

# Love to Code: Light Sensor

Add-on

# #1

Written by Jie Qi

Illustrations by K-Fai Steele



# Before You Begin

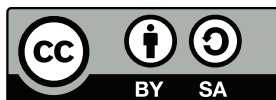
This is an add-on booklet to the Love to Code Creative Coding Kit. We wrote this booklet assuming you already have the reusable components included in the Creative Coding Kit, as well as the Light Sensor Add-on Materials Kit.

Make space for this add-on booklet by removing the pages currently inside your Chibi Book powered binder. Once you've slipped the add-on pages into the binder, you're ready to start!

If you haven't purchased the Love to Code Creative Coding Kit or the Light Sensor Add-on Materials Kit, you can get one via [chibitronics.com/lovetocode](http://chibitronics.com/lovetocode).

**Love to Code: Light Sensor** Copyright © 2018 by Jie Qi  
Some rights reserved.

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License:  
<https://creativecommons.org/licenses/by-sa/4.0/>



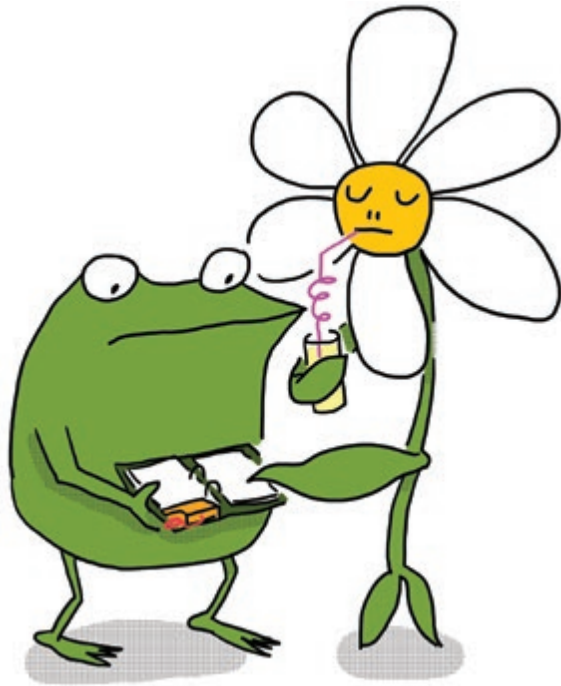
Publisher: Sutajio Ko-Usagi Pte Ltd  
dba Studio Kosagi, in Singapore  
[info@chibitronics.com](mailto:info@chibitronics.com)

Illustrator: K-Fai Steele  
Editor: Andrew "bunnie" Huang  
Technical Editor: Natalie Freed

ISBN: 978-981-11-6339-5



# Love to Code: Light Sensor



“Look! When I put my leaf over your circuit, nothing happens because your circuits are not light sensitive,” said George.

“Well then, how do we make it light sensitive?” asked Fern.

“With a light sensor, of course! *Do I have to explain everything?*” sighed George.

Fern grumbled. “Yes, George.”

# PROGRAM A LIGHT SENSOR

In Chapter 3 of *Love to Code Volume 1*, we learned how to make our circuit detect the press of a button. Just as our bodies have many different senses such as sight, hearing, and touch, circuits can also sense the world in different ways. In this activity, we will use a light sensor to make our projects react to light!

## You will need:



Chibi Chip



cable



programming device



Internet connection



USB Power connection



copper tape



conductive fabric patch



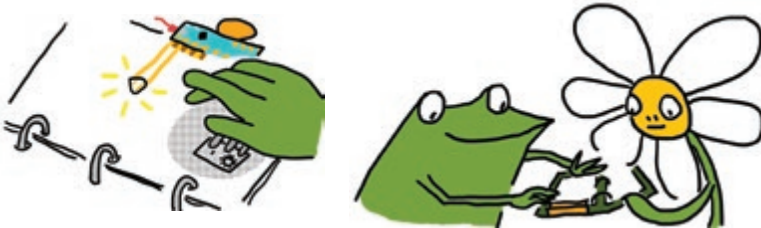
LED sticker



light sensor

1. Make the light sensor circuit on page **LS-4**. Instead of sticking the light sensor permanently to the page as with the LED stickers, leave the wax paper backing on. Place (don't stick!) the light sensor over the copper tape. Then, use three conductive fabric patches to stick it in place. This way the light sensor sticker can be reused in future projects!

2. Upload the **Basic Light Sensor** example code to your Chibi Chip! Visit **ltc.chibitronics.com** and select Examples → Sensors → Basic Light Sensor



Tah-Dah! When the light sensor (the little bump inside the sun shape) is made dark by covering it, pin 0 will turn on. Likewise, if it's exposed to light, pin 0 will turn off.

LIGHT SENSOR CIRCUIT NOT RESPONDING? FIRST, CHECK IF THE CODE UPLOADED PROPERLY:



Was the PROG button pressed before uploading?



