Intro

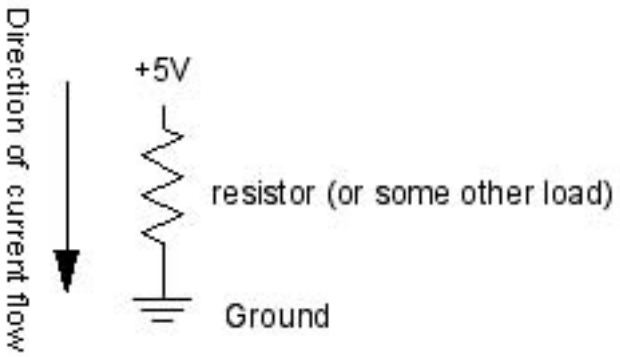
- Electricity
- Analog / digital circuits
- Various sensors
- Controlling a high amperage load through a transistor
- Serial communication
- What next? (more sensors, serial communication, visualizing input, wireless, controlling AC loads?...)

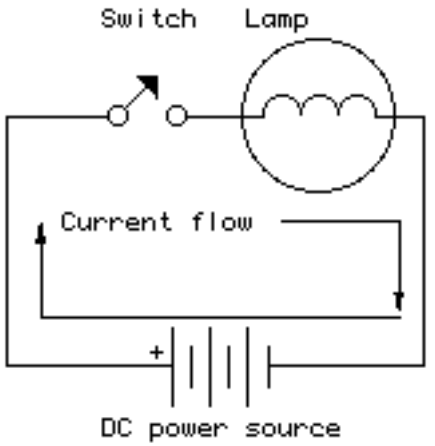
Electricity

- Electricity: flow of electrons through a conductive material
- Current (Amps): measure of how many electrons are flowing
- Voltage (volts): the electrical energy (analogy: pressure)
- Resistance (Ohms): a material's ability to RESIST or oppose current
- ANALOGY: water flowing through a hose:
  - current = HOW MUCH
  - Voltage = PRESSURE
  - Resistance = SIZE of the hose

V=IR

- A <u>circuit</u>: a closed loop containing a source of electrical energy (battery) and a load (a light bulb, a motor, etc.)
- Voltage = resistance * amperage



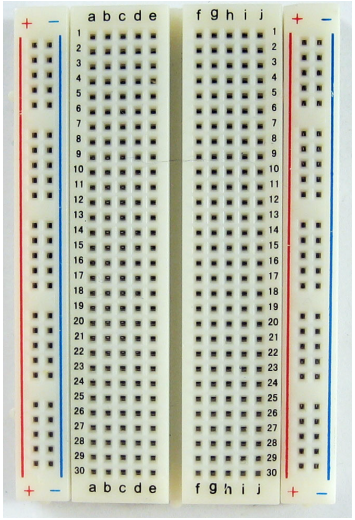


Current follows the path of least resistance

A simple circuit

Let’s get started

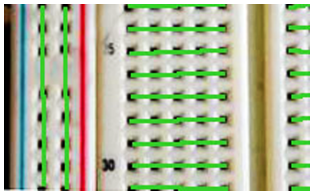
THE SOLDERLESS BREADBOARD



The breadboard is basically a chunk of plastic with a bunch of holes in it. But there is something special going on:

There is electrical conductivity. Basically this means that even though you can’t see it, if you poked inside, there are metal strips that connect the ROWS and the COLUMNS together. LIKE THIS:

here’s a CLOSE UP



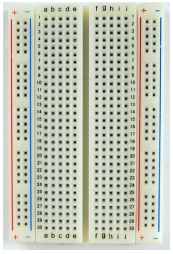
columns

rows

LED blink

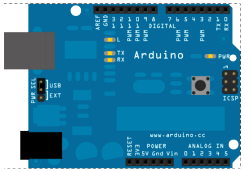
What we will need to set up the breadboard for a blinking LED

1. Breadboard



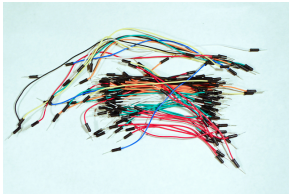
2. LED

3. Arduino

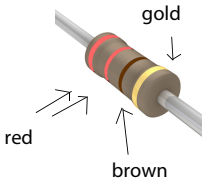


4. USB

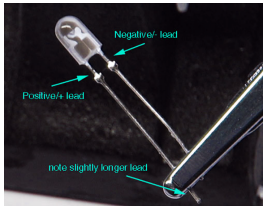
5. wires



6. 1 220 ohm resistor



Step 1. LED REVIEW



LED = LIGHT EMITTING DIODE

LED's WILL NOT work if placed backwards, so if your LED doesn't light up when you think it will, try reversing it!

