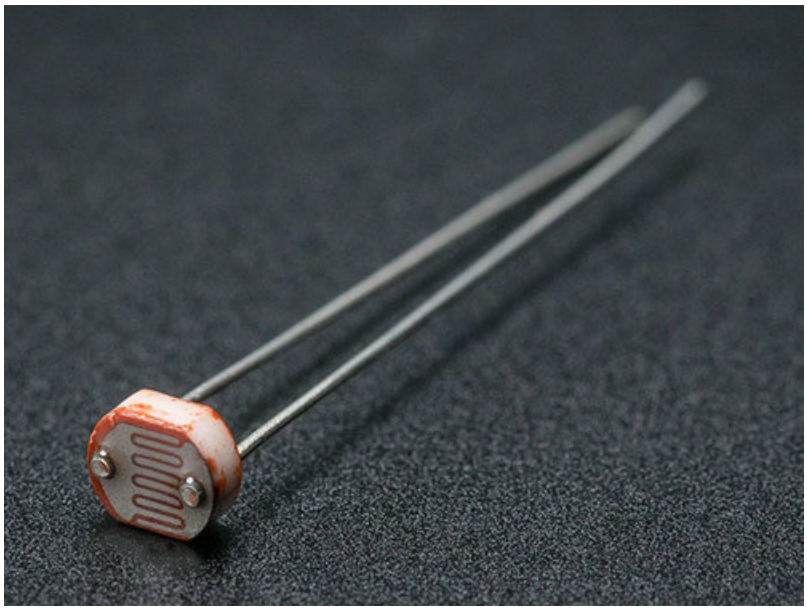
adafruit learning system

Photocells

Created by lady ada

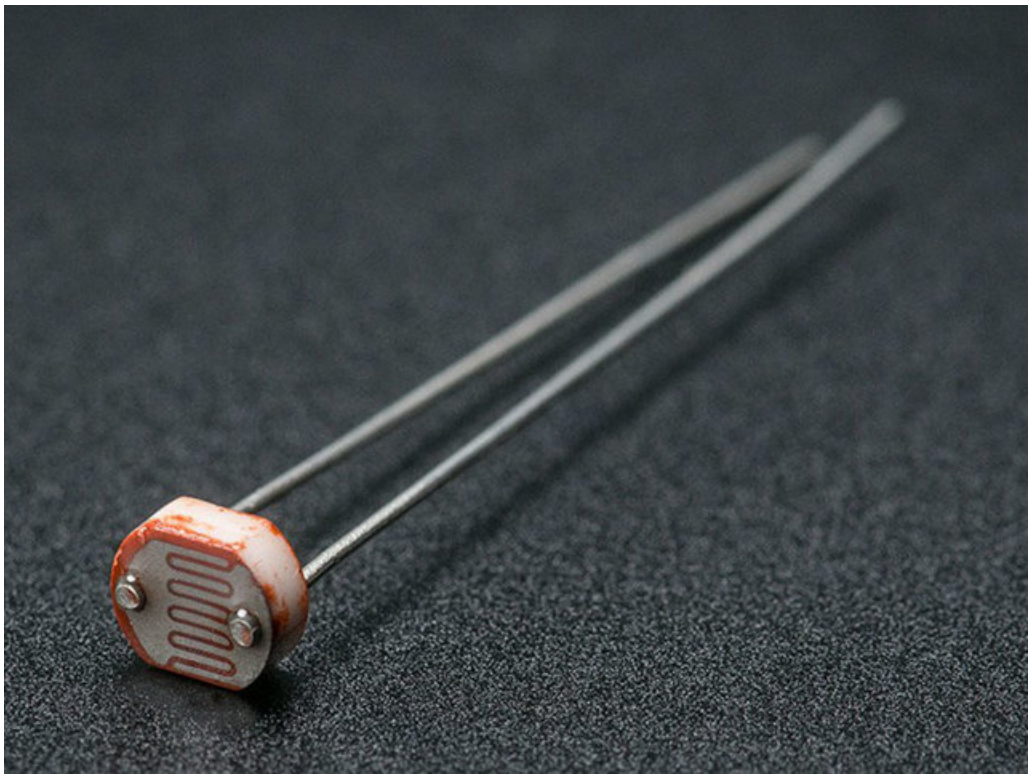


Last updated on 2017-12-26 08:23:04 PM UTC

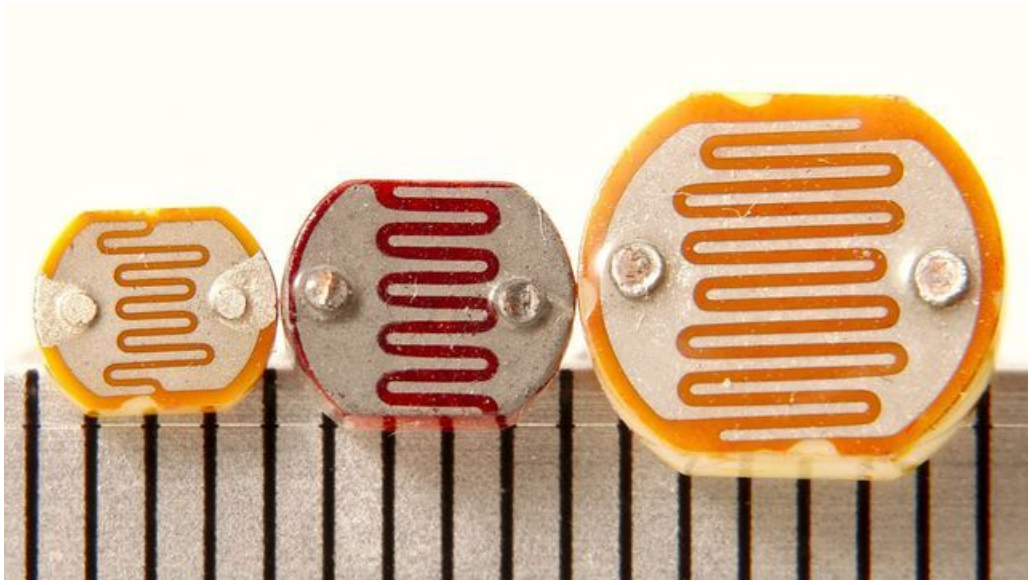
Guide Contents

Overview

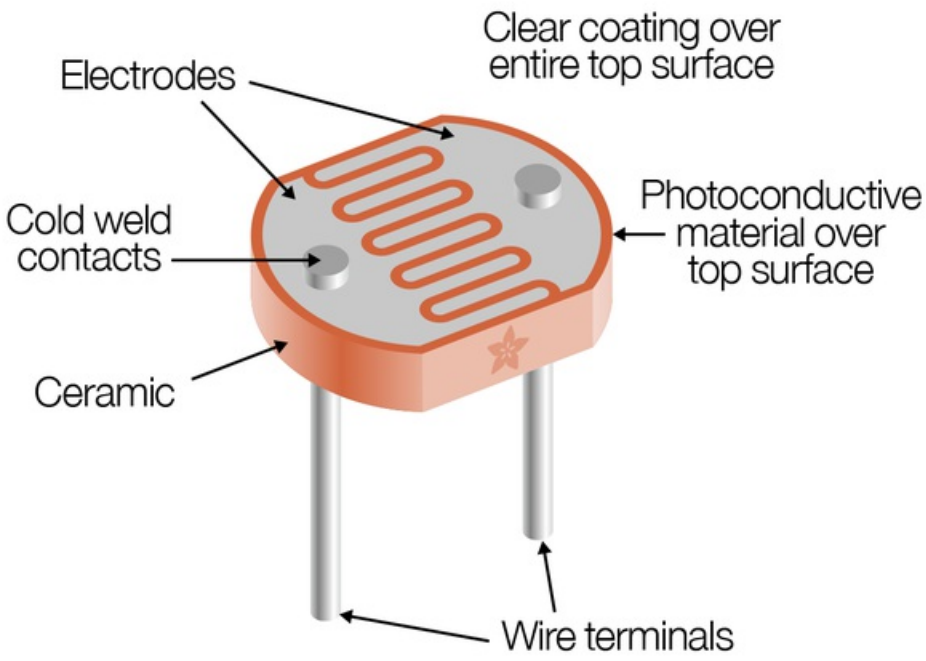
Photocells are sensors that allow you to detect light. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they often appear in toys, gadgets and appliances. They are often referred to as CdS cells (they are made of Cadmium-Sulfide), light-dependent resistors (LDR), and photoresistors.



Photocells are basically a resistor that changes its resistive value (in ohms Ω) depending on how much light is shining onto the squiggly face. They are very low cost, easy to get in many sizes and specifications, but are very innacurate. Each photocell sensor will act a little differently than the other, even if they are from the same batch. The variations can be really large, 50% or higher! For this reason, they shouldn't be used to try to determine precise light levels in lux or millicandela. Instead, you can expect to only be able to determine basic light changes.



For most light-sentsitive applications like "is it light or dark out", "is there something in front of the sensor (that would block light)", "is there something interrupting a laser beam" (break-beam sensors), or "which of multiple sensors has the most light hitting it", photocells can be a good choice!



Some Basic Stats

These stats are for the photocell in the Adafruit shop which is very much like the PDV-P8001 . Nearly all photocells will have slightly different specifications, although they all pretty much work the same. If there's a datasheet, you'll want to refer to it