Make sure the volume is all the way up!



No orange sound bar animation? Refresh the browser and try uploading again

IF THE CODE UPLOAD LOOKS GOOD, LET'S TRY CHECKING THE CIRCUIT:



Make sure the stripes on the Chibi Chip line up and contact the copper tape in the circuit.



Is the LED over pin 5 not turning on? Try going to a brighter spot or shining a brighter light on the light sensor!



Is pin 5 not turning off when it's dark? Check the fabric patch connections and make sure the light sensor is securely connected.



Is the LED over pin 0 turning on, but not the LED sticker on the page? Press harder on the LED sticker for a stronger connection.

CIRCUIT STILL NOT WORKING? CHECK OUT THE DEBUGGING CHAPTER FROM *LOVE TO CODE VOLUME 1* FOR MORE TIPS!



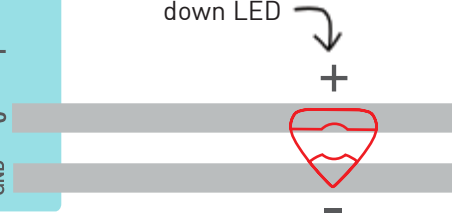
# LIGHT SENSOR TEMPLATE

1. Stick copper tape over the gray lines.



2. Place the light sensor over the footprint, but **do not remove the wax paper backing!**

4. Stick down LED



3. Stick conductive fabric patches over the copper tape and the light sensor's golden metal pads, like this:



This holds the light sensor in place while connecting it to the circuit!

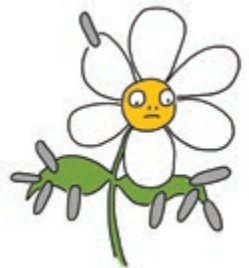
5. Clip the Chibi Chip

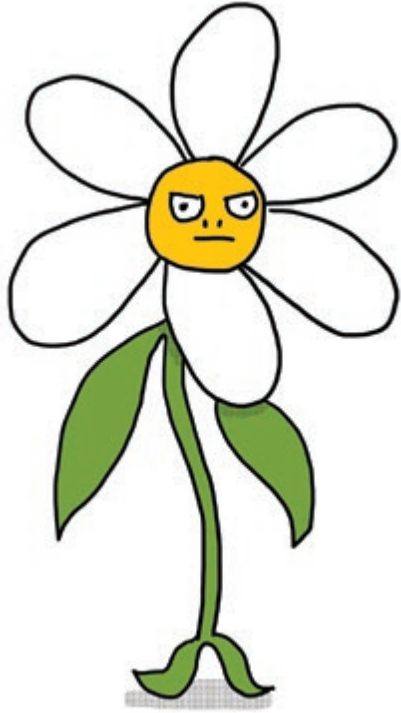


Connecting a light sensor with fabric patches lets us reuse the sensor sticker in other circuits! Just peel off the old fabric patches and use new patches to connect the light sensor to a new circuit.



Fabric patches can be used to connect other stickers, like LEDs, making them reusable too! This trick also works to recycle old LED stickers that have lost their stickiness. Be careful when peeling off old LED stickers, though! The stickers won't work if they are bent too much.





When you shine a light at George, something fades away! What is it?



# DECODE THE CODE

Let's look at the example code for the light sensor to see how it works:

```
// Love to Code
// Volume 1: Basic Light Sensor

int lightLevel = 0;

void setup() {
  pinMode(0); // pin 0 turns on and off the LED
  pinMode(5); // pin 5 connects to our light sensor
}

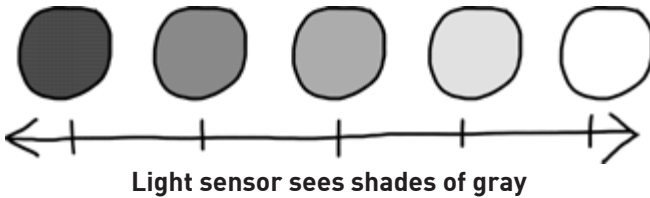
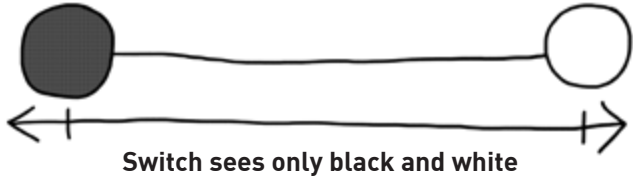
void loop(){
  lightLevel = read(5); // read light sensor from pin 5

  if(lightLevel == 1) {
    digitalWrite(0, LOW); // if bright, turn OFF pin 0
  } else {
    digitalWrite(0, HIGH); // otherwise, turn ON pin 0
  }
}
```

It works just like the switch code from page **3-7** of *Love to Code Volume 1!* We read the light sensor like a switch. When it's bright, the sensor reads 1 and when it's dark, the sensor reads 0. With this information, we can use the **lightLevel** variable and an **if()** statement to control the light on pin 0.

