features a tutorial titled "SIMPLE CAMERA SLIDER ELECTRONICS" by IanCumming in Technology > Arduino. The tutorial has 1,663 views, 20 likes, and is marked as "Featured". Below the title is a video player showing a close-up of an Arduino Uno board connected to a breadboard. The breadboard contains several red LEDs and various electronic components. The video player has a play button in the center and a video title "04 Arduino simple 'camera slider' electronics". The video player is set against a dark background with the "sparkfun" logo visible on the breadboard.



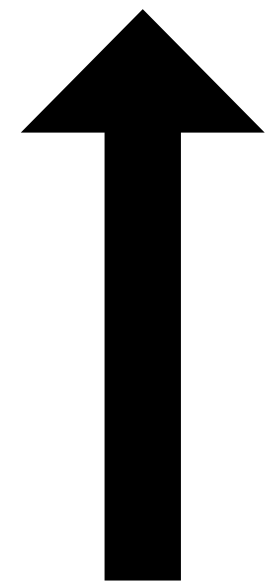
progress



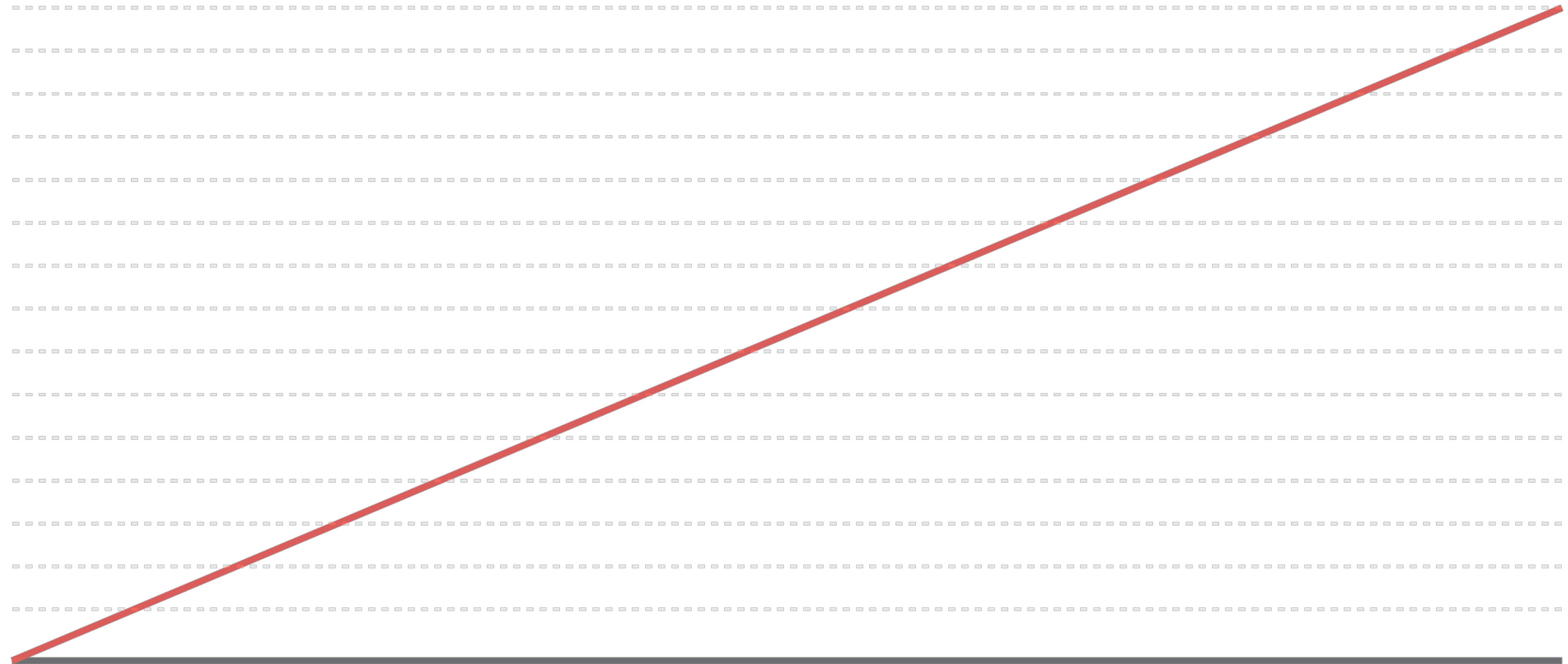
time



ideality



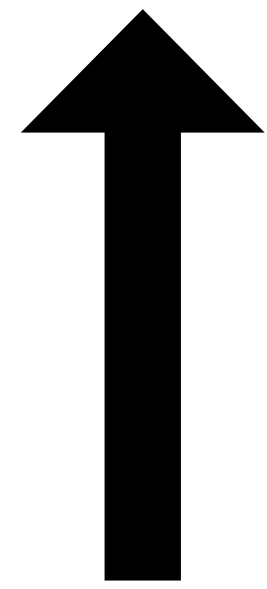
progress



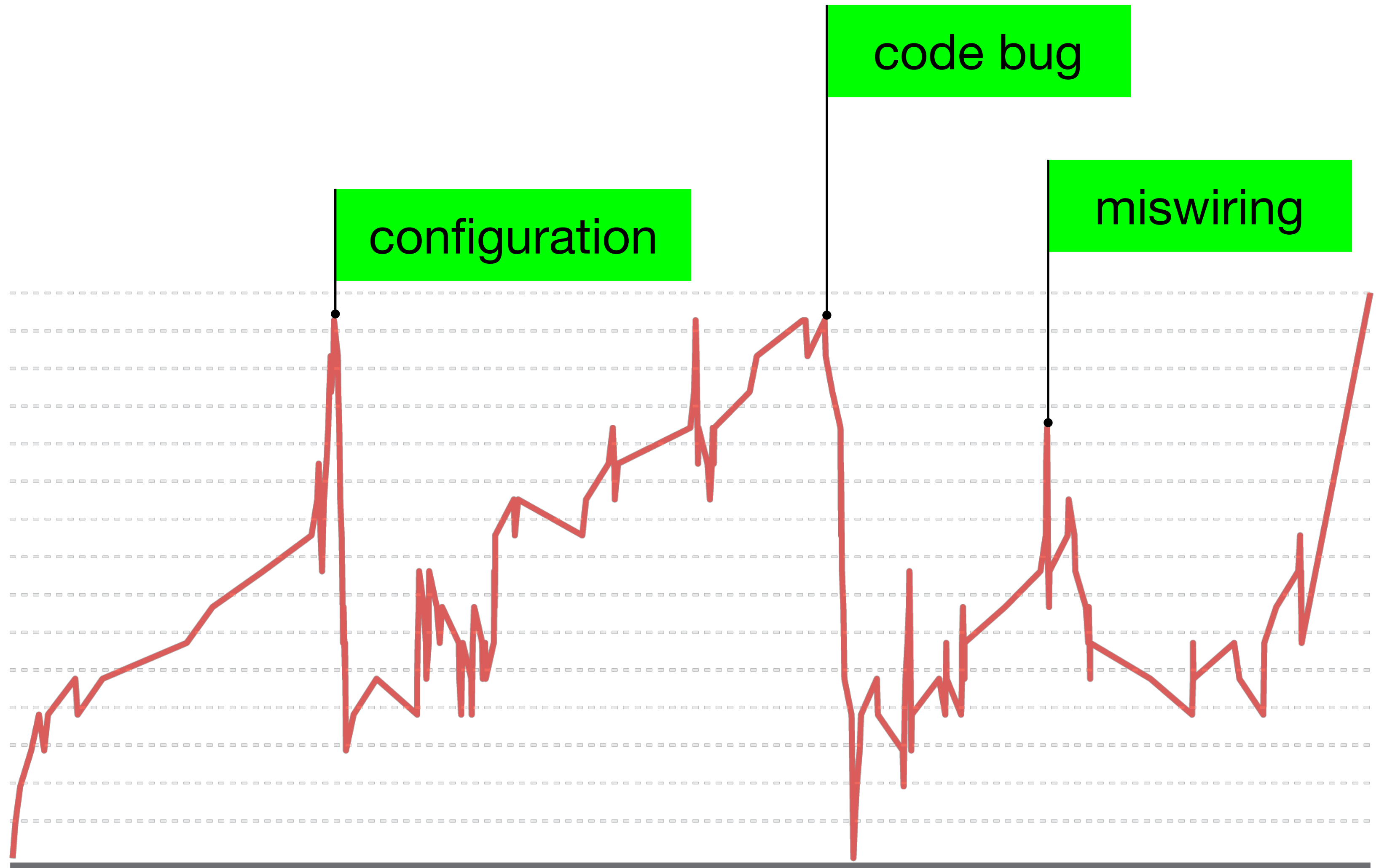
time



reality



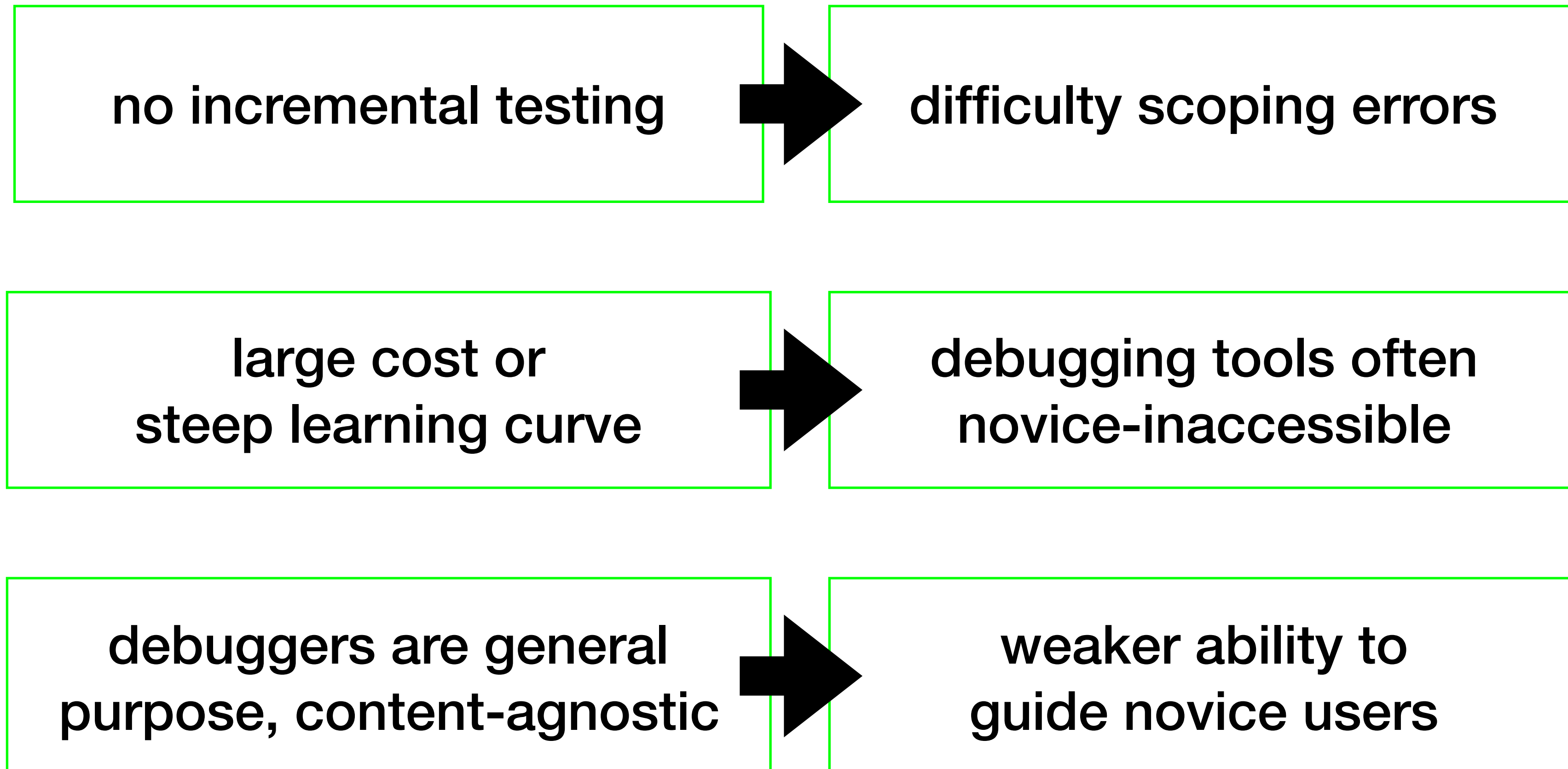
progress



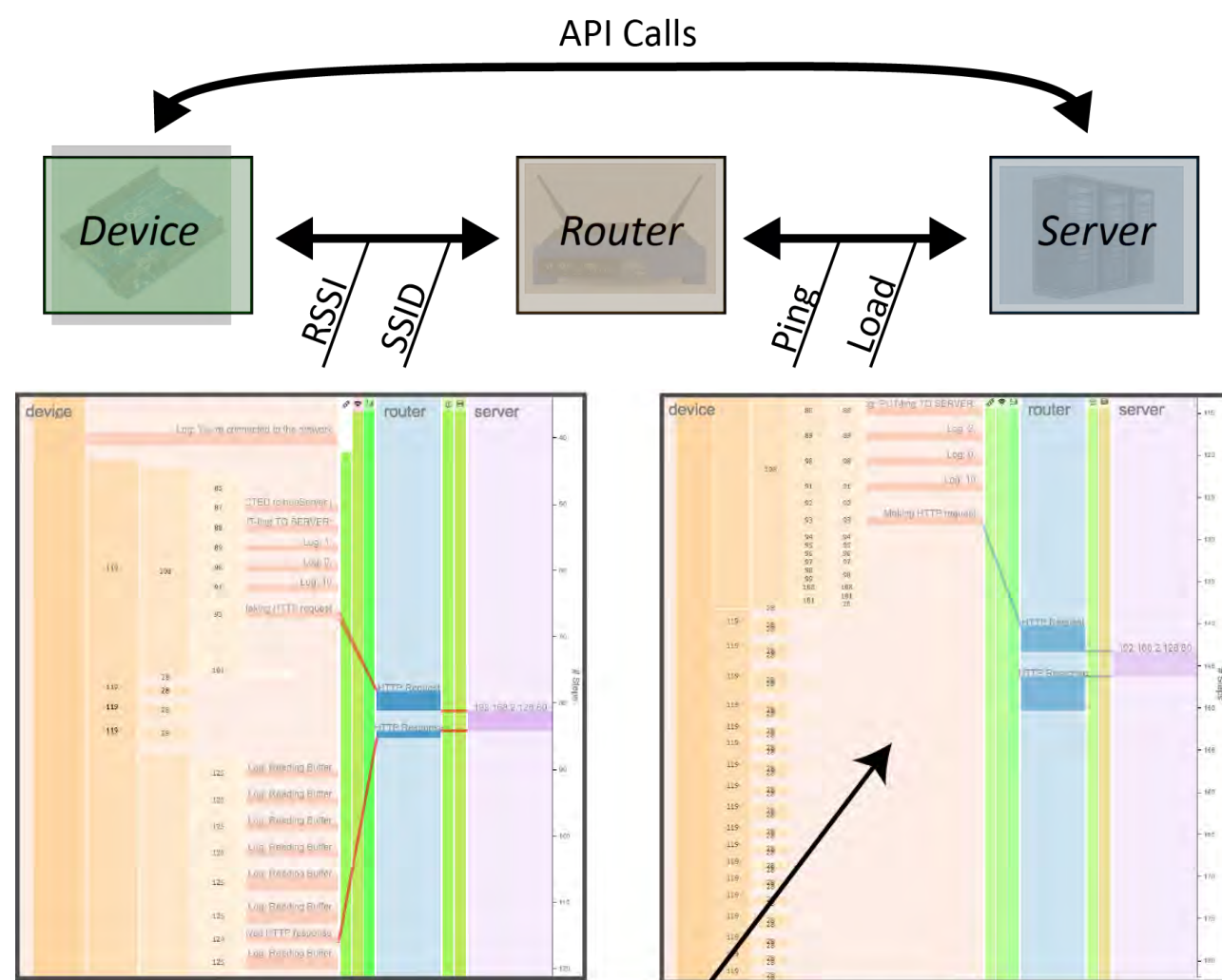
time



challenges

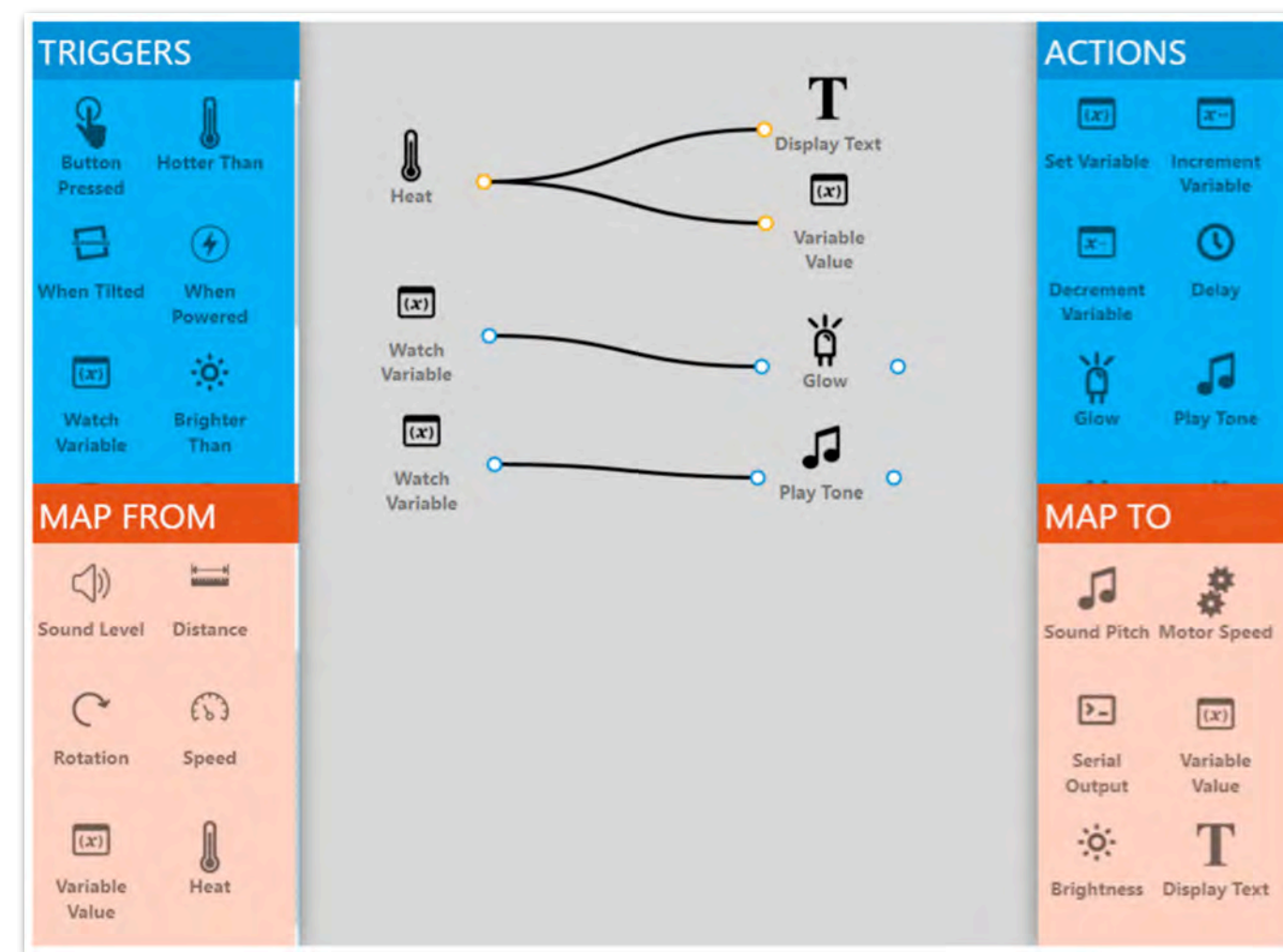


related



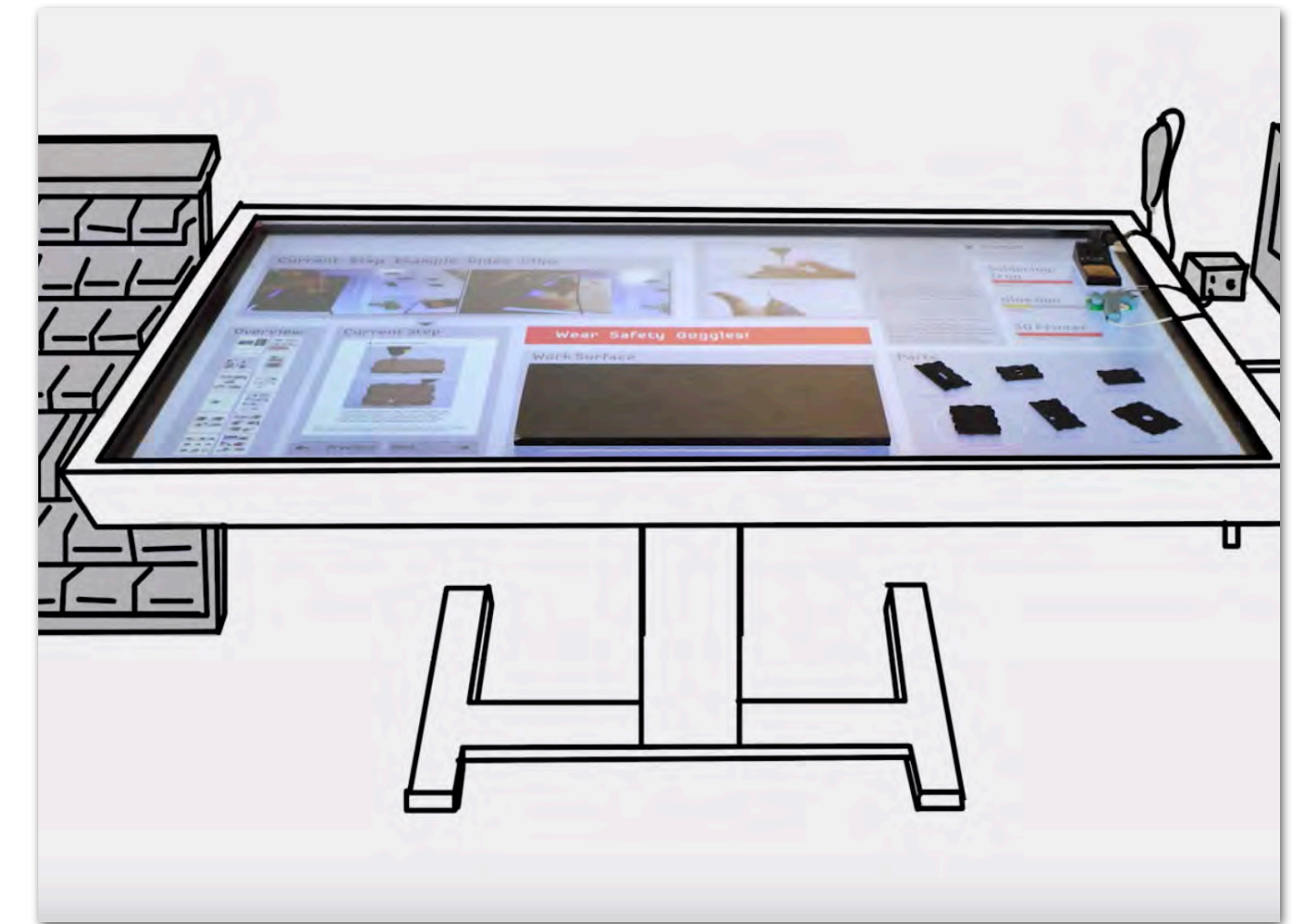
WiFröst

McGrath et al. UIST 2018



Trigger-Action-Circuits

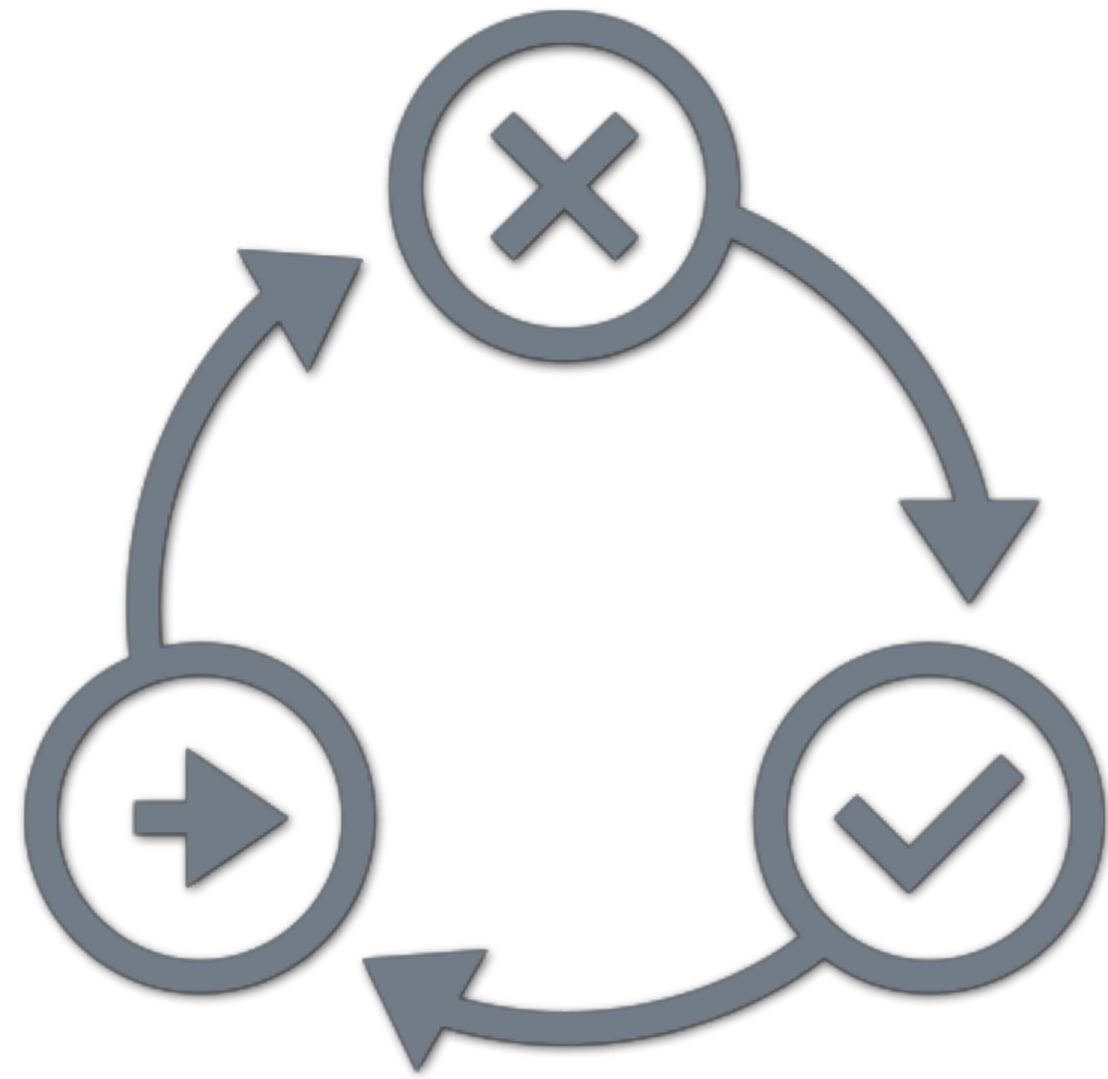
Anderson et al. UIST 2017



Smart Makerspace

Knibbe et al. ITS 2015

related

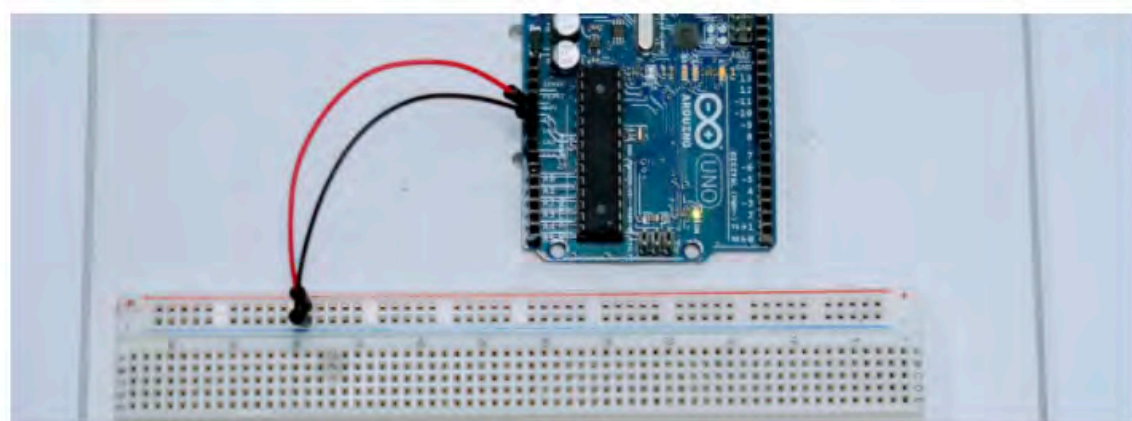


Test Driven Development
Kent Beck

incremental testing with
specific test cases

mitigating fear,
inspiring confidence

Step 5: Connecting Power and Ground



From the Arduino to the breadboard, connect:

- **GND** (ground) line to the blue rail
