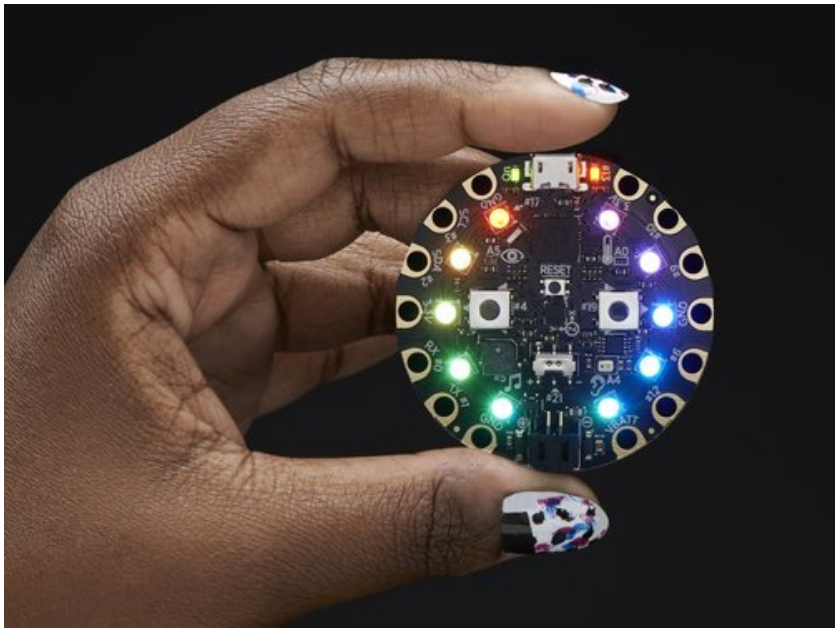
Circuit Playground Lesson #0

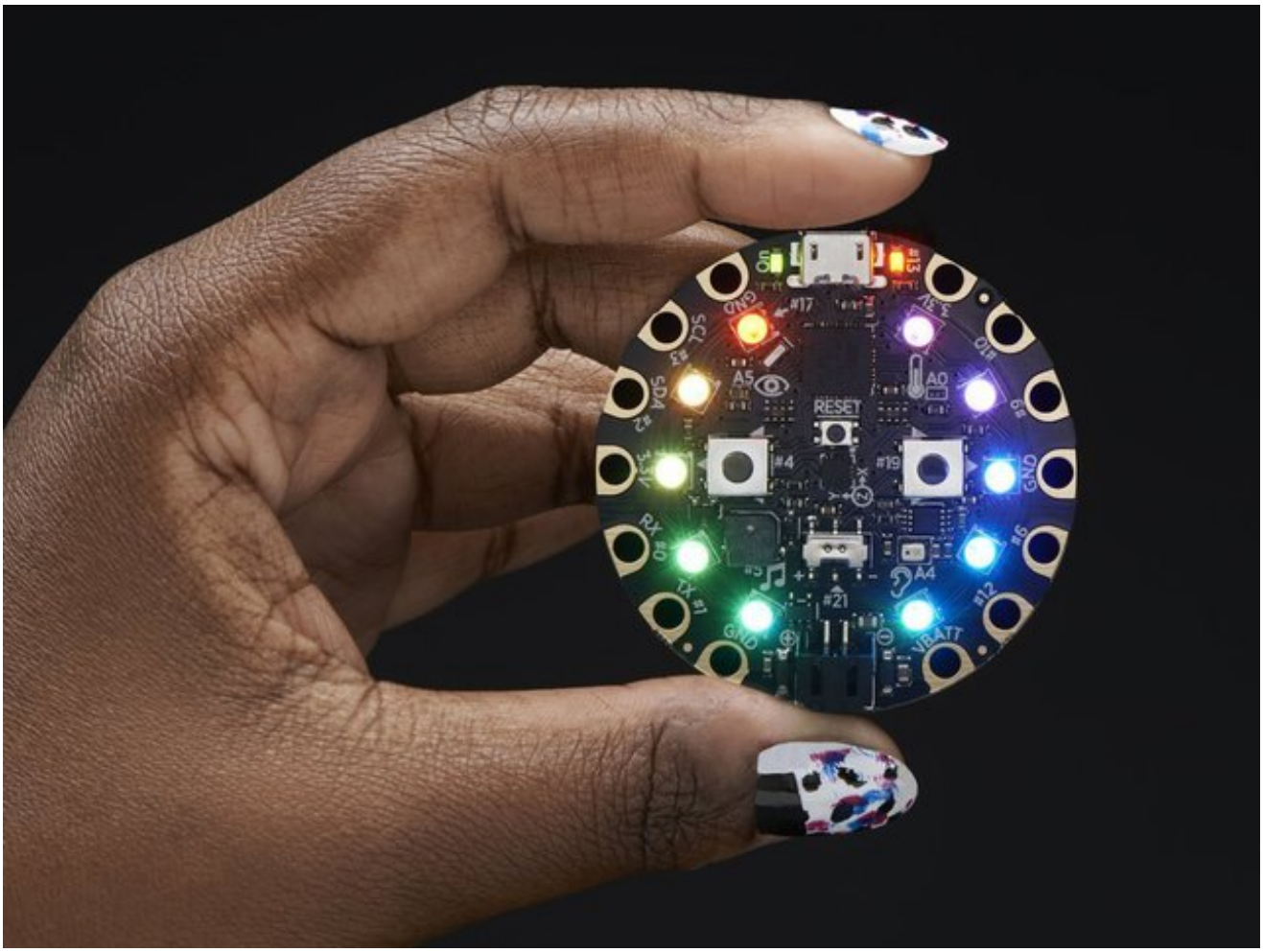
Created by lady ada



Last updated on 2016-08-30 07:36:12 PM UTC

Guide Contents

Intro



Hi there!

If you're here, it's because you want to learn how to build and make stuff with electronics! (If, rather than learn electronics, you'd like to look at pictures of cats instead, please check https://www.adafruit.com/galleries/cats-of-engineering (http://adafru.it/oAd))

And, you're in luck: there's *never* been a better time.

Gone are the days where you need thousands of dollars of equipment and lots physics/math background. Nowadays, if you want to learn to work with electronics, you can jump right in for $20 or less, and any sort of computer. And we're talking about learning *a lot* of electronics - from the basics of analog to the complexities of firmware. With a good

pack of parts, you can build a base of knowledge that will take you from your first blinking LED to someone who can start prototyping and inventing custom products.

Who is this for?

Anyone with a computer they can install software on, an Arduino or compatible and the ability to type and click a mouse. That's pretty much the minimum.

You don't need to know a lot of physics or math, and just like an Art Degree isn't required for making art and being creative, **you *don't* need to have a computer science degree**. It helps if you're comfortable using computers but that's a skill most people pick up through life.

If you know how to program already - great! If not, don't worry, we'll teach you enough to be dangerous.

What is a Circuit Playground?

Circuit Playground is the name of the little round electronic circuit board that you are going to use as a tool to investigate and explore programming & electronics. Think of it as like a Swiss Army Knife or multi-tool for learning electronics as an art form! It is an all-in-one board so you can jump in quickly and do a whole lotta projects (It can slice, it can dice, it has lots of blinky LEDs!)

Circuit Playground can light up multicolored LEDs (make beautiful lighting effects), make sounds (like a synthesizer!), detect touch or color, sense the enviroment like temperature, sound, and motion, and more!

By programming your Playground with software, you can combine all these abilities. For example, it can use the sound sensor to listen to music, and then pulse the LEDs to react to the melody and rhythm it hears. Or, you can use the motion sensor to detect when someone moves your purse and play a loud alarm to scare them off!

Circuit Playground works with the Arduino software. You can use the Arduino IDE software to program Circuit Playground. The main chip in Circuit Playground is natively supported, which means you can use the thousands of projets and tutorials seamlessly!

