Arduino for the Classroom

Let's Talk Science Instructional Design Team

Titus Ferguson, Shawn Nock

UnPacked Digital Literacy

Goals

By the end of this class, you'll:

- Be familiar with basic electionics concepts
- Assemble a kit of sensors into a complete system
- Know how to program the Arduino platform to collect and plot the sensor data

Enabling Exploration, Creativity, and Excellence In Art+Make+Tech

Challenging and embracing ideas related to new technologies and social platforms through the education, entertainment and engagement of our membership and the community-at-large.

- 121Studios: Coworking for Creatives
- STEAM Outreach & Education
- ExplodeConf
- Nuit Blanche

What's in your kit?

Kit Contents

Arduino Uno R3

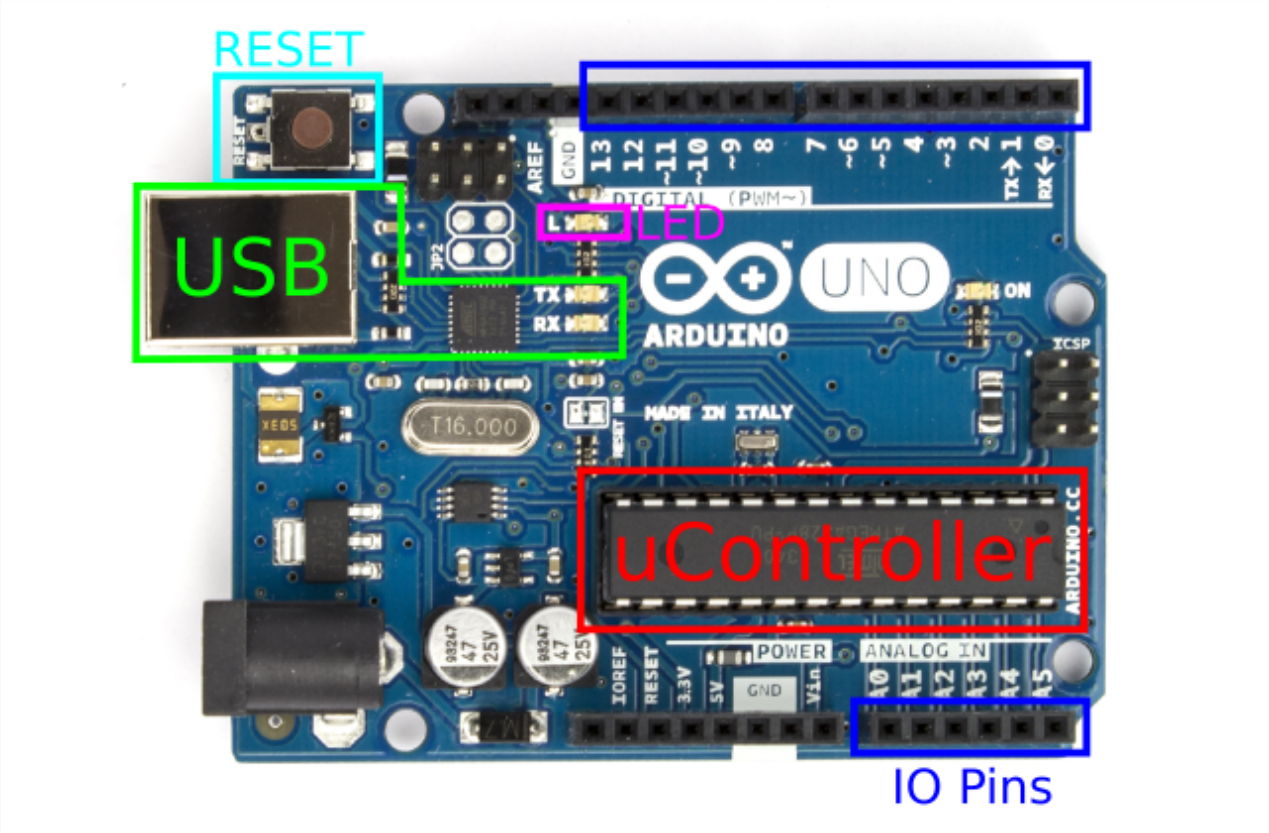
Solderless Breadboard

Connecting wires

Sensors

- Temperature / Humidity
- Light Sensor
- CO2 Gas Sensor

What is Arduino?



It's a kit (on a board) with the bare minimum components to easily use the μC hardware. They do the basic, boring design needed for any board, so users only need to add the neat stuff.

Arduino Software

The Arduino folks also adapted an *Integrated Development Environment* (IDE) to their boards. This IDE allows us to easily write programs for their boards and then transfer the programs to the μC.

Get the Arduino IDE:

`https://www.arduino.cc/en/Main/Software`

Get the libraries and sample code for this workshop:

https://monadnock.ca/lts-workshop-Nov2017.zip

Circuit Basics

Current

Current is the flow of charge through a circuit. Measured in Amperes (A).

Resistance / Impedance

Circuits have a resistance to current flow that depends on the parts in the circuit.

Measured in Ohms (Ω)

Voltage

Voltage is a potential (akin to a pressure) pushing the current through a circuit. Current is said to flow from higher (+) voltage to lower (-) voltage.

Measured in Volts (V)

Ohm's Law; Light

Voltage, Current and Resistance are related to each other.

- As voltage increases, current increases
- As voltage decreases, current decreases
- As resistance increases, current decreases
- As resistance decreases, current increases

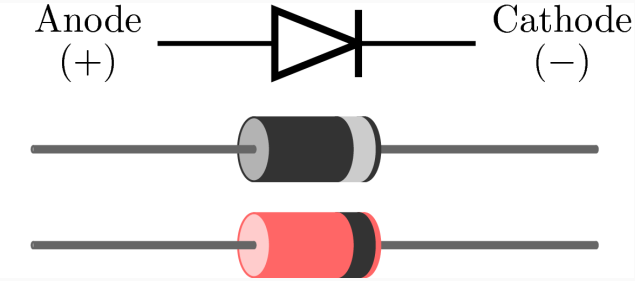
V, Ω, A: **The Water Analogy**

If charge were water, then:

- resistance = obstacles blocking flow
- current = flow rate
- voltage = change in height *or* pressure.