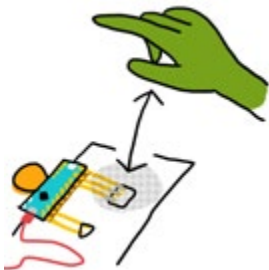
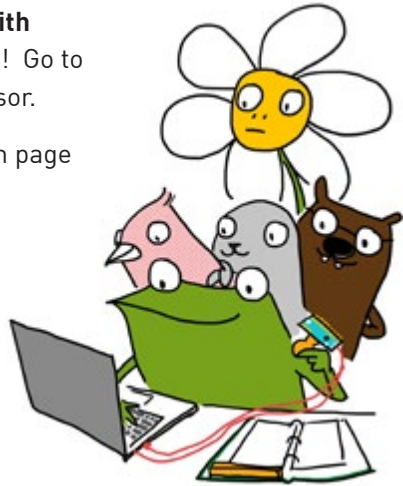


The cool thing about a light sensor is that unlike a switch, which can only sense fully on and fully off, a light sensor can sense light levels in between. One way to think about it is that a switch can only see black and white, while a light sensor can see black, white and shades of gray in between.



To see this in action, upload the **Fade With Sensor** example code to your Chibi Chip! Go to Examples → Sensors → Fade With Sensor.

Then, clip the Chibi Chip to the circuit on page **LS-4**.

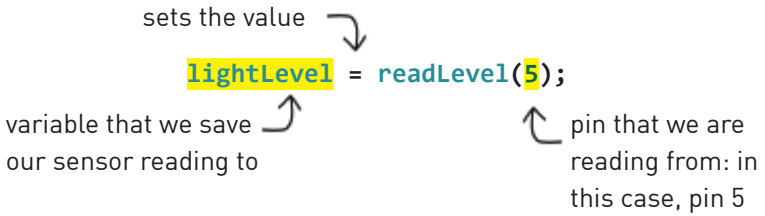


Now the more the light sensor is covered, the dimmer pin 0 becomes. Alternatively, shining a light at the sensor will make pin 0 become brighter!

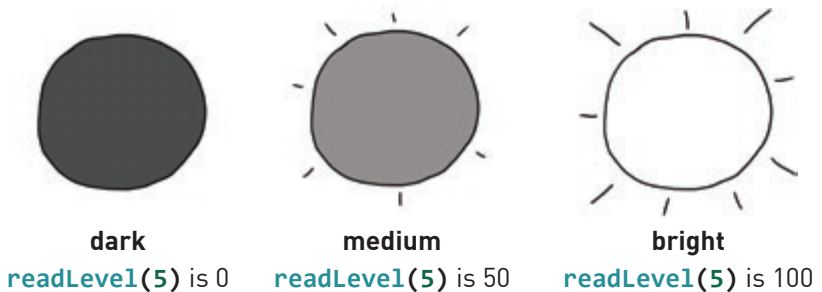
Let's take a look at the `loop()` function of our **Fade With Sensor** example code to see how it works:

```
void loop () {  
    lightLevel = readLevel(5); // read sensor level at pin 5  
    setLevel(0, lightLevel); // then set pin 0 to  
                             // the same "brightness"  
    pause(10); // tiny pause for the brightness to show  
}
```

First we read the voltage from the light sensor on pin 5 with `readLevel()` and then set the variable `lightLevel` to that value:



`readLevel()` gives us a number from **0** when it's completely dark to **100** when it's completely bright. To get information from the sensor, we use `readLevel(5)`, since the light sensor is connected to pin 5.



Then we set the brightness of pin 0 to the same brightness as the light sensor using `setLevel(0, lightLevel)`. Finally, we wait 10 milliseconds for the new brightness level to show.

# PLAY WITH CODE

Time to mess around with the light sensor code! We've already tried making pin 0 brighter with more light. How do we make a pin brighter with less light, so that it lights up when surroundings are dark? We just need a little math to do this! Try editing the `loop()` function code like this:

```
void loop(){  
  lightLevel = 100 - readLevel(5);  
  setLevel(0, lightLevel);  
  pause(10);  
}
```

After uploading this new code, the darker the light sensor gets, the brighter the light on pin 0 becomes! That's because now we set the light level to the opposite of the light sensor reading by subtraction:



dark:

`readLevel(5)` is 0  
`lightLevel` is 100 minus 0  
so, `lightLevel` is 100



medium:

`readLevel(5)` is 50  
`lightLevel` is 100 minus 50  
so, `lightLevel` is 50



bright:

`readLevel(5)` is 100  
`lightLevel` is 100 minus 100  
so, `lightLevel` is 0



Cool huh? Try making your own light patterns by playing with the light sensor! Can you get multiple pins to respond to the light sensor?