Best of all, no soldering required! You do not need *anything* to get started other than a Circuit Playground, a common USB cable and any computer (Windows/Mac/Linux/Chromebook)

Who are you?

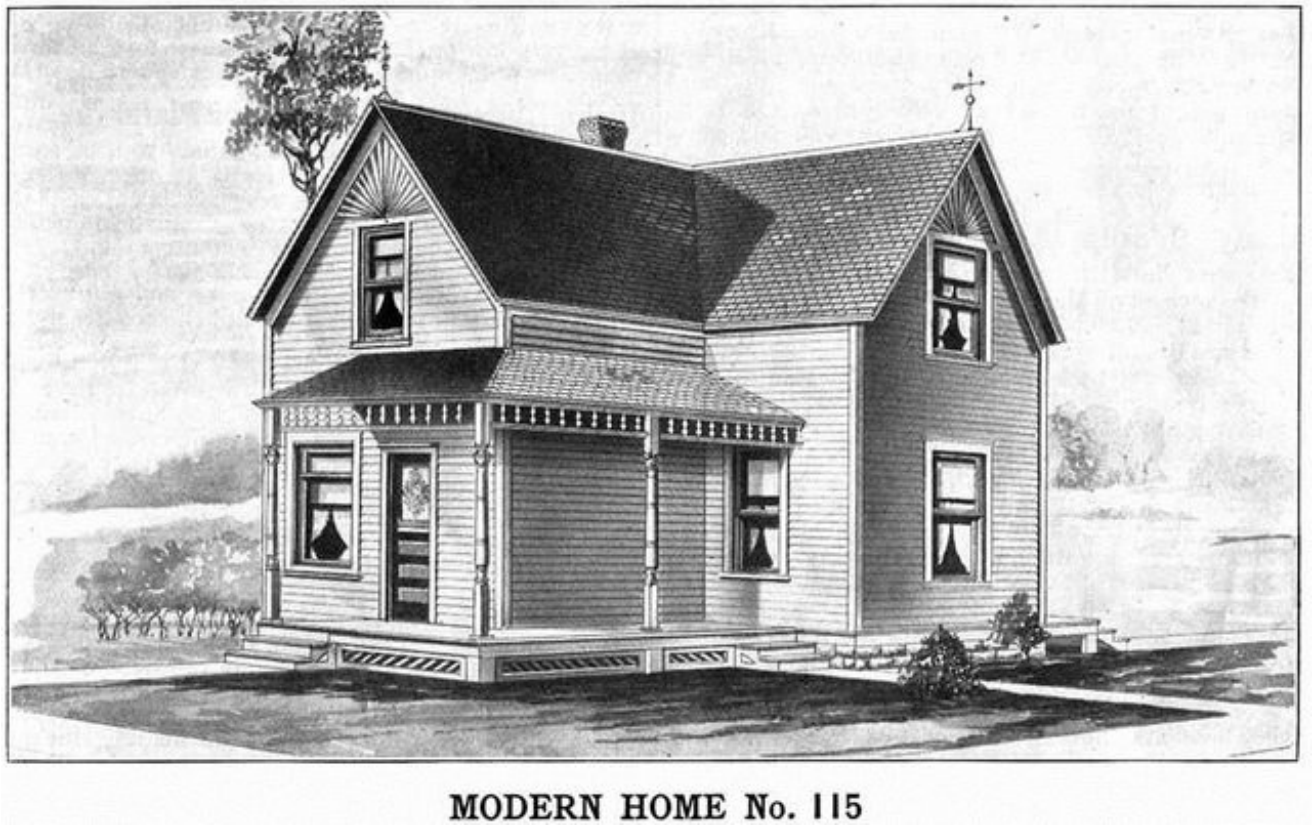
Great question. This is me:

I'm Ladyada, and I love to teach people how to build stuff and how they can be creative with technology.

So, are you ready?

Let's do this thing!

About This Lesson



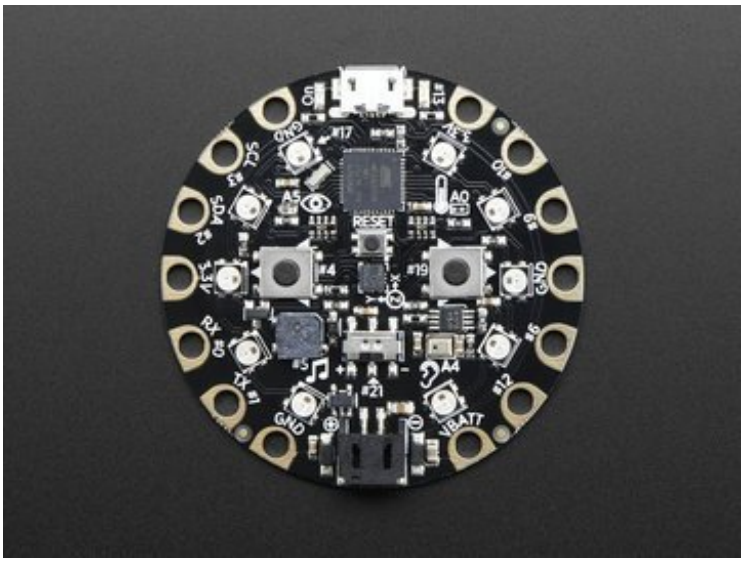
This lesson won't teach any electronics, really. Its more for making sure that everything is setup and *ready* for the future lessons. It will verify the Circuit Playground is working as intended and that the computer you are using is compatible.

Think of this tutorial as the **'home base'** of your journey. If things ever get into a weird spot *come back here and re-verify this lesson!*

One of the most important skills you'll have to learn is that when things go wrong (and they will, *tons*) come back to the most basic assumptions. This is a little bit of the "are you sure its on" of electronics. It's surprising how many skilled engineers will spend hours debugging a circuit to realize that...it wasn't plugged in!

Lesson Parts

Required:

- 

  A **Circuit Playground**! Of course, this is essential, and is required for all lessons.

  Available at Adafruit (http://adafru.it/3000)

- 

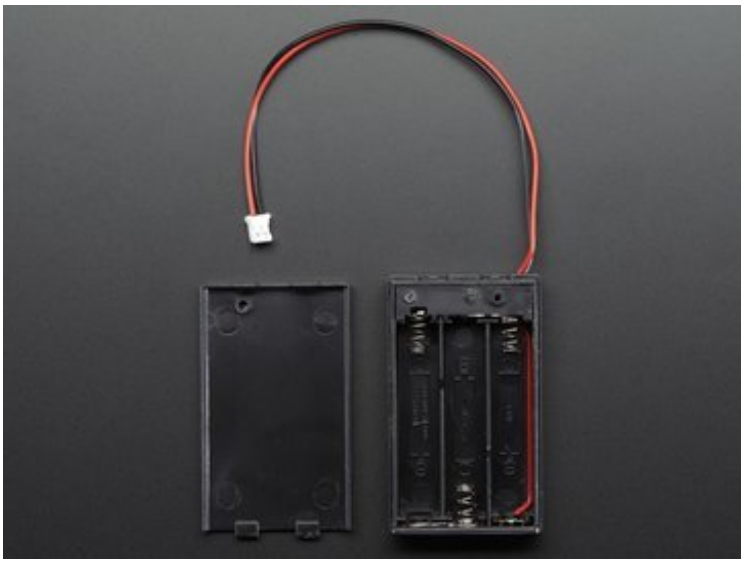
  Micro USB Cable any length. You probably have one of these around, they're the most common USB cable.

  Make sure its a data/sync cable!

  USB cable available at Adafruit (http://adafru.it/592)

A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times students have nearly given up due to a flakey USB cable!

Not Required but... Recommended!