- **5V** (power) line to the red rail



Pass the tests to continue.
Pass the all tests in the right-hand pane to continue.

Back Next

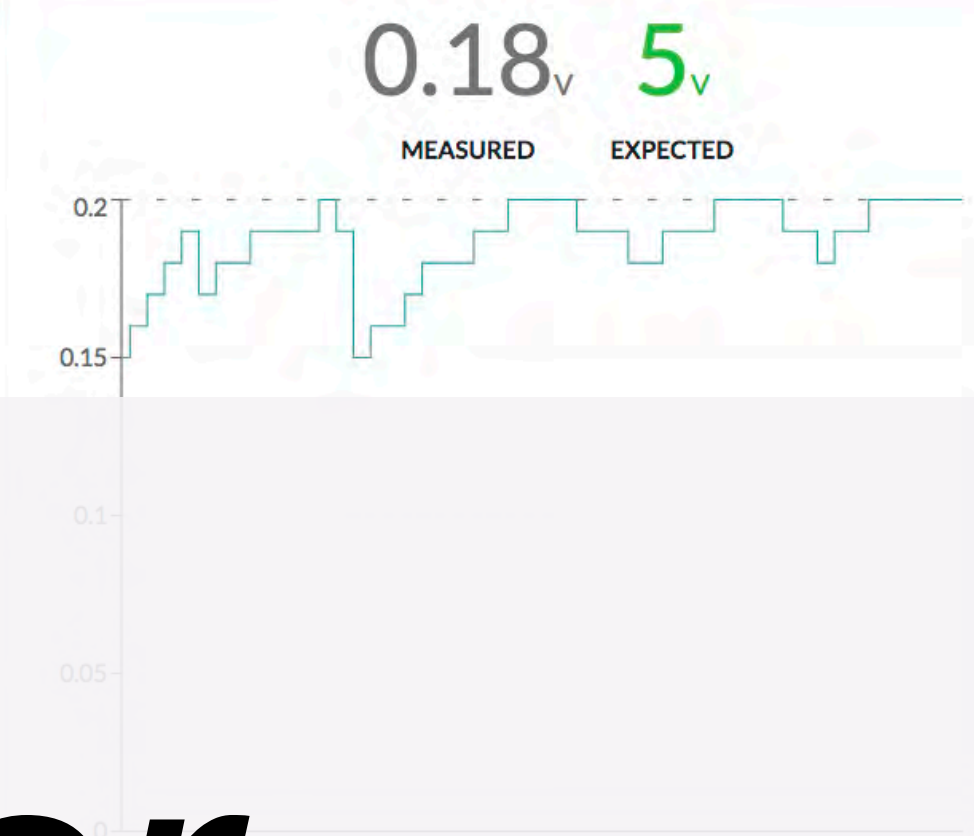
```
1 // Starter code - this is a comment.  
2  
3  
4 void setup() {  
5  
6 }  
7  
8 void loop() {  
9  
10 }  
11
```

Testing

Unpassed Tests
Pass the tests to continue.

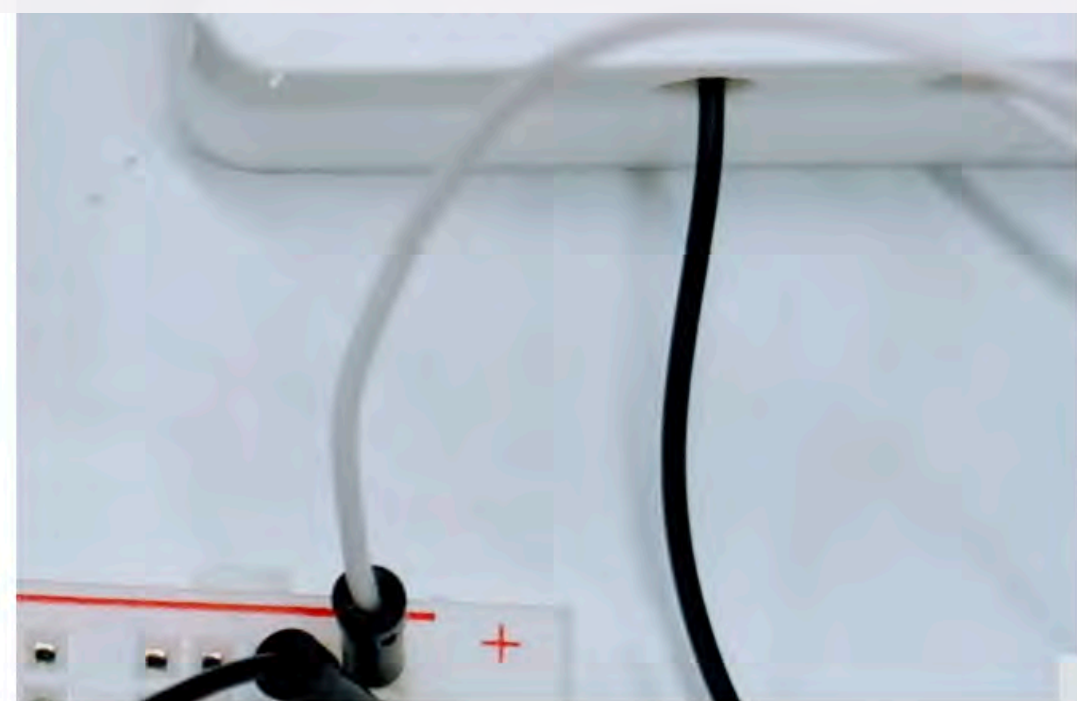
Next

Breadboard Power Check



Measuring...
Analyzing probe values...

Measure the breadboard power with the testing device's two probes.
Connect the white pin to **5V** (red/power), and the black pin to **0V/GND** (blue/ground).
Then, press the button below to check the voltage.

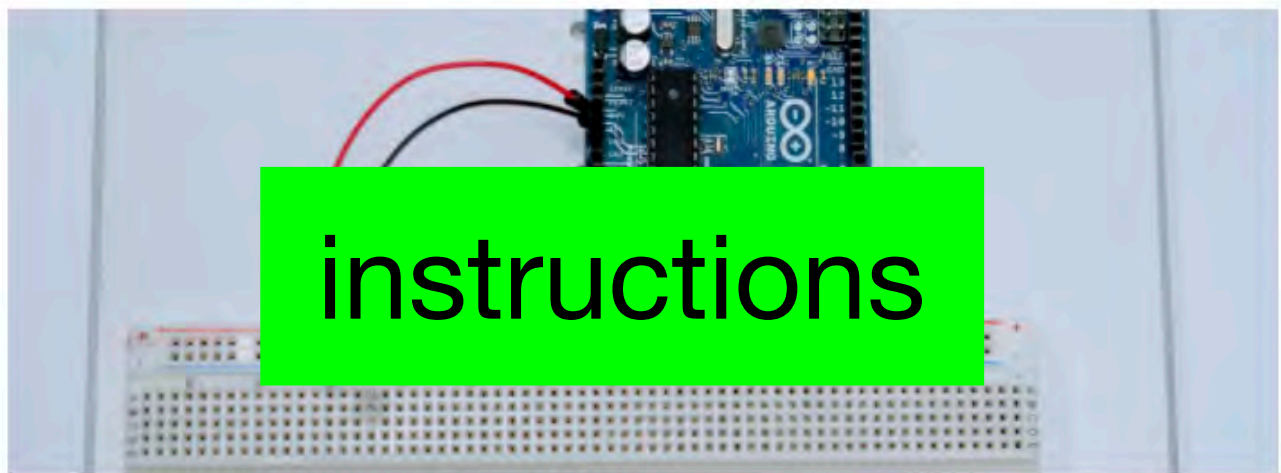


Guide Exit

ElectroTutor

ElectroTutor

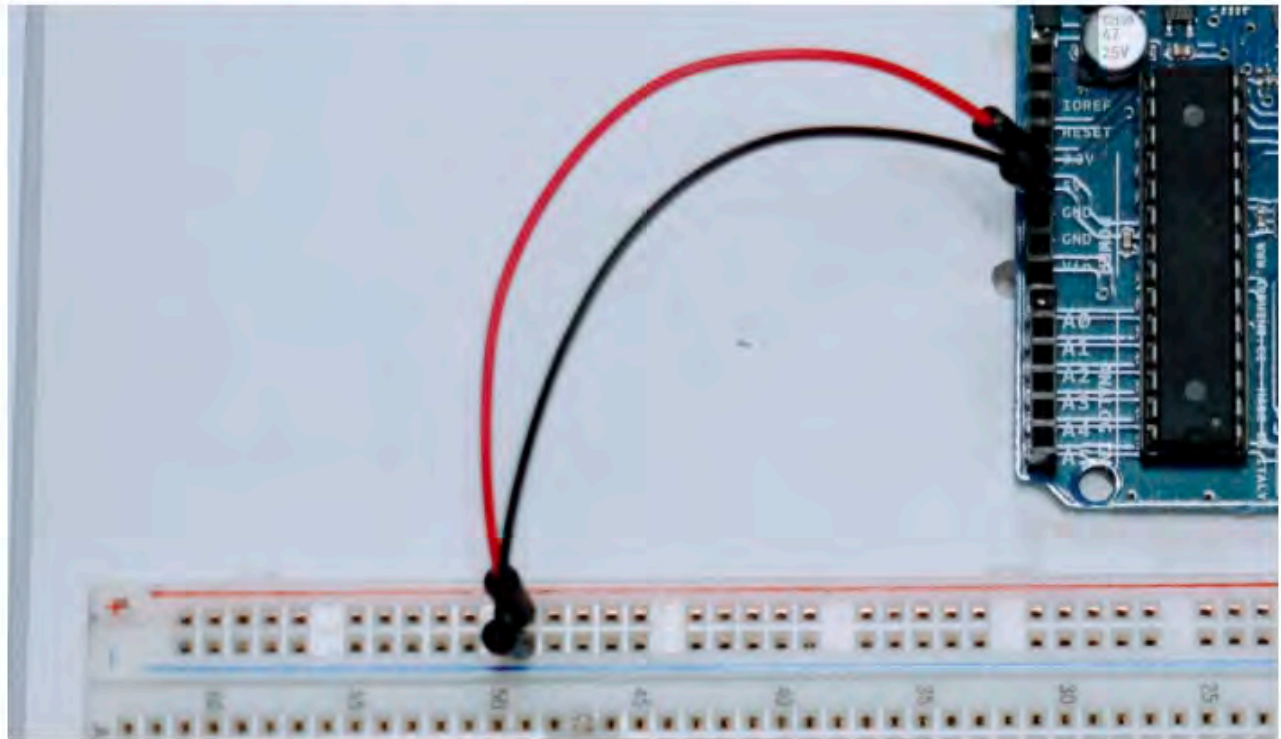
Step 5: Connecting Power and Ground



instructions

From the Arduino to the breadboard, connect:

- **GND** (ground) line to the blue rail
- **5V** (power) line to the red rail



Pass the tests to continue.
Pass the all tests in the right-hand pane to continue.

Back Next

```
1 // Starter code - this is a comment.  
2  
3  
4 void setup() {  
5  
6 }  
7  
8 void loop() {  
9  
10 }  
11
```

editor

Testing

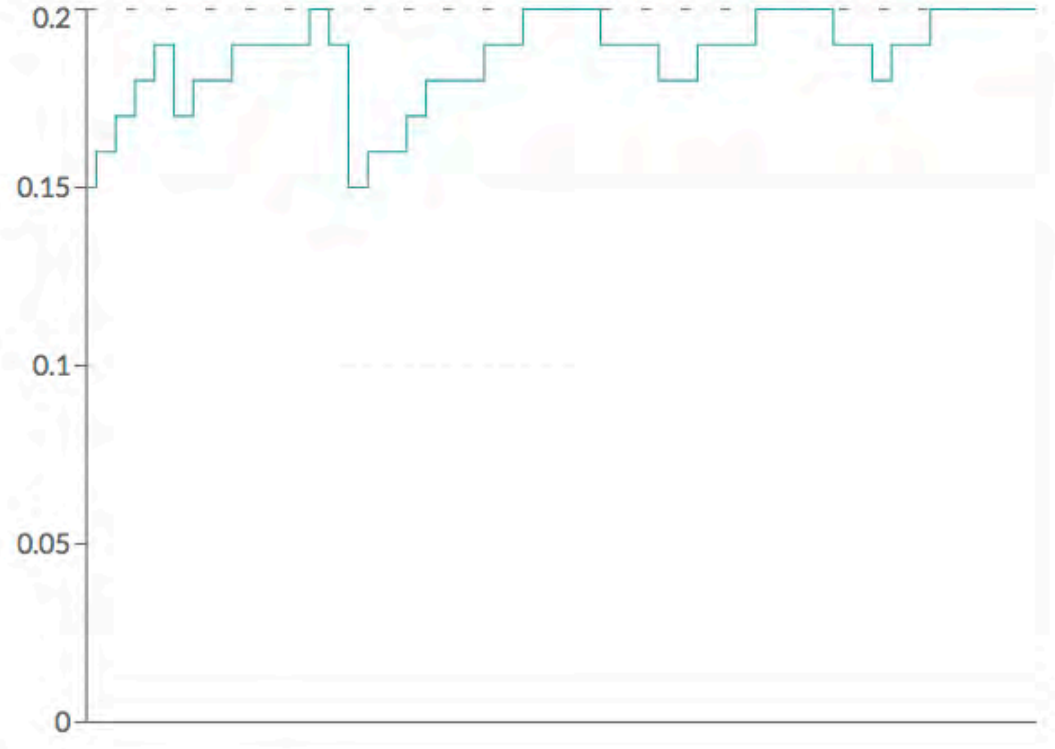
Unpassed Tests
Pass the tests to continue.

Breadboard Power C

test pane

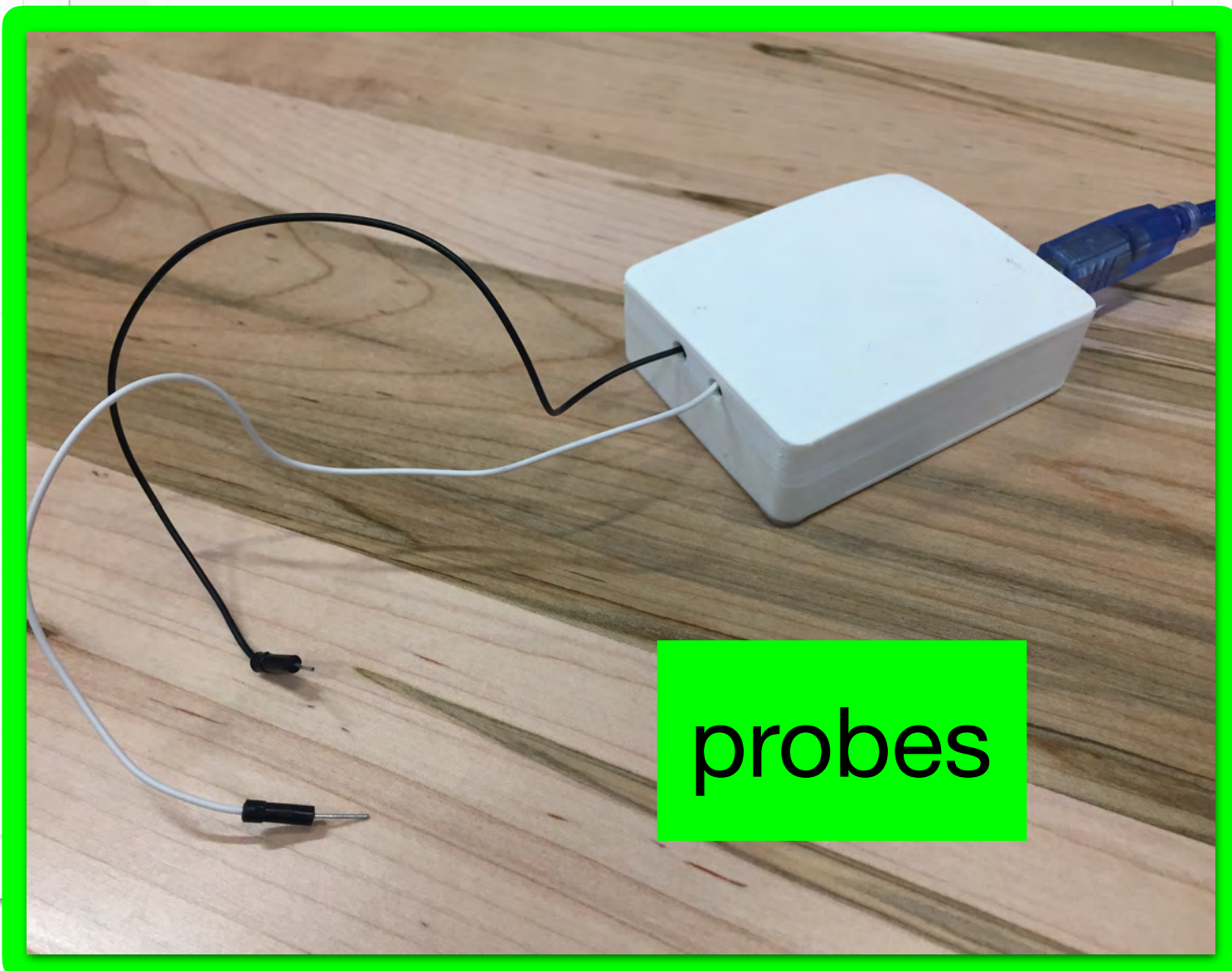
0.18_v 5_v

MEASURED EXPECTED



Measuring...
Analyzing probe values...