Light sensor readings can also be put inside `if()` conditions. Give this a try with the Light-o-meter activity on page **LS-12**!

For this activity, first load the **Light-o-meter** example code in Examples → Sensors → Light-o-meter. Then, complete the circuit template on page **LS-12** by reusing your light sensor from page **LS-4** with new fabric patches.



WHEN PEELING UP THE LIGHT SENSOR STICKER, GENTLY PEEL OFF THE OLD FABRIC TAPE, AND BE CAREFUL NOT TO BEND THE STICKER TOO MUCH, AS THAT WILL BREAK CONNECTIONS INSIDE THE STICKER! ALSO, USE FRESH PIECES OF FABRIC TAPE FOR EVERY PROJECT TO ENSURE A RELIABLE CONNECTION.



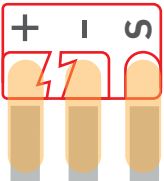
YEP!



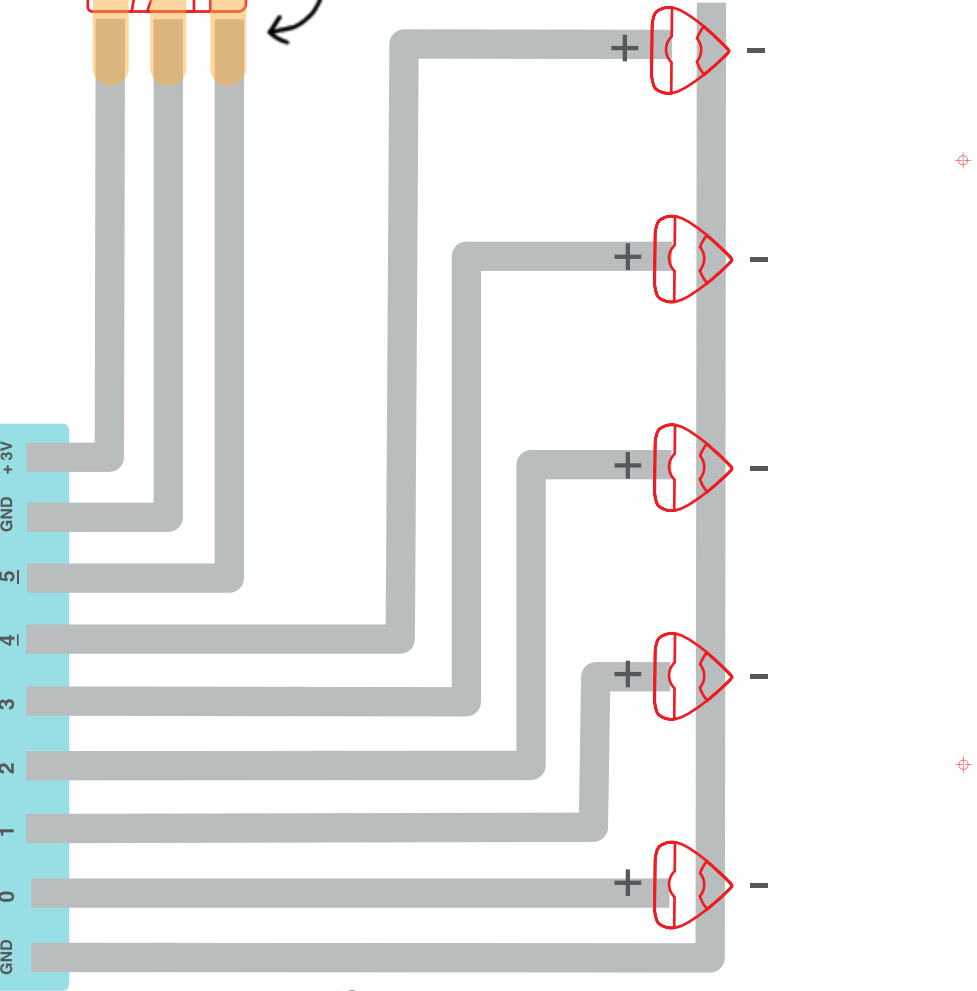
NOPE.

# LIGHT-O-METER

The brighter the light sensor's reading, the more LEDs will light up!



Reuse your light sensor from page **LS-4** and connect it to this circuit using new fabric patches!