- 
  If you'd like to power your Playground for portable use, you can use a 3 x AAA battery holder with a JST PH-2 connector on it. These holders even come with an on/off switch so you can save battery when not in use.
- 

Of course, 3 AAA batteries are required as well. Circuit playground works best with Alkaline or rechargeable NiMH

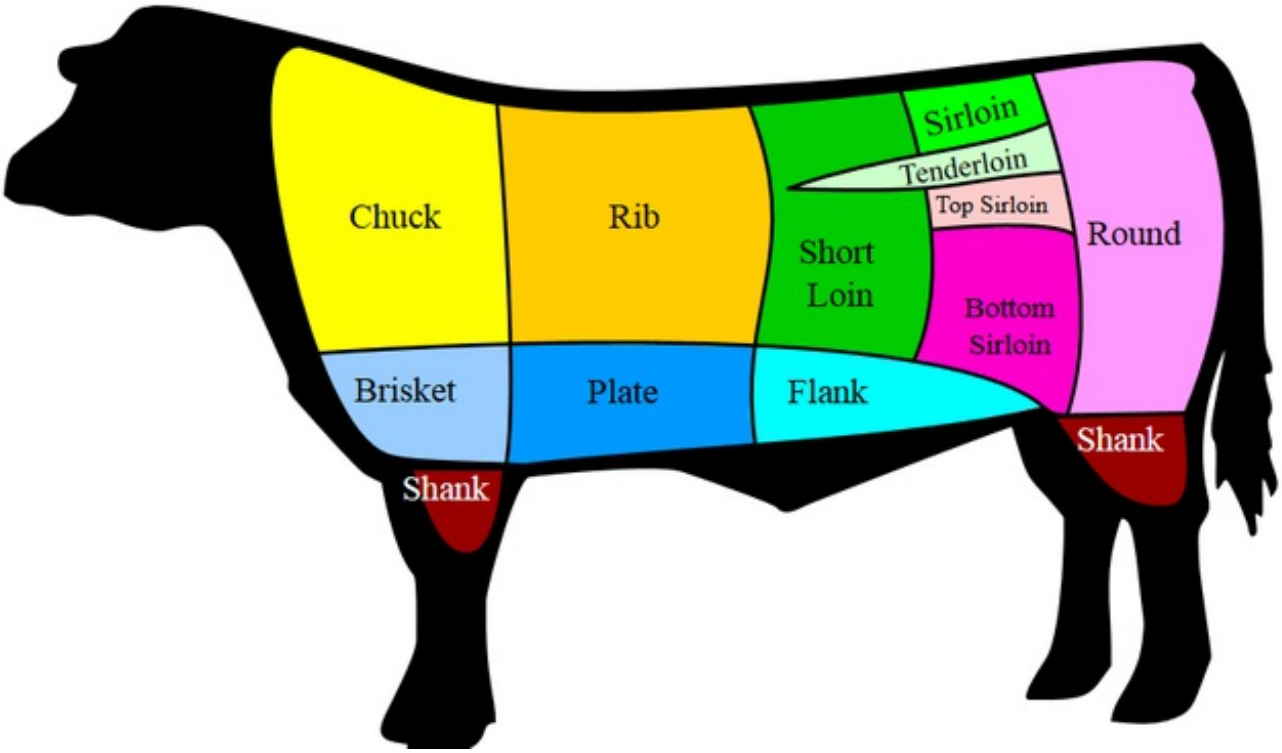
- 

batteries.

You can also use other AAA batteries such as Zinc or Lithium. NiCad batteries are not recommended.

Please use rechargeable batteries if you can, to reduce waste!

Take a Tour!

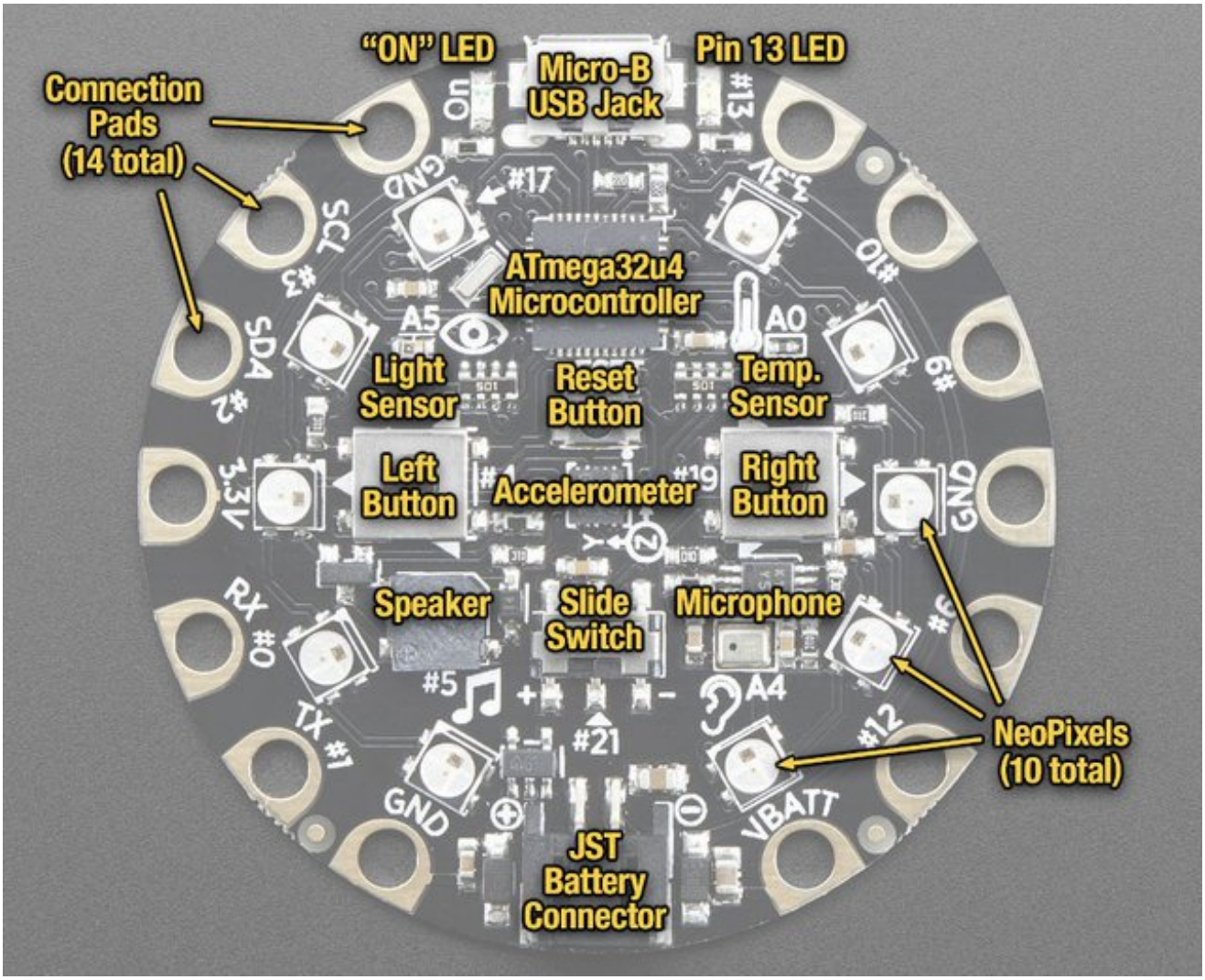


It's handy to know the names for parts of a cow when talking about cuts of meat!