- **Size:** Round, 5mm (0.2") diameter. (Other photocells can get up to 12mm/0.4" diameter!)
- **Price:** $1.00 at the Adafruit shop
- **Resistance range:** 200KΩ (dark) to 10KΩ (10 lux brightness)
- **Sensitivity range:** CdS cells respond to light between 400nm (violet) and 600nm (orange) wavelengths, peaking at about 520nm (green).
- **Power supply:** pretty much anything up to 100V, uses less than 1mA of current on average (depends on power supply voltage)
- **Datasheet** and another **Datasheet**
- Two **application notes on using** and **selecting photocells** where nearly all of these graphs are taken from

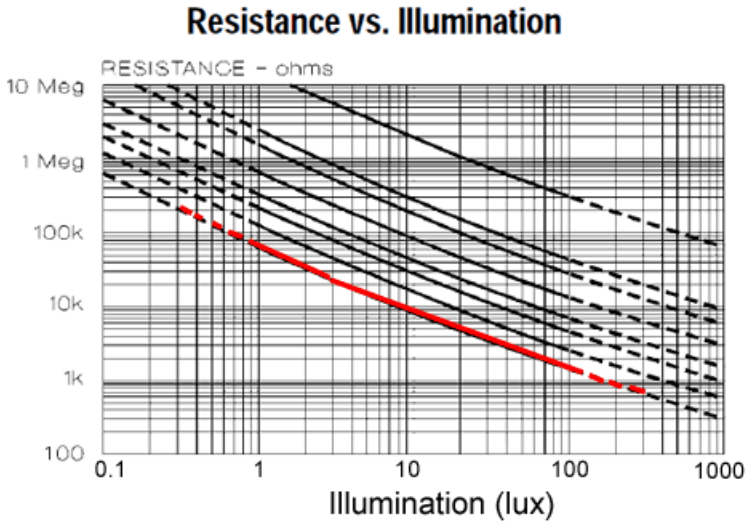
Problems you may encounter with multiple sensors

If, when adding more sensors, you find that the readings are inconsistent, this indicates that the sensors are interfering with each other when switching the analog reading circuit from one pin to the other. You can fix this by doing two delayed readings and tossing out the first one.

See this post for more information

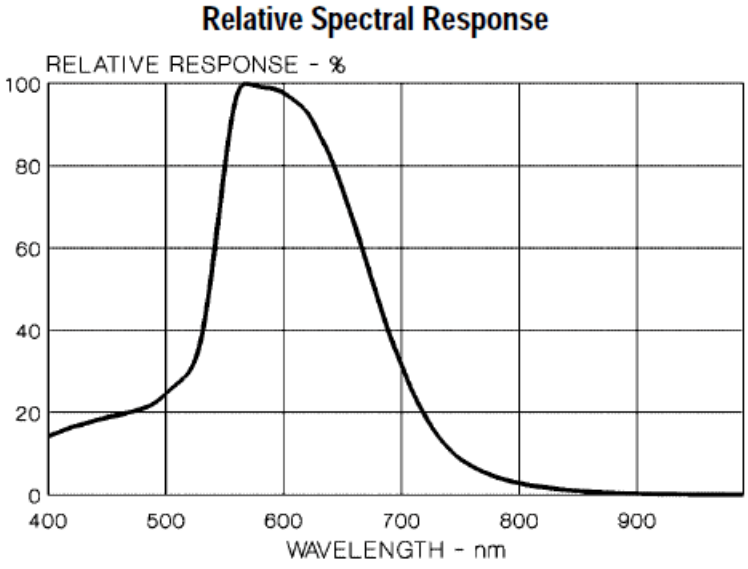
Measuring Light

As we've said, a photocell's resistance changes as the face is exposed to more light. When its dark, the sensor looks like an large resistor up to 10MΩ, as the light level increases, the resistance goes down. This graph indicates approximately the resistance of the sensor at different light levels. Remember each photocell will be a little different so use this as a guide only!



Note that the graph is not linear, its a log-log graph!

Photocells, particularly the common CdS cells that you're likely to find, are not sensitive to all light. In particular they tend to be sensitive to light between 700nm (red) and 500nm (green) light.