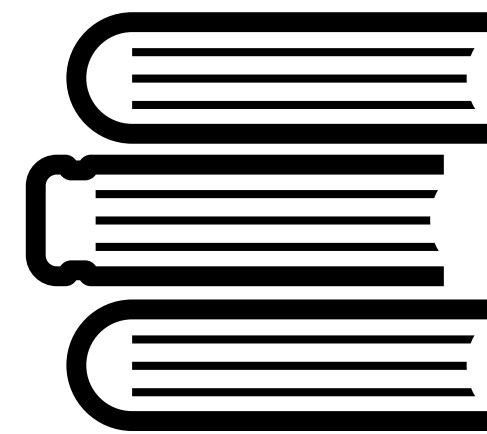
Measure the breadboard power with the testing device's two probes.
Connect the white pin to **5V** (red/power), and the black pin to **0V/GND** (blue/ground).
Then, press the button below to check the voltage.



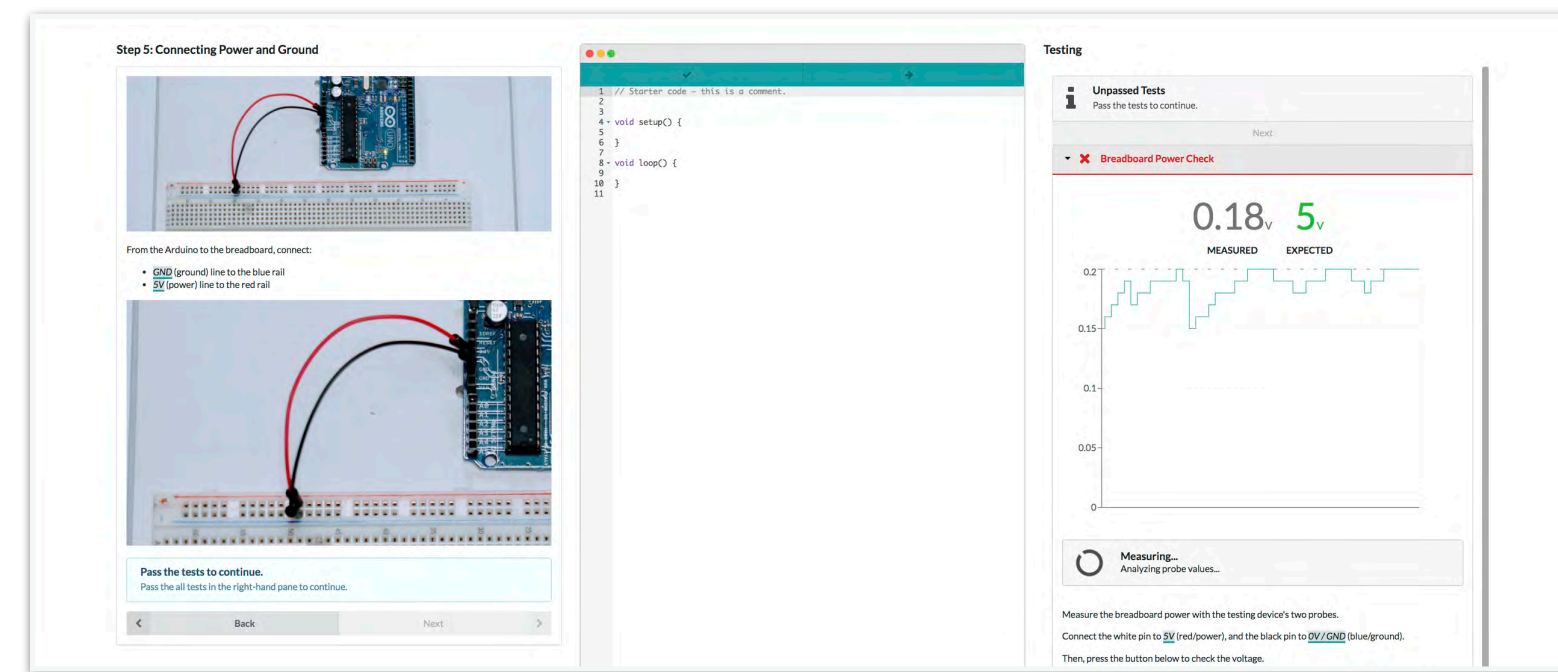
probes

system

tutorial
content



user interface



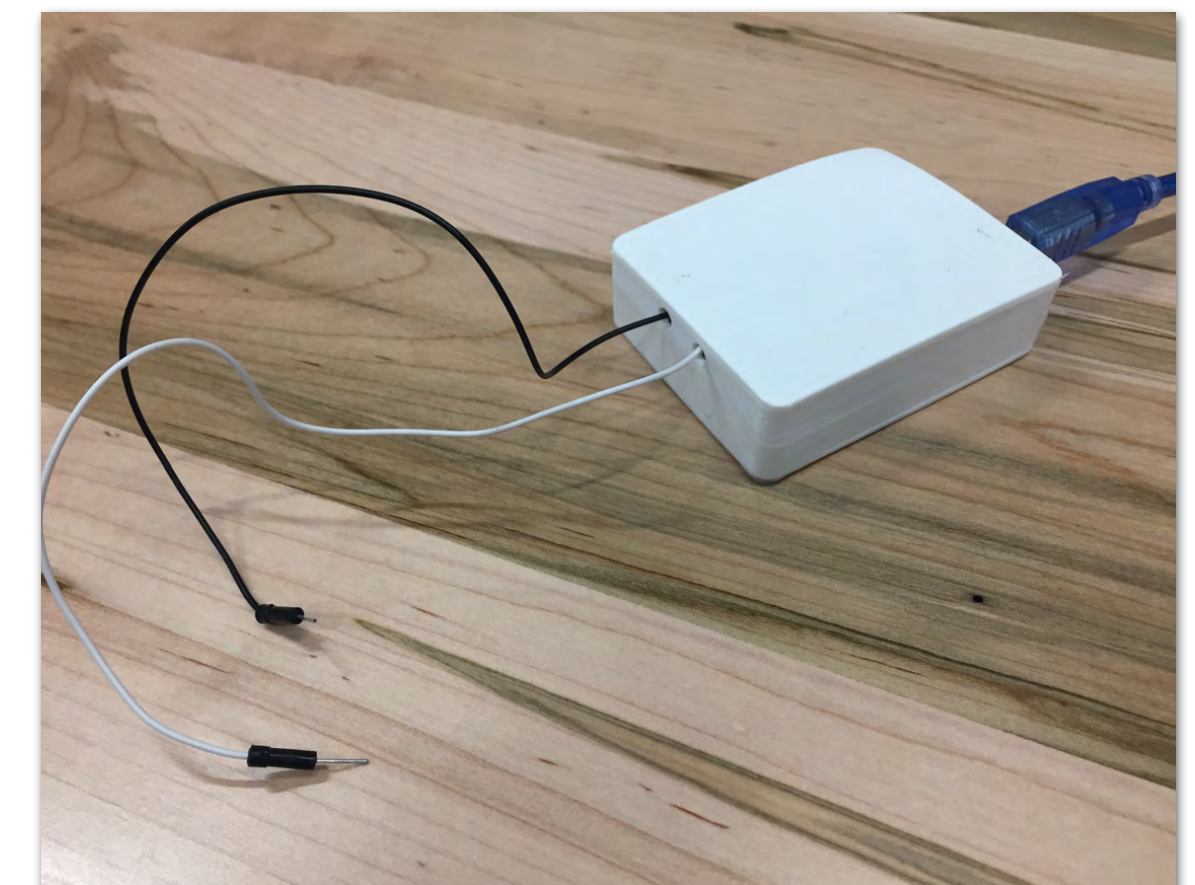
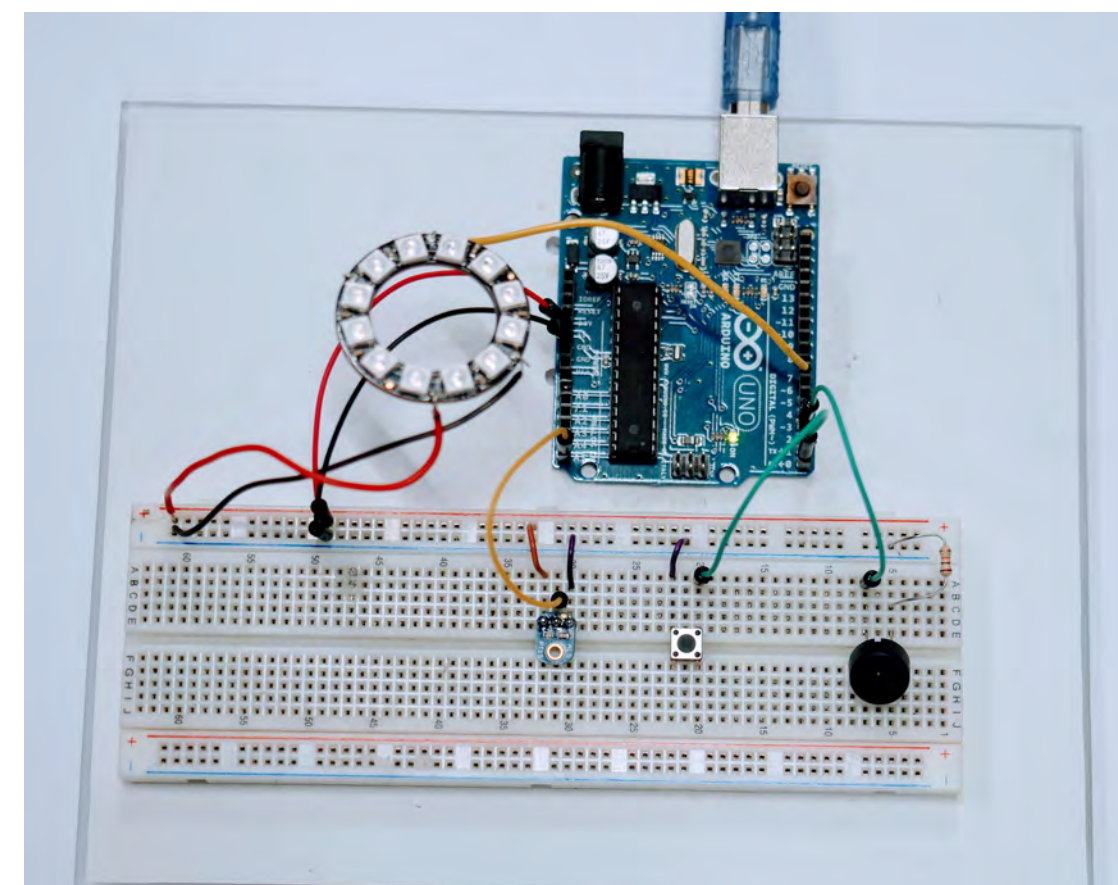
backend



serial

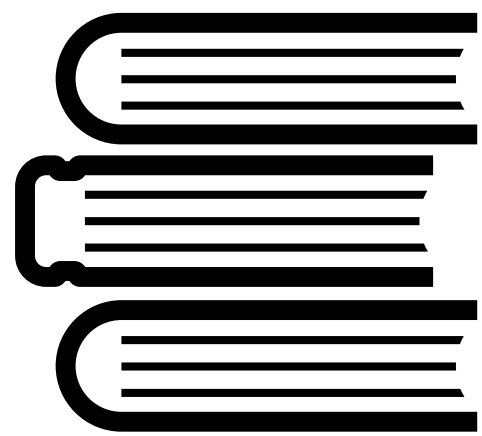
tutorial project

probes



system

tutorial
content



<YAML>



```
title: Neopixel Ring Wiring
description: |
  ![neopix](neopixel.jpg)

  Connect the *Power (red)* and *Ground (black)* ends of the Neopixel ring to the positive and ground rails.

  ![neopix](neopixel-crop1.jpg)

  Connect the *Data In* yellow LED pin to the *digital 7* on the Arduino.

  ![neopix](neopixel-crop2.jpg)

tests:
  title: Hardware Wiring Check
  description: |
    The system will now check to see that you have wired up the LED ring correctly.

    If the lights are wired up correctly, you will see a glowing pattern.

    Does the LED ring rotate though the rainbow?

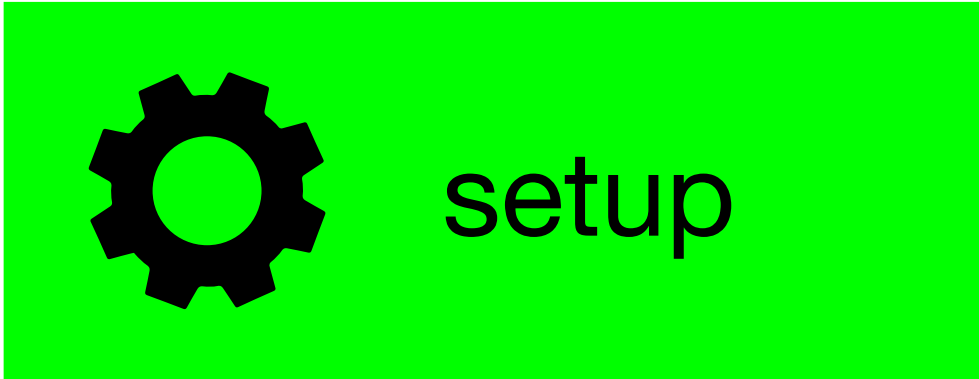
    Click below to setup the test.

    ![neopix](rainbow.gif)

onerror: Make the sure data pin is connected to pin 7, and the power/ground pins are connected to 5V and GND respectively.
jsondata: '{"file":"3_lights.ino"}'
form: autoupload
```

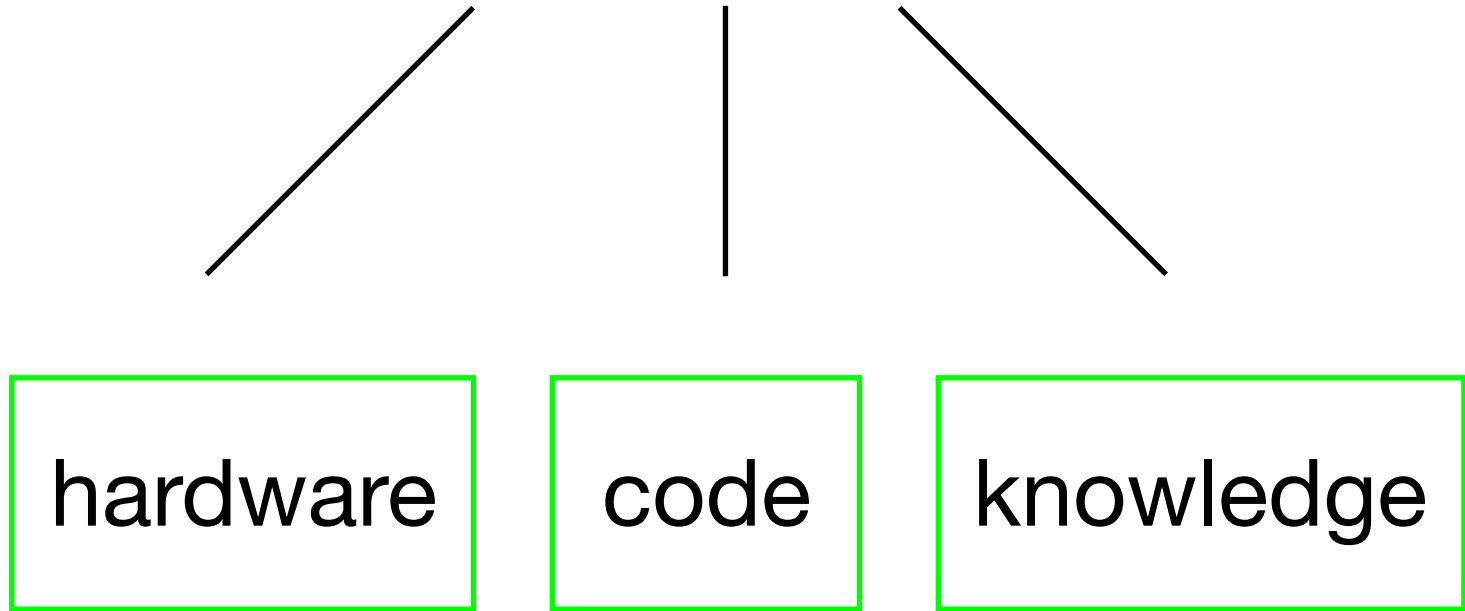
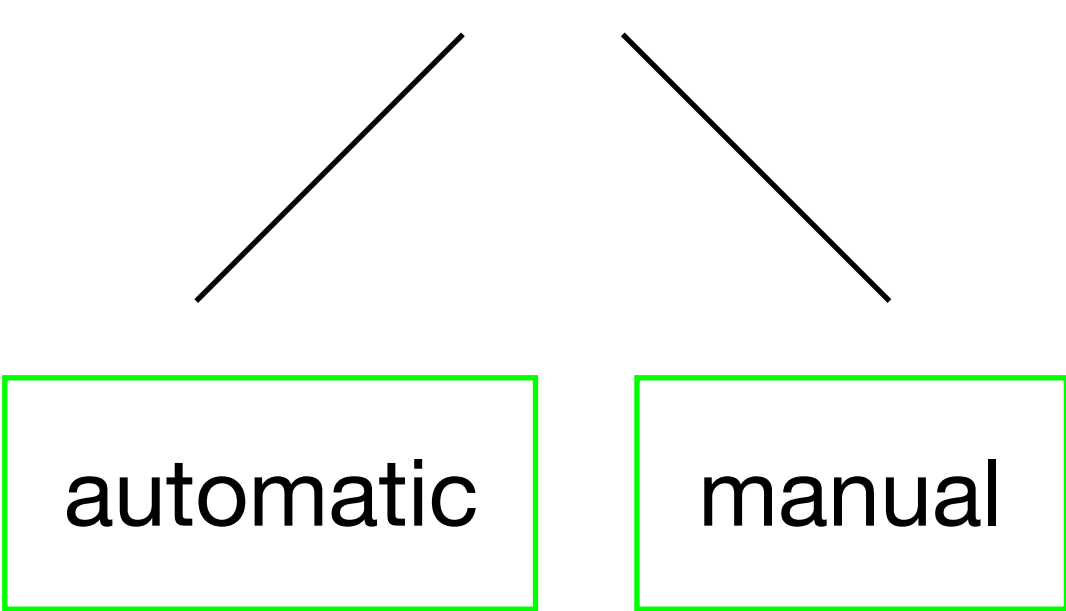
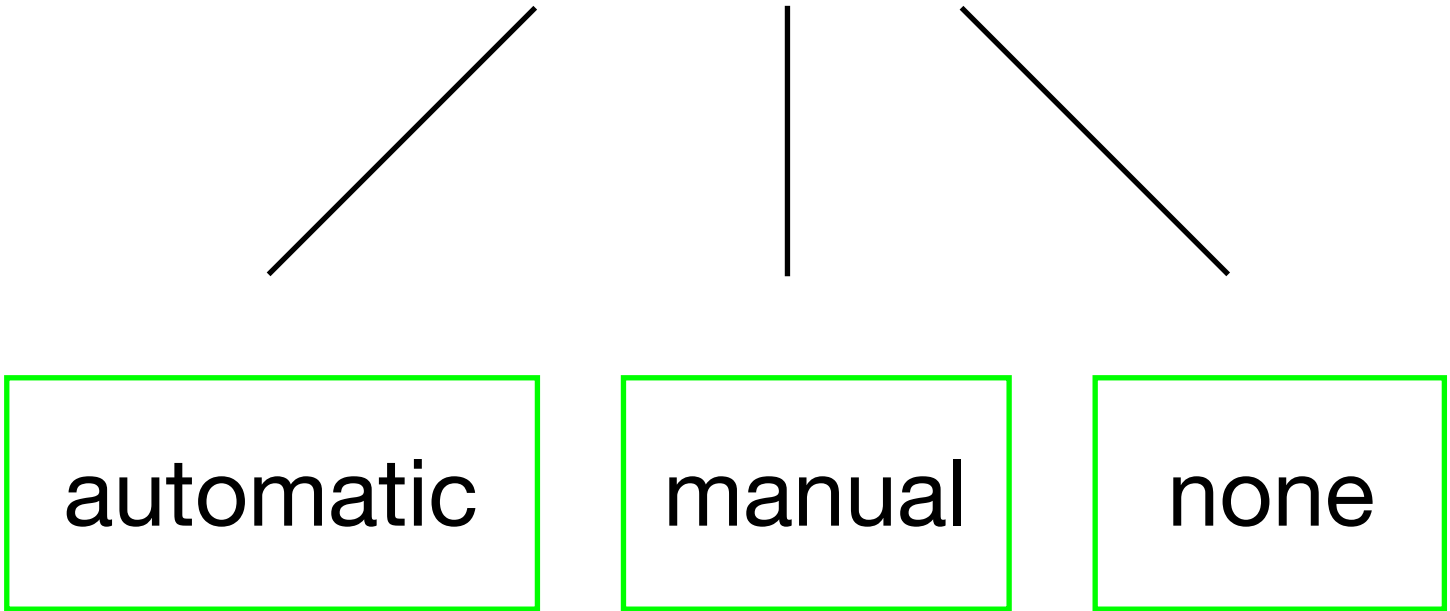
} <Markdown>

test taxonomy

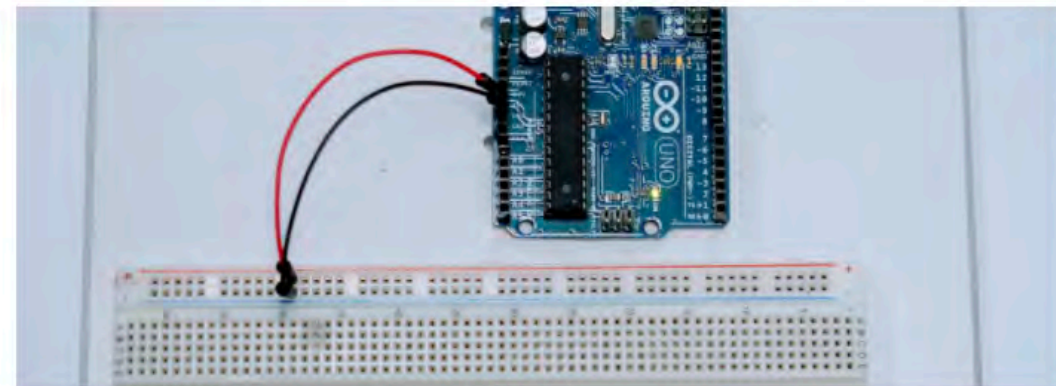


types

A green rectangular box containing the text "types".



Step 5: Connecting Power and Ground



From the Arduino to the breadboard, connect:

- **GND** (ground) line to the blue rail
- **5V** (power) line to the red rail



Pass the tests to continue.
Pass the all tests in the right-hand pane to continue.