Here we'll take a tour and point out the names of the Circuit Playground. You'll want to refer back to this page a ton, so keep it handy when we say something like "JST connector" or "Reset button"

The Parts of a Circuit Playground

Here's a rough version of the parts of an Circuit Playground we'll refer to. Each part is covered in more detail in the next sections

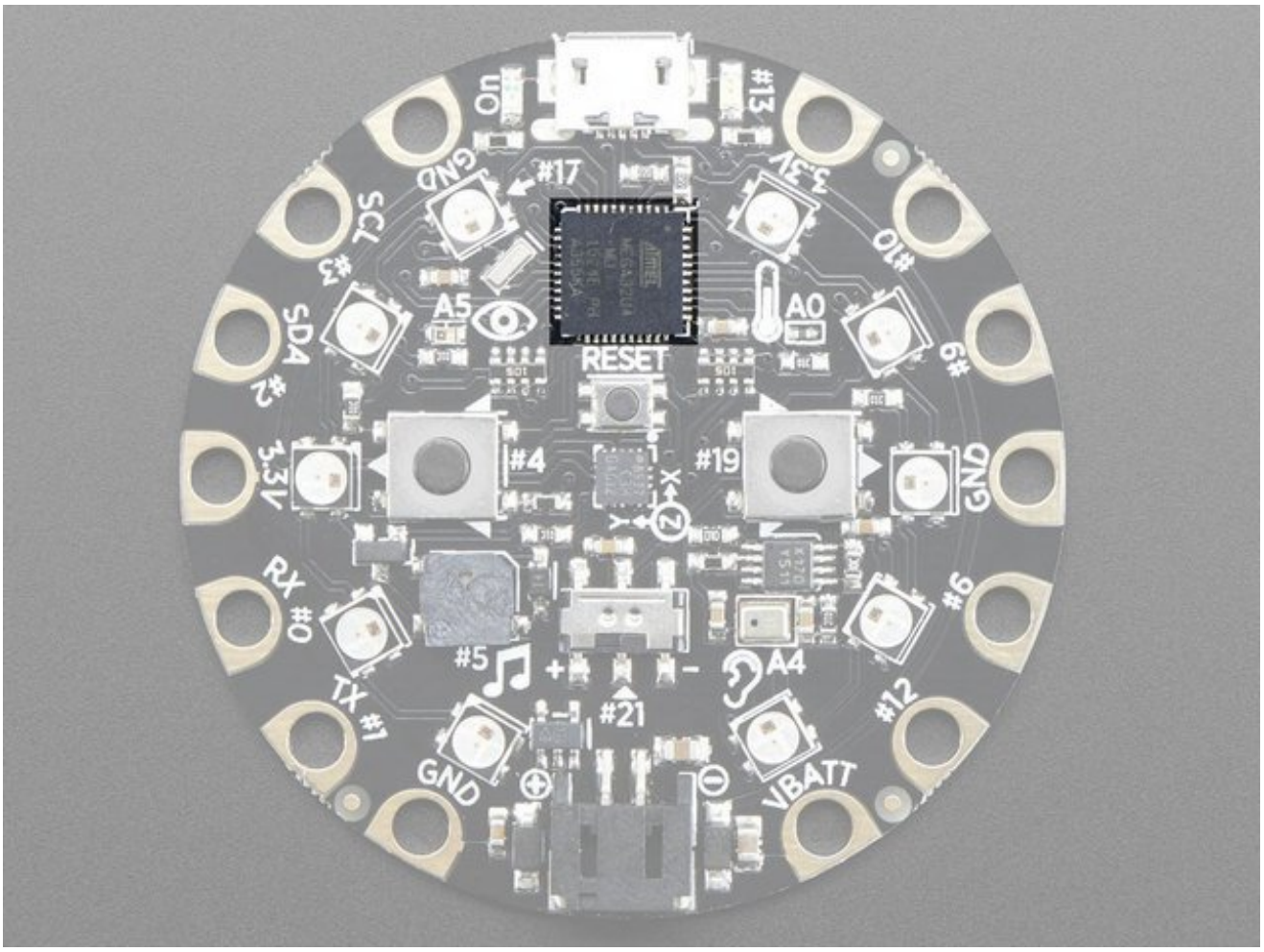

"ON" LED
Micro-B USB Jack
Pin 13 LED
Connection Pads (14 total)
ATmega32u4 Microcontroller
Light Sensor
Reset Button
Temp. Sensor
Left Button
Accelerometer
Right Button
Speaker
Slide Switch
Microphone
NeoPixels (10 total)
JST Battery Connector
On
#13
GND
#17
3.3V
SCL
#3
#10
A5
A0
SDA
#2
#9
3.3V
#1
#19
GND
RX
#0
#6
TX
#1
#5
A4
#21
#12
GND
VBATT

Microcontroller

Don't feel like you have to understand this part fully! Skim it for now, and consider it a resource for you when you want to take a deeper dive into understanding the hardware!

Main Chip / Microcontroller

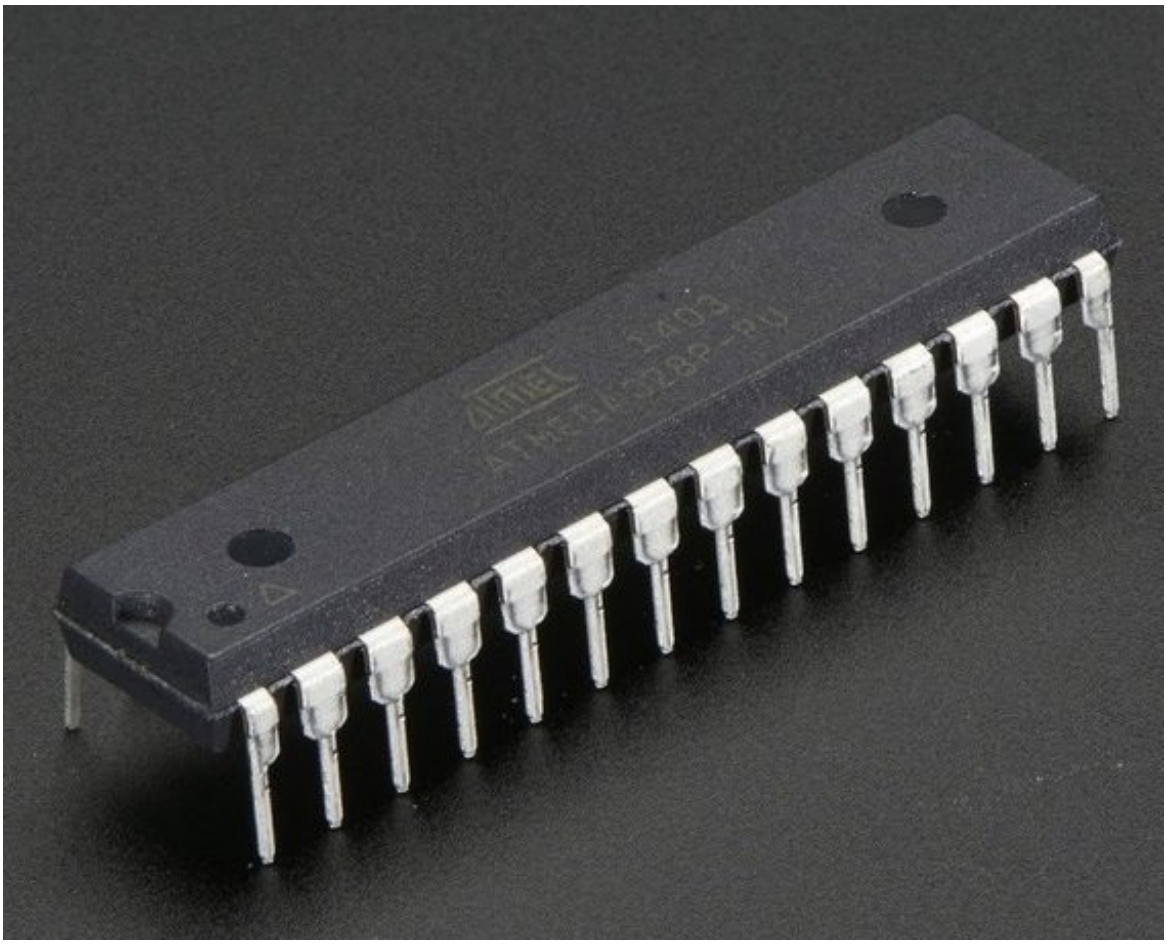
This is the 'brains' of the Circuit Playground. The thing that you program when you program! It's what runs the code, the **CPU** (Central Processing Unit). Kinda like the processor that runs in your computer but much much much simpler and smaller.