Diode

- One way value for current[1]
- LED ≡ Light Emitting Diode
- Band marks (-)[2]
- Longer leg marks (+)



[1] https://learn.sparkfun.com/tutorials/diodes

[2] https://learn.sparkfun.com/tutorials/polarity/diode-and-led-polarity

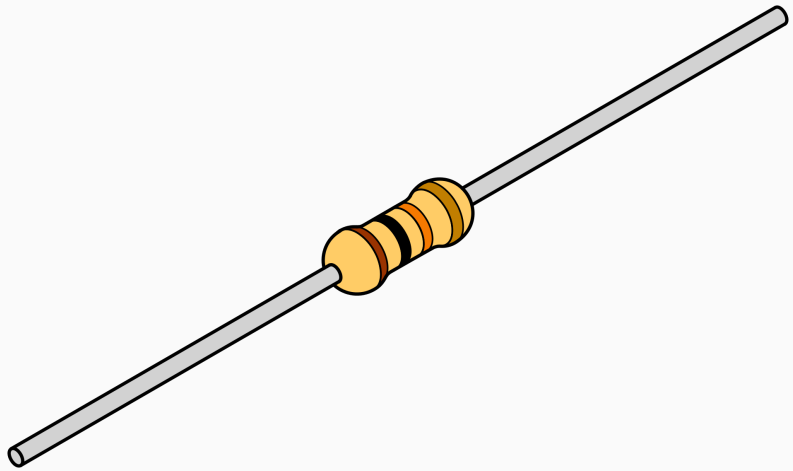
Diode Problems

- Diodes don't limit current
- Diodes aren't perfect (some current turned to heat)
- Too much current $\rightarrow$ Too much heat $\rightarrow$

What's that smell?

Resistor

- *Resists*/limits the flow of current
- Needed for LEDs: $\approx 1000\,\Omega$
- Button Pull-up/down: $\geq 10\,\mathrm{k}\Omega$
- Color coded, Google it



Buttons

- Buttons connect *or* disconnect two wires/parts
- Momentary Switch: Normally Closed (NC), Normally Open (NO)
- Toggle Switch

Circuits

A circuit is a completed loop from HIGH potential (voltage) to LOW, which causes current to flow through some other components along the way.

Transducers

Often these *other* components are *transducers*, which convert electrical energy into another sort of energy:

Speaker	Electrical $\rightarrow$ Sound
Microphone	Sound $\rightarrow$ Electrical
LED	Electrical $\rightarrow$ Light
LED	Light $\rightarrow$ Electrical
Piezoelectric	Electrical $\rightarrow$ Motion

Power

The power supply provides the energy to drive the system.

Can be a:

- Voltage Regulator (converts one potential to another)
- Batteries
- Solar Panel

In our circuits, your computer is converting it's power source to 5 V and delivering power to our circuit via USB.

μController

Microcontroller (μC) is a *processor*, *memory* and a few *peripherals* on a standalone chip.

Processor	is a group of transistors that understands a dozen or so commands (ADD, SUB, JUMP..)
Memory	a circuit that can hold values.
Peripherals	Vary chip to chip, but often include timers, communications and ADC, DAC.

Seems complicated, but really simple. They read a command from the start of memory, then execute the command. At the end of the command, read the next command from the next memory cell and repeat[1]

[1] some commands change the address of the next fetched command

Digital Signals

- Vcc: The power supply of the circuit elements
- GND: The reference voltage (usually 0 V)
- Connecting a part to Vcc = Logical 1 or High
- Connecting to GND = Logical 0 or Low
- Connecting various pins to Vcc or Ground is all the μC can do to talk to the world [2]

[2] w/o fancy peripherals or dirty tricks

μController INPUT and OUTPUT

Most of the pins on the Arduino can be set for INPUT or OUTPUT mode.

- INPUT mode pins listen for a signal (High or Low) from another device
- OUTPUT mode pins drive the pin High or Low

Floating Pins

What's happens if an INPUT mode pin tries to read the value of a pin that is connected to nothing? Is that a 1 or 0?

No one knows!

It's dependant of transient charges, static, nearby electric fields, the phase of the moon, . . . Whenever you want to check a digital signal, make sure that something is *driving* it (ensuring Vcc or GND).

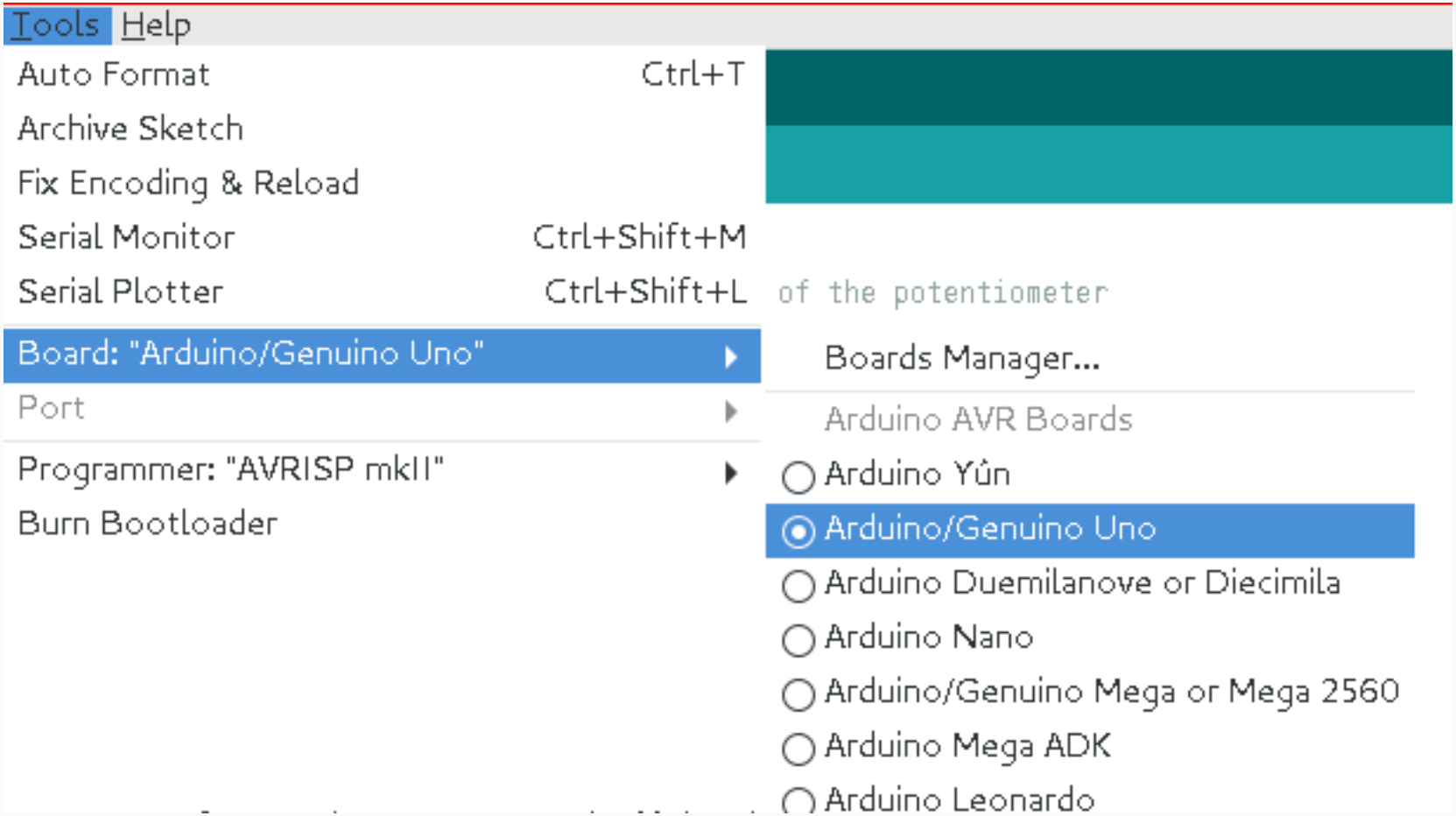
μC + Digital Signals as Switches

If one end of an LED is connected to ground, and the other end is connected to an OUTPUT pin on a μController, then:

- If the μC sets the pin High (Vcc, 5 V) then current will flow from the pin through the LED and turn it on.
- If μC sets the pin Low (GND, 0 V) then the current will not flow and the LED is off.