Turn copper tape at corners with this folding trick:



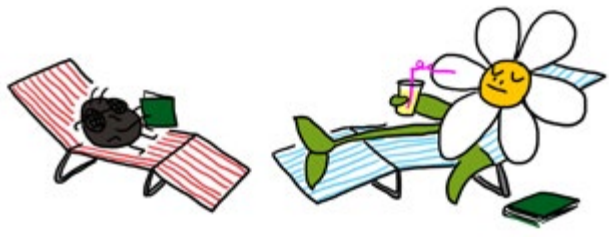
1. Fold tape back, so the sticky side faces up

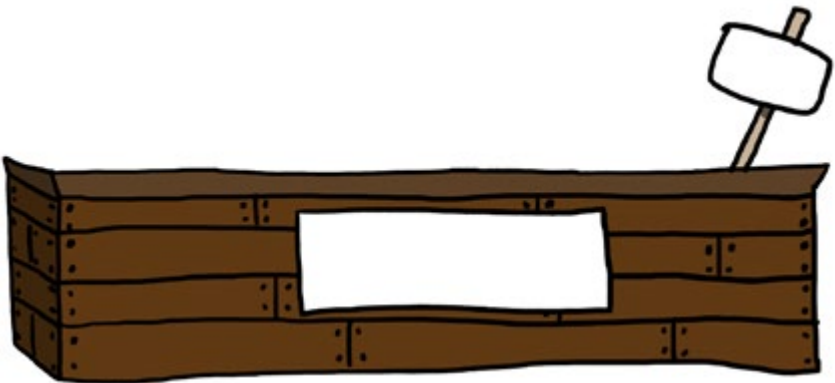


2. Flip and turn the tape



3. Flatten the corner!



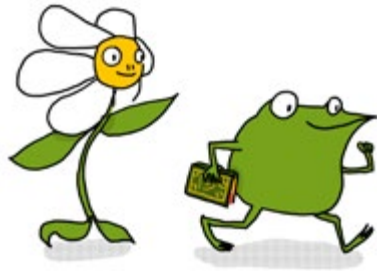


LS-14 What grows when you shine light on the sun? Draw it above!



# SENSING THE ENVIRONMENT

The cool thing about a light sensor is that not only can it be triggered by covering it with hands or shining a light on it, the sensor also responds to light level changes in the environment!



Where's a sunny spot for growing plants? Try taking your circuit to different places to find out! For example:



near a window



underneath a table



in the bathroom



outdoors

Try checking the same places both at night, and in the afternoon! What differences do you notice?

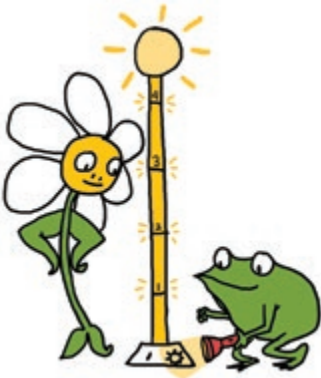
# DECODE THE CODE

Let's take a look at how our **Light-o-meter** code works! Here's the **loop()** function of our example code:

```
void loop(){
  lightLevel = readLevel(5); // read light level
  if(lightLevel > 0) { // if level is greater than 0,
    on(0); // turn on pin 0
  } else {
    off(0); // otherwise turn off pin 0
  }
  if(lightLevel > 20) { // if level is greater than 20,
    on(1); // turn on pin 1
  } else {
    off(1); // otherwise turn off pin 1
  }
  if(lightLevel > 40) { // if level is greater than 40,
    on(2); // turn on pin 2
  } else {
    off(2); // otherwise turn off pin 2
  }
  if(lightLevel > 60) { // if level is greater than 60
    on(3); // turn on pin 3
  } else {
    off(3); // otherwise turn off pin 3
  }
  if(lightLevel > 80) { // if level is greater than 80,
    on(4); // turn on pin 4
  } else {
    off(4); // otherwise turn off pin 4
  }
}
```



We use `if()` statements and the `>` “greater than” **operator** to create a light level **threshold** for each LED pin. An operator is a symbol that tells the Chibi Chip to do some kind of math or logical operation. In this case, the light level needs to be greater than the threshold number in order for the pin to turn on. The higher the threshold, the brighter the light level must be before that pin will turn on.



light greater than 80

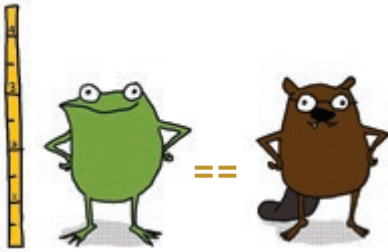


light not greater than 40

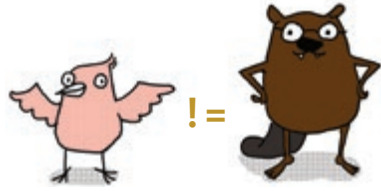
It's just like the height limit at a scary ride in the amusement park. In order to go on the ride, you have to be taller than the threshold height.