Basically, blue light wont be nearly as effective at triggering the sensor as green/yellow light!

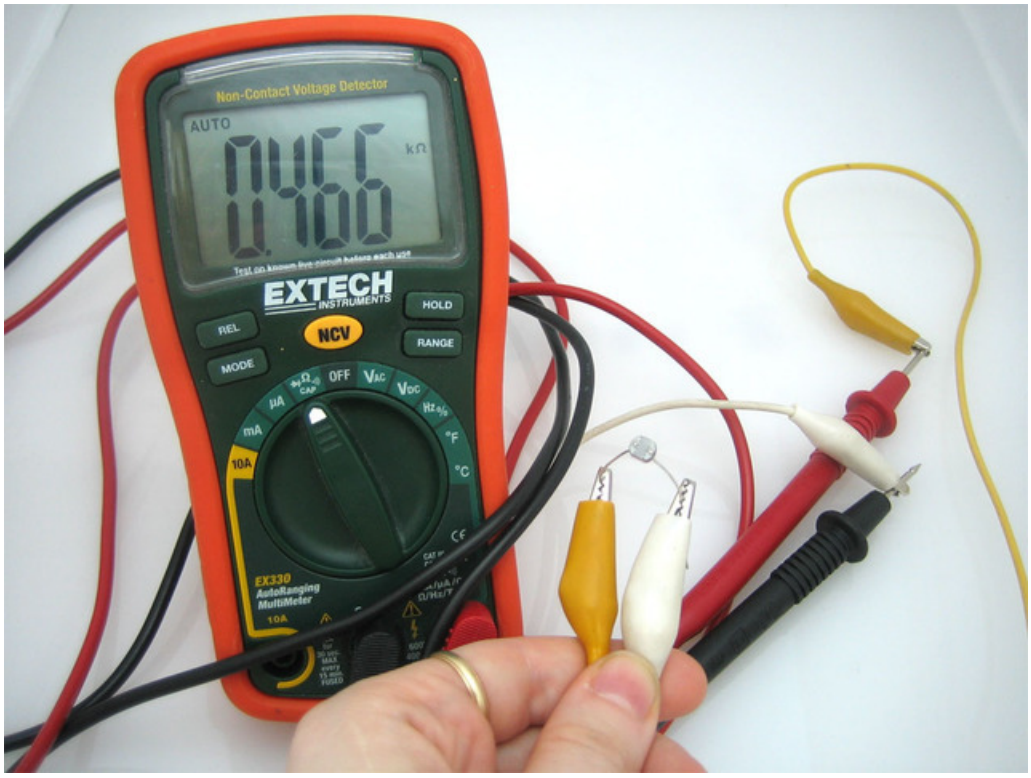
What the Heck is Lux?

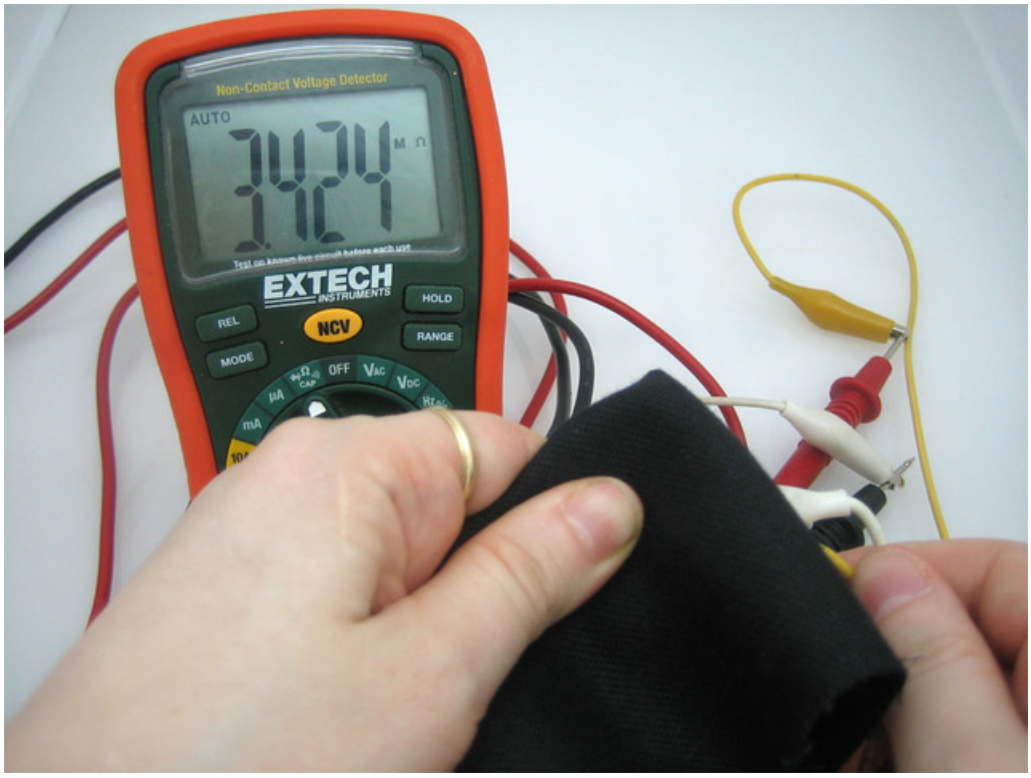
Most datasheets use lux to indicate the resistance at certain light levels. But what is lux ? Its not a method we tend to use to describe brightness so its tough to gauge. Here is a table adapted from a Wikipedia article on the topic!

Illuminance	Example
0.002 lux	Moonless clear night sky
0.2 lux	Design minimum for emergency lighting (AS2293).
0.27 - 1 lux	Full moon on a clear night
3.4 lux	Dark limit of civil twilight under a clear sky
50 lux	Family living room
80 lux	Hallway/toilet
100 lux	Very dark overcast day
300 - 500 lux	Sunrise or sunset on a clear day. Well-lit office area.
1,000 lux	Overcast day; typical TV studio lighting
10,000 - 25,000 lux	Full daylight (not direct sun)
32,000 - 130,000 lux	Direct sunlight