The chip has very thin little *legs* that go around it, you can see the little silver dots that connect each leg of the microcontroller to the circuit board. Those dots both mechanically *and* electrically connect the CPU - so it can send data to and from your computer and those

sensors and lights.

Each one of those mini legs are called*pins* because older microcontrollers used to look like this:



Where each of the legs really do look like sharp pins. Those older microcontrollers are really big, though. And so over time the 'controllers got hit with a shrink ray until they looked like this:

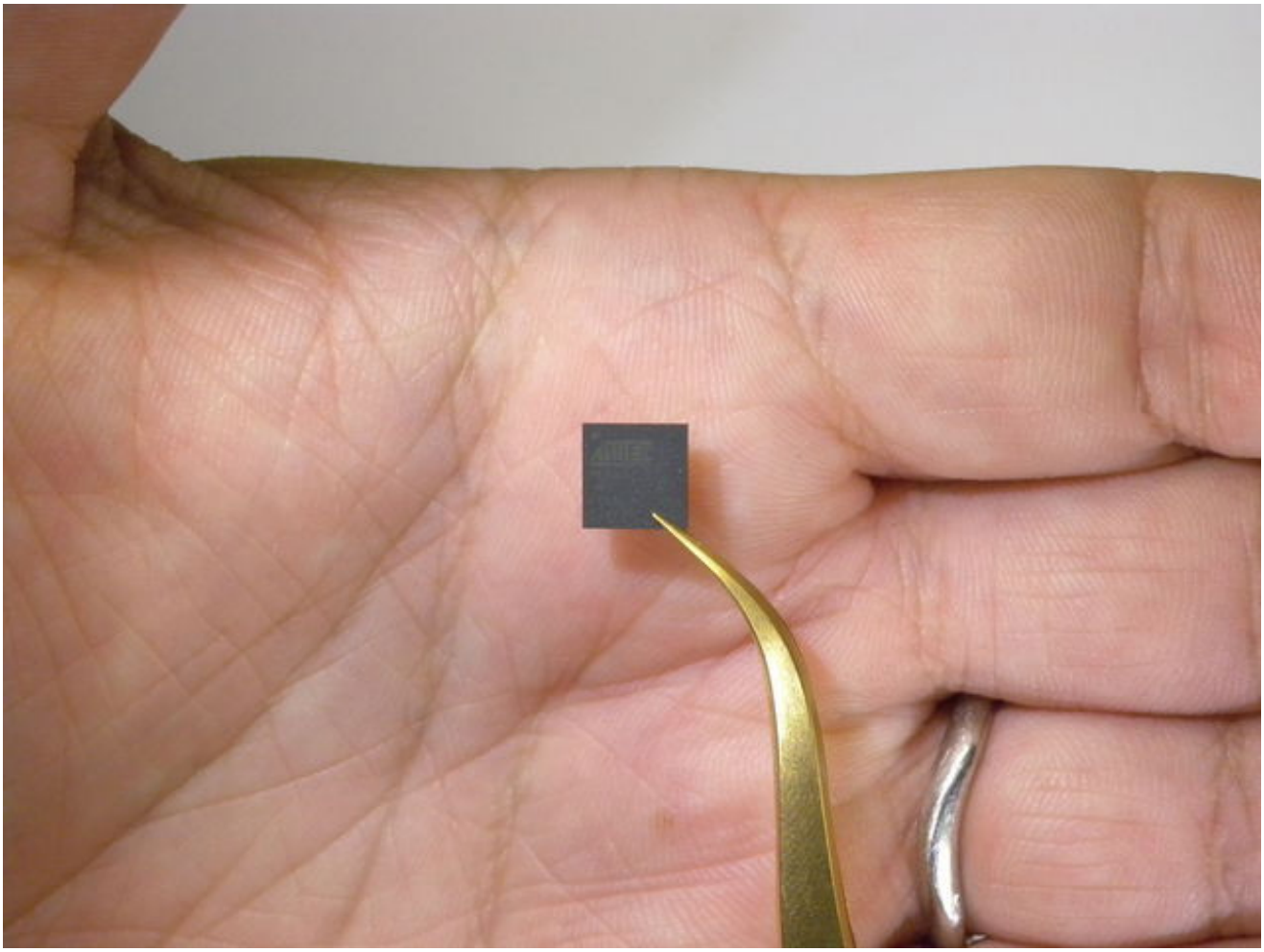


Image by Osamu (http://adafru.it/oVe)

Yep that little square has all the circuitry of the bigger chip, but so small you have to use tweezers to pick it up!

Our Pal, the ATMEGA32u4

The Circuit Playground **microcontroller** is called the ATmega32u4 and it has:

- 44 Pins
- Powered by 3 Volts
- Requires about 0.1 Watts of power
- Runs at 8 MHz
- 32 KB of flash storage
- ~2 KB of RAM
- Costs about $5 per

Compare this to, at the time of writing, a common**computer** chip, the Intel i5-6400

- 1151 Pins
- Powered by 1.35 Volts
- Requires about 35 Watts of power
- Runs at 2800 MHz
- No internal Flash storage, but most computers have at least 250 GB = 250,000 MB = 250,000,000 KB* of storage
- No internal RAM but most computers have at least 4 GB = 4,000 MB = 4,000,000* KB of RAM
- Costs about $200

(yes I know its not exactly 1000)*

So, clearly if you want an ultra powerful computer processor that can play the latest games, an i5 is the way to go. But its expensive, and requires a ton of power, and you need to have a full motherboard to run it so it's kinda big. If you just want to do some simple tasks, a microcontroller like the '32u4 is peachy. Also, its quite handy that it has the RAM and storage inside of it - its not a lot but that means you don't need to hook up a hard drive to this chip, its very compact and complete.

Simplicity & Sturdiness

What's cool about the microcontroller is that unlike your computer which requires an operating system (Mac OS X or Windows) and booting up, the microcontroller is 'barebones'. When you plug it in, it immediately runs whatever code you asked it to do.

And, you don't have to worry about a diskette or hard drive or cd-rom getting scratched or damaged. The storage inside of the chip lasts for a *really long time.* You could program your Arduino, leave it alone for decades, even hundreds of years & come back and power it up with a post-apocalyptic-cyber-battery and it would work just as new.