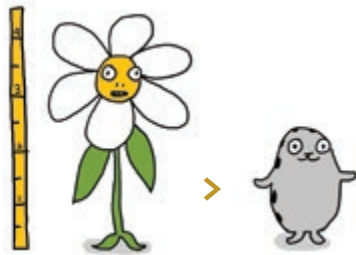
There are lots of ways to make comparison conditions for `if()` statements, not just with `==` **equals to** and `>` **greater than**. Here's a list of comparison operators, also known as relational operators:



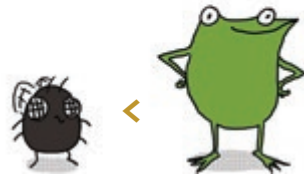
`==` **equal to**  
(in height)



`!=` **not equal to**



`>` **greater than**



`<` **less than**



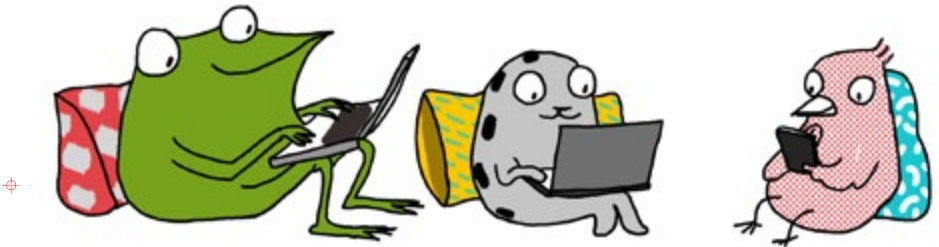
`>=` **greater than or equal to**



`<=` **less than or equal to**

# PLAY WITH CODE

Can you make the plant on page **LS-14** grow taller the darker it gets? Try changing your code so that the light meter goes higher the less light there is!



Here's a hint: try playing around with the comparison operators and the thresholds so that pins turn on when light levels are below the threshold number.

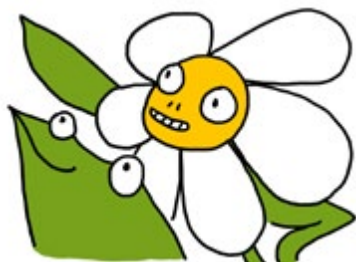


STILL STUMPED? CHECK OUT THE  
DARK-O-METER EXAMPLE CODE!

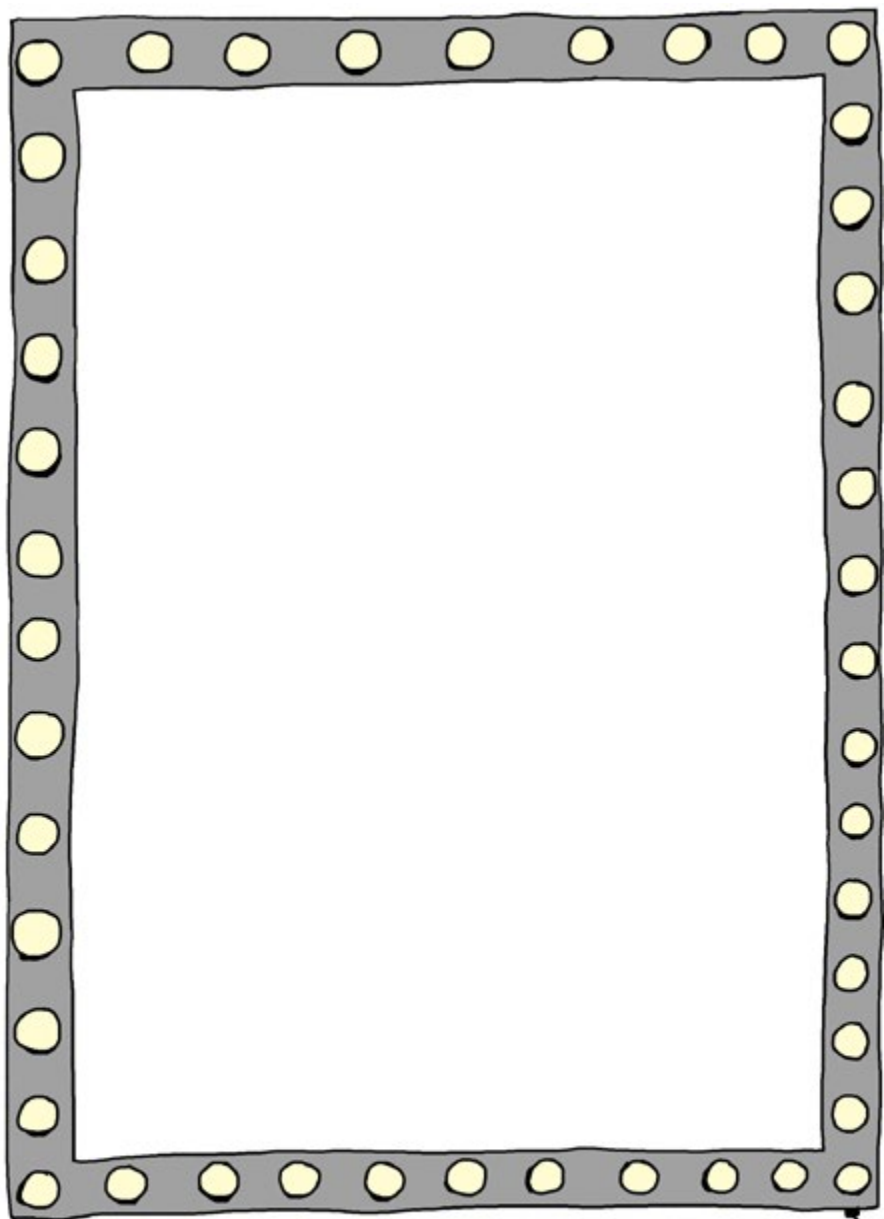
# LET'S PLAY!