Testing a Photocell

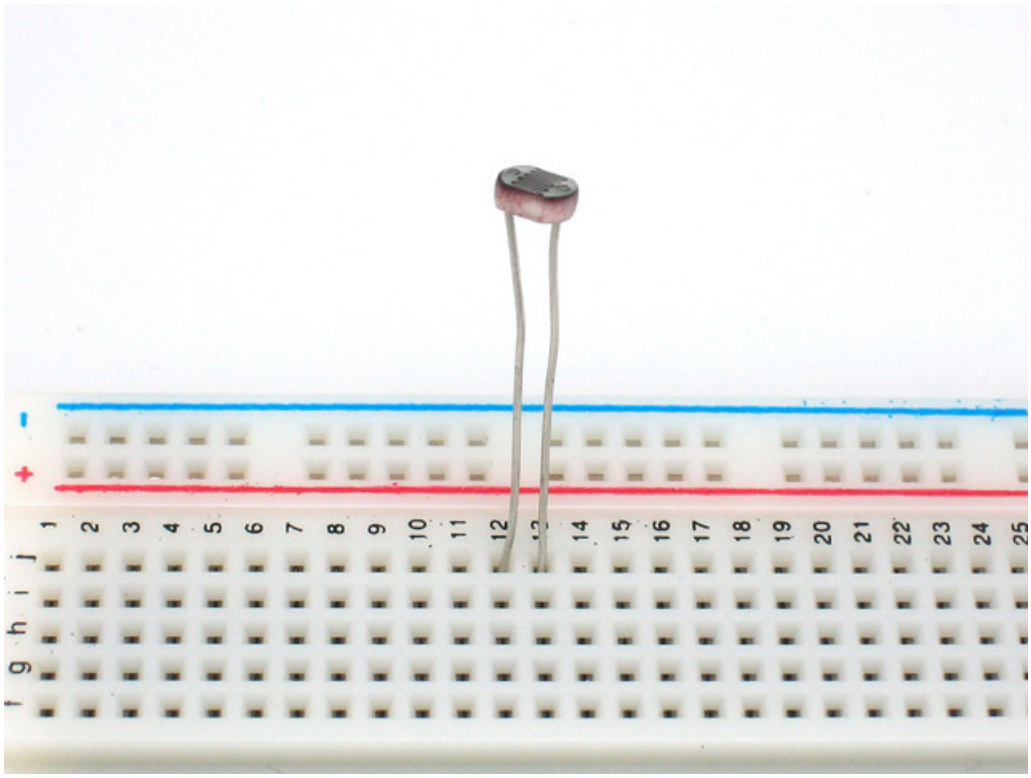
The easiest way to determine how your photocell works is to connect a multimeter in resistance-measurement mode to the two leads and see how the resistance changes when shading the sensor with your hand, turning off lights, etc. Because the resistance changes a lot, an auto-ranging meter works well here. Otherwise, just make sure you try different ranges, between 1MΩ and 1KΩ before 'giving up'.





Connecting a Photocell

Because photocells are basically resistors, they are non-polarized. That means you can connect them up 'either way' and they'll work just fine!

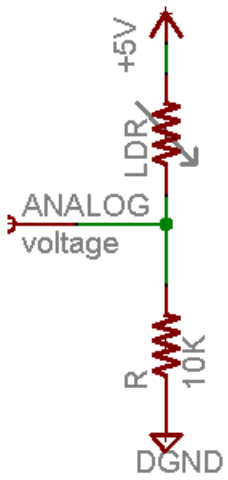


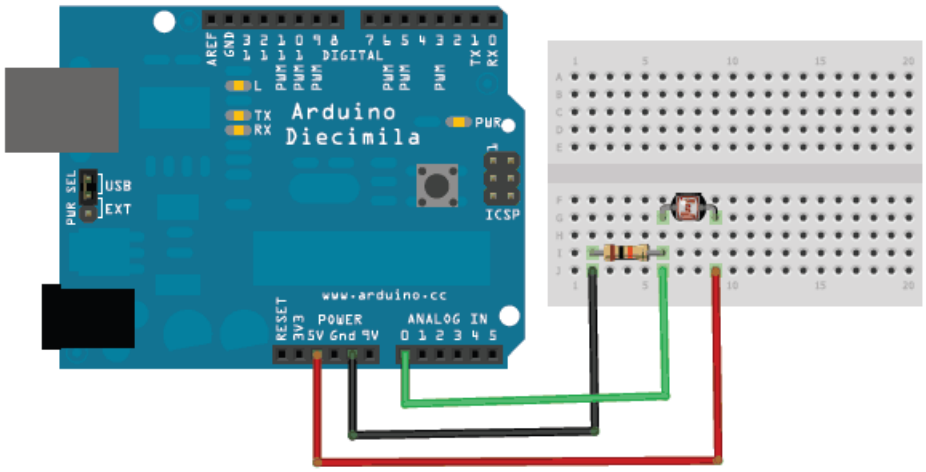
Photocells are pretty hardy, you can easily solder to them, clip the leads, plug them into breadboards, use alligator clips, etc. The only care you should take is to avoid bending the leads right at the epoxied sensor, as they could break off if flexed too often.

Using a Photocell

Analog Voltage Reading Method

The easiest way to measure a resistive sensor is to connect one end to Power and the other to a **pull-down** resistor to ground. Then the point between the fixed pulldown resistor and the variable photocell resistor is connected to the analog input of a microcontroller such as an Arduino (shown)





For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. In this configuration the analog voltage reading ranges from 0V (ground) to about 5V (or about the same as the power supply voltage).

The way this works is that as the resistance of the photocell decreases, the total resistance of the photocell and the pulldown resistor decreases from over 600KΩ to 10KΩ. That means that the current flowing through both resistors *increases* which in turn causes the voltage across the fixed 10KΩ resistor to increase. It's quite a trick!

Ambient light like...	Ambient light (lux)	Photocell resistance (Ω)	LDR + R (Ω)	Current thru LDR +R	Voltage across R
Dim hallway	0.1 lux	600KΩ	610 KΩ	0.008 mA	0.1 V
Moonlit night	1 lux	70 KΩ	80 KΩ	0.07 mA	0.6 V
Dark room	10 lux	10 KΩ	20 KΩ	0.25 mA	2.5 V
Dark overcast day / Bright room	100 lux	1.5 KΩ	11.5 KΩ	0.43 mA	4.3 V
Overcast day	1000 lux	300 Ω	10.03 KΩ	0.5 mA	5V

This table indicates the approximate analog voltage based on the sensor light/resistance w/a 5V supply and 10KΩ pulldown resistor.

If you're planning to have the sensor in a bright area and use a 10KΩ pulldown, it will quickly*saturate*. That means that it will hit the 'ceiling' of 5V and not be able to differentiate between kinda bright and really bright. In that case, you should replace the 10KΩ pulldown with a 1KΩ pulldown. In that case, it will not be able to detect dark level differences as well but it will be able to detect bright light differences better. This is a tradeoff that you will have to decide upon!