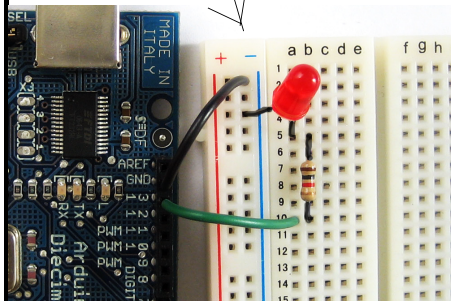
Step 2. Setup the Breadboard

Connect a BLACK wire from a ground ("GND") pin on Arduino to the ground column on the breadboard (the one marked in BLUE, with the minus "-" sign

Connect a BLUE or GREEN wire from pin 13 to any row on the breadboard

Connect a resistor from one row to the LONG leg of the LED

Connect the LED from the RESISTOR to GROUND (the blue striped column on the breadboard)



NOTE: RESISTORS can be connected EITHER way (unlike LED's)

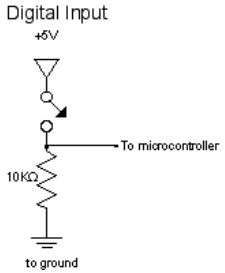
The color of the wires DOES NOT matter. It just helps to be neat.

Uploading a sketch

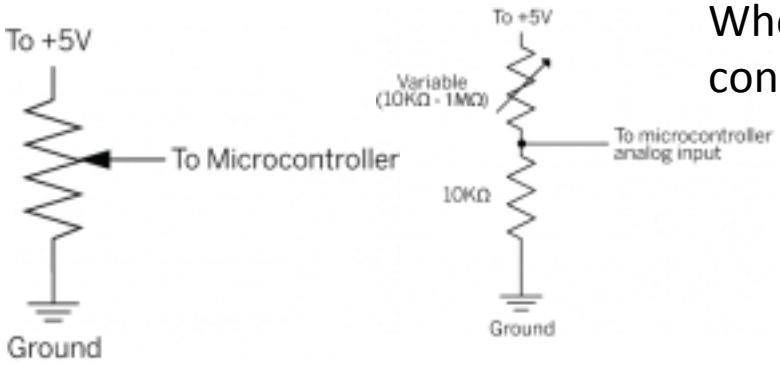
e image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Review

Analog vs. Digital



Digital inputs have two states: off and on.
If voltage is flowing, the circuit is on.
If it's not flowing, the circuit is off.

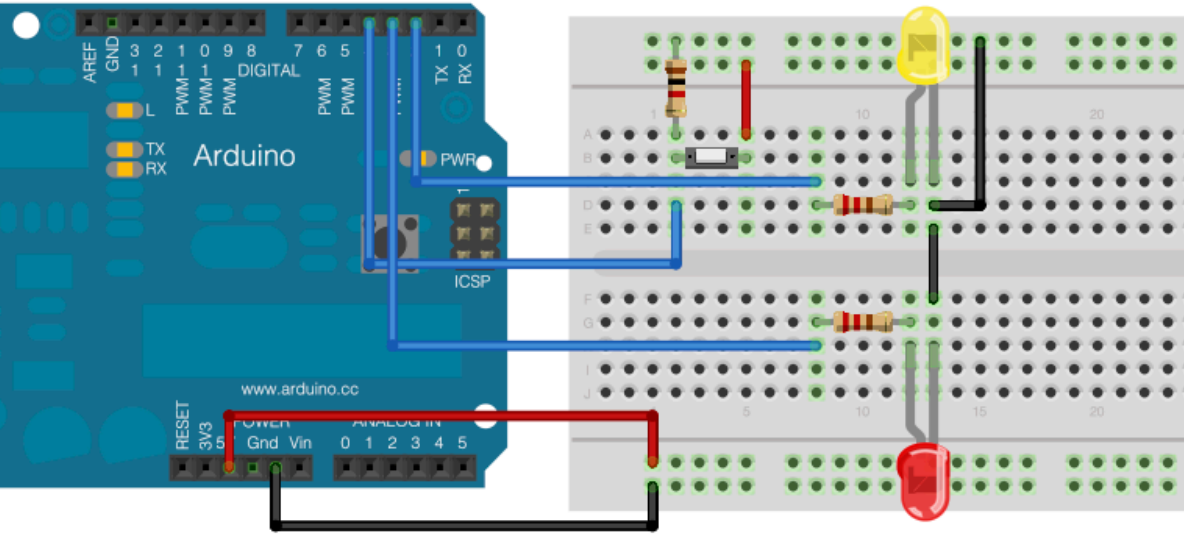


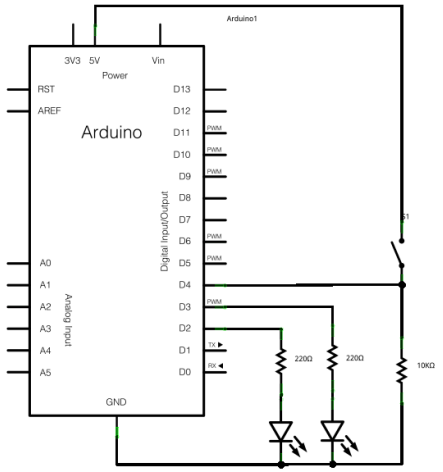
When we want to measure variably changing conditions like this, we need analog inputs.