Time to create your own light-reactive artwork! First, draw a scene on page **LS-22** and then build a light-reactive circuit for your artwork here!





*Light Sensor* **LS-21**



Draw a light-reactive scene here!

LS-22





# THINK OUTSIDE THE BOOK

Nice work! You've learned so many cool circuits and programs. Now you're ready to take your skills to the next level by taking your projects off the page and outside of this book. Here are some fun things to try!

✦ A fabulous hat that sparkles light patterns when a bright light shines at it!



A spooky haunted house that comes to life when it gets dark!

✦ A magic wand that triggers a circuit from far away, like a remote control. Hint: send signals with a light on the wand, and receive it on the other side with a light sensor!



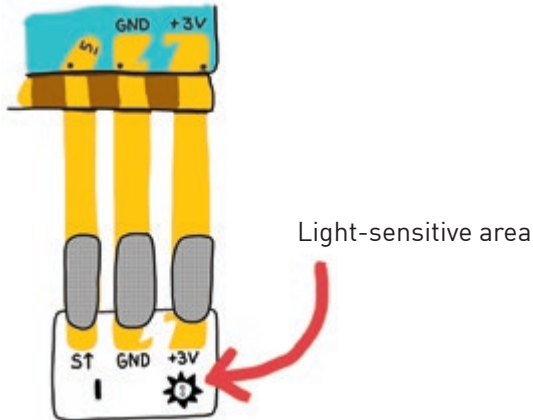


# DOWN THE RABBIT HOLE: DIGITAL AND ANALOG

Light sensors are pretty nifty, huh? Let's dig into how they work!



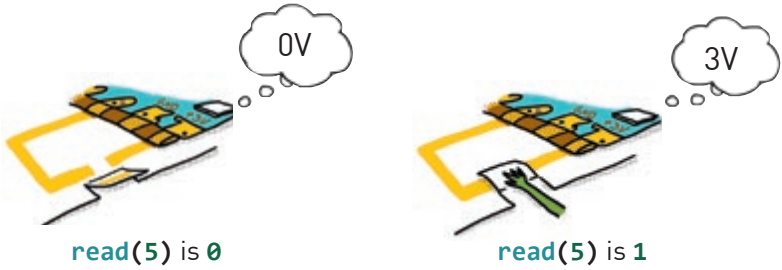
First, the light sensor has to be connected to GND and +3V so that it has power. Once powered, the light sensor produces a voltage depending on the light level and sends it to the **S** pad. The S stands for **signal**.



Similar to a switch, a light sensor sends a changing voltage to an input pin that tells the Chibi Chip information about the world. However, with a switch the voltage depends on whether someone is pressing or not, while with the light sensor the voltage changes depending on the amount of light that hits the light sensor.

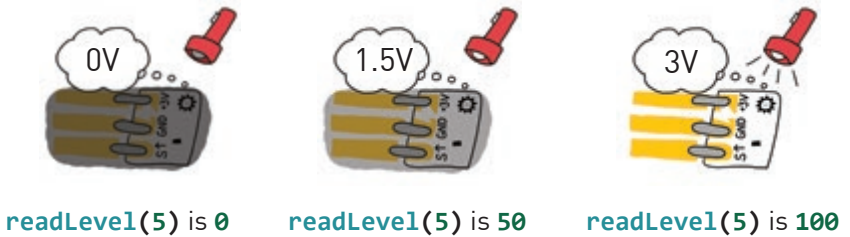
**LS-24**

A big difference between a switch and a light sensor is that a switch can only be 0V or 3V, while a light sensor can be 0V, 3V or any voltage in between. A switch can only read **0** for OFF and **1** for ON, which is called a **digital** reading. Digital means that we can only work reliably with values that are fully on (3V) or fully off (0V), and nothing in between.