You can also use the "Axel Benz" formula by first measuring the minimum and maximum resistance value with the multimeter and then finding the resistor value with: Pull-Down-Resistor = squareroot(Rmin * Rmax), this will give you slightly better range calculations

Ambient light like...	Ambient light (lux)	Photocell resistance (?)	LDR + R (?)	Current thru LDR+R	Voltage across R
Moonlit night	1 lux	70 KΩ	71 KΩ	0.07 mA	0.1 V
Dark room	10 lux	10 KΩ	11 KΩ	0.45 mA	0.5 V
Dark overcast day / Bright room	100 lux	1.5 KΩ	2.5 KΩ	2 mA	2.0 V
Overcast day	1000 lux	300 Ω	1.3 KΩ	3.8 mA	3.8 V
Full daylight	10,000 lux	100 Ω	1.1 KΩ	4.5 mA	4.5 V

This table indicates the approximate analog voltage based on the sensor light/resistance w/a 5V supply and 1K pulldown resistor.

Note that our method does not provide linear voltage with respect to brightness! Also, each sensor will be different. As the light level increases, the analog voltage goes up even though the resistance goes down:

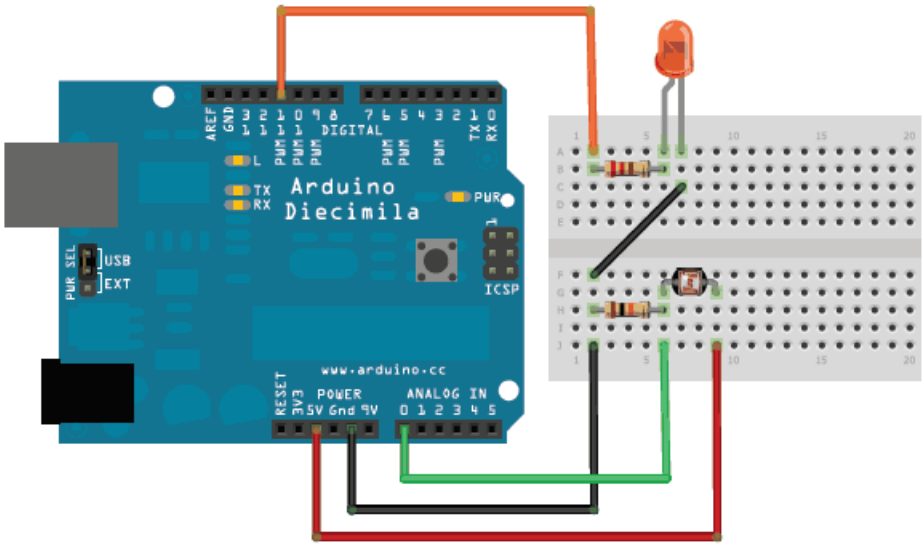
Vo = Vcc (R / (R + Photocell))

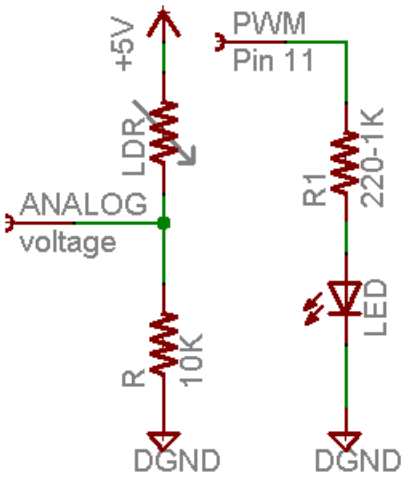
That is, the voltage is proportional to the **inverse** of the photocell resistance which is, in turn, inversely proportional to light levels.

Arduino Code

Simple Demonstration of Use

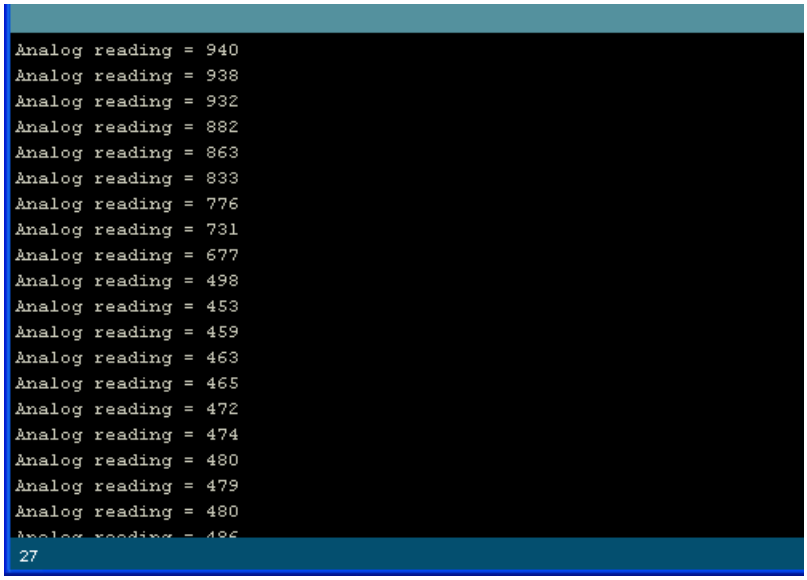
This sketch will take the analog voltage reading and use that to determine how bright the red LED is. The darker it is, the brighter the LED will be! Remember that the LED has to be connected to a PWM pin for this to work, I use pin 11 in this example.





These examples assume you know some basic Arduino programming. If you don't, maybe spend some time reviewing the basics at the Arduino tutorial?

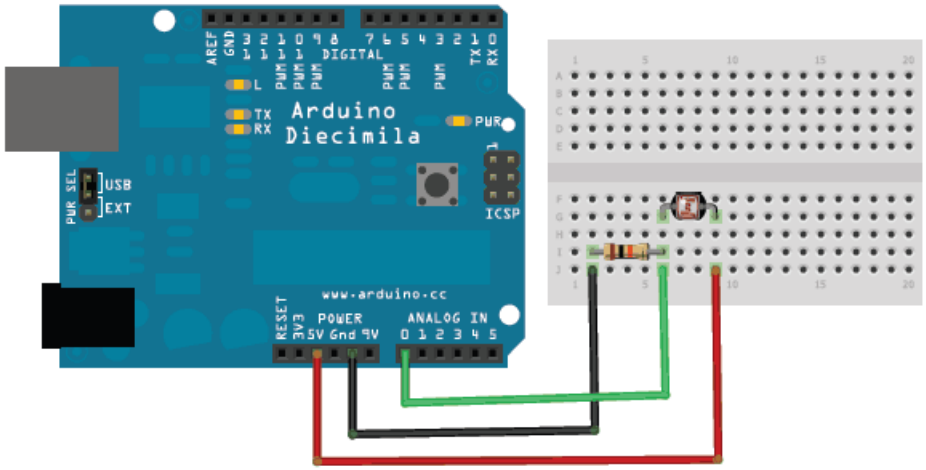
```
/* Photocell simple testing sketch.

Connect one end of the photocell to 5V, the other end to Analog 0.
Then connect one end of a 10K resistor from Analog 0 to ground
Connect LED from pin 11 through a resistor to ground
For more information see http://learn.adafruit.com/photocells */

int photocellPin = 0;     // the cell and 10K pulldown are connected to a0
int photocellReading;     // the analog reading from the sensor divider
int LEDpin = 11;          // connect Red LED to pin 11 (PWM pin)
int LEDbrightness;        //
void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.println(photocellReading);     // the raw analog reading

  // LED gets brighter the darker it is at the sensor
  // that means we have to -invert- the reading from 0-1023 back to 1023-0
  photocellReading = 1023 - photocellReading;
  //now we have to map 0-1023 to 0-255 since thats the range analogWrite uses
  LEDbrightness = map(photocellReading, 0, 1023, 0, 255);
  analogWrite(LEDpin, LEDbrightness);

  delay(100);
}
```