Let's start programming

Configure Arduino



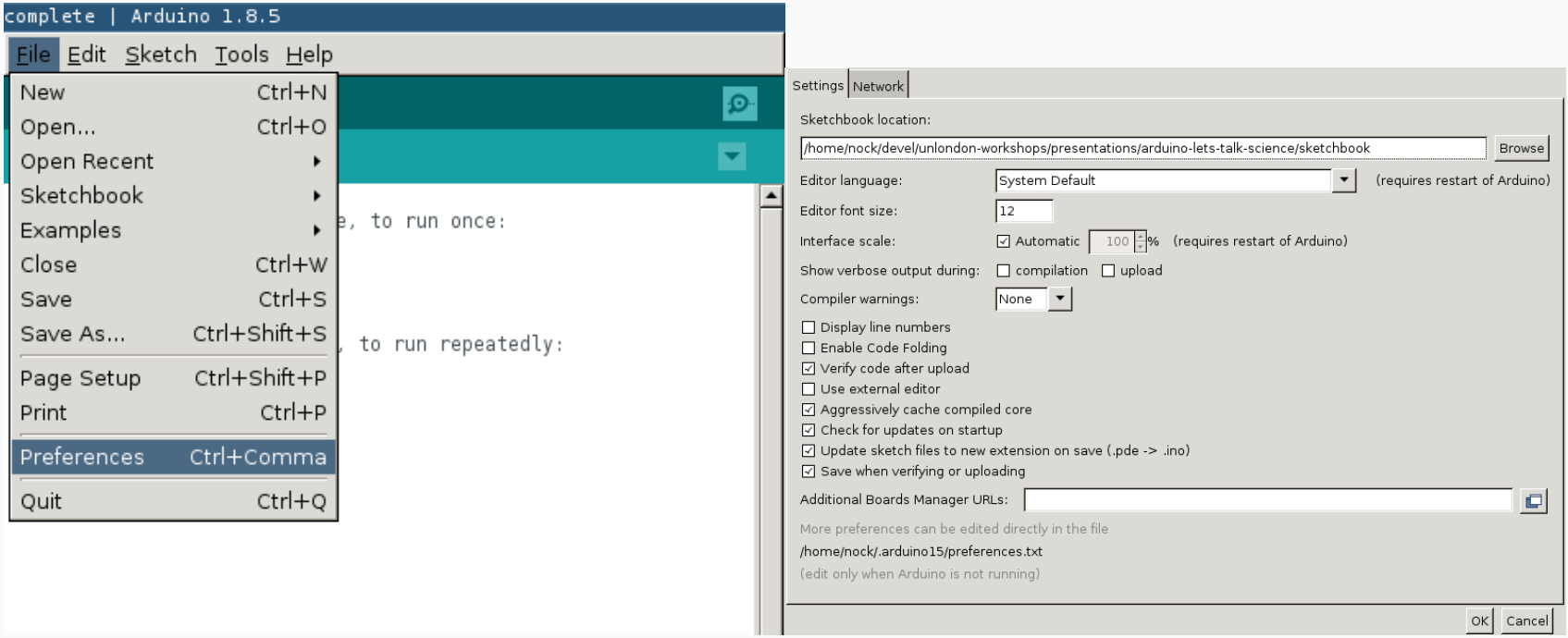
- Board: Arduino/Genuino UNO
- Port: ...

Fetch the Class Code

Download and extract:
`https://monadnock.ca/static/lts-workshop-Nov2017.zip`

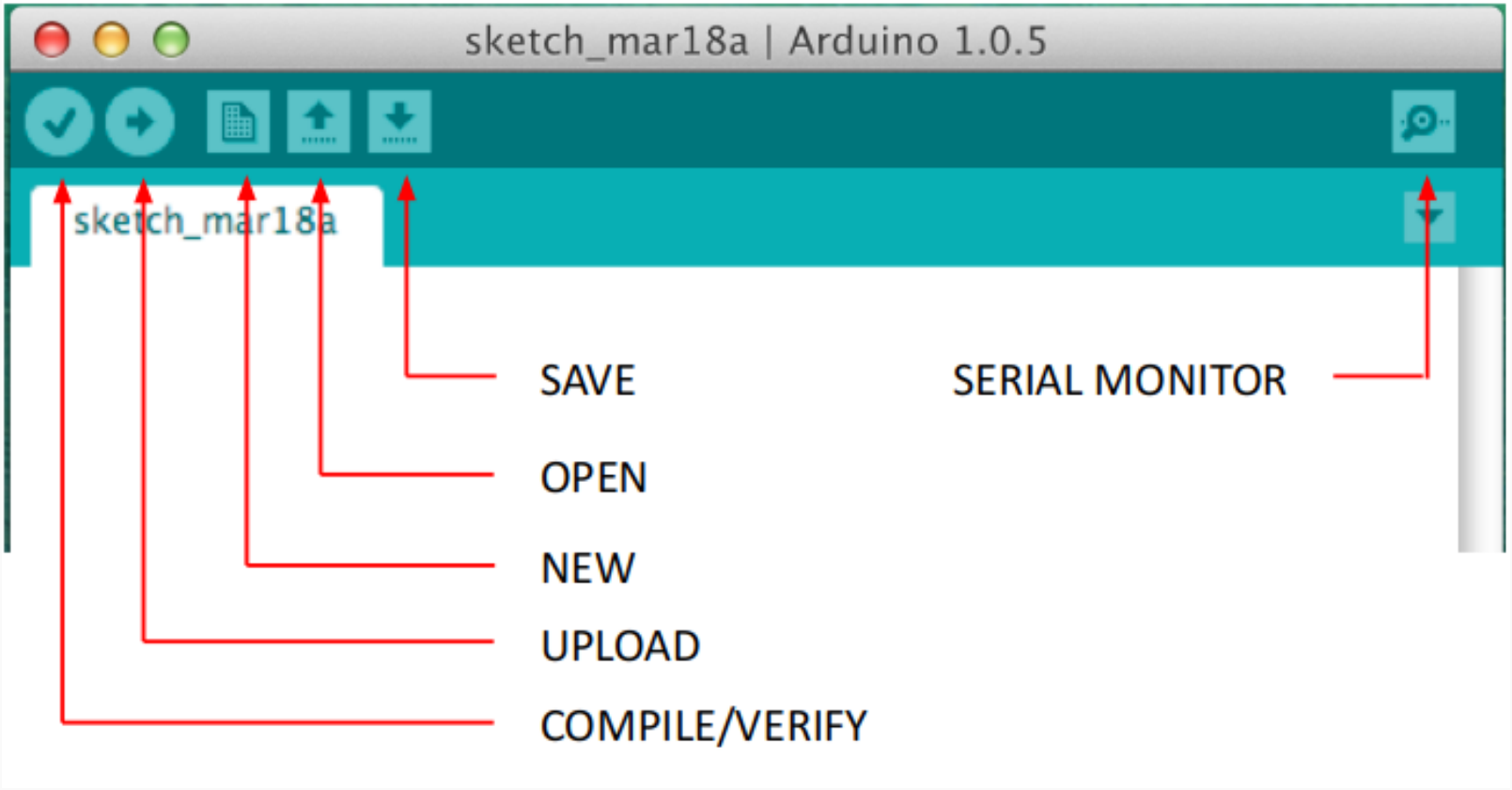
- File→Preferences
- Browse for sketchbook
- Point it at the `sketchbook` subfolder of the extracted download
- You should now see a list of projects in the File→Sketchbook menu.