### using `read(pin)` for digital readings

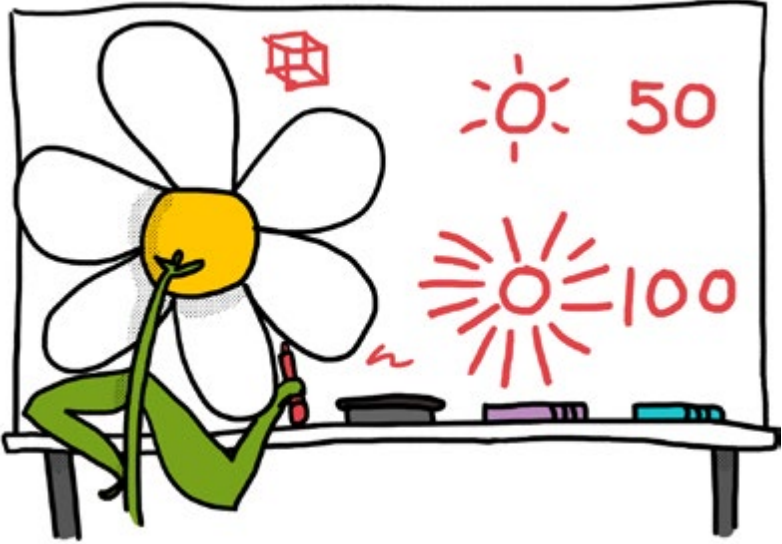
Meanwhile the light sensor can read any brightness level from **0** for completely dark to **100** for fully bright, which is called an **analog** reading. Analog means that we can work with values that are in between fully on and fully off — in this case, voltages that anywhere in between 0V and 3V.



### using `readLevel(pin)` for analog readings

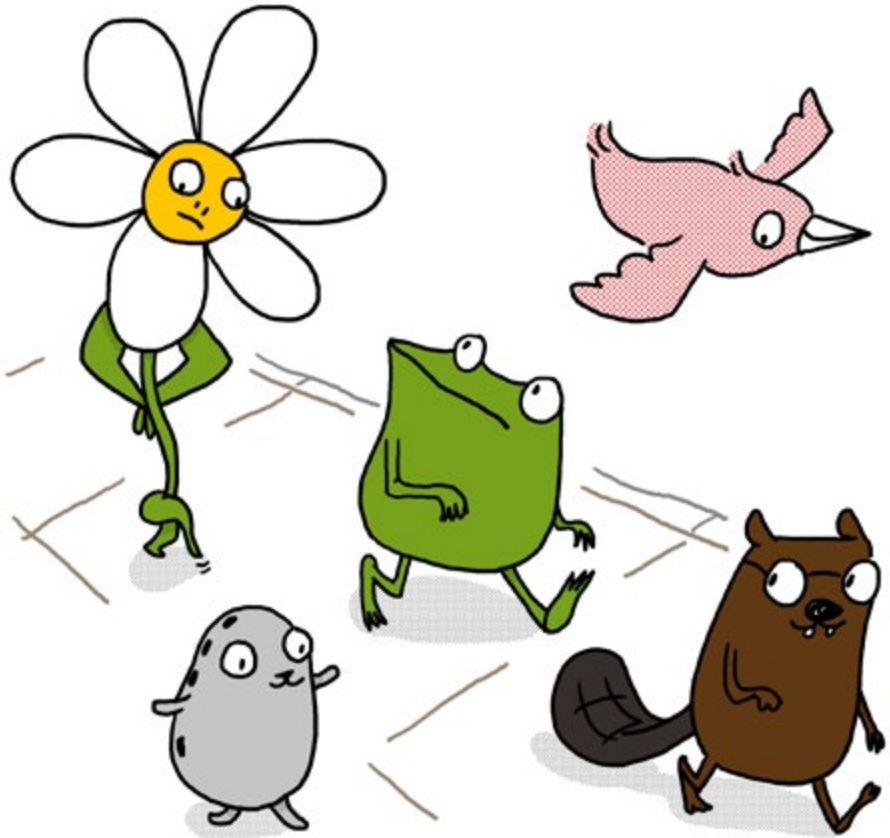
The terms digital and analog aren't just for sensors and inputs. `on()` and `off()` are digital output functions: they turn LEDs either fully on, or fully off. Likewise, `setLevel()` is an analog output function because it can make an LED appear to be at various light levels, not just fully on or fully off.

The fun thing about working with analog values and functions is that it gives us more information about the world. For example, the light sensor tells us not only that there's light, but also how much light.



As we learn about more types of circuits and programs, we'll be using analog functions more and more because we can use the additional information along with neat math tricks to make more complex interactions. In other words, analog values allow us to make our projects do even more cool stuff!





“This is so cool,” said Fern. “Thanks for showing us how to use light sensors, George! See ya later!”

“Are you leaving already?” asked George, “We’re just getting started! Can we please keep hanging out?”

Fern thought for a second. George wasn’t always the sunniest flower to be around, but once they started to learn things together he was actually pretty cool. She glanced at her friends, and then knew her answer.

“Sure!” Fern declared.



Happy making and see you again soon!

# STAY TUNED!

Thanks for joining us on this adventure! Stay tuned to **[chibitronics.com/lovetocode](https://chibitronics.com/lovetocode)** for new chapters on even more circuits and programming techniques with Fern and friends!