Examples: thermistors, photocells, force sensing resistors, flex sensors, …

A digital circuit

Here's a circuit for a program that reads the digital input on pin 4.
Then it turns on the LED on pin 2 if the input is high (i.e. the switch is on),
or turns on the LED on pin 3 is the input is low (the switch is off):



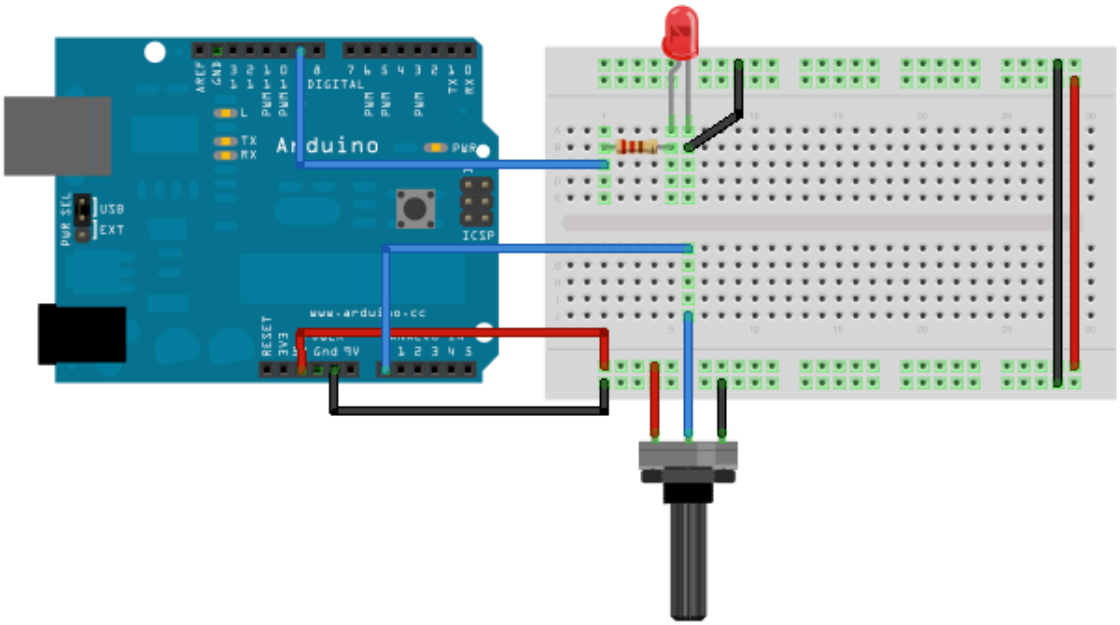


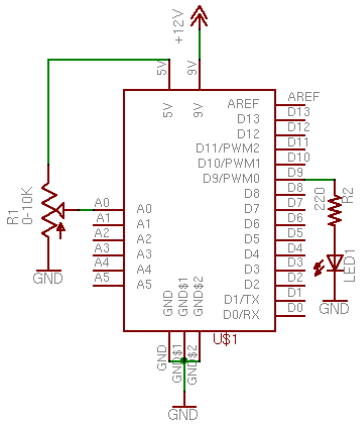
The switch code

```
// declare variables:
int switchPin = 4;      //  digital input pin for a switch
int yellowLedPin = 2;   //  digital output pin for an LED
int redLedPin = 3;      //  digital output pin for an LED
int switchState = 0;    // the state of the switch

void setup() {
  pinMode(switchPin, INPUT);       // set the switch pin to be an input
  pinMode(yellowLedPin, OUTPUT);   // set the yellow LED pin to be an output
  pinMode(redLedPin, OUTPUT);      // set the red LED pin to be an output
}

void loop() {
  // read the switch input:
  switchState = digitalRead(switchPin);

  if (switchState == 1) {
    // if the switch is closed:
    digitalWrite(yellowLedPin, HIGH);    // turn on the yellow LED
    digitalWrite(redLedPin, LOW);       // turn off the red LED
  }
  else {
    // if the switch is open:
    digitalWrite(yellowLedPin, LOW);   // turn off the yellow LED
    digitalWrite(redLedPin, HIGH);     // turn on the red LED
  }
}
```

An analog circuit

When you run this code, the LED should dim up and down as you turn the pot, and the value of the pot should show up in the debugger pane.