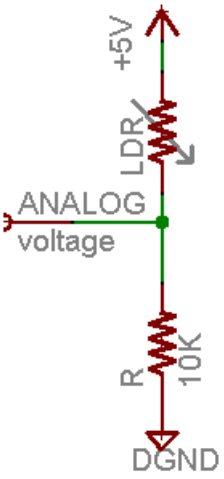


You may want to try different pulldown resistors depending on the light level range you want to detect!

Simple Code for Analog Light Measurements

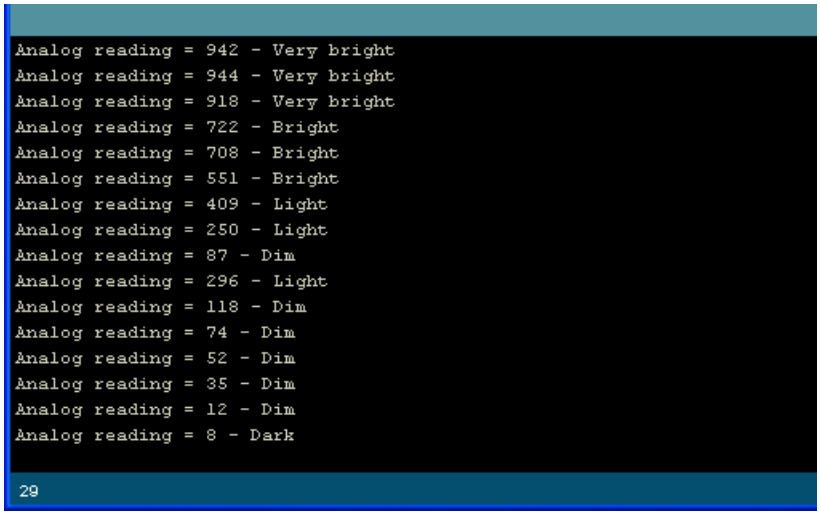
This code doesn't do any calculations, it just prints out what it interprets as the amount of light in a qualitative manner. For most projects, this is pretty much all thats needed!


Arduino
Diecimila
www.arduino.cc
USB
EXT
PWR SEL
ICSP
POWER
ANALOG IN


+5V
LDR
ANALOG
voltage
R
10K
DGND

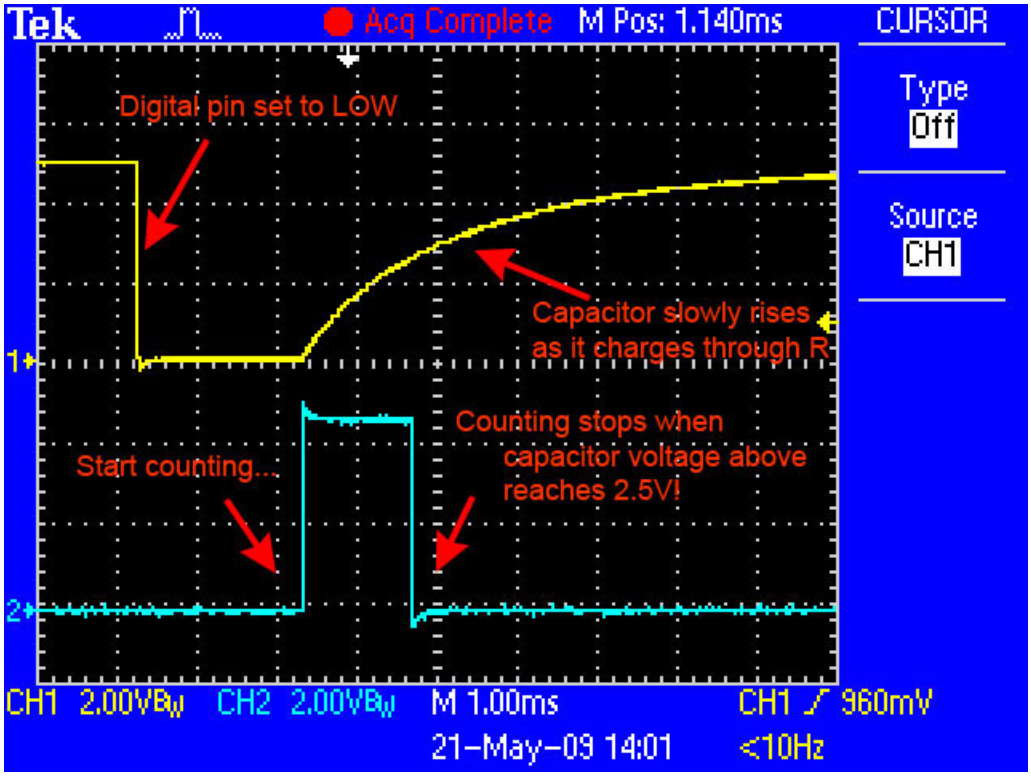
```
/* Photocell simple testing sketch.

Connect one end of the photocell to 5V, the other end to Analog 0.
Then connect one end of a 10K resistor from Analog 0 to ground

For more information see http://learn.adafruit.com/photocells */

int photocellPin = 0;     // the cell and 10K pulldown are connected to a0
int photocellReading;     // the analog reading from the analog resistor divider

void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.print(photocellReading);     // the raw analog reading

  // We'll have a few threshholds, qualitatively determined
  if (photocellReading < 10) {
    Serial.println(" - Dark");
  } else if (photocellReading < 200) {
    Serial.println(" - Dim");
  } else if (photocellReading < 500) {
    Serial.println(" - Light");
  } else if (photocellReading < 800) {
    Serial.println(" - Bright");
  } else {
    Serial.println(" - Very bright");
  }
  delay(1000);
}
```

To test it, I started in a sunlit (but shaded) room and covered the sensor with my hand, then covered it with a piece of blackout fabric.