Back Next

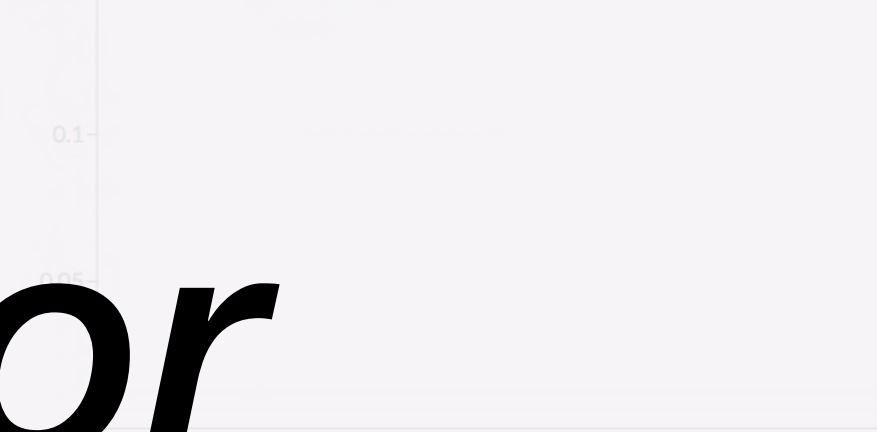
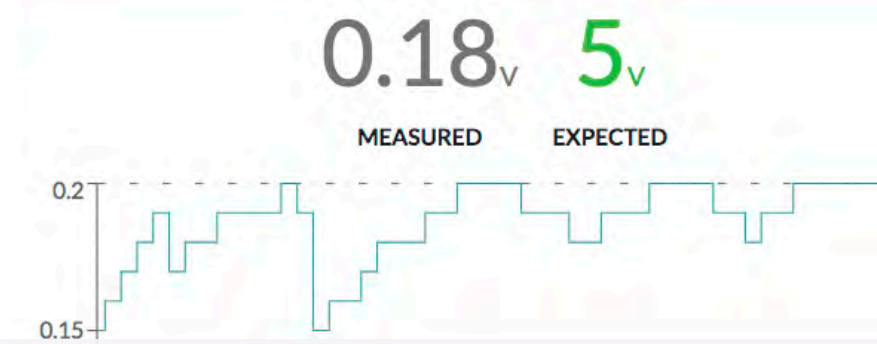
```
1 // Starter code - this is a comment.  
2  
3  
4 void setup() {  
5 }  
6 }  
7  
8 void loop() {  
9 }  
10 }  
11
```

Testing

Unpassed Tests
Pass the tests to continue.

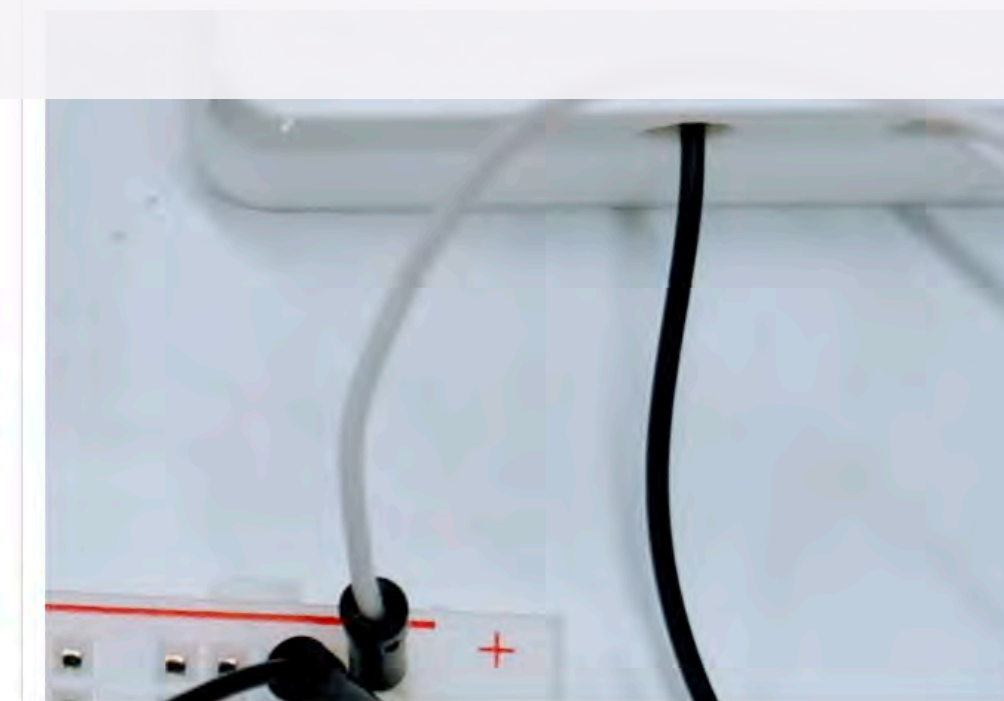
Next

✖ Breadboard Power Check



Measuring...
Analyzing probe values...

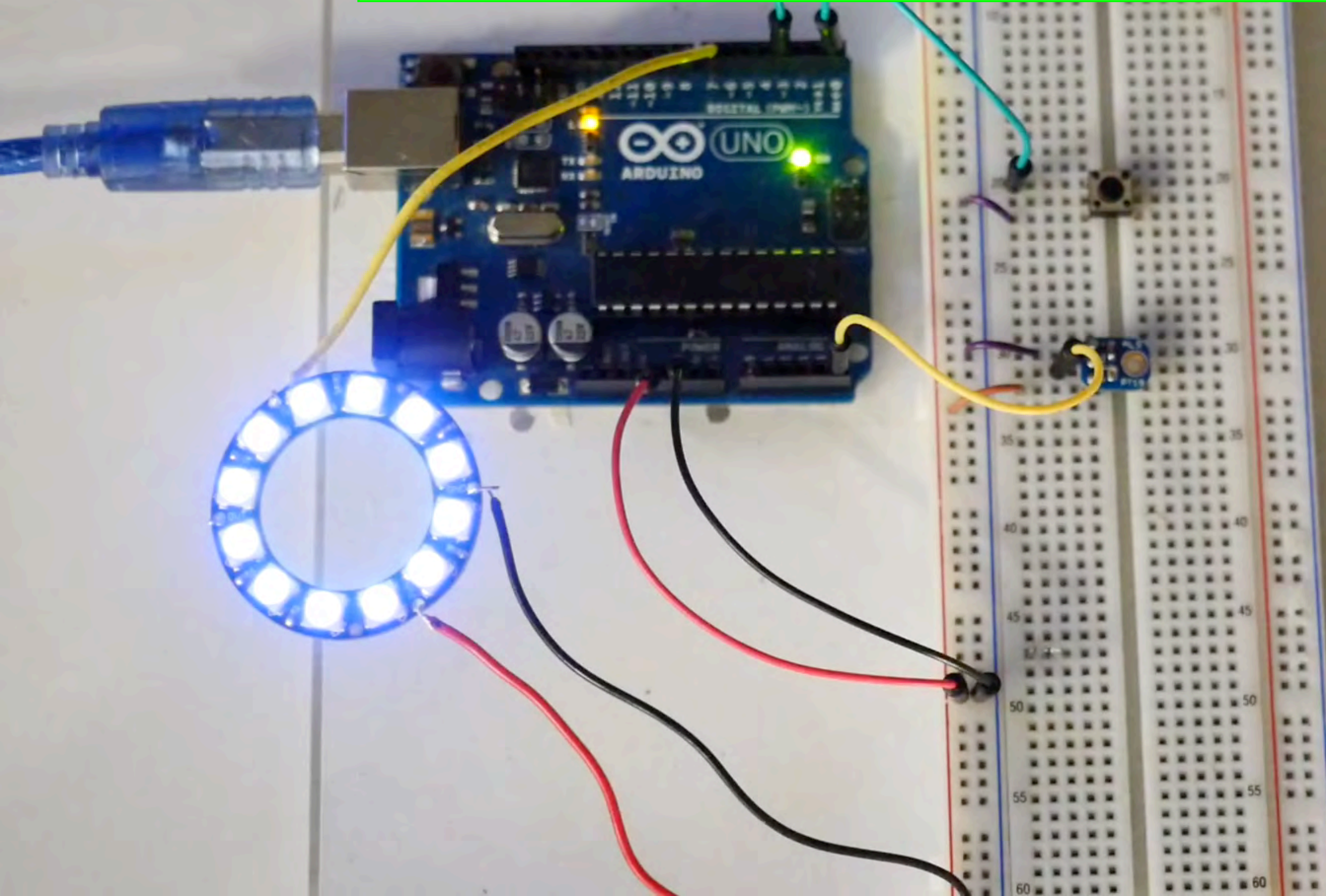
Ensure the breadboard power with the testing device's two probes.
Connect the white pin to **5V** (red/power), and the black pin to **GND** (blue/ground).
Then, press the button below to check the voltage.



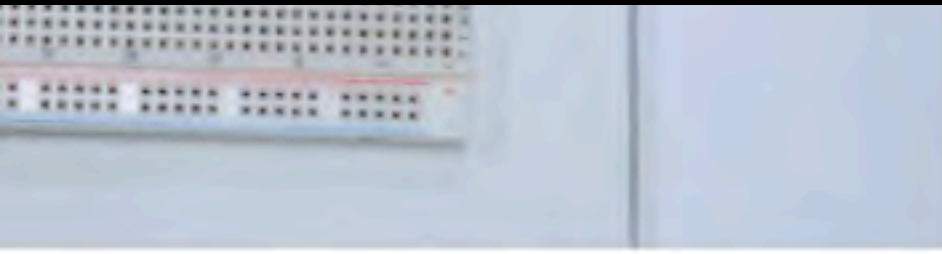
Guide Exit

ElectroTutor Test Examples

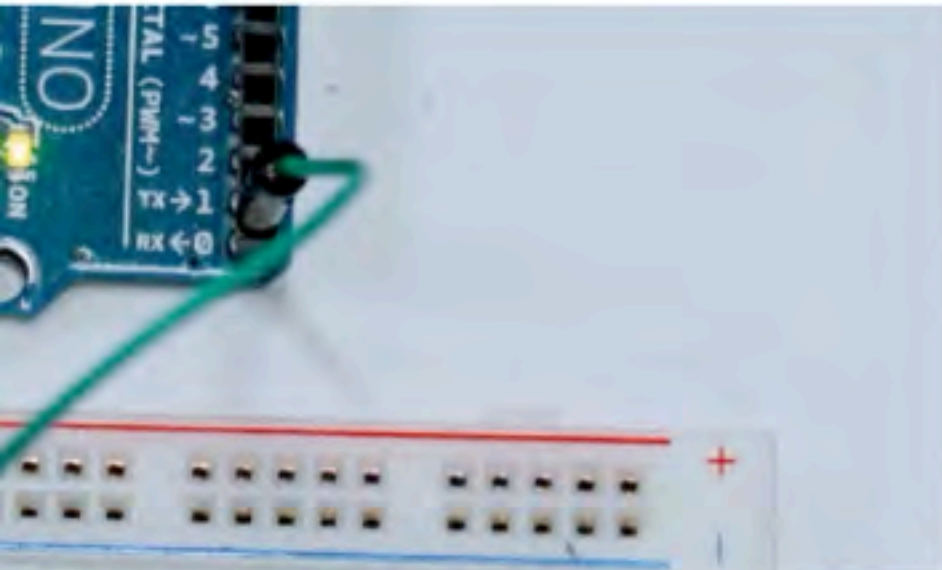
project: light-sensitive alarm clock



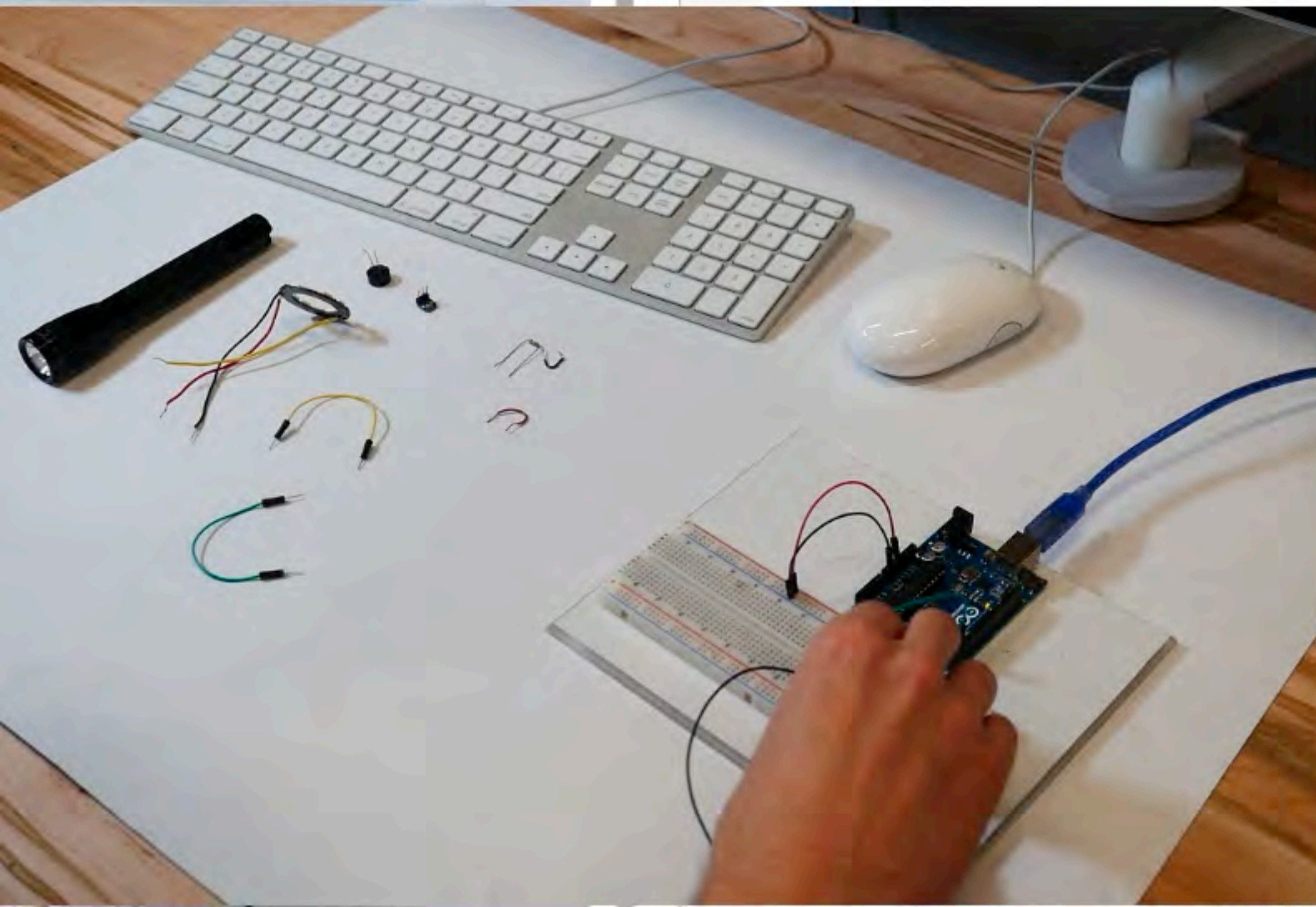
runtime hardware test



ured.
ue/ground) with a jumper wire.
rduino board.



```
1 // Starter code - this is a comment.  
2  
3  
4 void setup() {  
5 }  
6 }  
7  
8 void loop() {  
9 }  
10 }  
11
```



Failed Tests
Pass the tests to continue.

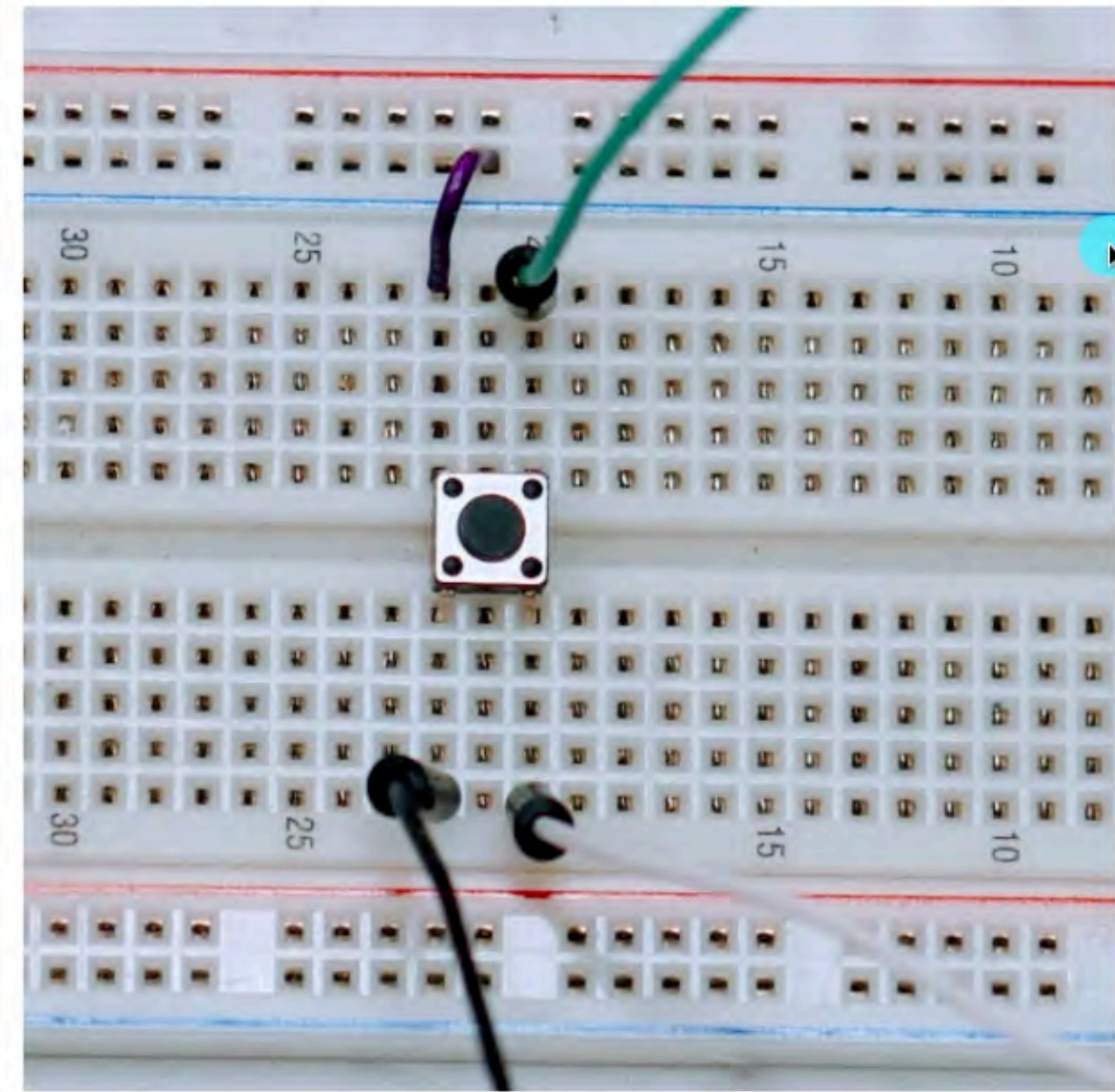
Next

Button Press Check

Measure the voltage across the button using the tester.

Connect the black wire to GND (ground) and the white wire to the Arduino-connected side.

The voltage should drop as you press the button down - hold it down for AT LEAST 2 seconds.



Click to Measure Voltage Signal

Voltage Knowledge Test

runtime code value test

```
9 #define thresh_lo 100
10 #define thresh_hi 900
11
12 #define button 2
13 #define outLED 13
14
15 #define buzzer 4
16 #define freq 220
17
18 int color = 0;
19 Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB + NEO_KHZ800);
20 uint32_t white = strip.Color(16, 16, 16); // dim white color
21
22 void setup() {
23
24   pinMode(light, INPUT);
25   pinMode(button, INPUT_PULLUP);
```

Next

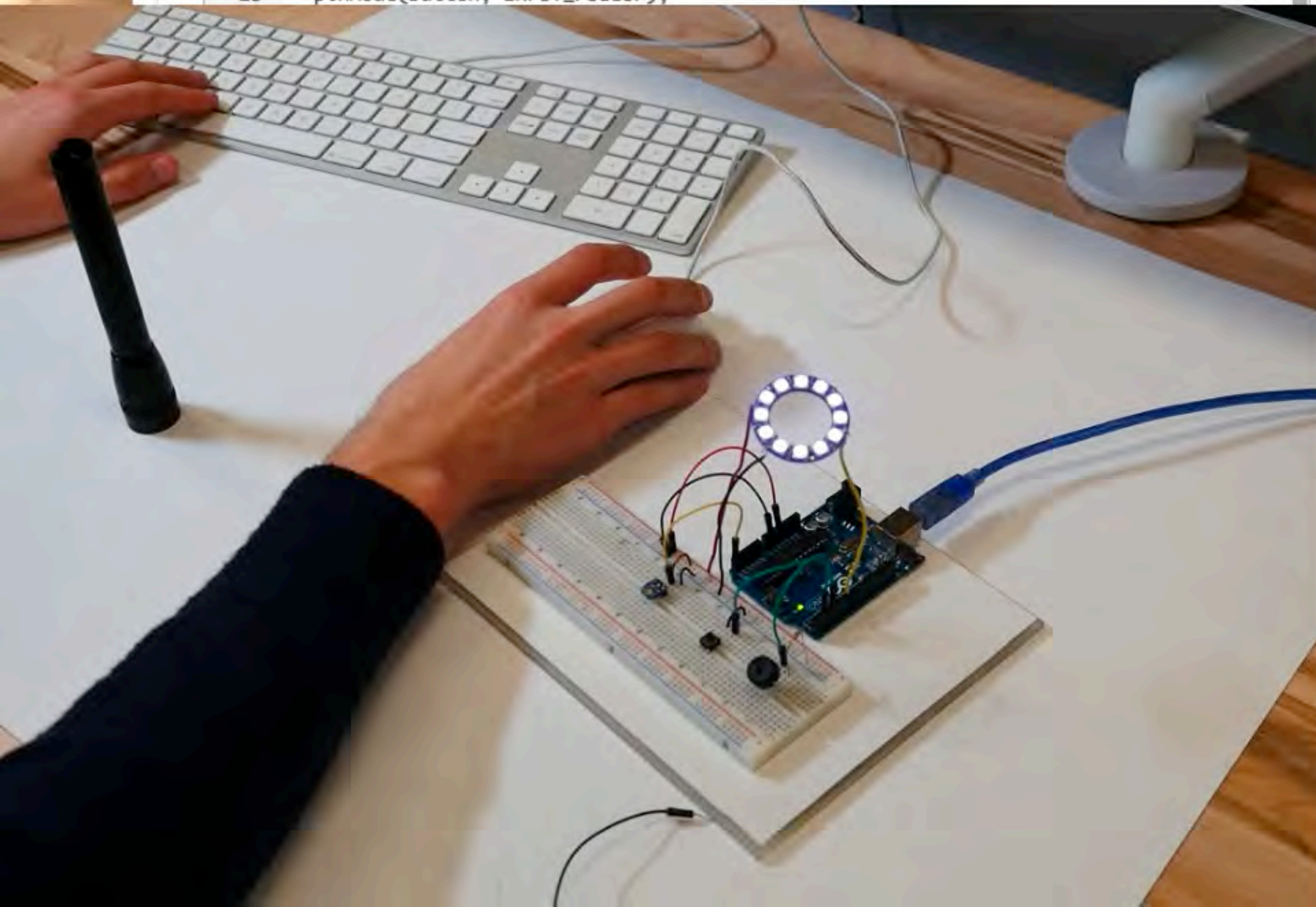
▶ ✓ Code Selection Test - Sensor Reading

▼ ⓘ Software Trigger Check

The system will now check to see that you programmed the Arduino board correctly.