Analog input code

```
int potPin = 0;    // Analog input pin that the potentiometer is attached to
int potValue = 0;   // value read from the pot
int led = 6;    // PWM pin that the LED is on.  n.b. PWM 0 is on digital pin 9

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  // declare the led pin as an output:
  pinMode(led, OUTPUT);
}

void loop() {
  potValue = analogRead(potPin); // read the pot value
  analogWrite(led, potValue/4);  // PWM the LED with the pot value (divided by 4 to fit in a byte)
  Serial.println(potValue);      // print the pot value back to the debugger pane
  delay(10);                     // wait 10 milliseconds before the next loop
}
```

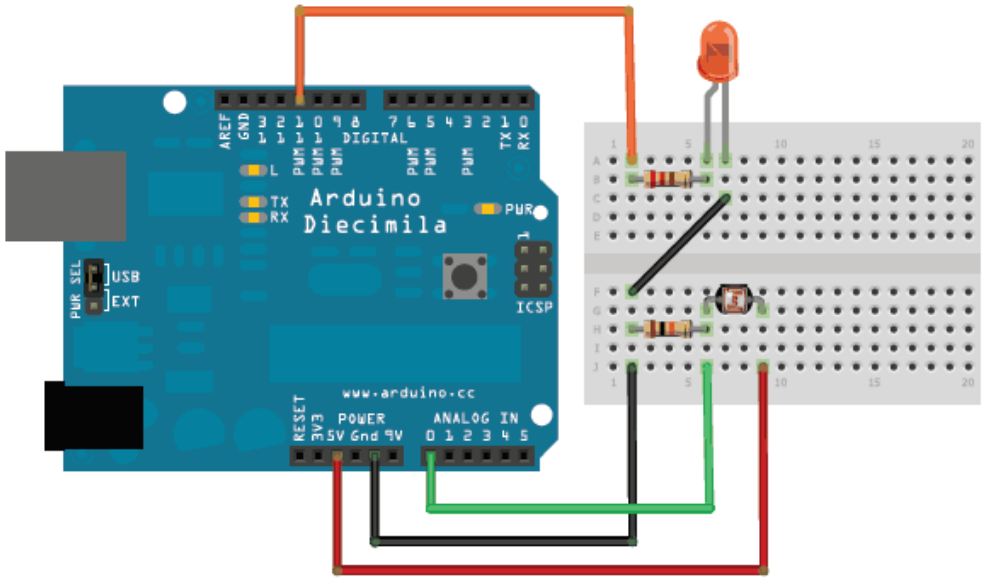
Another sensor, and the map() function

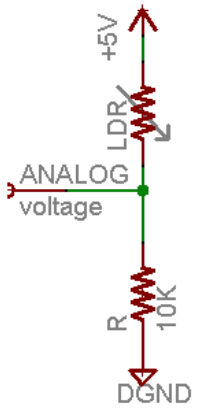
- Let's replace the potentiometer with a photocell (or any variable resistor)
- Add a fixed resistor? (voltage divider)
- Look at the serial monitor: find out the RANGE of the analog sensor - note the maximum and minimum values
  - Before, a potentiometer was giving us analog input values of 0 – 1023, the full range of the analogRead() function;
  - Now, we must modify the code using the map() function
  - Map the input (min – max sensor reading) to the output (0-255 – because 0 – 255 is the range of Arduino's analogWrite() function

Map the incoming values:

```
potValue = analogRead(potPin); // read the pot value
Serial.println(potValue);      // print the pot value back to the debugger pane
int brightness = map(potValue, 600, 950, 0, 255);
analogWrite(led, brightness);
```

Photocell circuit





Code for one photocell and one LED

```
int potValue = 0;   // value read from the pot
int led = 6;    // PWM pin that the LED is on.  n.b. PWM 0 is on digital pin 9

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  // declare the led pin as an output:
  pinMode(led, OUTPUT);
}

void loop() {

  potValue = analogRead(potPin); // read the pot value
  Serial.println(potValue);      // print the pot value back to the debugger pane
  int brightness = map(potValue, 600, 950, 0, 255);
  analogWrite(led, brightness);

  delay(10);                     // wait 10 milliseconds before the next loop
}
```

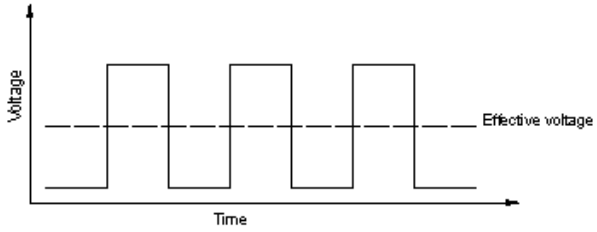
NOTE: you may have to modify the map() function, depending on your sensor