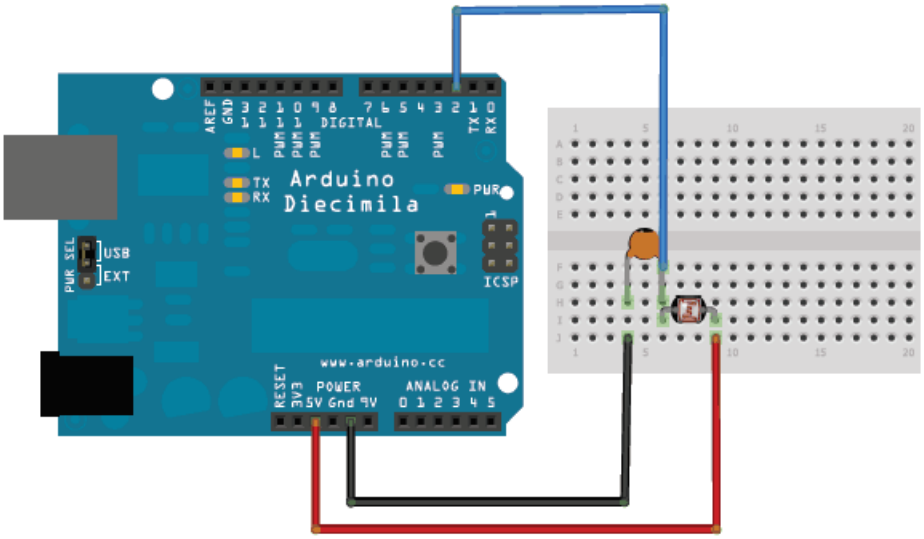
BONUS! Reading Photocells Without Analog Pins

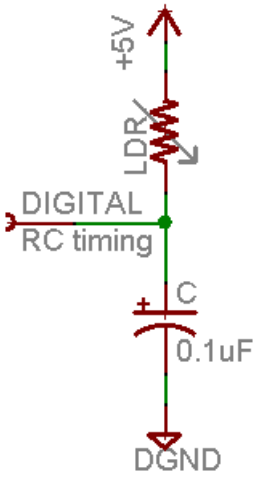
Because photocells are basically resistors, its possible to use them even if you don't have any analog pins on your microcontroller (or if say you want to connect more than you have analog input pins). The way we do this is by taking advantage of a basic electronic property of resistors and capacitors. It turns out that if you take a capacitor that is initially storing no voltage, and then connect it to power (like 5V) through a resistor, it will charge up to the power voltage slowly. The bigger the resistor, the slower it is.



This capture from an oscilloscope shows whats happening on the digital pin (yellow). The blue line indicates when the sketch starts counting and when the couting is complete, about 1.2ms later.

This is because the capacitor acts like a bucket and the resistor is like a thin pipe. To fill a bucket up with a very thin pipe takes enough time that you can figure out how wide the pipe is by timing how long it takes to fill the bucket up halfway.

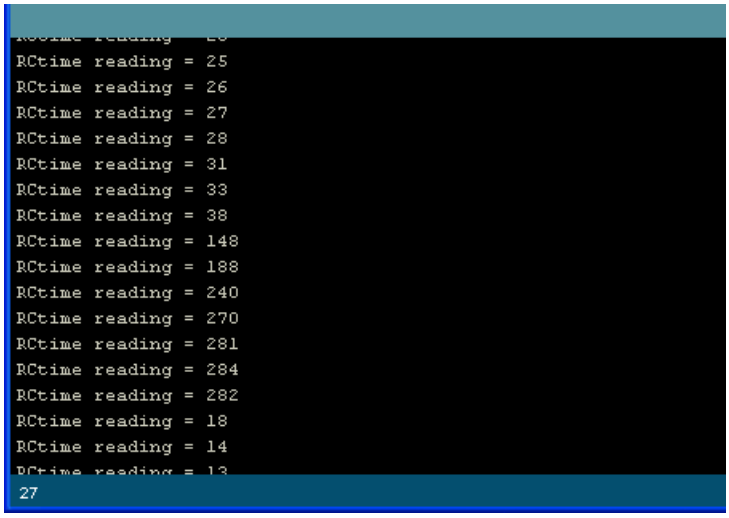




In this case, our 'bucket' is a 0.1uF ceramic capacitor. You can change the capacitor nearly any way you want but the timing values will also change. 0.1uF seems to be an OK place to start for these photocells. If you want to measure brighter ranges, use a 1uF capacitor. If you want to measure darker ranges, go down to 0.01uF.

```
/* Photocell simple testing sketch.
Connect one end of photocell to power, the other end to pin 2.
Then connect one end of a 0.1uF capacitor from pin 2 to ground
For more information see http://learn.adafruit.com/photocells */

int photocellPin = 2;     // the LDR and cap are connected to pin2
int photocellReading;     // the digital reading
int ledPin = 13;    // you can just use the 'built in' LED

void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);  // have an LED for output
}
```

```
void loop(void) {
  // read the resistor using the RCtime technique
  photocellReading = RCtime(photocellPin);

  if (photocellReading == 30000) {
    // if we got 30000 that means we 'timed out'
    Serial.println("Nothing connected!");
  } else {
    Serial.print("RCtime reading = ");
    Serial.println(photocellReading);     // the raw analog reading

    // The brighter it is, the faster it blinks!
    digitalWrite(ledPin, HIGH);
    delay(photocellReading);
    digitalWrite(ledPin, LOW);
    delay(photocellReading);
  }
  delay(100);
}

// Uses a digital pin to measure a resistor (like an FSR or photocell!)
// We do this by having the resistor feed current into a capacitor and
// counting how long it takes to get to Vcc/2 (for most arduinos, thats 2.5V)
int RCtime(int RCpin) {
  int reading = 0;  // start with 0

  // set the pin to an output and pull to LOW (ground)
  pinMode(RCpin, OUTPUT);
  digitalWrite(RCpin, LOW);

  // Now set the pin to an input and...
  pinMode(RCpin, INPUT);
  while (digitalRead(RCpin) == LOW) { // count how long it takes to rise up to HIGH
    reading++;      // increment to keep track of time

    if (reading == 30000) {
      // if we got this far, the resistance is so high
      // its likely that nothing is connected!
      break;           // leave the loop
    }
  }
  // OK either we maxed out at 30000 or hopefully got a reading, return the count

  return reading;
}
```