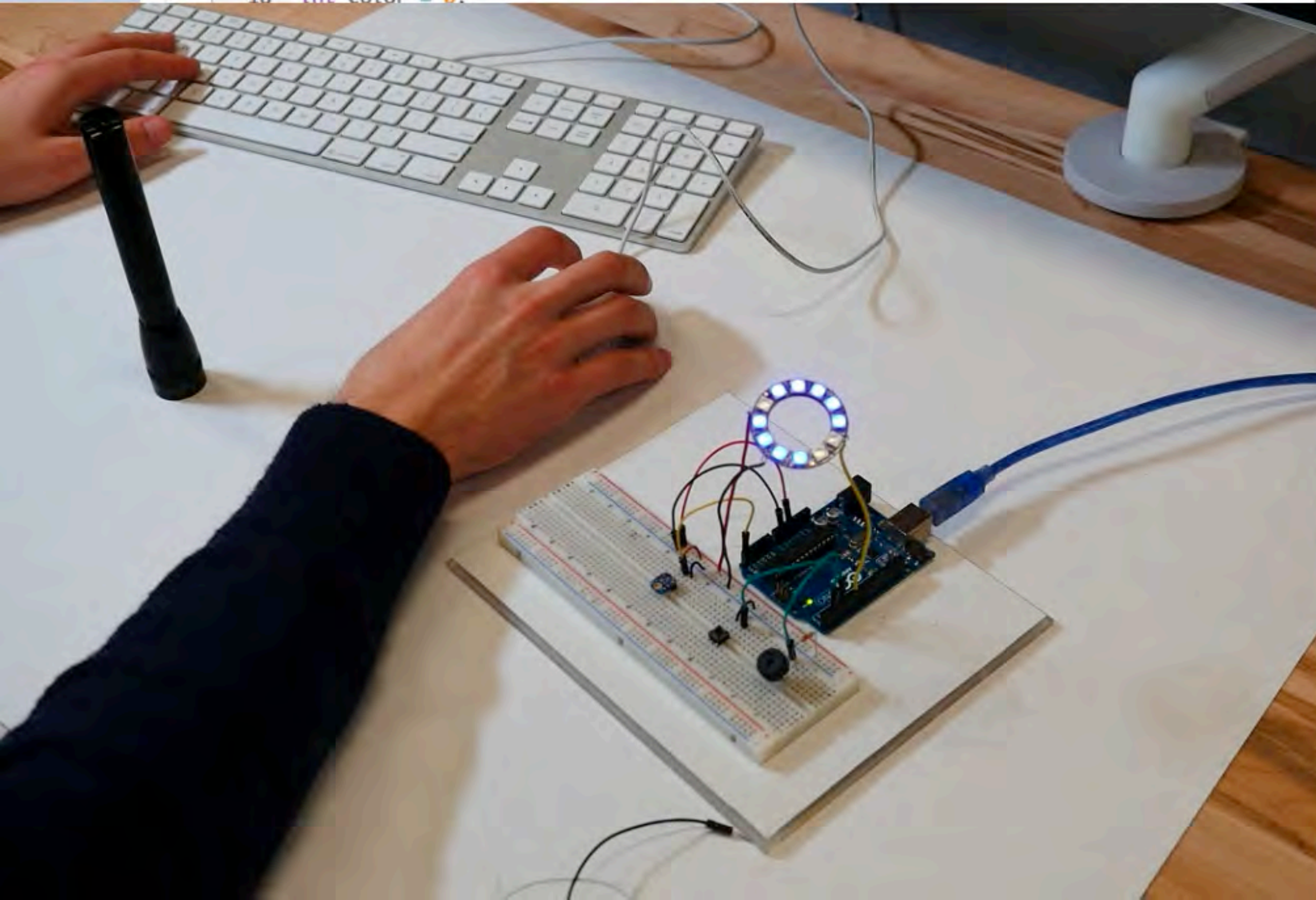
Once loaded, shine the light sensor to raise the level above the threshold.

[Click to Measure Code Variable](#)



wiring test

```
5 #define PIN 7
6 #define PIXELS 12
7
8 #define light A5
9 #define thresh_lo 100
10 #define thresh_hi 900
11
12 #define button 2
13 #define outLED 13
14
15 #define buzzer 4
16 #define freq 220
17
18 int color = 0;
```



Unpassed Tests

Pass the tests to continue.

Next

✖ Hardware Configuration Check

Check that the sensor is connected and configured as 'INPUT', not 'INPUT_PULLUP'.

We will now check that your hardware is configured corrected.

Try flashing the light sensor to turn on the alarm.

Try pressing the button to turn off the alarm after it is activated.

Did both of these functions work as expected?

[Click to Setup and Begin Test](#)

compilation test

Include this library code for the LED ring at the VERY top of the file:

```
#include <Adafruit_NeoPixel.h>
```

Then, define the pin the Neopixel ring's data pin is connected to, and the number of "pixels" in the ring.

```
#define PIN 7  
#define PIXELS 12
```

Before the setup() function, create the strip of LEDs and a base white color.

```
int color = 0;  
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB + NEO_KHZ800);  
uint32_t white = strip.Color(16, 16, 16); // dim white color
```

Initialize the LED strip just before the end of setup() function.

```
strip.begin(); // This initializes the NeoPixel library.  
strip.setBrightness(16);
```

In between the setup() and loop() functions, insert these two light helper functions.

The first function, wheel, takes in a number (specifically a byte, an 8-bit number), and returns a color that we can use to set the LED.

The second function, rainbow, loops over all of the LEDs in the ring and updates their color, using the wheel function.

```
uint32_t Wheel(byte WheelPos) {  
  WheelPos = 255 - WheelPos;  
  if (WheelPos < 85) {
```

```
1 // Starter code - this is a comment.  
2  
3 #include <Adafruit_NeoPixel.h>  
4  
5 #define PIN 7  
6 #define PIXELS 12  
7  
8 #define button 2  
9 #define outLED 13  
10  
11 #define buzzer 4  
12 #define freq 220  
13  
14 void setup() {  
15  
16 int color = 0;  
17 Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB + NEO_KHZ800);  
18 uint32_t white = strip.Color(16, 16, 16); // dim white color  
19  
20 pinMode(button, INPUT_PULLUP);  
21 pinMode(outLED, OUTPUT);  
22  
23 strip.begin(); // This initializes the NeoPixel library.  
24 strip.setBrightness(16);  
25 }  
26  
27 uint32_t Wheel(byte WheelPos) {  
28   WheelPos = 255 - WheelPos;  
29   if (WheelPos < 85) {  
30     return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);  
31   }  
32   if (WheelPos < 170) {  
33     WheelPos -= 85;  
34     return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);  
35   }  
36   WheelPos -= 170;  
37   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);  
38 }  
39  
40 (int start, int wait) {  
41   start;  
42   q = 0; q < 3; q++) {  
43     (int i = 0; i < strip.numPixels(); i = i + 3) {  
44       strip.setPixelColor(i + q, Wheel((i + j) % 255)); //turn every third  
45       pixel on  
46     }  
47     p.show();  
48     y(wait);  
49     (int i = 0; i < strip.numPixels(); i = i + 5) {  
50       strip.setPixelColor(i + q, 0); //turn every third pixel off  
51     }  
52   }  
53 }  
54  
55 lor, 2);  
56 olor+2) % 255;
```

```
(int start, int wait) {  
  start;  
  q = 0; q < 3; q++) {  
    (int i = 0; i < strip.numPixels(); i = i + 3) {  
      strip.setPixelColor(i + q, Wheel((i + j) % 255)); //turn every third  
      pixel on  
    }  
    p.show();  
    y(wait);  
    (int i = 0; i < strip.numPixels(); i = i + 5) {  
      strip.setPixelColor(i + q, 0); //turn every third pixel off  
    }  
  }  
}
```

```
p.show();  
y(wait);
```

```
(int i = 0; i < strip.numPixels(); i = i + 5) {  
  strip.setPixelColor(i + q, 0); //turn every third pixel off
```

```
{  
  lor, 2);  
  olor+2) % 255;
```

Unpassed Tests

Pass the tests to continue.

Next

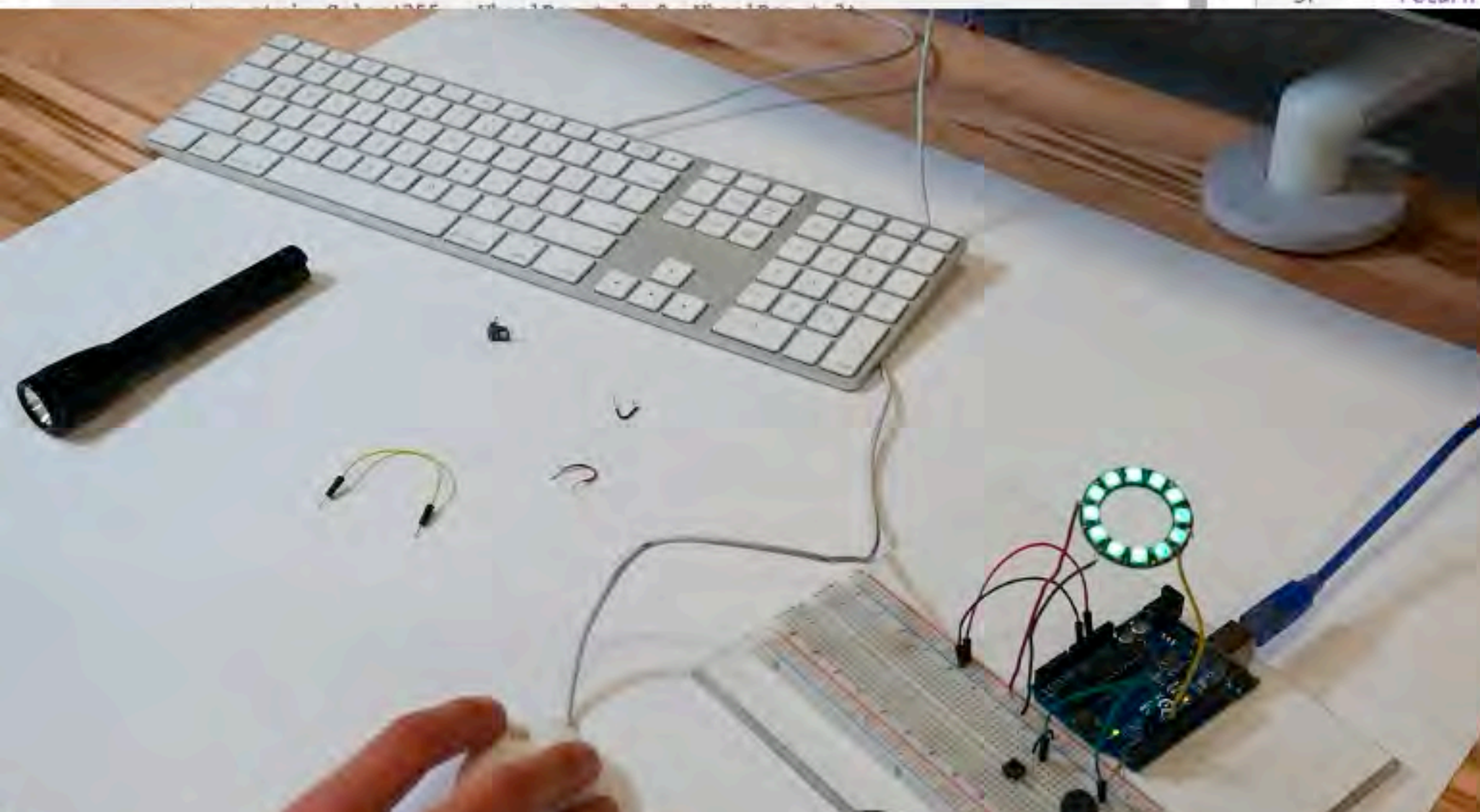
✓ Code Selection Test - Library Inclusion

✗ Compilation Check

Test your code to see if there are any errors in it.

```
code: In function 'void setup()':  
code:17:5: warning: unused variable 'color' [-Wunused-variable]  
int color = 0;  
^  
code:19:10: warning: unused variable 'white' [-Wunused-variable]  
uint32_t white = strip.Color(16, 16, 16); // dim white color  
^  
code: In function 'uint32_t Wheel(byte)':  
code:31:16: error: 'strip' was not declared in this scope  
return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);  
^  
code:35:16: error: 'strip' was not declared in this scope  
return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);  
^  
code:38:12: error: 'strip' was not declared in this scope  
return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);  
^  
code: In function 'void rainbow(int, int)':  
code:44:29: error: 'strip' was not declared in this scope  
for (int i = 0; i < strip.numPixels(); i = i + 3) {  
^  
code:47:9: error: 'strip' was not declared in this scope  
strip.show();  
^  
code: In function 'void loop()':  
code:58:11: error: 'color' was not declared in this scope  
rainbow(color, 2);  
^  
code: In function 'uint32_t Wheel(byte)':  
code:39:1: warning: control reaches end of non-void function [-Wreturn-type]  
}  
^  
[ERROR] Took 3.83 seconds  
[compile] Error 1
```

Click to Attempt Compilation



code selection test

Step 9: Tone Generation Code

To stop the piezo's beeping, press the [Upload](#) button in the Arduino IDE.

Next, in software, add code to set up the system to buzz whenever the button is pressed.

Add these statements at the top of the file to define the buzzer pin and note frequency.

```
#define buzzer 4
#define freq 220
```

Then, *just before the end* of `loop()` function, set the tone to play when the button is pressed:

```
if (press) {
  tone(buzzer, freq, 10);
  delay(10);
}
```

Pass the tests to continue.

Pass the all tests in the right-hand pane to continue.



Back

Next



```
1 // Starter code - this is a comment.
2
3 #define button 2
4 #define outLED 13
5
6 #define buzzer 4
7 #define freq 220
8
9 void setup() {
10   pinMode(button, INPUT_PULLUP);
11   pinMode(outLED, OUTPUT);
12 }
13
14
15
16 void loop() {
17   int press = !digitalRead(button);
18   digitalWrite(outLED, press);
19 }
20
21
22
```

Testing



Step Tests

Click to start tests.

Begin



code selection test

```
1 // Starter code - this is a comment.
2
3 #define button 2
4 #define outLED 13
5
6 #define buzzer 4
7 #define freq 220
8
9 void setup() {
10
11     pinMode(button, INPUT_PULLUP);
12     pinMode(outLED, OUTPUT);
13
14 }
15
16 void loop() {
17
18     int press = !digitalRead(button);
19     digitalWrite(outLED, press);
20
21 }
22
23 if (press) {
24     tone(buzzer, freq, 10);
25     delay(10);
26 }
```



Testing

Unpassed Tests
Pass the tests to continue.

Next

Code Selection Test

Highlight the part of your code which turns on the buzzer.

Make sure to include the buzzer code line as described in the left pane.

```
int press = !digitalRead(button);
```

Check Code Selection

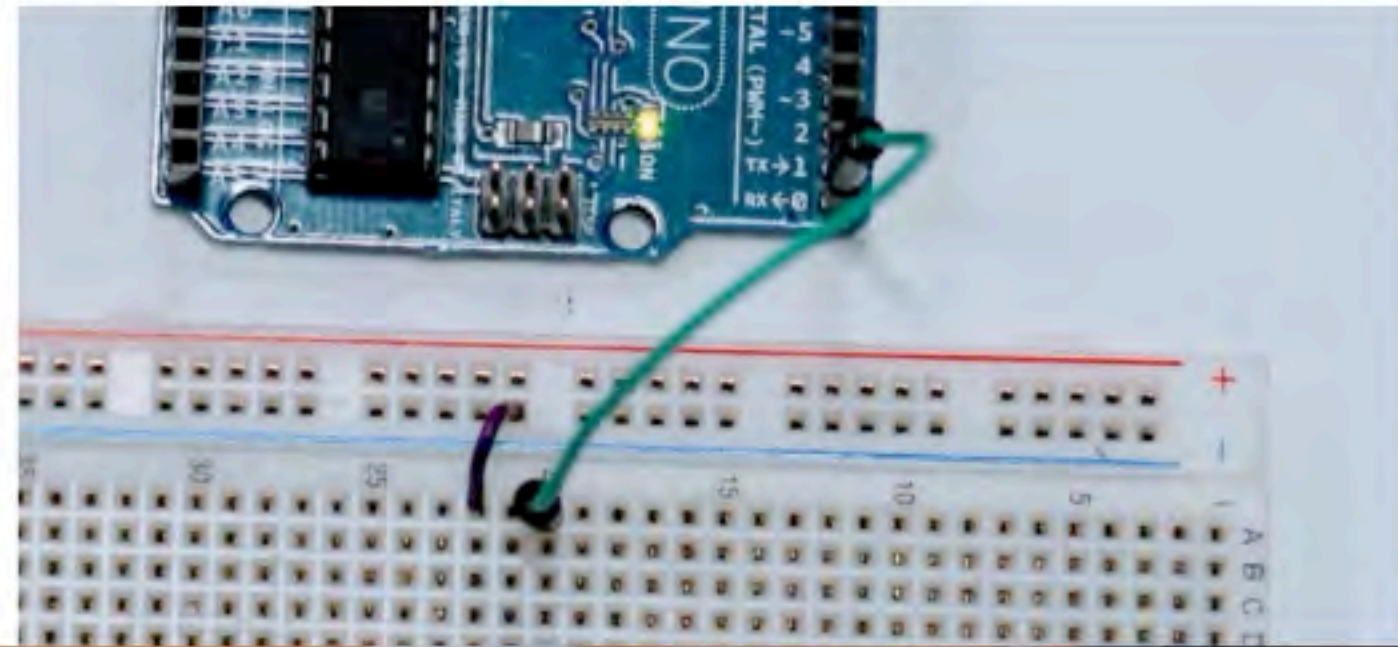
knowledge test



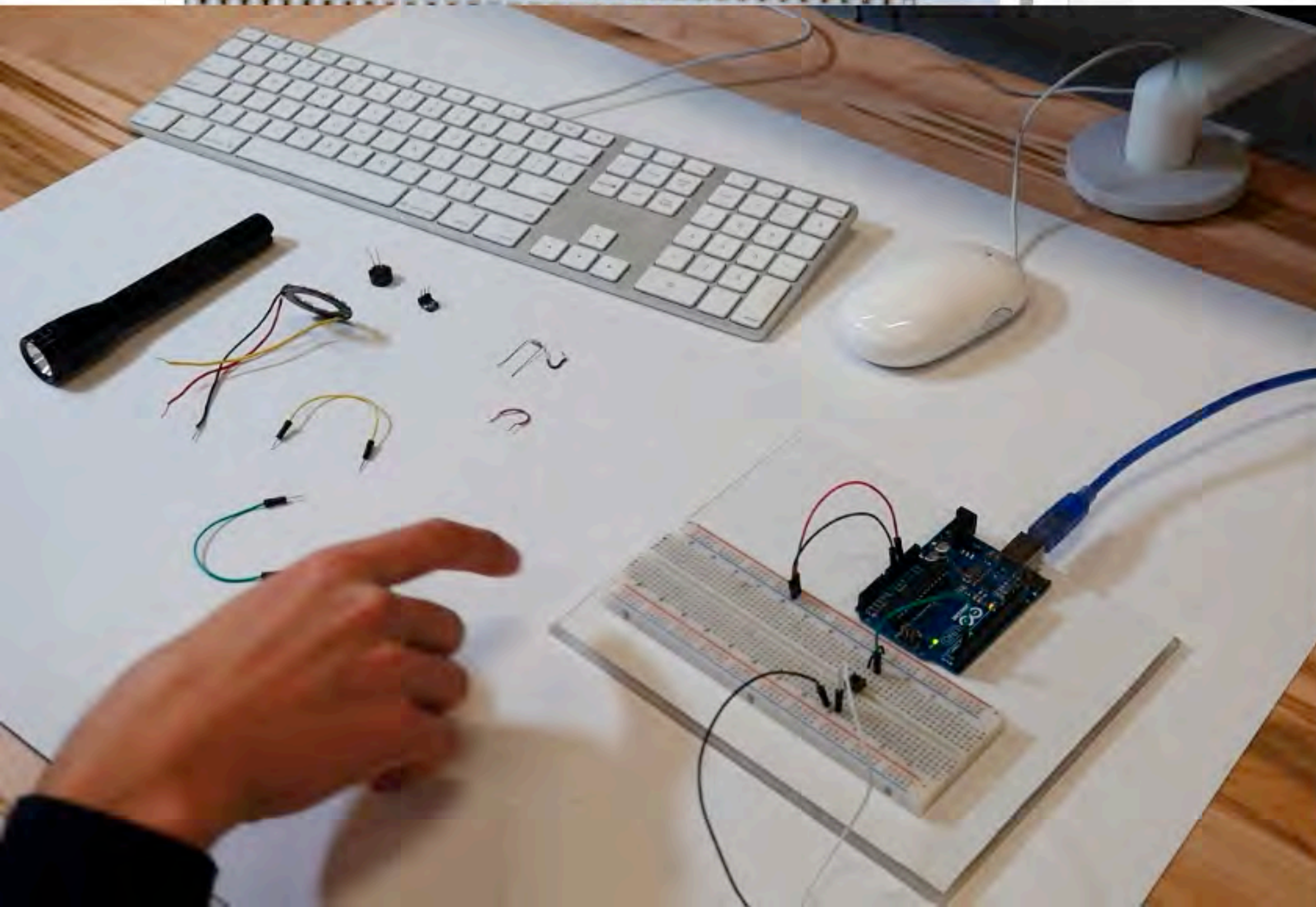
Insert the button onto the breadboard, as pictured.

Connect the one side of the button to **GND** (blue/ground) with a jumper wire.

Connect the other side to **digital pin 2** on the Arduino board.



```
3  
4 void setup() {  
5  
6 }  
7  
8 void loop() {  
9  
10 }  
11
```



Next

✔ **Button Press Check**

ⓘ **Voltage Knowledge Test**

What voltage difference exists between the power (red) and ground (blue) rails?

0V

10V

5V

3.3V

Submit Choice

ⓘ **Hardware Wiring Check**

REMOVE THE WHITE AND BLACK PROBES FROM THE CIRCUIT BEFORE DOING THIS TEST

The system will now check to see that you have wired up the board correctly.

Does the Arduino light turn on as you push the button?

Click the button below to begin the test.

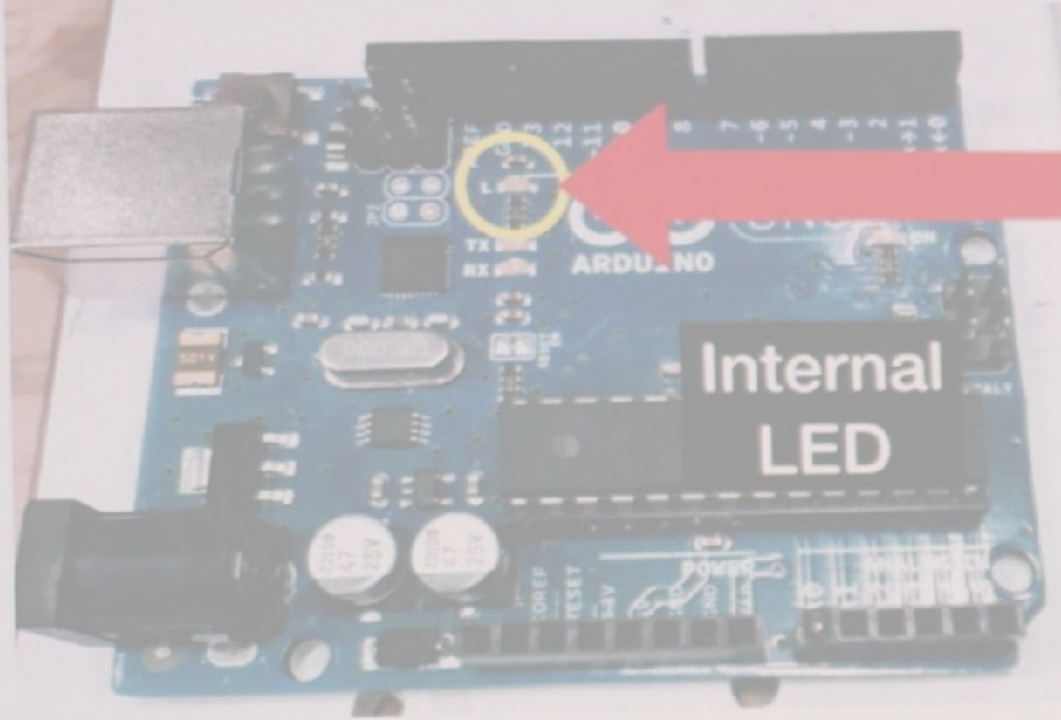
Click to Setup and Begin Test


```
pinMode(button, INPUT_PULLUP);
pinMode(outLED, OUTPUT);
```

Last, add this code to the `loop()` function to read the button's value.

```
int press = digitalRead(button);
digitalWrite(outLED, press);
```

We use a `!` to negate the reading, since when we press the button we pull the pin value to ground (0V). This code will also copy the inverted button state to the onboard Arduino LED, turning it on (high) when we press the button.