Setup the Sensor Libraries (Sketchbook)



For the *Sketchbook Location* browse to the extracted lts-workshop-Nov2017/sketchbook directory.

The Code Environment

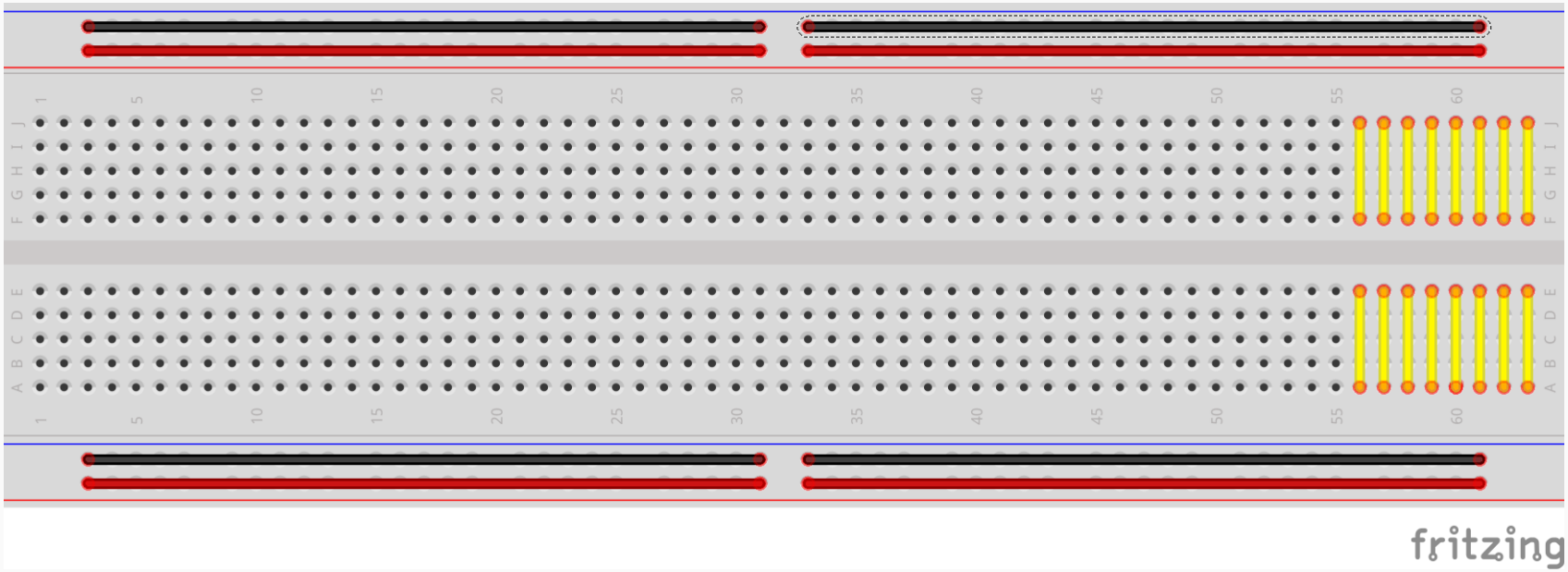


Your first Program

```
#define LED 13

/* the setup function runs once on reset / power */
void setup() {
  pinMode(LED, OUTPUT);
}

/* loop() repeats until reset or power off */
void loop() {
  digitalWrite(LED, HIGH);   // turn on LED
  delay(1000);               // wait for a second
  digitalWrite(LED, LOW);    // turn the off LED
  delay(1000);
}
```