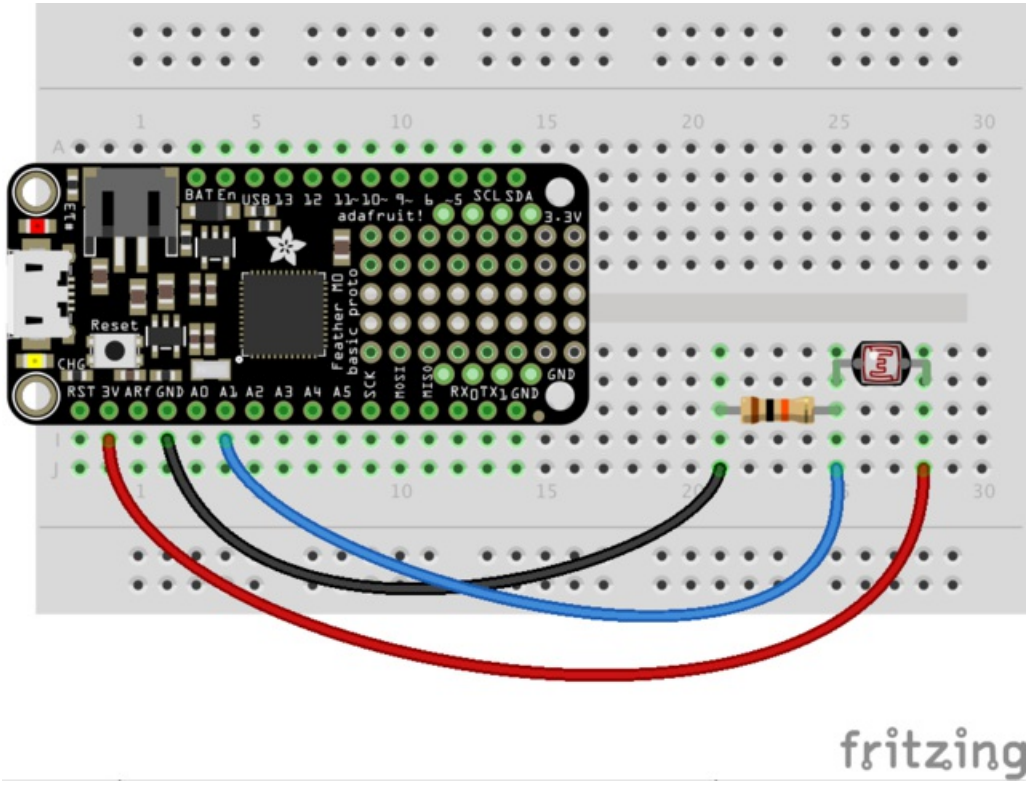

RCtime reading = 25
RCtime reading = 26
RCtime reading = 27
RCtime reading = 28
RCtime reading = 31
RCtime reading = 33
RCtime reading = 38
RCtime reading = 148
RCtime reading = 188
RCtime reading = 240
RCtime reading = 270
RCtime reading = 281
RCtime reading = 284
RCtime reading = 282
RCtime reading = 18
RCtime reading = 14
27

CircuitPython

It's easy to read how much light a photocell sees with CircuitPython and its built-in analog input support. By wiring the photocell to an analog input of your board you can read the voltage from it and see how it changes as the amount of light hitting the sensor changes too.

First wire up a photocell to your board as shown on the previous page for Arduino. You'll want to setup the same voltage divider with a **10 kilo-ohm resistor** circuit and feed the output into any analog input on your board (note the special method of reading photocells without an analog input is not currently supported by CircuitPython).

Here's an example of wiring a photocell to a Feather M0:



- **Board 3.3V** to **one leg of the photocell** (doesn't matter which leg). Note you want to use the voltage from your board that corresponds to the maximum analog input voltage. For Feather boards this is 3.3V, but for other boards it might be higher or lower--consult your board documentation to be sure.
- **10 kilo-ohm resistor** to the **other leg of the photocell.**
- **Board GND** to the **other leg of the 10 kilo-ohm resistor**.
- **Board A1** (or any other analog input) to the **junction of the photocell & 10 kilo-ohm resistor**.

Next connect to the board's serial REPL so you are at the CircuitPython >>> prompt.

Now import the **board** and **analogio** modules that allow you to read an analog input. Be sure you've read the CircuitPython analog I/O guide for more background on using analog inputs too!

```
import board
import analogio
```

Create an analog input for the A1 pin connected to the photocell & resistor junction:

```
photocell = analogio.AnalogIn(board.A1)
```

At this point you can read the value property to get a reading of the light seen by the photocell. Try it:

```
photocell.value
```

```
>>> photocell.value
58503
```

Try covering the photocell with your hand to block the light it can see and read the value again:

```
photocell.value
```

```
>>> photocell.value
22648
>>>
```

Notice the value changed! When the sensor sees less light the value is reduced. The more light the sensor sees, the higher the value.

You might wonder, what's the range of possible values? It turns out for an analog input in CircuitPython the maximum values range from 0 to 65535 (or the maximum 16-bit unsigned integer value). If you shine an extremely bright light on the photocell you might see a value near 65k, and if you completely block the sensor you might see a value down near 0.

If you're curious you can also convert this value into a voltage that's higher or lower depending on how much light is hitting the sensor. Let's make a function to do this:

```
def analog_voltage(adc):
    return adc.value / 65535 * adc.reference_voltage
volts = analog_voltage(photocell)
print('Photocell voltage: {0}V'.format(volts))
```

```
>>> def analog_voltage(adc):
...     return adc.value / 65535 * adc.reference_voltage
...
>>> volts = analog_voltage(photocell)
>>> print('Photocell voltage: {0}V'.format(volts))
Photocell voltage: 2.98317V
>>>
```

Cool! Notice the voltage increases up to near 3.3 volts as the light hitting the photocell increases. If you cover the photocell up and read the voltage you'll see it falls down near 0 volts.

You can use either the raw value or voltage to check how much light is hitting the photocell. Both will change proportionally to the amount of light hitting the sensor.

Here's a complete program that reads the photocell value and prints both the value and voltage every second. Save this as **main.py** on your board and open the serial output to see the printed values. Try shining light on the sensor or covering it up to see how the value and voltage change!

```
import time

import board
import analogio


# Initialize analog input connected to photocell.
photocell = analogio.AnalogIn(board.A1)

# Make a function to convert from analog value to voltage.
def analog_voltage(adc):
    return adc.value / 65535 * adc.reference_voltage

# Main loop reads value and voltage every second and prints them out.
while True:
    # Read the value, then the voltage.
    val = photocell.value
    volts = analog_voltage(photocell)
    # Print the values:
    print('Photocell value: {0} voltage: {1}V'.format(val, volts))
    # Delay for a second and repeat!
    time.sleep(1.0)
```

That's all there is to reading a photocell using an analog input with CircuitPython!

Example Projects

Noisemaker that changes frequency based on light level.

Motor value and directional control with photoresistors and microcontroller

Line-following robot that uses photocells to detect the light bouncing off of white/black stripes

Another robot, this one has two sensors and moves towards light (they're called Braitenberg vehicles)

Using a photocell and pocket laser pointer to create a breakbeam sensor

Buy a Photocell

Buy a Photocell (http://adafru.it/161)