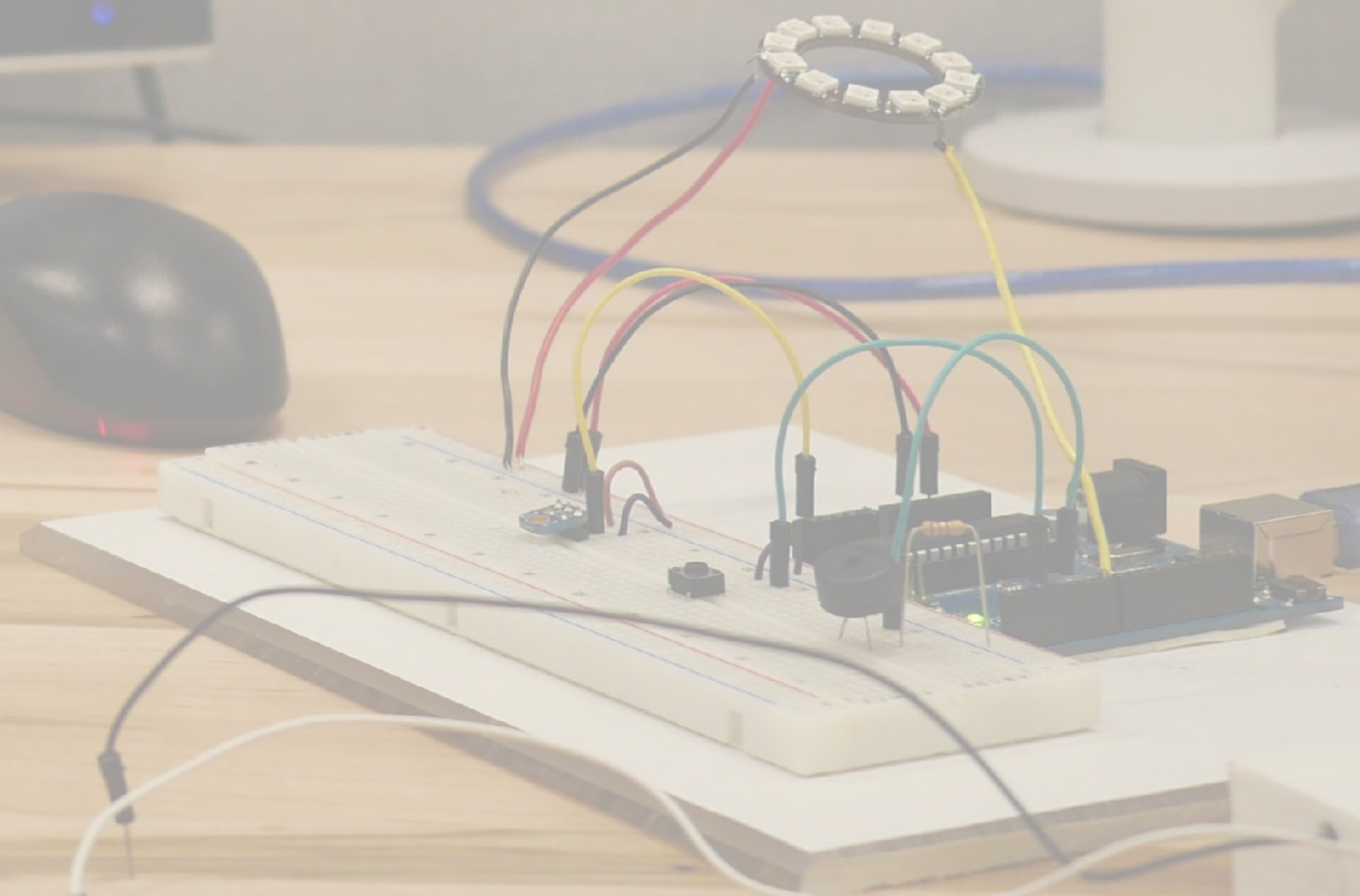
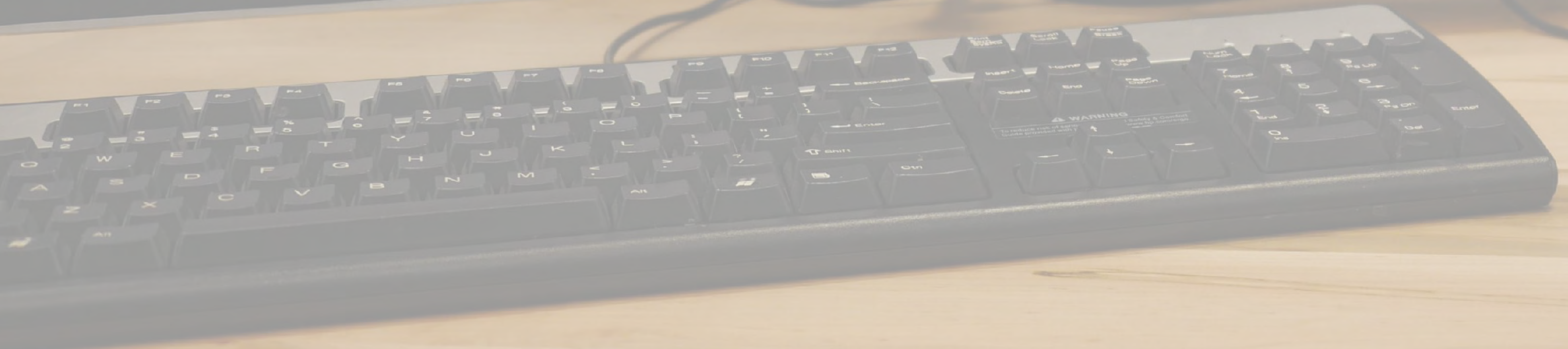


Internal LED

Pass the tests to continue.
Pass the all tests in the right-hand pane to continue.

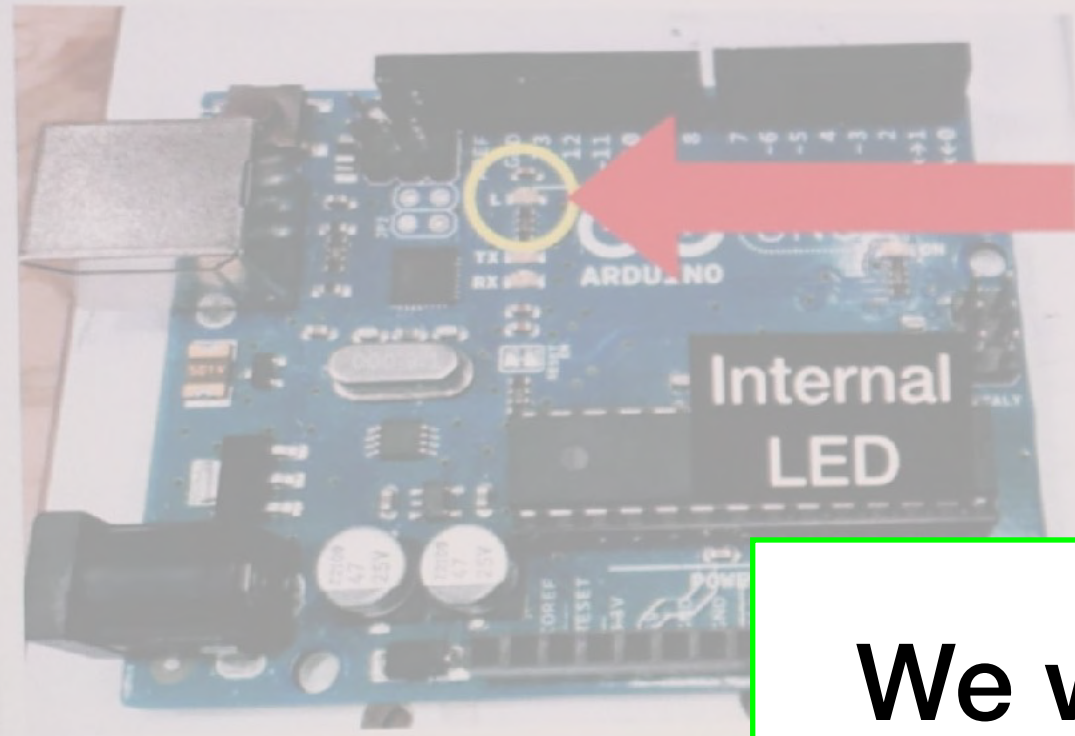
< Back Next >

evaluation



evaluation

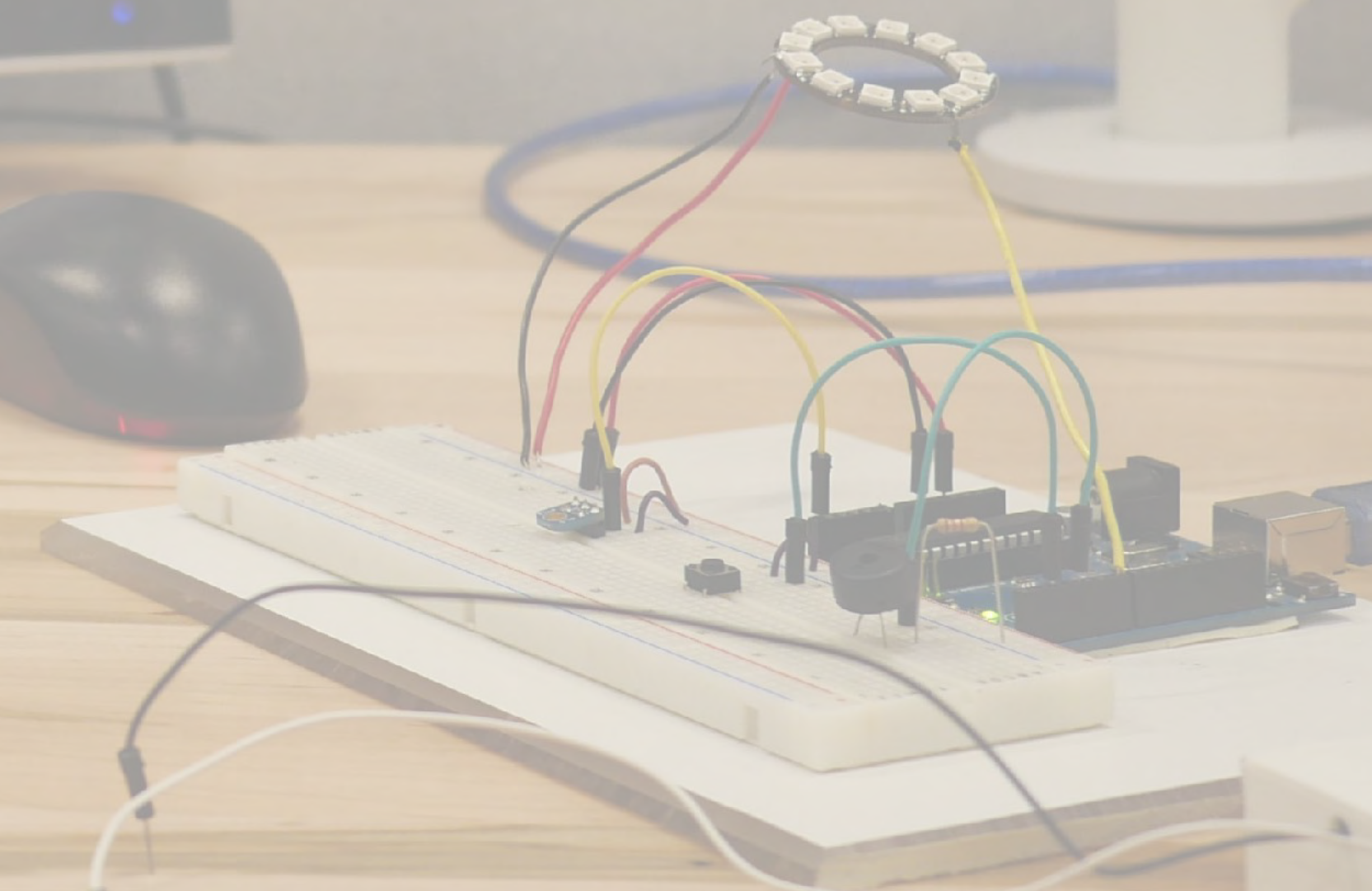
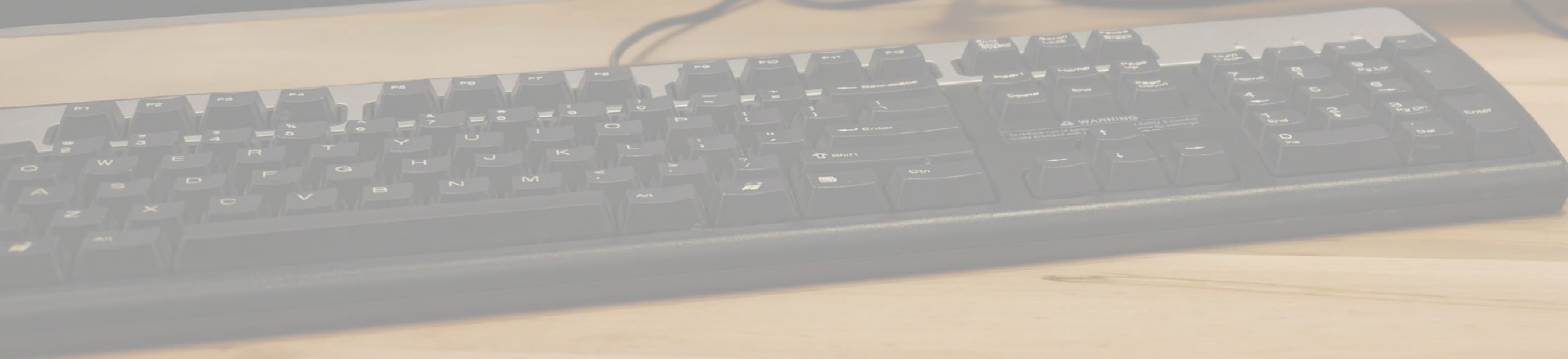
We were interested in seeing how ElectroTutor would shape participants' *usage patterns*, *knowledge retention*, and *subjective experiences* while working with the tutorial.



Pass the tests to continue.
Pass the all tests in the right-hand pane to continue.

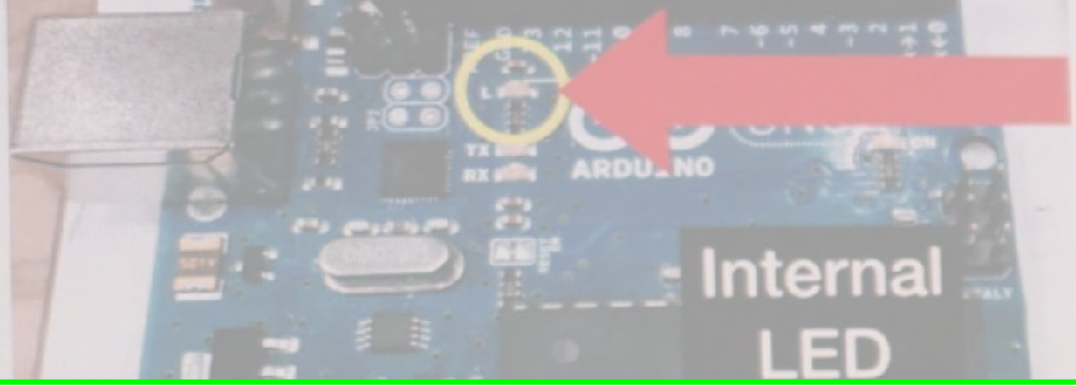
< Back

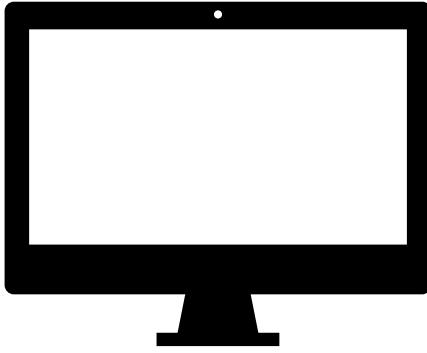
```
(Thu Apr 5 02:23:41 2018) Processing uno (platform: atmelavr) board: uno; framework: arduino  
(SUCCESS) Took 4.03 seconds
```

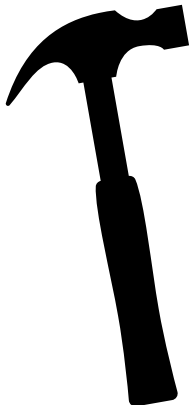


evaluation

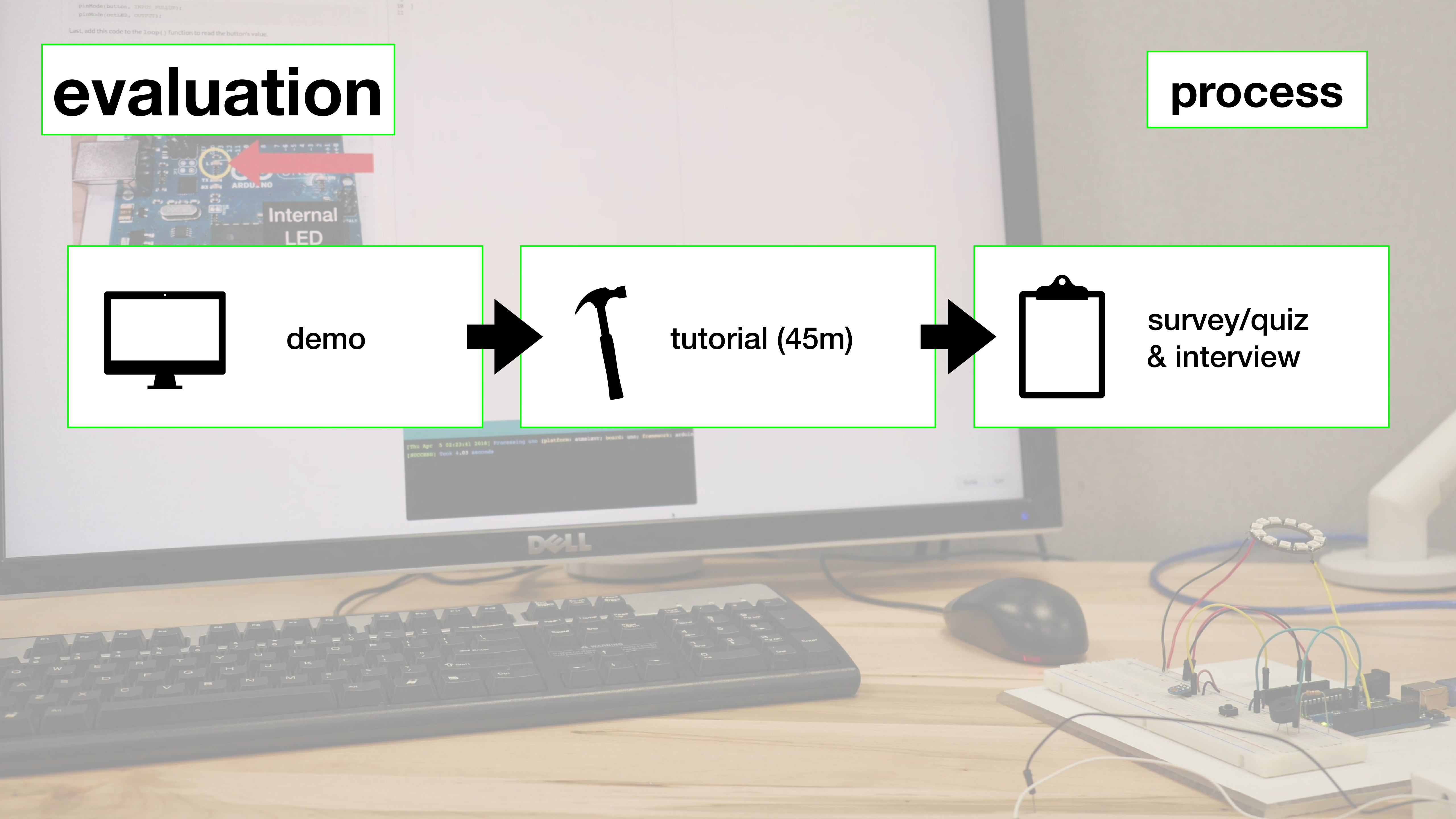
process



 demo

 tutorial (45m)

 survey/quiz & interview



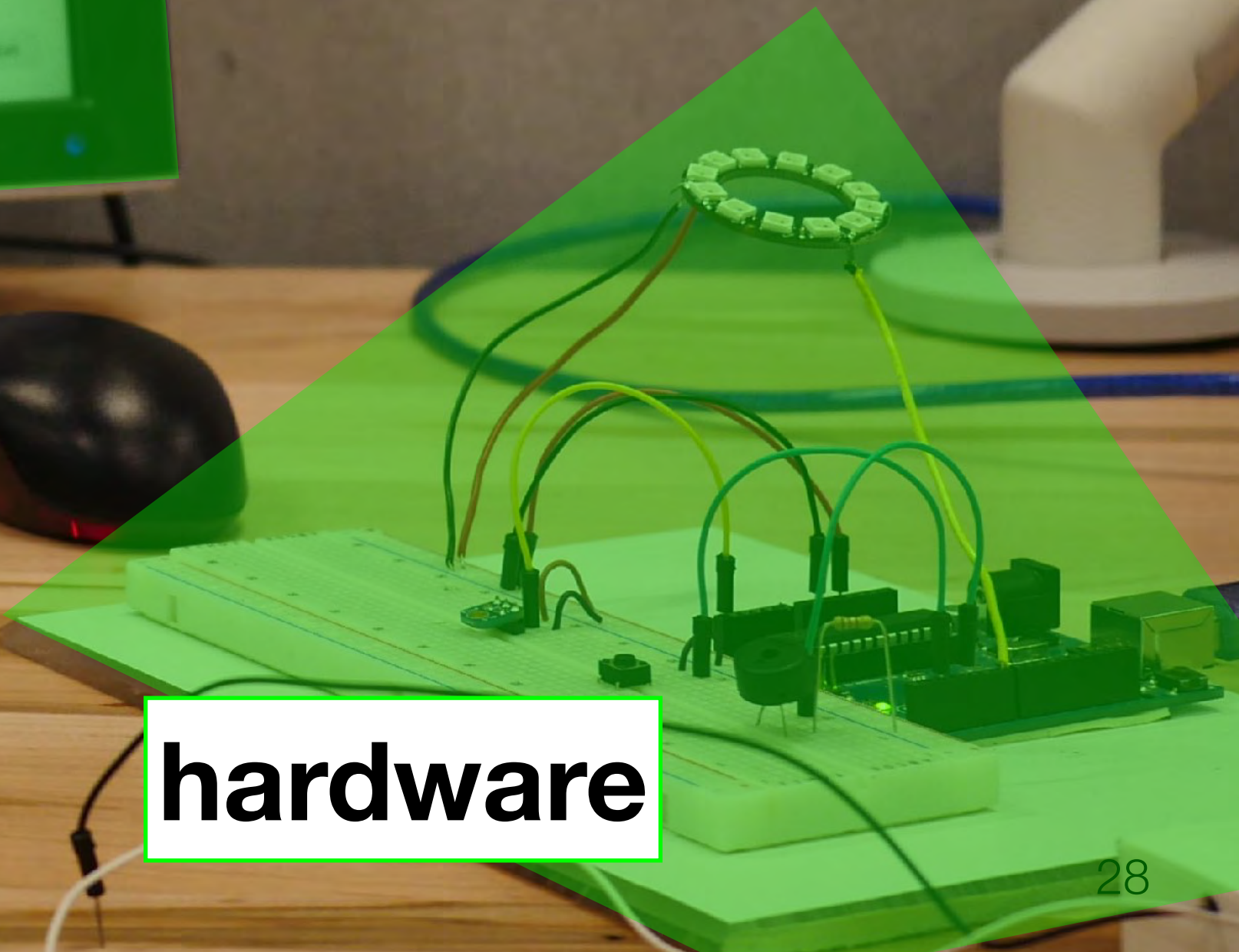
evaluation

setup



software

```
(Thu Apr 3 02:23:41 2018) Processing uno (platform: atmelavr) board: uno; framework: arduino
[SUCCESS] Took 4.03 seconds
```