Lets Add Some Parts

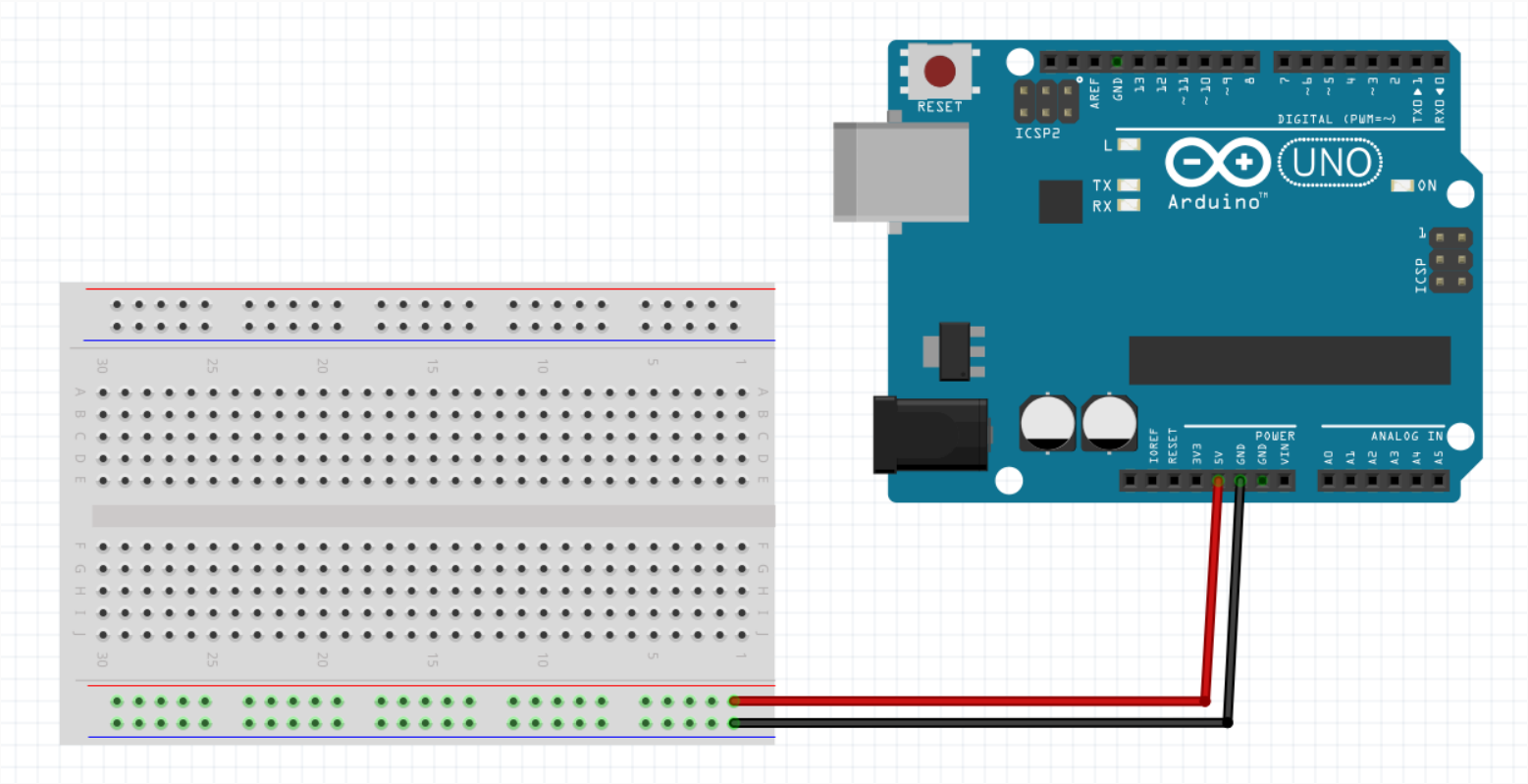
Breadboard



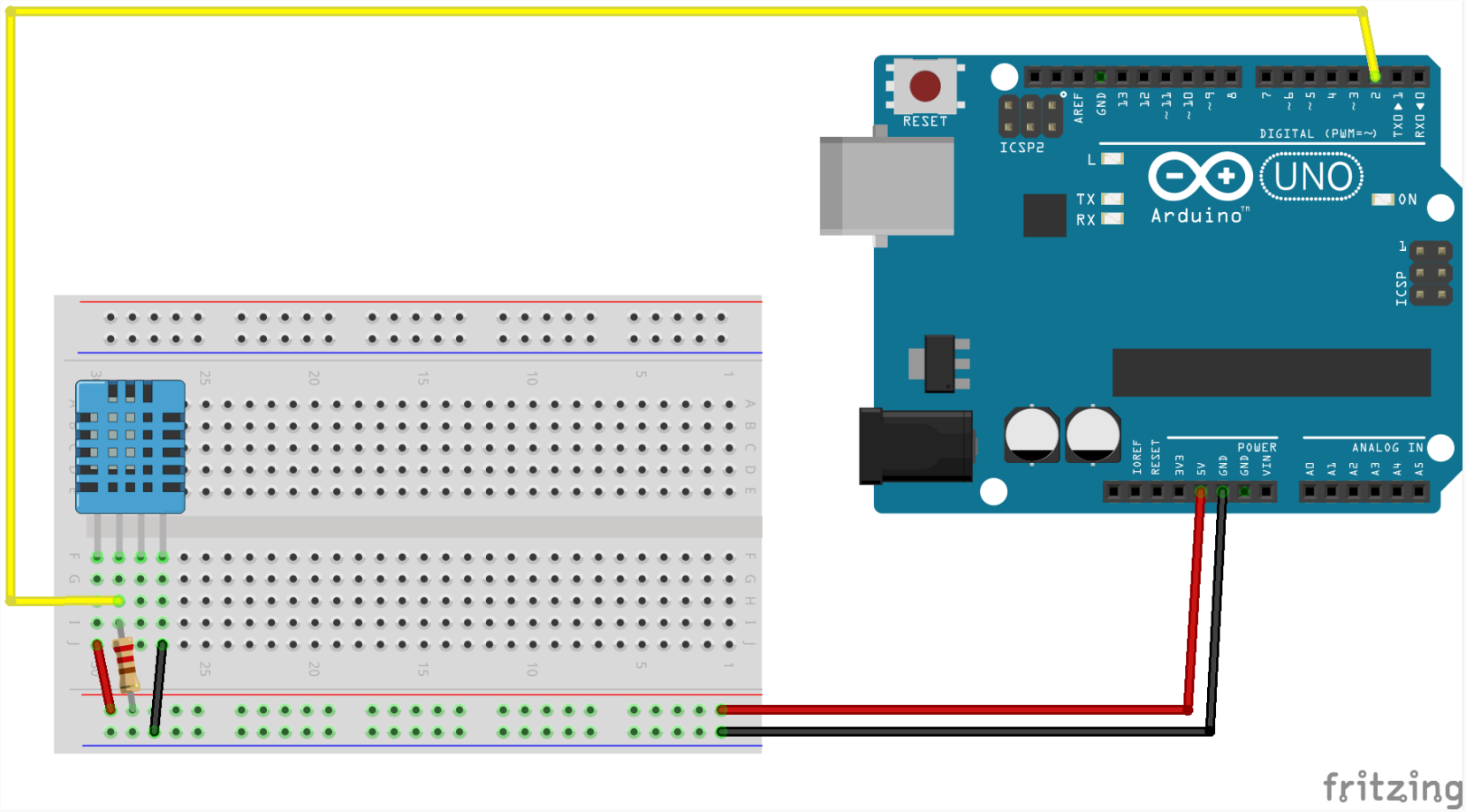
- Connectors gently pinch component leads, wires.
- Have internal connections

Power Up the Rails

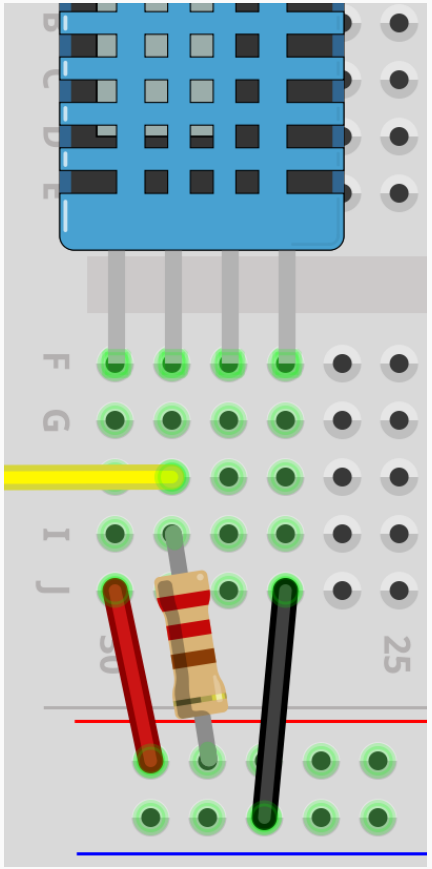
We use the long rows to distribute power. The Arduino outputs 5 V on the pin marked `5V`, the reference (GND) is marked `GND`.