Excersize

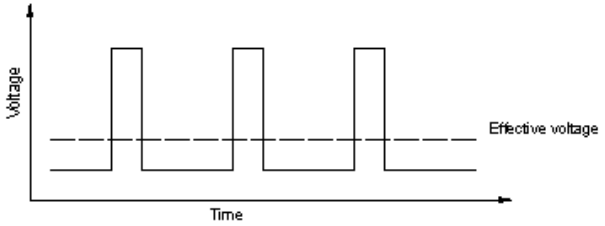
- Use the map() function to make the LED get less bright as the photocell gets darker, and brighter with MORE light…

Analog Output: Pulse Width Modulation (PWM)

- Most microcontrollers “fake” an analog voltage output by producing a series of voltage pulses at regular intervals, and varying the width of the pulses (PWM)
- DUTY CYCLE = ratio of the time the pin is HIGH to the time it is LOW

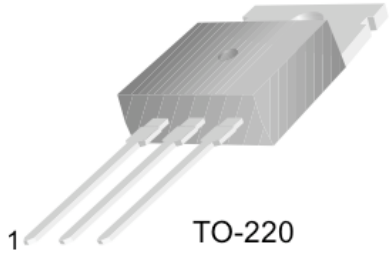


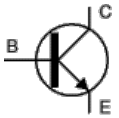
Here the duty cycle is 50%, making the voltage output about half



The duty cycle is less because it is OFF for longer

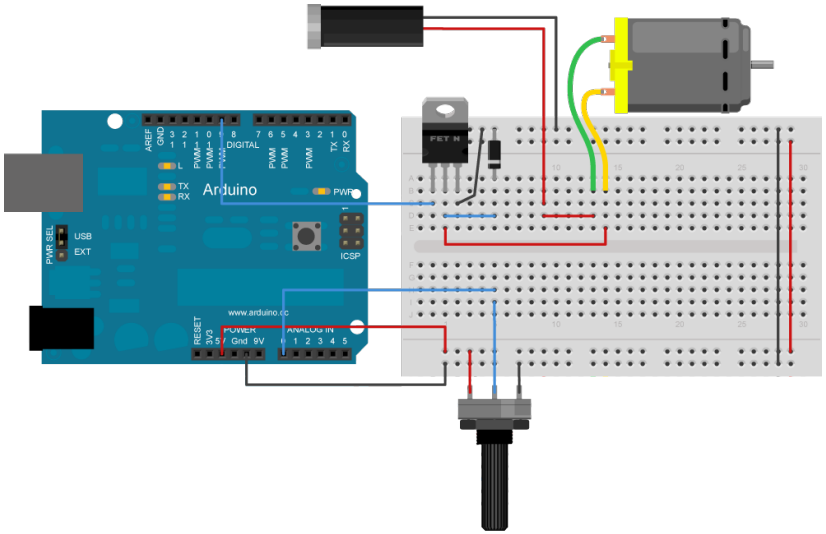
Powering a high-current load using a transistor





1.Base 2.Collector 3.Emitter

An electronic switch



Transistor circuit code

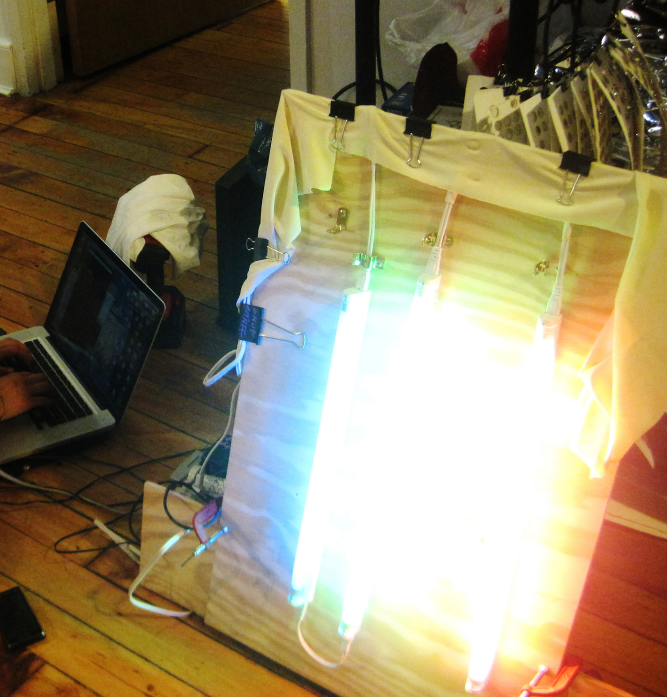
```
//try changing the speed of the motor or the intensity
//of the lamp using the potentiometer. Try this code:
 const int potPin = 0;           // Analog in 0 connected to the potentiometer
 const int transistorPin = 9;    // connected to the base of the transistor
 int potValue = 0;               // value returned from the potentiometer

 void setup() {
   // set  the transistor pin as output:
   pinMode(transistorPin, OUTPUT);
 }

 void loop() {
   // read the potentiometer, convert it to 0 - 255:
   potValue = analogRead(potPin) / 4;
   // use that to control the transistor:
   analogWrite(9, potValue);
 }
```