hardware

evaluation

conditions

ElectroTutor

tests enabled, pass to progress

Control

no tests, no progress restrictions

evaluation

participants

participants

12 total (10 male, 2 female, ages 20-54)

study design

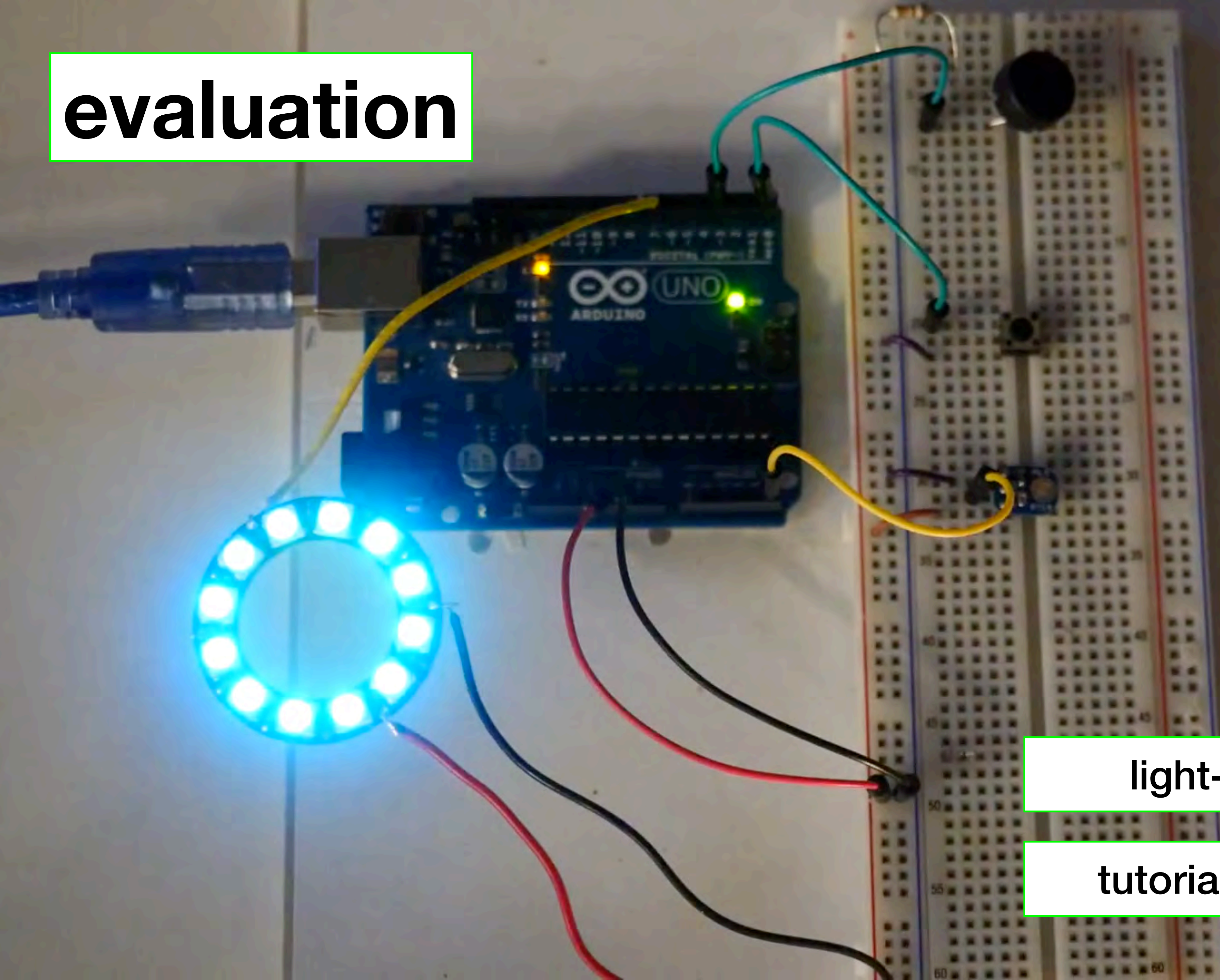
between-subjects (6 ElectroTutor, 6 control)