DHT11: Temperature and Humidity Sensor



DHT11: Zoom



Connect left pin to Vcc rail (+) and right pin to GND (−) rail.

Second from left connects to Arduino pin #2 *and* a resistor to Vcc

Pullup / Pulldown Resistors

The DHT11 uses an *Open Collector* output that switches between *Not Connected* and *Connected to Ground*. When it's not connected, the pin is left *floating*. How do we make a floating pin look like a High signal?

Solution:

Connect the pin to Vcc so that it reads High; use a resistor to prevent short circuit (limit current).

DHT11: Run example code

We can test the sensor by running sample code from the library.

Open the sample: File→Examples→DHT Sensor Library→DHTtester

Once the sample is loaded click the *upload* button (looks like right arrow)

Arduino Serial Monitor

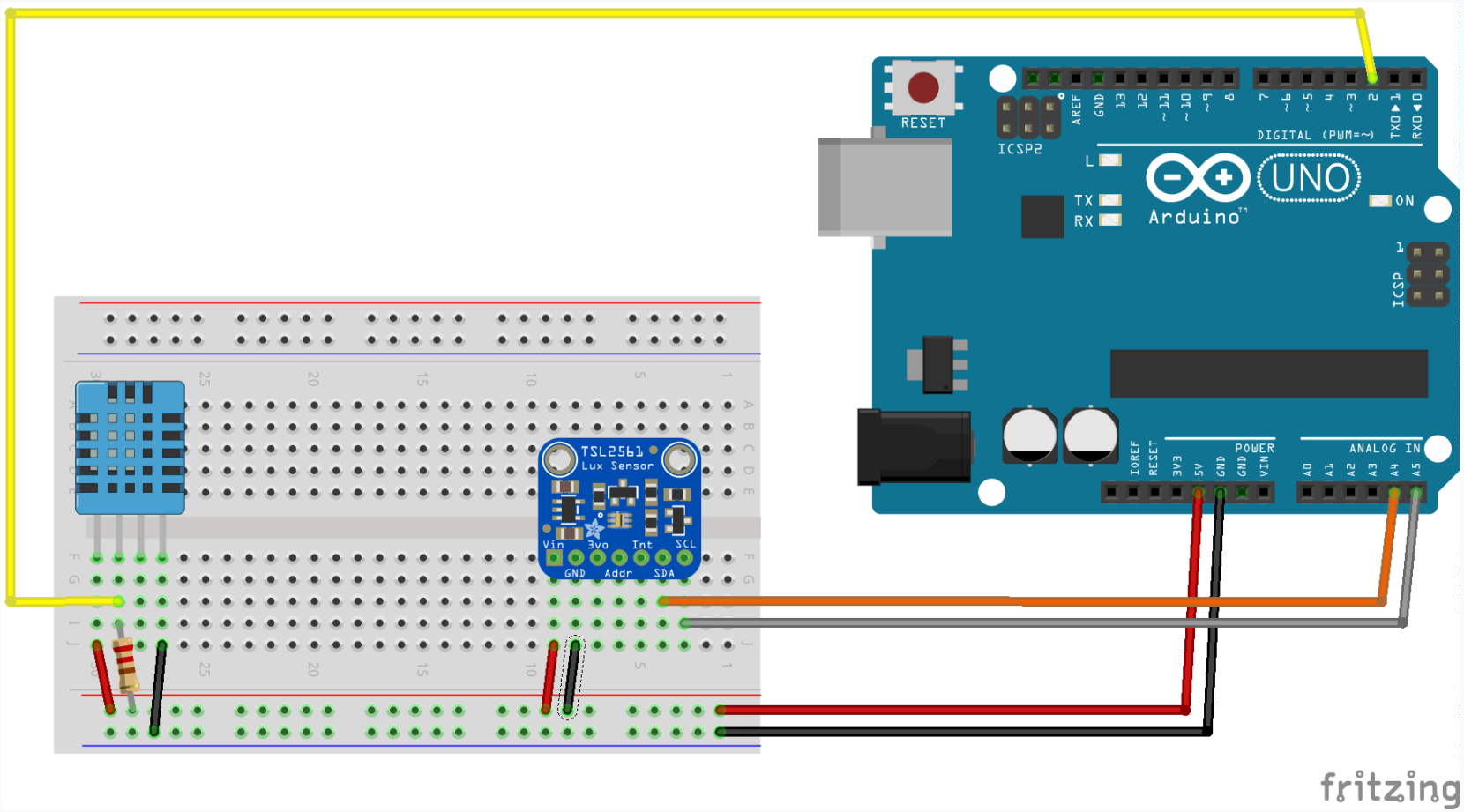
Once the sample code is uploaded, you can open the *Serial Monitor* to view the output. It can be found under:

Tools→Serial Monitor

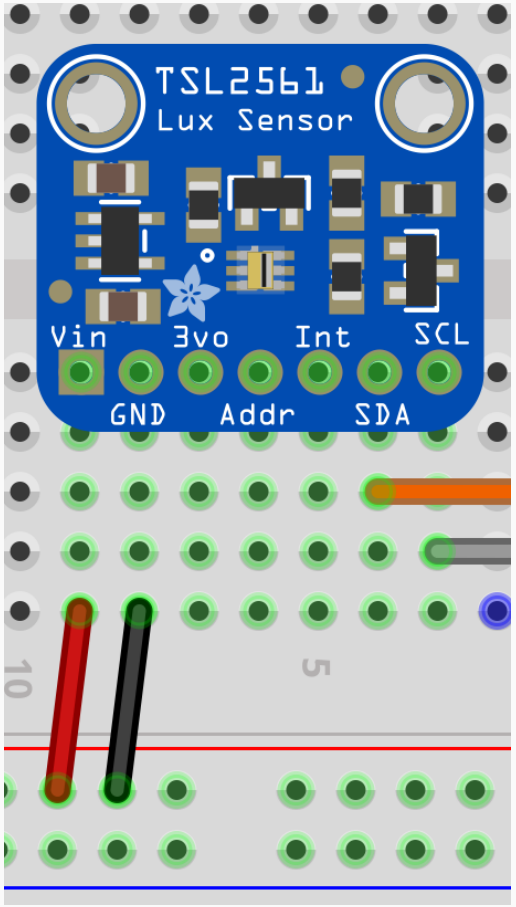
DHTtester Output

```
DHDHTxx test!
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 45.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.23 *C 66.61 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
Humidity: 44.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 19.20 *C 66.57 *F
```

TSL2561: Ambient Light Sensor



TSL2561: Zoom



Connect Vin to (+) rail, GND to (−) rail, SDA to Arduino pin A4 and SCL to A5

TSL2561: Run example code

We can test the sensor by running sample code from the library.