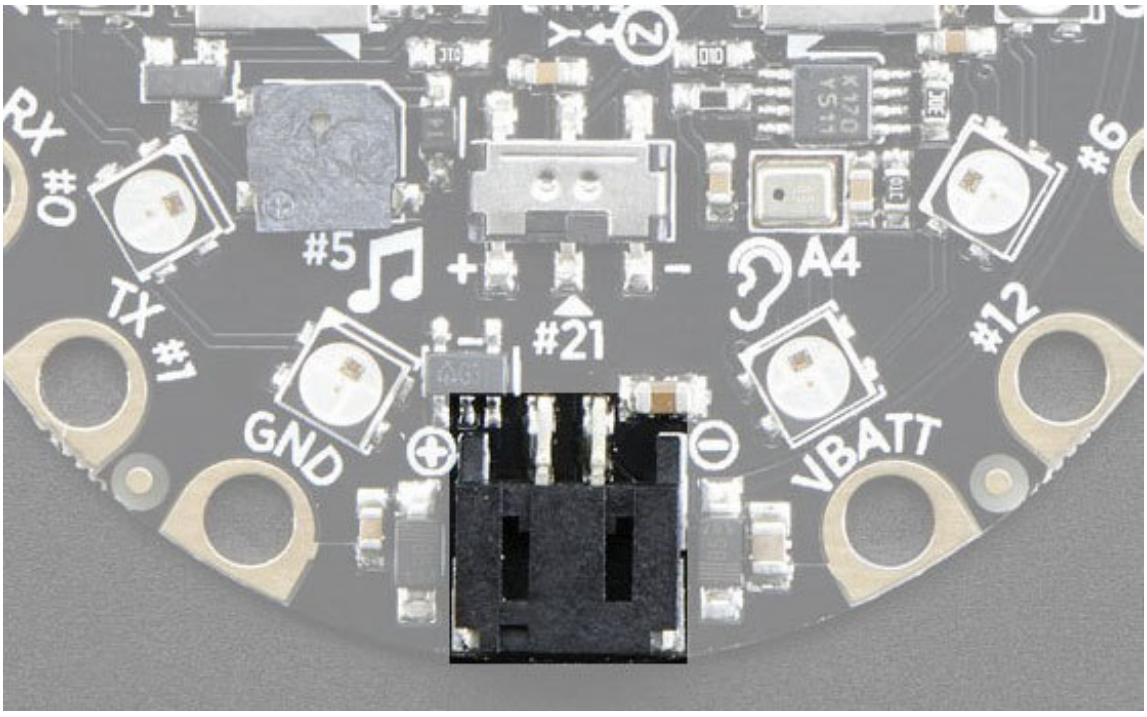
Battery Jack & Supply

Don't feel like you have to understand this part fully! Skim it for now, and consider it a resource for you when you want to take a deeper dive into understanding the hardware!

Battery Power Jack

There are two ways to power your Circuit Playground: you can use the USB connector to connect to a computer or portable USB power pack *or* you can plug in a battery pack. USB can be used to power and program. The battery connection can only be used for power - but it's great for when you want to take your Circuit Playground out into the world

This is the Battery Power Jack:

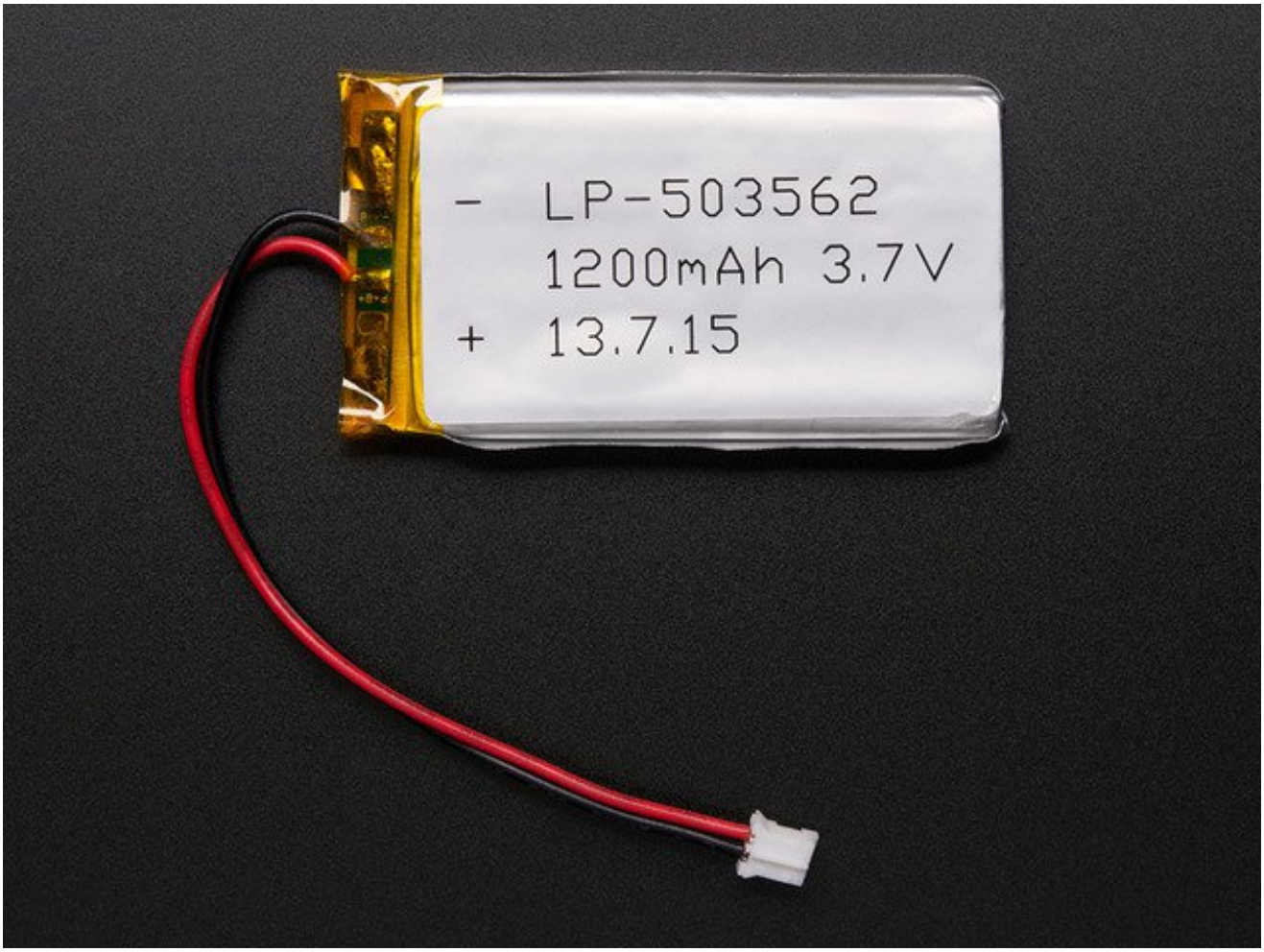


The technical specifications for the jack is:

JST PH type 2-pin connector Pin #1 Power and Pin #2 Ground

You can use this with any battery pack that can provide at least 3.5V DC and up to 6.5V DC. In particular it works really well with 3 x 1.5V batteries, like this 3 x AAA battery holder

Advanced users can connect up a single-cell rechargeable lithium ion or lithium polymer battery