skill

no electronics work, varied software skill

evaluation

project



light-sensitive alarm clock

tutorial has 16 steps, 24 tests

results

completion

completion rate

timing

time to completion

knowledge

post-tutorial quiz

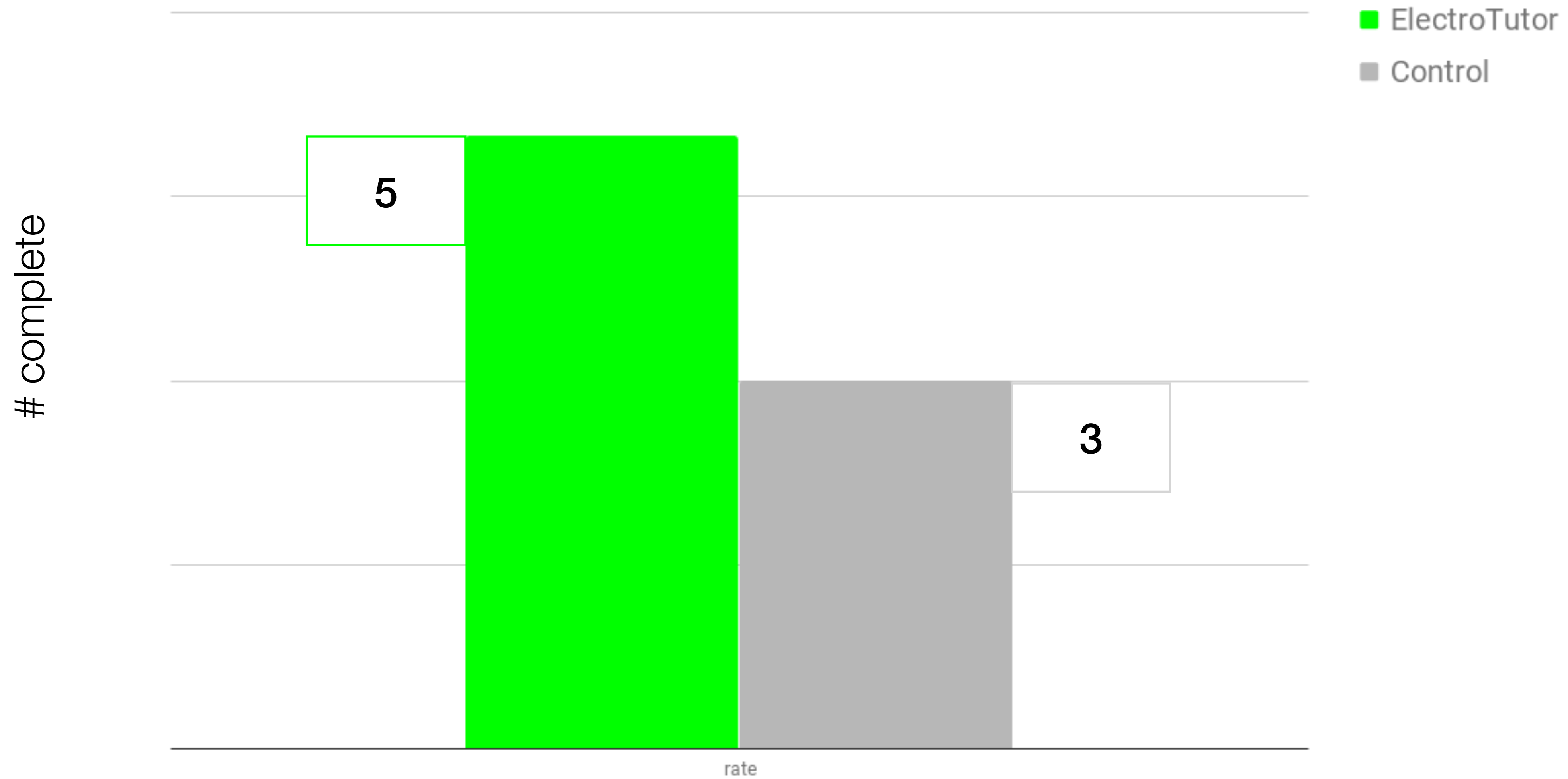
back steps

progression patterns

results

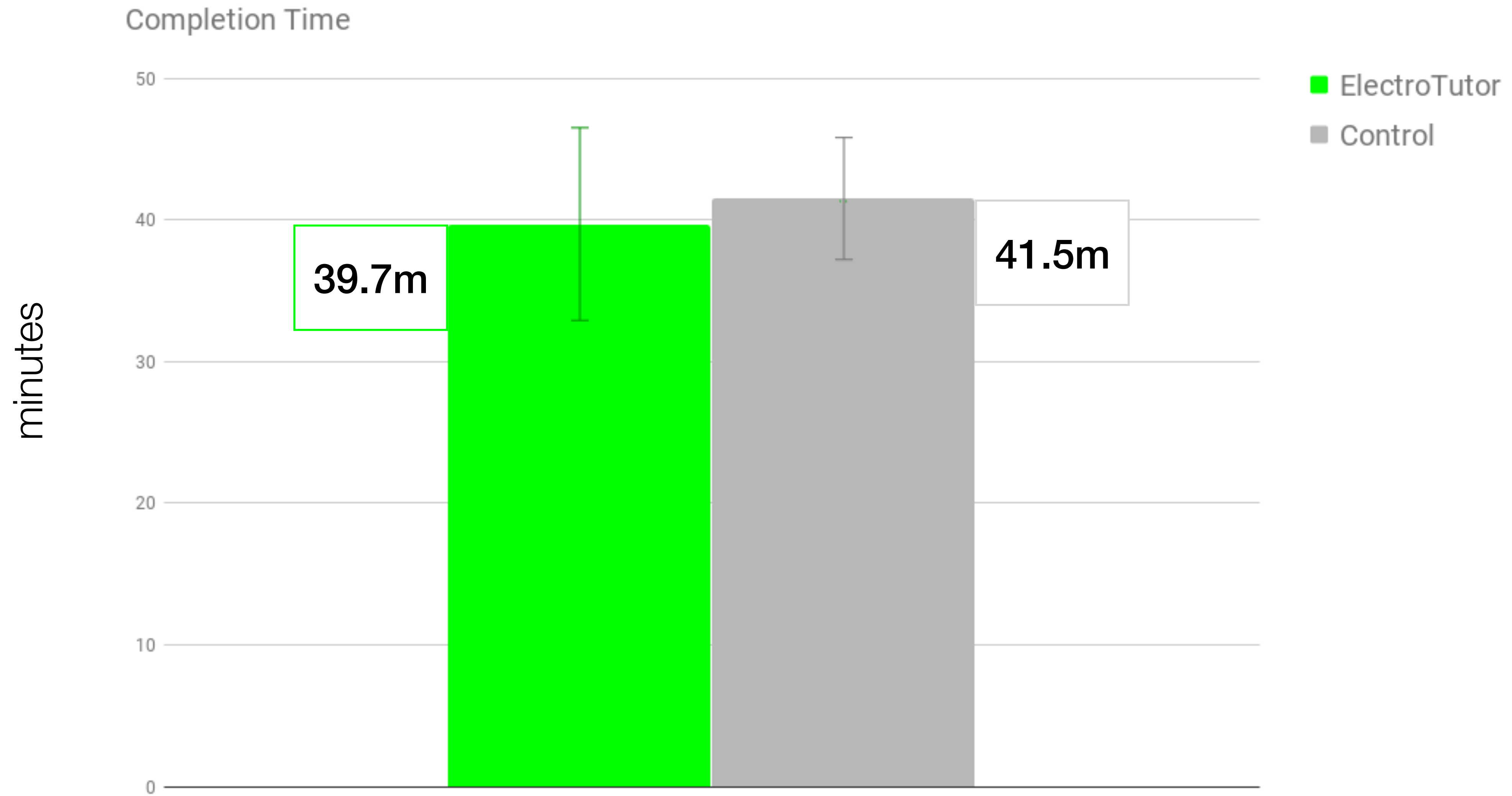
max = 6

completion



results

timing

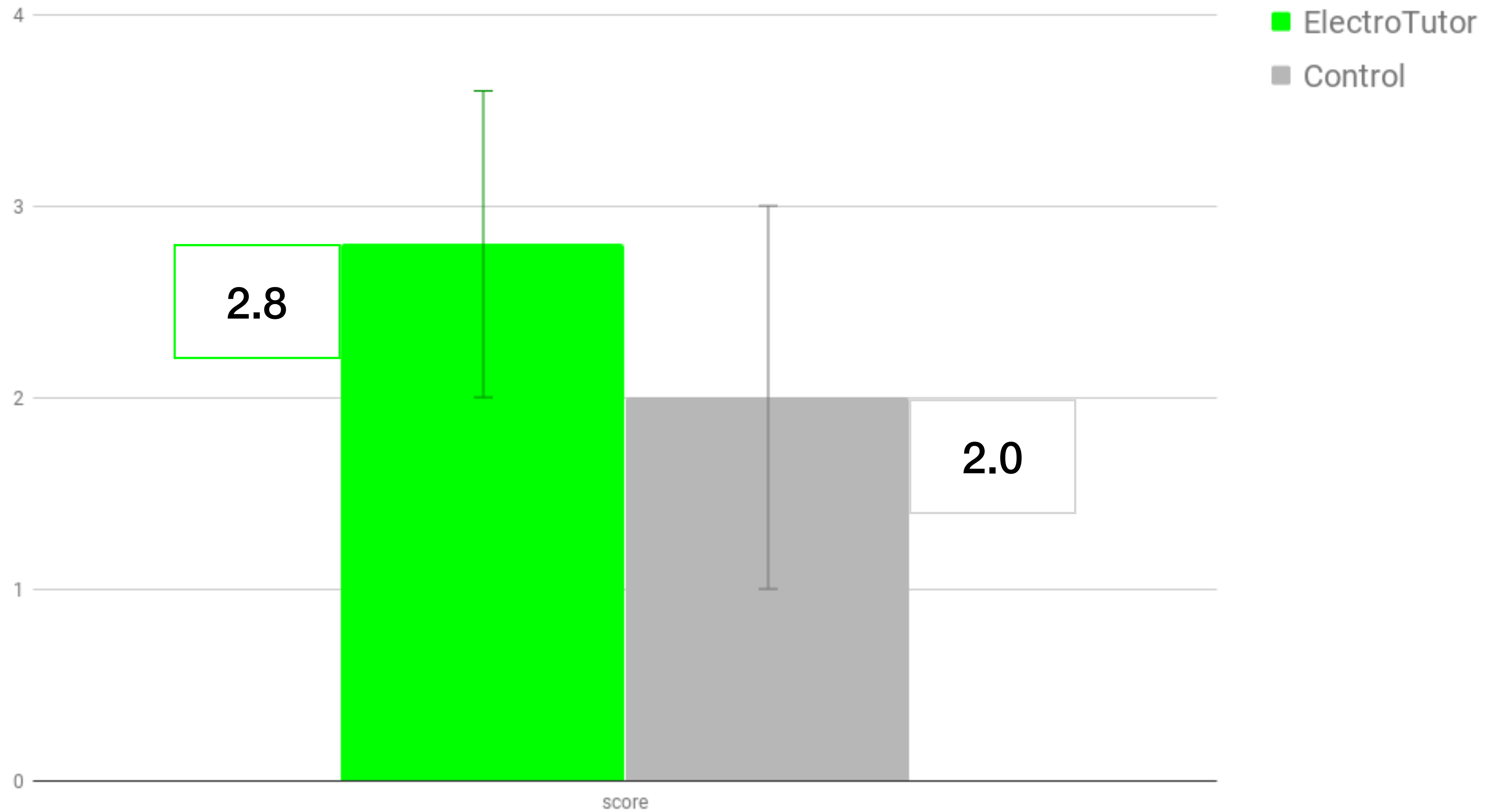


results

max score = 4

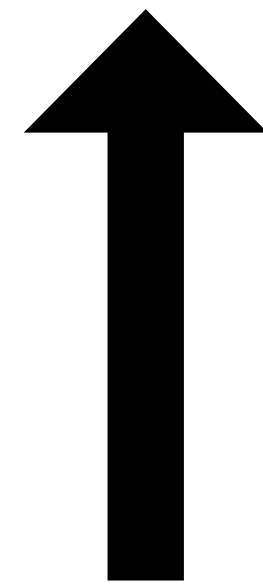
knowledge

Knowledge Test

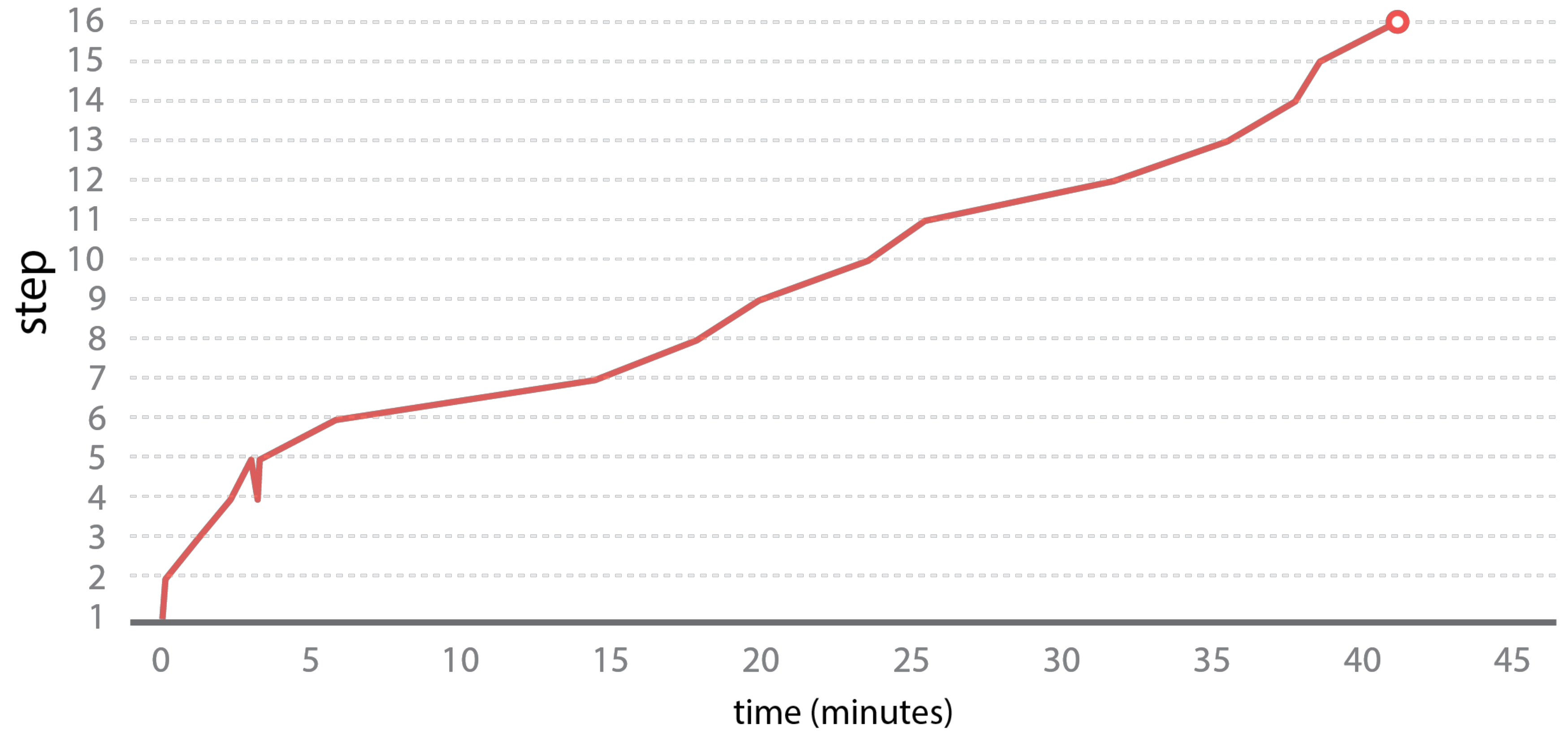


results

back steps



progress

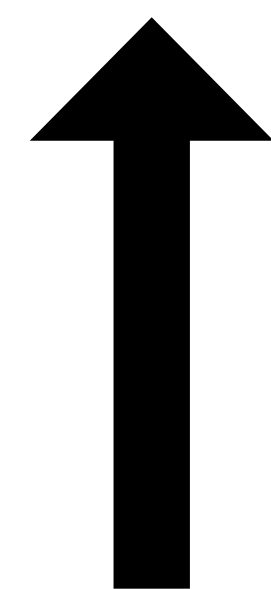


time

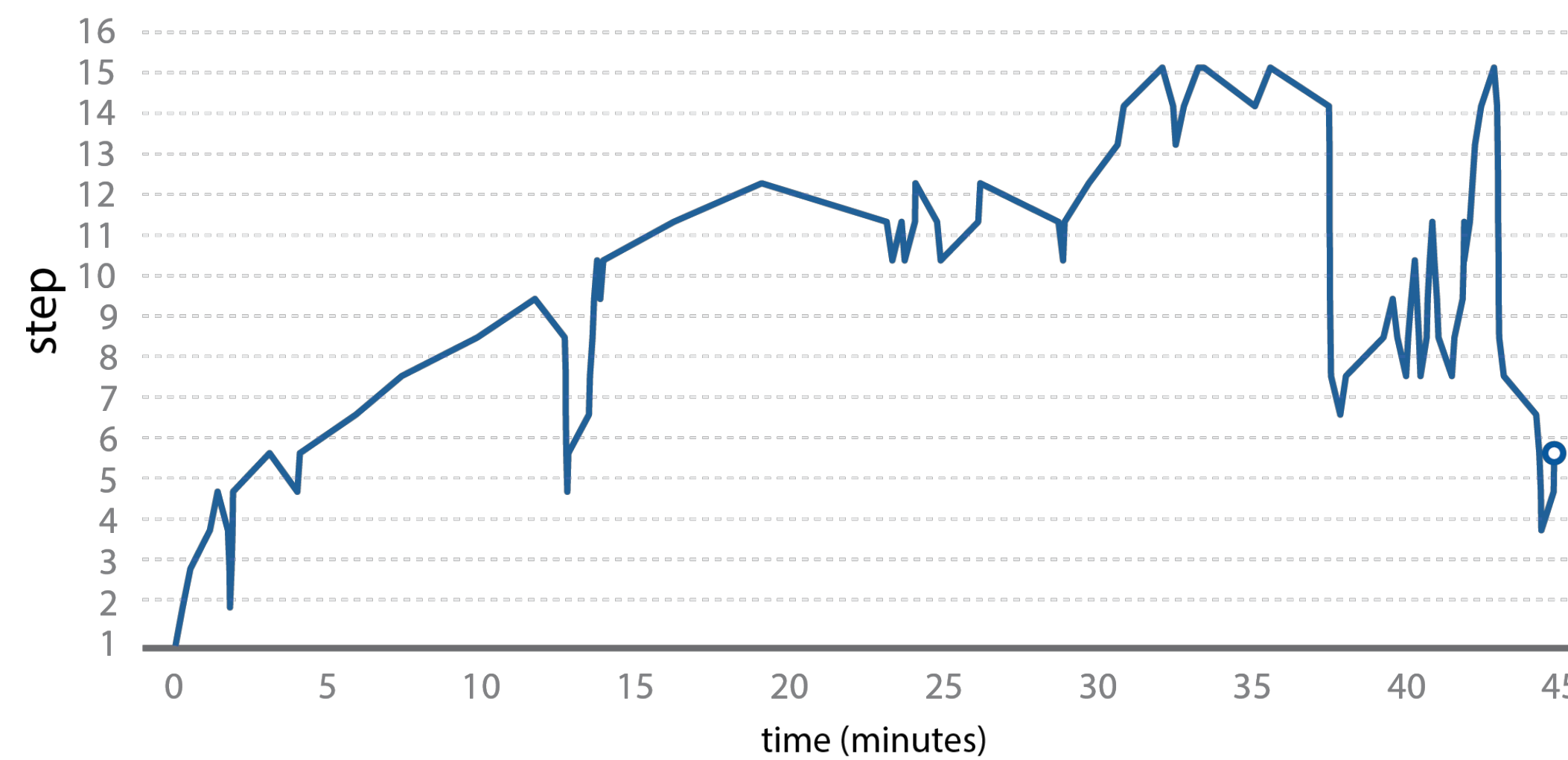
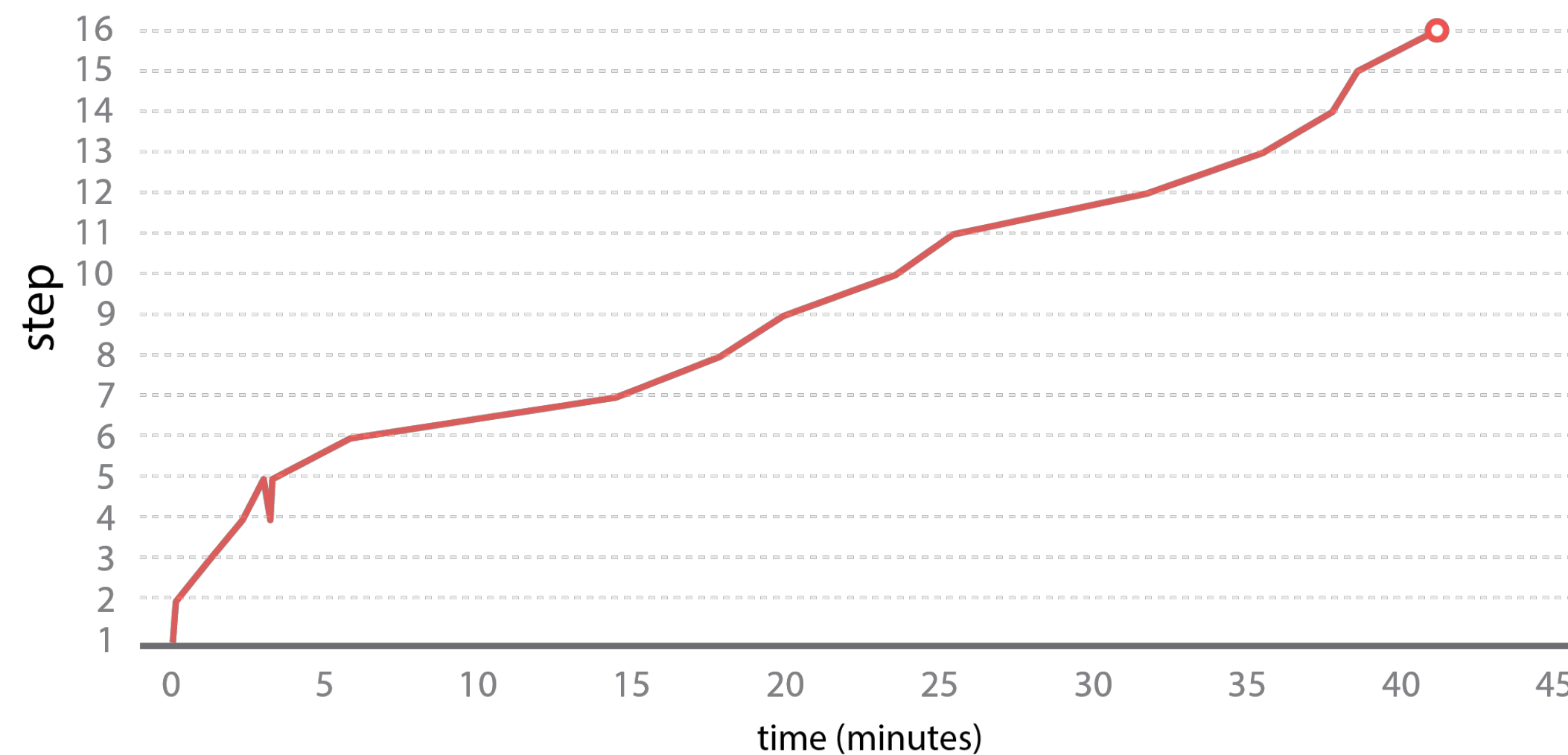


results

back steps



progress

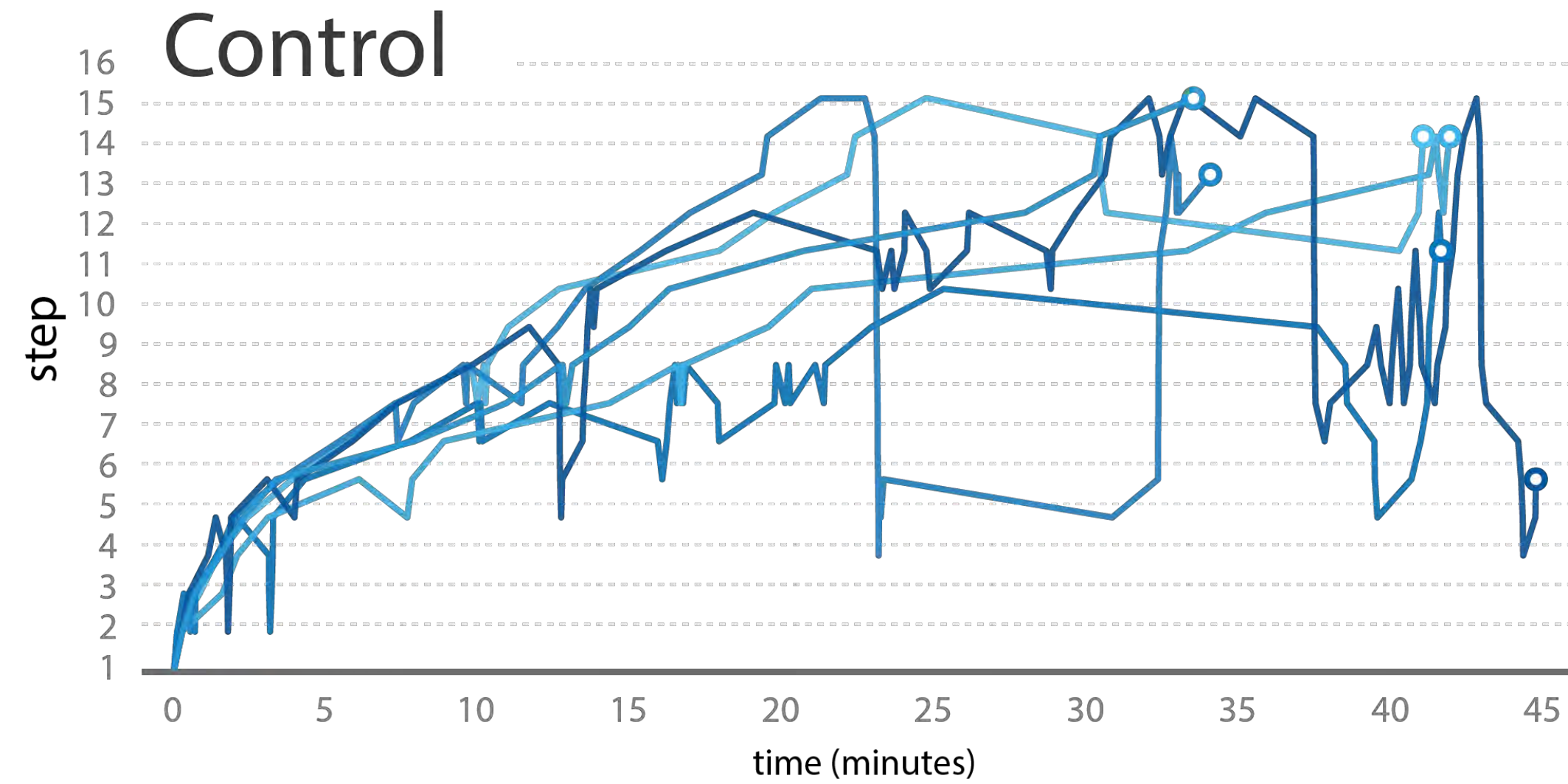
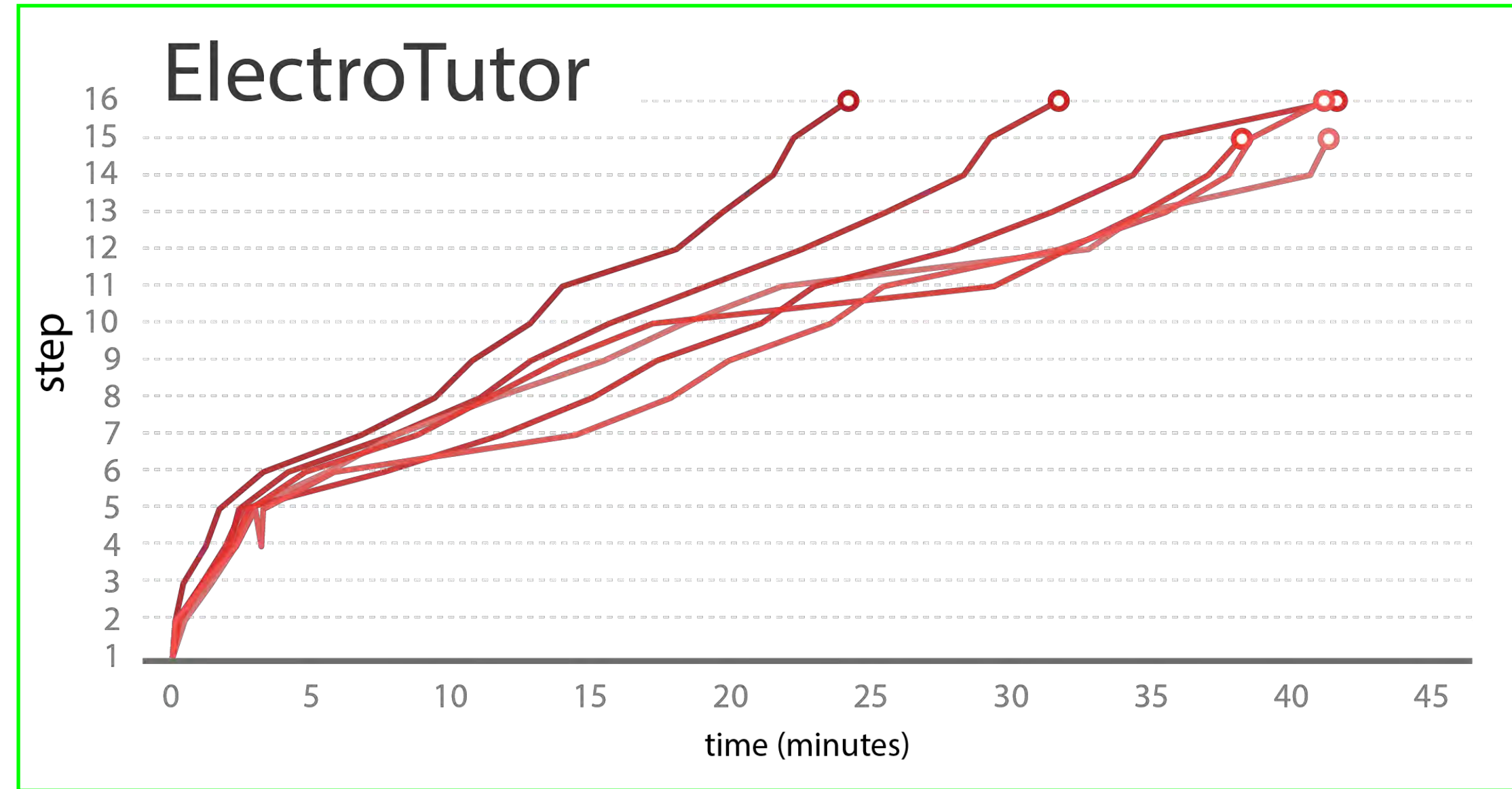


time



results

back steps



↑
progress

time →

feedback

ElectroTutor

tests enabled, pass to progress

It's fun making things happen. I was never good at electronics in university, so... it's fun and breaks it down really nicely to make sure that ***you feel very comfortable where you are before you go onto the next thing.***

The tests were definitely helpful. I especially liked the fact that tests were done in a granular manner at each step along the tutorial, so that ***I felt confident throughout the tutorial.***

feedback

Control

tests disabled, no restrictions

I struggled with the hardware part, I was not that familiar with it. It's kind of like, I am following the instructions but ***I am just not sure if I am doing it the right way.***

Did a good job of guiding me to successful completion, but ***did not really educate me on what I was doing.***

future work

1. tutorial authorship interfaces
2. activity review dashboards
3. skill-aware dynamic tutorials
4. supporting formal learning



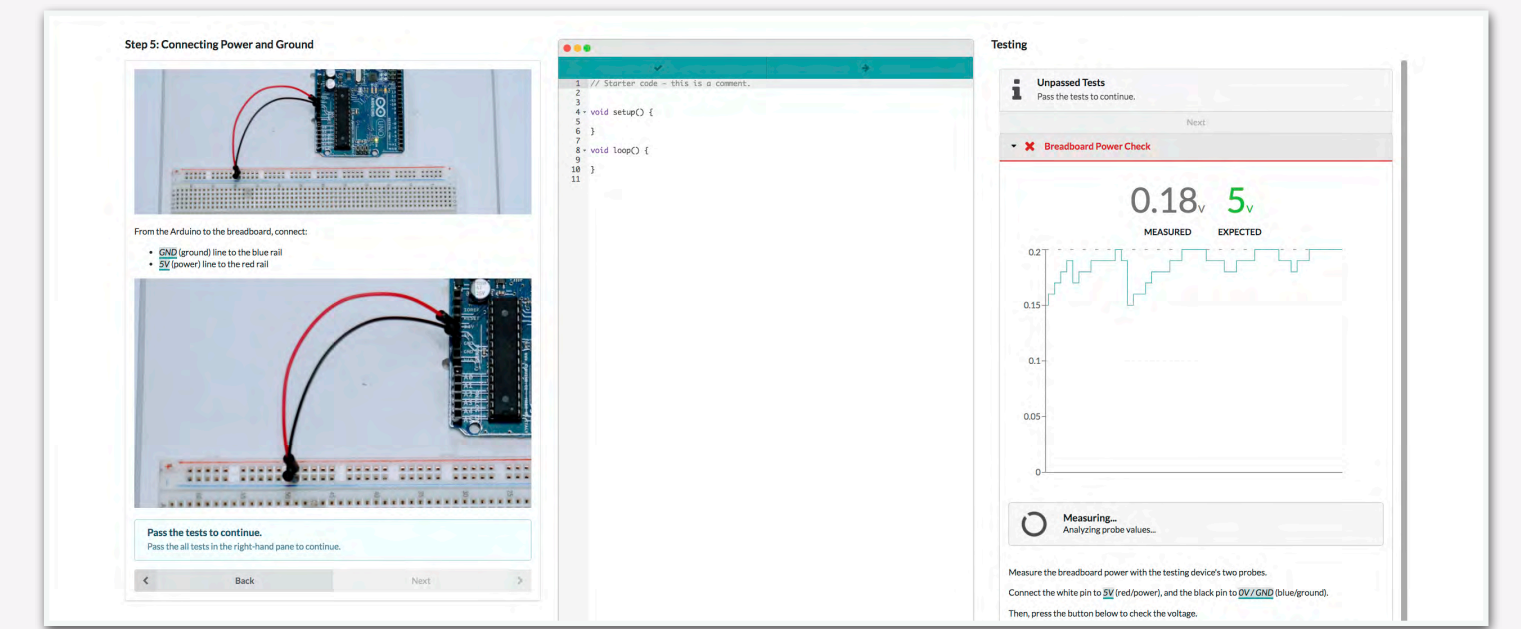
acknowledgments

Thanks to my collaborators, the Autodesk User Interface Research Group, and David Mellis.



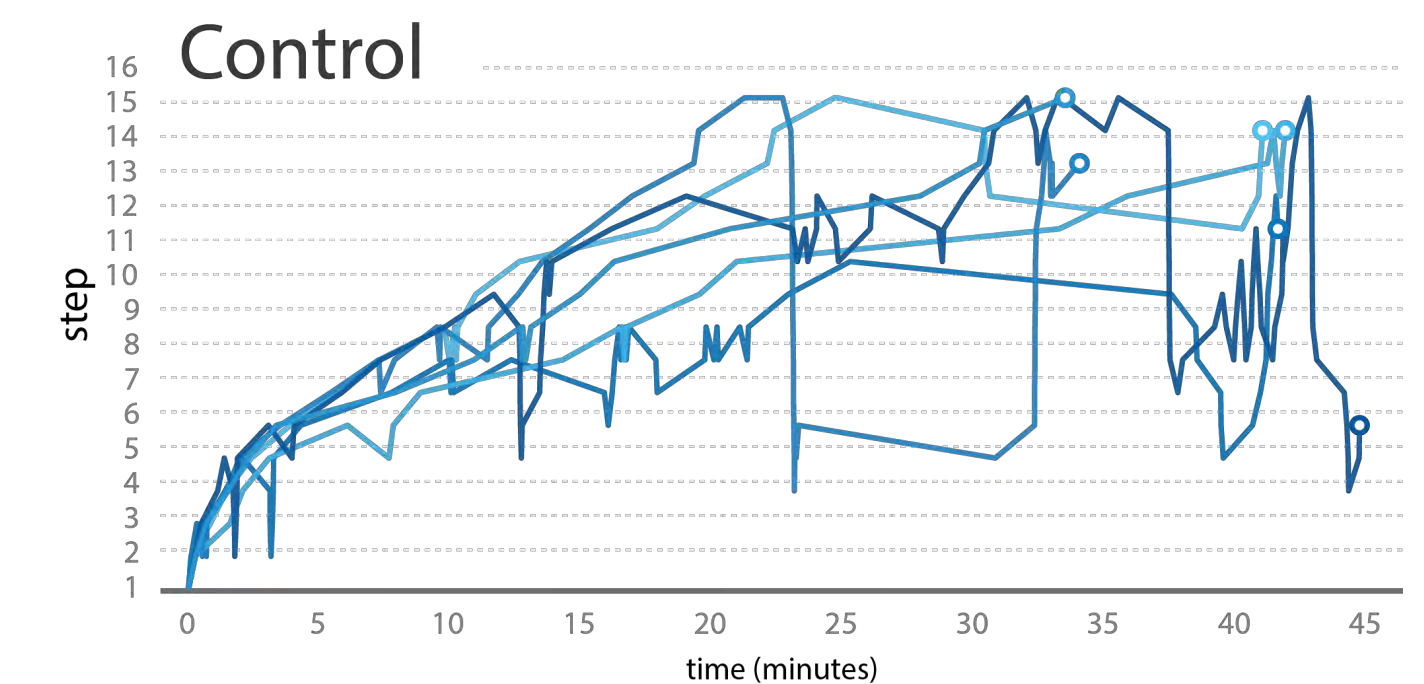
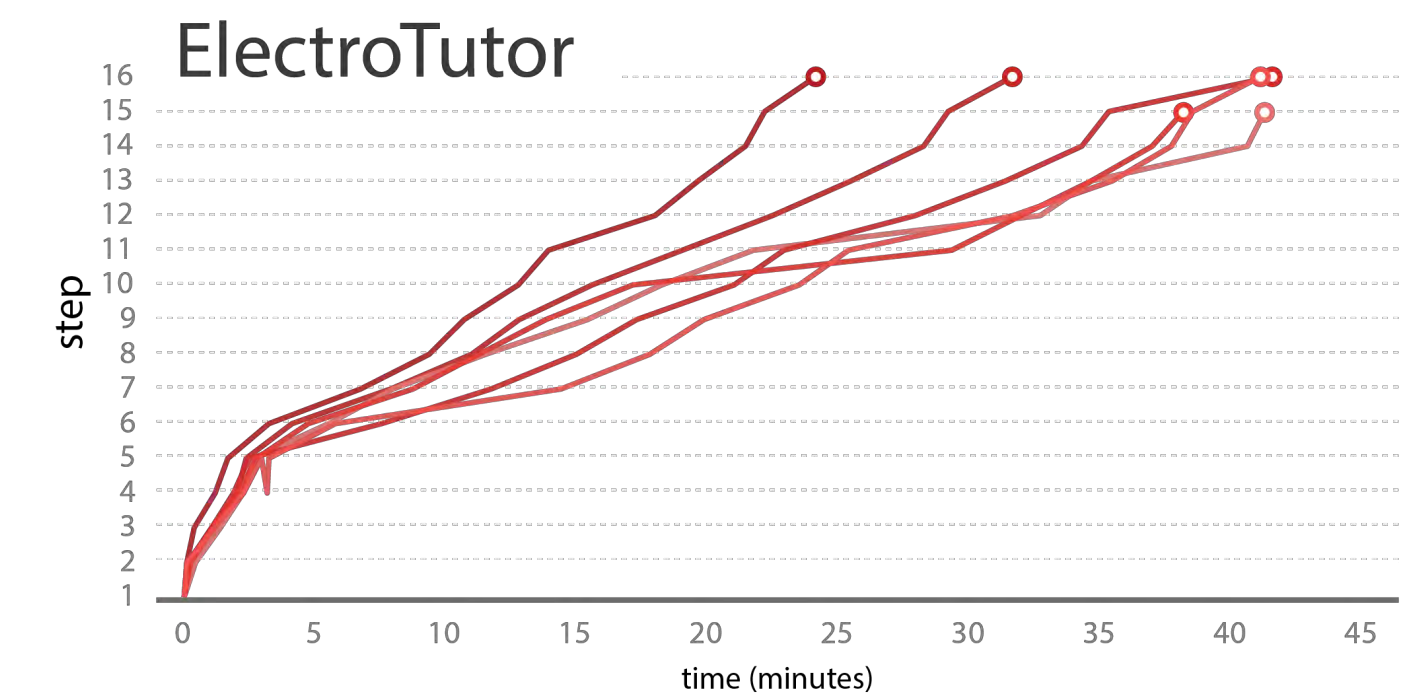
ElectroTutor:

Test-Driven Physical Computing Tutorials



ElectroTutor is a novel system that *integrates tests into physical computing tutorials*. Novices can benefit from incremental tests in physical computing tutorials.

Incremental tests aid novices in problem scoping, and preliminary work points towards the potential for *improved tutorial completion rates*.



Jeremy Warner

Ben Lafreniere

George Fitzmaurice

Tovi Grossman

Thanks!

<http://bit.ly/electrotutor>



@jeremywrnr



jwrnr@berkeley.edu