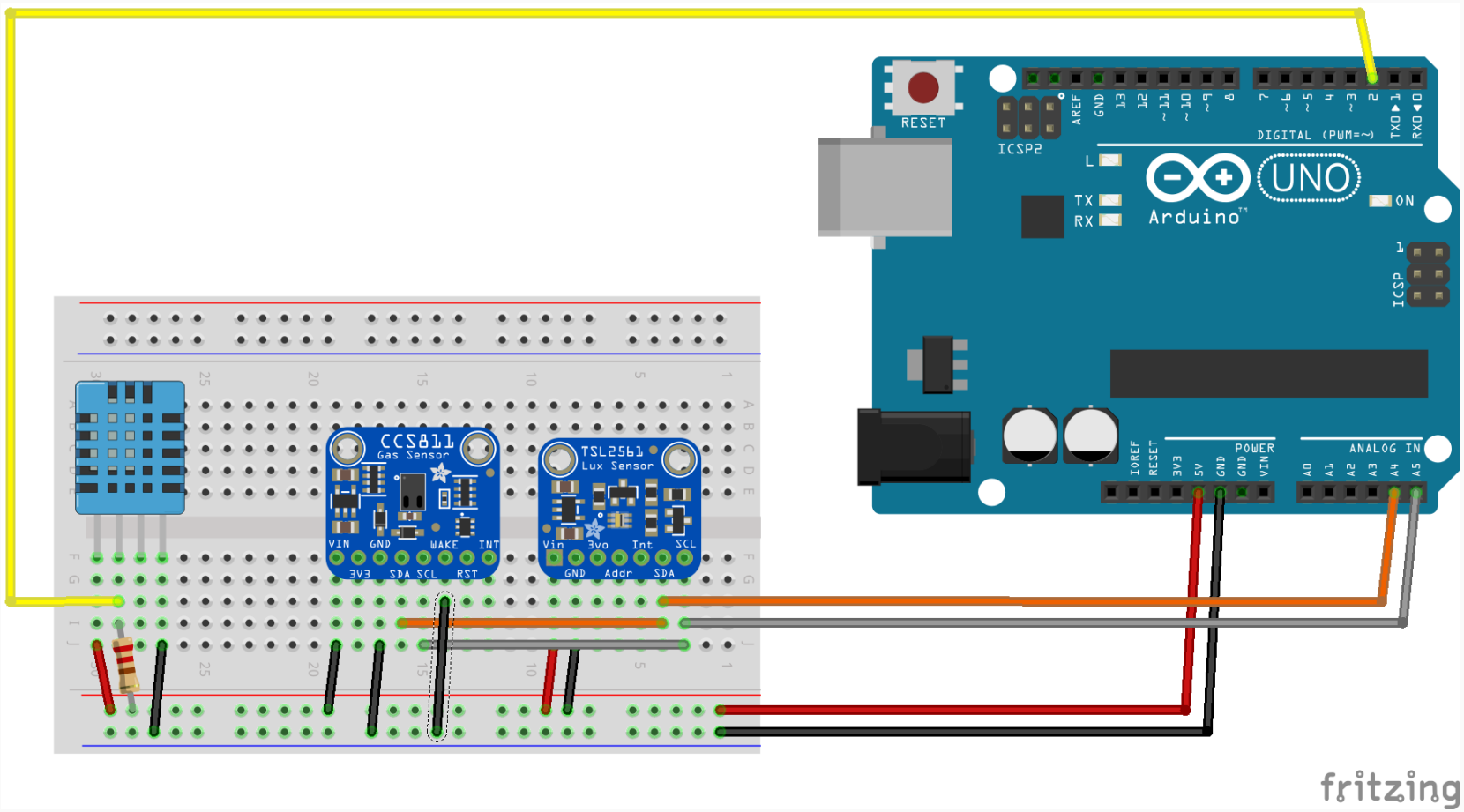
Open the sample: File→Examples→Adafruit TSL2561→sensorapi

Once the sample is loaded click the *upload* button (looks like right arrow)

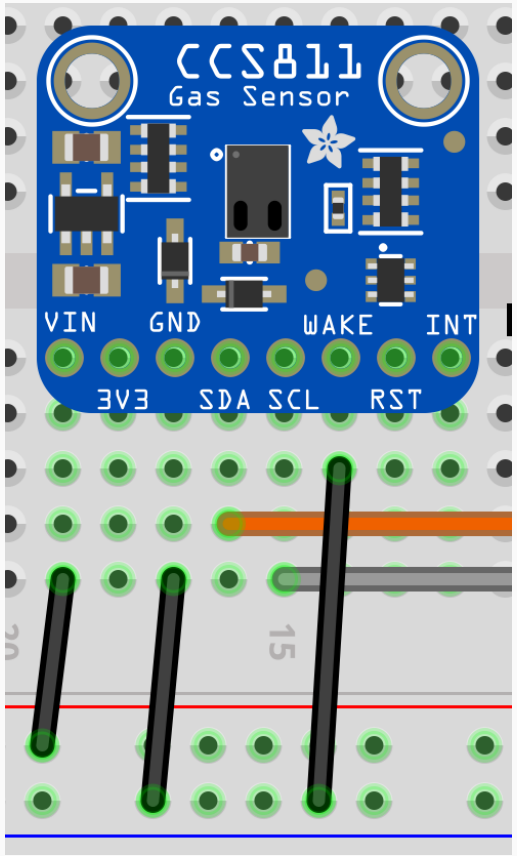
TSL2561 Sample Output

```
Light Sensor Test

-------------------------¶Light Sensor Test


------------------------------------
Sensor:       TSL2561
Driver Ver:   1
Unique ID:    12345
Max Value:    17000.00 lux
Min Value:    0.00 lux
Resolution:   1.00 lux
------------------------------------

------------------------------------
Gain:         Auto
Timing:       13 ms
------------------------------------

28.00 lux
28.00 lux
28.00 lux
28.00 lux
30.00 lux
30.00 lux
```

CCS811: Gas Sensor



CCS811: Zoom



Vin to (+) rail, GND *and* WAKE to (−) rail, SDA to TSL2561 SDA, SCL to TSL2561 SCL

CCS811: Run example code

We can test the sensor by running sample code from the library.