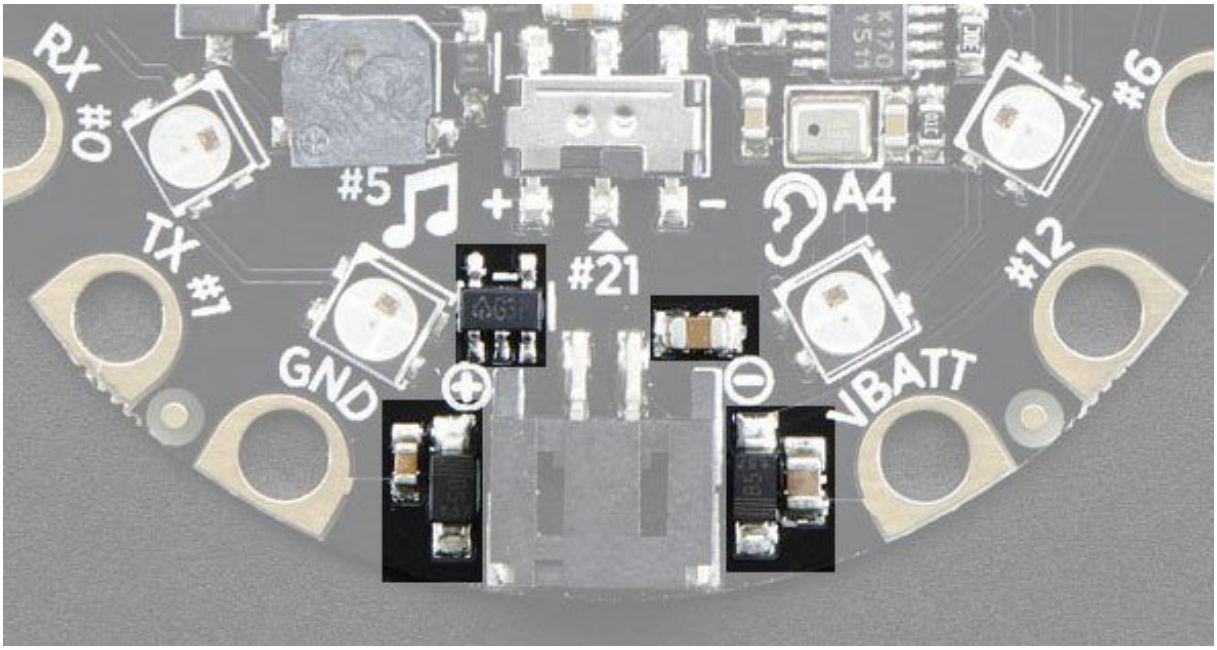


Lithium Ion or Polymer batteries are more difficult to use and require much more care than rechargeable AAA's so we don't recommend them.

Circuit Playground does not have battery charging built in so no matter what rechargeable battery you have, you will need a separate charger as well.

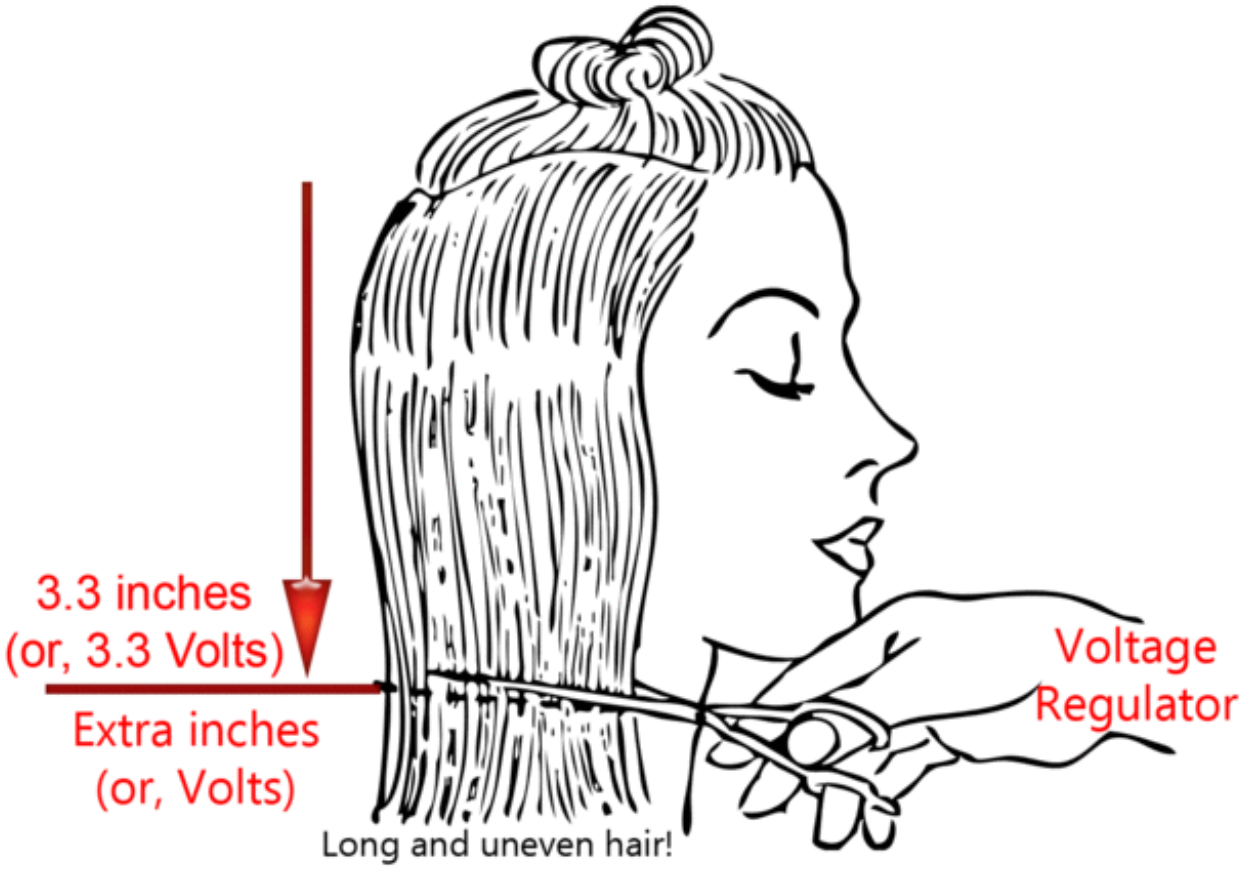
Onboard Power Supply

Circuit Playground is designed for beginners so it has some protection and *regulation* circuitry so that it is flexible about how it is powered. In particular there is a polarity protection diode (to avoid destroying the board if you have a backwards-connected battery). It also has an **onboard 3.3V Power Supply:**



This means you can power it from anywhere between 3.5V up to 6.5V and it will automatically *regulate* it down to a clean 3.3V

Think of it like a barber. When you go to the hair salon, you have long uneven hair (just like the large, uneven power that is fed into the Circuit Playground from a battery pack). The stylist takes out the scissors and says **OK how long do you want your hair?** and you reply **3.3 inches long!**



>snip snip< and your hair is cut straight off, leaving a very clean line.

Just make sure all of your ~~hair~~, err... **battery voltage** is *at least 3.5 Volts* because the regulator needs some extra length to work with.

The Circuit Playground will work off of battery or USB, and can have both plugged in at once - it will automatically switch to whichever is the higher voltage

Output Current

When powering Circuit Playground, you can draw at *most 5*00 milliAmps of current. This is not a guarantee because you also have to make sure that the regulator doesn't overheat. If you're using some other voltage input, the max current you can pull continously is *approximately*

250 milliWatt / (Input Voltage - 0.3 - 3.3 V) = in milliAmps

So for 5V USB in, the max for *continuous* current is 250/(5-0.3-3.3) = 178 milliAmps