Controlling AC current

- The RELAY: A mechanical switch

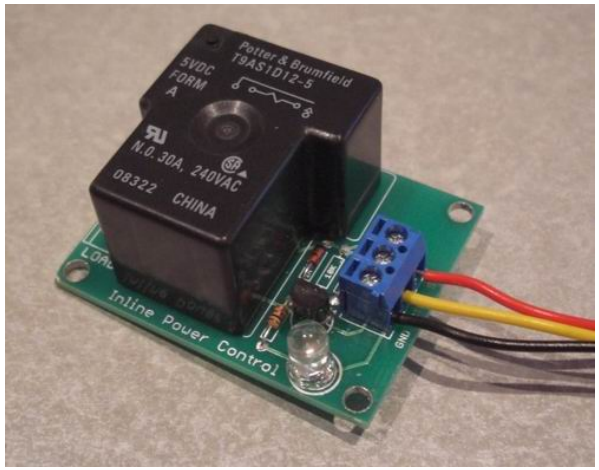
  - slow response, so unlike a transistor, we cannot control the output voltage (in the past example, the transistor was opening and closing rapidly and the motor was getting slower)

  - use a small (5V) input from arduino to allow AC current to flow through, thereby turning on an AC load using Arduino

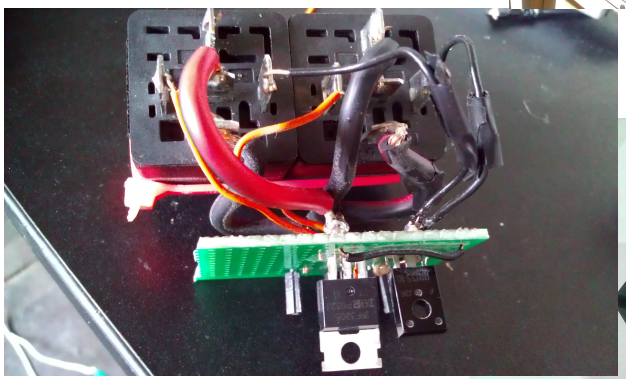


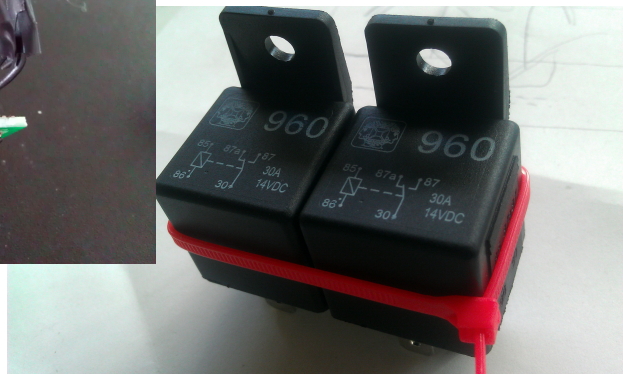
Two 12V relays used to reverse the direction of A motor (a motor driver does this, however)

For an RC sailboat using Two motors from a drill



A relay used to switch on AC current to turn on fluorescent lights





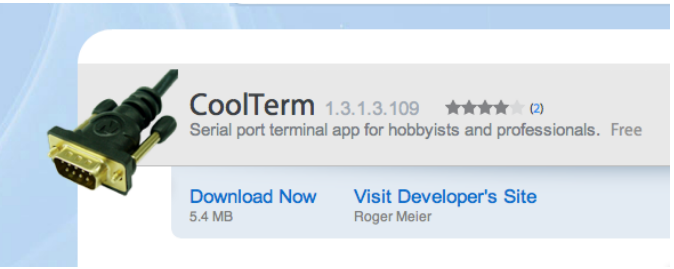
Visualizing the data

Sometimes it's nice to see sensor data in a graph

-CoolTerm (a serial port terminal application)

-Processing (an open-source programming language to create images, animations, interactions…)

We'll use CoolTerm for now:



http://www.macupdate.com/app/mac/31352/coolterm

Download and open CoolTerm.