Open the sample: File→Examples→Adafruit CCS811 Library→CCS811_test

Once the sample is loaded click the *upload* button (looks like right arrow)

CCS811 Sample Output

```
CCS811 test
CO2: 0ppm, TVOC: 0ppb   Temp:25.00
CO2: 0ppm, TVOC: 0ppb   Temp:25.00
CO2: 0ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 400ppm, TVOC: 0ppb   Temp:25.00
CO2: 423ppm, TVOC: 3ppb   Temp:25.00
```

Programming

Part 1; Includes

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include "Adafruit_CCS811.h"
#include "DHT.h"
```

Part 2; Definitions

```
#define DHTPIN 2
#define DHTTYPE DHT11
```

```
Adafruit_CCS811 ccs;
DHT dht(DHTPIN, DHTTYPE);
Adafruit_TSL2561_Unified tsl =
    Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
```

Programming Note: Variables

Declare a variable:

```
int button_state = HIGH;
```

`<type> <name> [= <initial value>];` (value optional)

It's a name, like a preprocessor `#define`, but the value can change at *runtime*

Part 4; Setup

```
void setup() {
  Serial.begin(9600);
  Serial.println("Let's Talk Science!");

  dht.begin();
  ccs.begin();
  tsl.begin();

  while(!ccs.available());
  float temp = ccs.calculateTemperature();
  ccs.setTempOffset(temp - 25.0);

  tsl.enableAutoRange(true);
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);
}
```

Part 4; Main Loop

```
loop () {
  delay(500);
  ...
}
```

Part 4a; Temperature / Humity Sensor Read

```
loop () {
  ...
  float humidity = dht.readHumidity();
  float temp = dht.readTemperature();

  Serial.print(humidity);
  Serial.print(" ");
  Serial.print(temp);
  Serial.print(" ");
  ...
}
```

Part 4b; Gas Sensor Read

```
loop () {
  ...
  ccs.readData();
  int co2 = ccs.geteCO2();
  int tvoc = ccs.getTVOC();

  Serial.print(co2/100);
  Serial.print(" ");
  Serial.print(tvoc/10);
  Serial.print(" ");
}
```

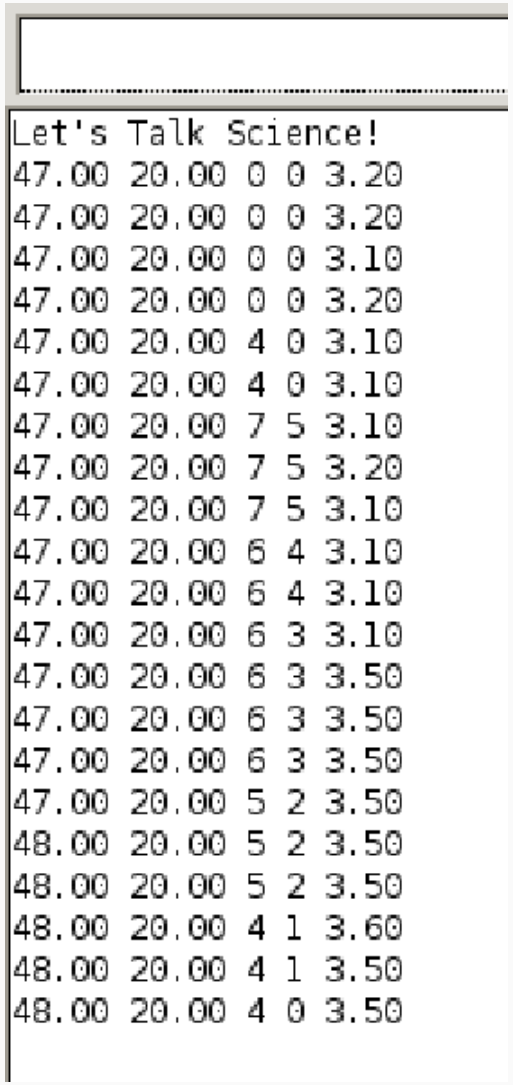
Part 4c; Light Sensor Read

```
loop () {
  ...
  sensors_event_t event;
  tsl.getEvent(&event);

  Serial.print(event.light/10);
  ...
}
```

Part 4d; Ending the output

```
loop () {
  ...
  Serial.println("");
}
```

Complete Sensor Output



Arduino Serial Plotter

Why such a terse, boring output format?

Arduino has a feature where it will plot data in this specific format. You can see the plotted output by closing the Arduino Serial Monitor and Opening *Arduino Serial Plotter* (Tools→Serial Plotter)

Arduino Serial Plotter Output

Undiscussed Issues

Obtaining the kit

- Educators would need to order several different parts (SKUs).
- Vendors keep very little stock (~4 dozen).
- Educators may need to "google around" to find parts, most likely from abroad (US or China)

Installing the Software

- School machines may prevent software installation
- School machines may not be conventional computers (iPads, Chromebooks)

Troubleshooting the Build

- Breadboards aren't rugged.
- Sockets wear out quickly
- Parts may need to be wiggled
- Some parts may be DoA
- Most troubleshooting requires a Multimeter (and training on how to use it)

Extra Credit

Ohm's Law

Ohm's Law relates current to potential and resistance.

$$V = IR$$
$$I = \frac{V}{R}$$
$$R = \frac{V}{I}$$

- V = Potential in Volts (V)
- I = Current in Amperes (A)
- R = Resistance in Ohms (Ω)

Ohm's Law: Example

The datasheet for an LED says that the maximum continuous current is 15 mA. Your circuit operates at 5 V[1]. How big should your resistor be?

$$\Omega = \frac{5\,\mathrm{V}}{0.015\,\mathrm{A}} = 333.\overline{3}\Omega$$

How much current for our *cheet sheet* value?

$$\mathrm{A} = \frac{5\,\mathrm{V}}{1\,\mathrm{k}\Omega} = 5\,\mathrm{mA}$$

[1] Actually, this calculation is inaccurate. LEDs will have a *forward voltage drop* of between 1.8 V and 3.3 V this should be subtracted from V above... but it's not critical.

Current Limits, Arduino

- No single pin should source more that 20 mA (40 mA is absolute max)
- Pins are ganged together in groups of 8, no group should source more than 150 mA total
- The whole board cannot source more than 200 mA total

Practically speaking, this means that the Arduino cannot drive speakers, most motors, or anything normally mains powered.

So. . . no Arduino smart blender?

You can control almost anything with an arduino, you just can't power it with the Arduino. There are various devices that let you switch highier powered devices:

- Transistors
- Relays
- Solid State Relays
- Triac

HIGHs and LOWs

Many different logic levels are in common use: 1.2 V, 1.8 V, 2.5 V, 3.3 V, and 5 V. The voltage cited is the *nominal* Vcc of the system.

A HIGH signal is generally any voltage $\geq \frac{2}{3} V_{cc}$.

A LOW signal is generally any voltage $\leq \frac{1}{3} V_{cc}$.

HIGHs and LOWs, pt. 2

In your travels, you're likely to see both 5 V and 3.3 V sensors and peripherals.

Since $3.3\,\mathrm{V} \geq \frac{2}{3} V_c c$ your Arduino will accept input from a 3.3 V peripheral without issue.

If you drive an output to 5 V while it's connected to a 3.3 V peripheral with an Arduino **it will blow up your peripheral**.[3]

[3] In the datasheet for the sensor, it'll have a section called *Absolute Maximums*. Generally 3.3 V parts won't accept more that $\approx 3.6\,\mathrm{V}$, but some will.

HIGHs and LOWs, pt. 3

Solutions:

- Level Shifter: A dedicated chip that translates between voltages. Available as uni or bidirectional.
- Buy a 3.3V Arduino Compatible. Arduinos are available that operate at the lower voltage.