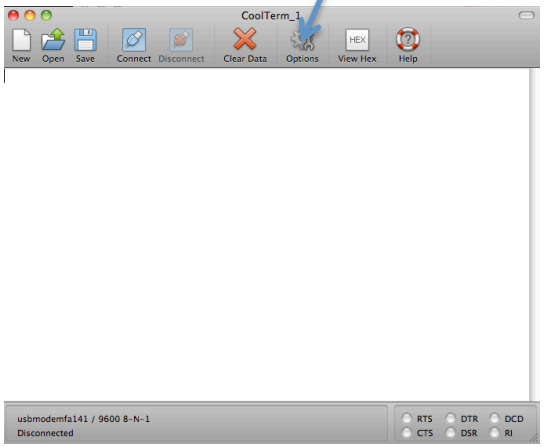
CoolTerm

- Let's go back to a sketch that reads sensor information (such as the one with the photocell) and prints it to the serial port

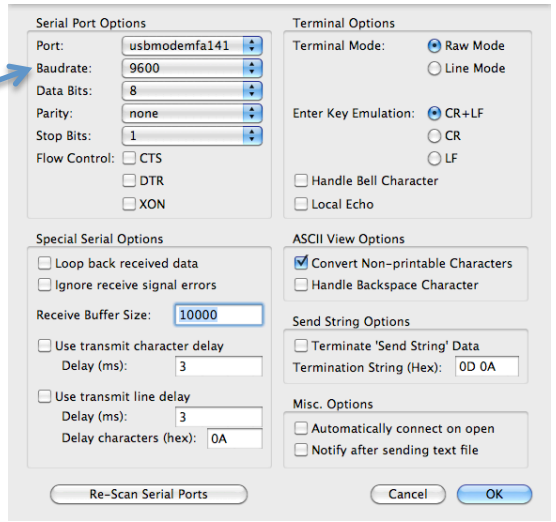
```
Serial.println(potValue);      // print the pot value back to the debugger pane
```

- With your arduino connected:

Click Options



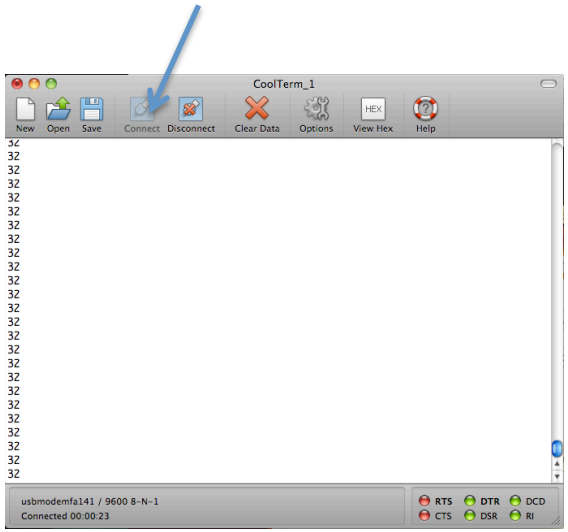
Make sure the Baudrate is set to 9600 (remember, in Arduino we set it to 9600 using Serial.begin(9600)



NOTE: you cannot have two ports open at the same time (so you can't view the Arduino's serial monitor while you are using CoolTerm

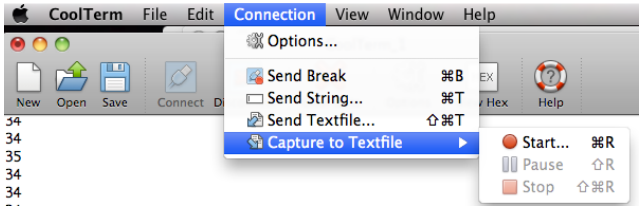
Start recording

Now click CONNECT



You should see data scrolling through

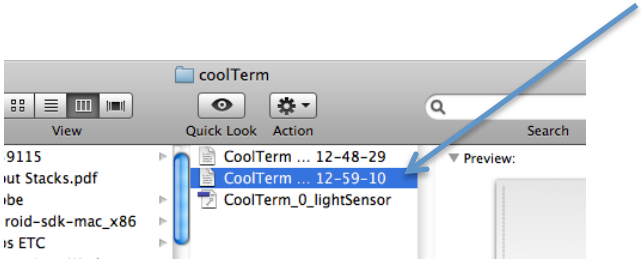
Click Connection → Capture to TextFile → Start



You can specify where to save the text file

Graph the data

Open the file with excel:



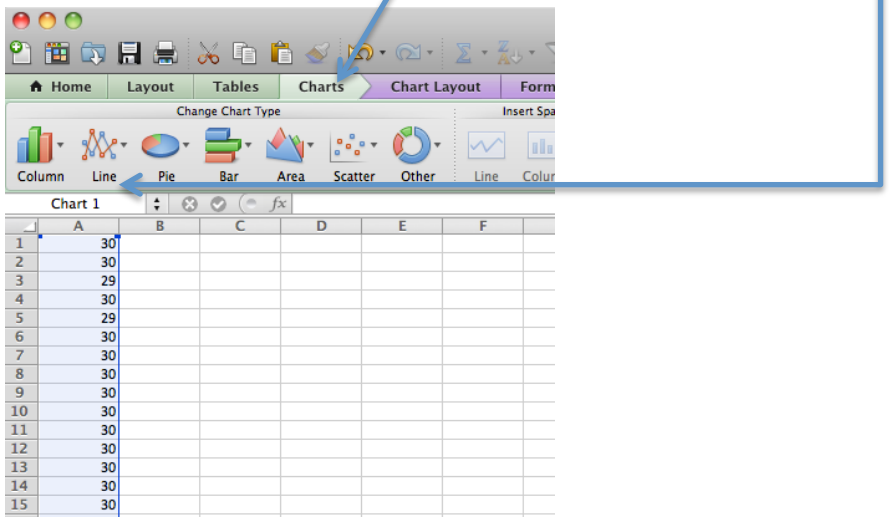
Your data should look something like this:
a long column of numbers (your sensor data)

	A	B
1	30	
2	30	
3	29	
4	30	
5	29	
6	30	
7	30	
8	30	
9	30	
10	30	
11	30	
12	30	

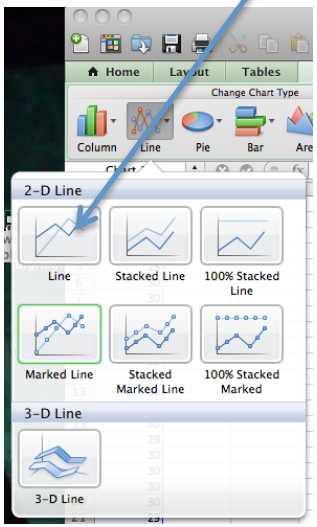
Graph the data

-Click on “Chart”

-Then click on “Line”

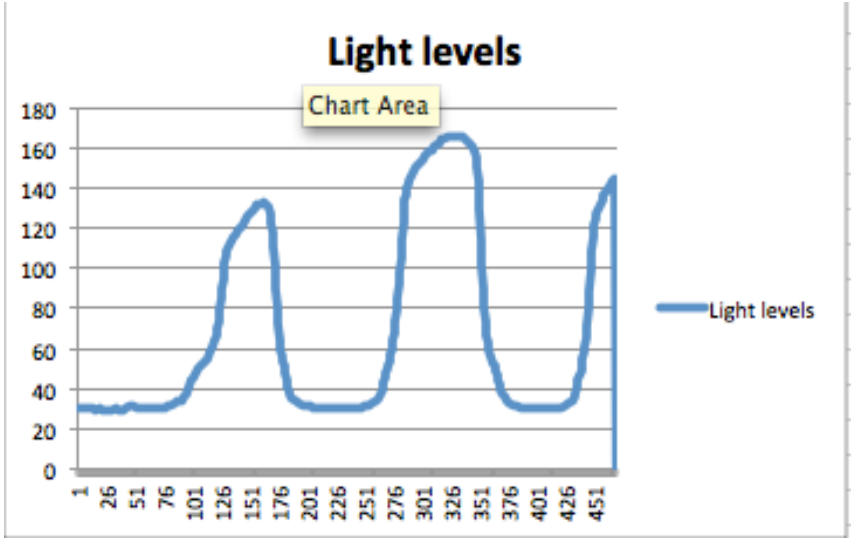


Then click on “Line”



And you should get something that looks like this:

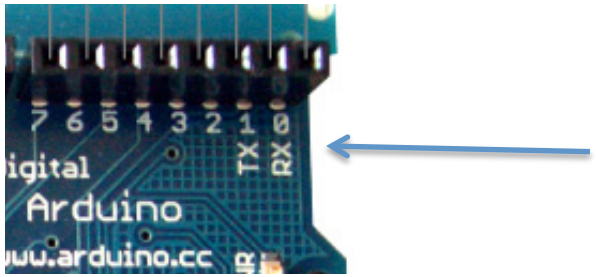
(This is from me moving
My hand near and far
The photocell)



Now you can work in excel to label the axes and format it how you like

Serial communication

- You've already used this via Arduino's serial monitor
- Computer → Arduino, agree upon rate (9600 baud)
- Sending a series of digital pulses back and forth (via a transmit wire and a receive wire)
- By interpreting several bits of data over time, the receiver can get a detailed message from the sender
- More complicated sensors: MIDI, GPS, accelerometer
- Digital pins 0 and 1 on Arduino are for transmitting and receiving data



What we did today

- Reviewed the basics of electricity
- Reviewed analog vs. digital input
- Made circuits that use analog and digital input
- Made a circuit that uses analog output (PWM)
- Controlled high-amperage loads using a transistor
- Visualized the data using CoolTerm