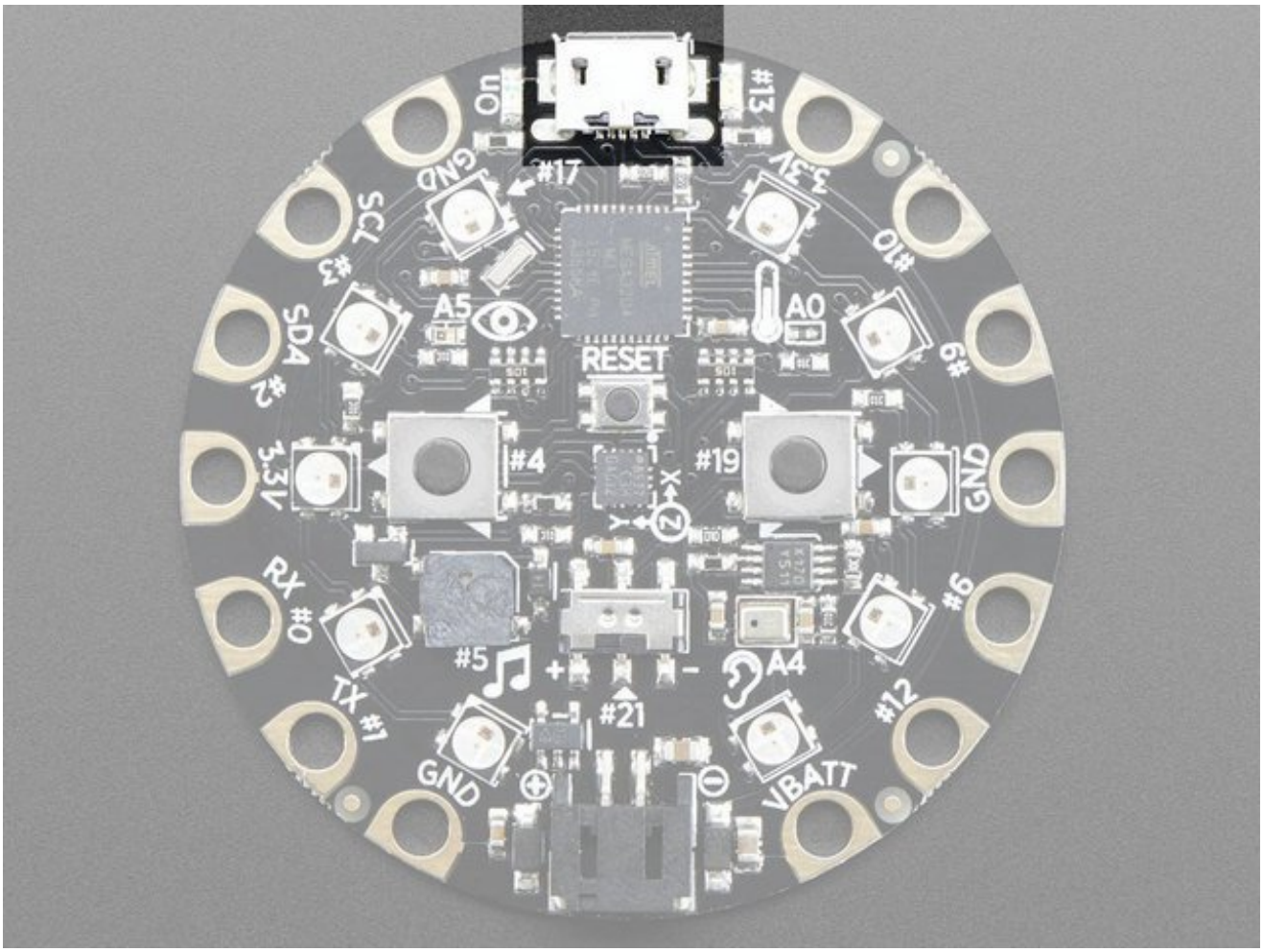
This is just a rough estimate and depends on if the power usage is continuous or just once

in a while.

USB Connection

Don't feel like you have to understand this part fully! Skim it for now, and consider it a resource for you when you want to take a deeper dive into understanding the hardware!

The USB Jack is next up in our tour



USB Jack

As we talked about in the beginning, this is how you connect your Arduino to your computer. You can use any computer with a USB port. You will need a cable to connect! This cable is usually not included.

Make sure you have a Micro USB cable

A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times students have nearly given up due to a flakey USB cable!

I can't stress it enough. Make sure you have a good USB cable. Naughty USB cables will really ruin your day, like a stone in your shoe. Just thow out bad cables and replace with a good one - they are designed to be disposable!

USB Inteface Chip??

If you're coming from any background with an Arduino you're used to having a secondary chip on the circuit board that acts as the **USB to Serial Interface Translator chip**. Circuit Playground does not have one of these! That's because the ATmega32u4 processor does the USB for you *natively* without the need for a second chip. There's a few tradeoffs, in

particular you may need to reset the chip if the USB crashes but you'll learn how to do that

Powering over USB

For permanent installation, you can power your Circuit Playground using a USB wall adapter - these come with almost every phone and gadget these days. They have a nice clean 5V output. Some have an output up to 5.5V but thats OK. Basically, if it has a USB connection it will power your 'play just fine!

You can *sometimes* power an Circuit Playground project off of a portable USB power pack but these packs are often designed to charge a phone and the 'Play uses so little power that it will cause the pack to think that it is "done charging" and auto-shutoff

So, try it out but your mileage may vary.

Basic LEDs

Don't feel like you have to understand this part fully! Skim it for now, and consider it a resource for you when you want to take a deeper dive into understanding the hardware!

Your Circuit Playground has some lights that it can use to give you an idea of what it is up to.

These lights, called **LEDs** (pronounced Ell Eee Dee), are on just about every electronic device you own. Often times they're used to let you know if something is on and if there's an error. For example, here's a cable modem with multiple LEDs.

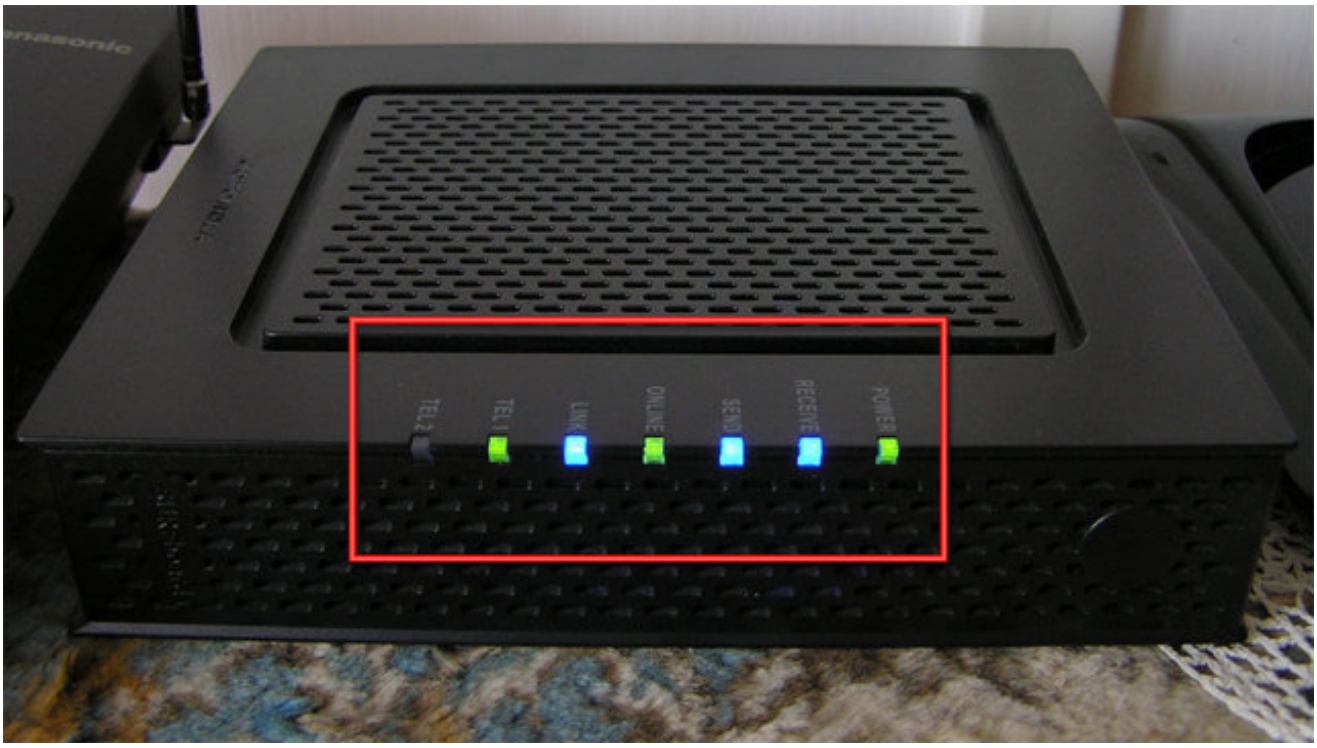
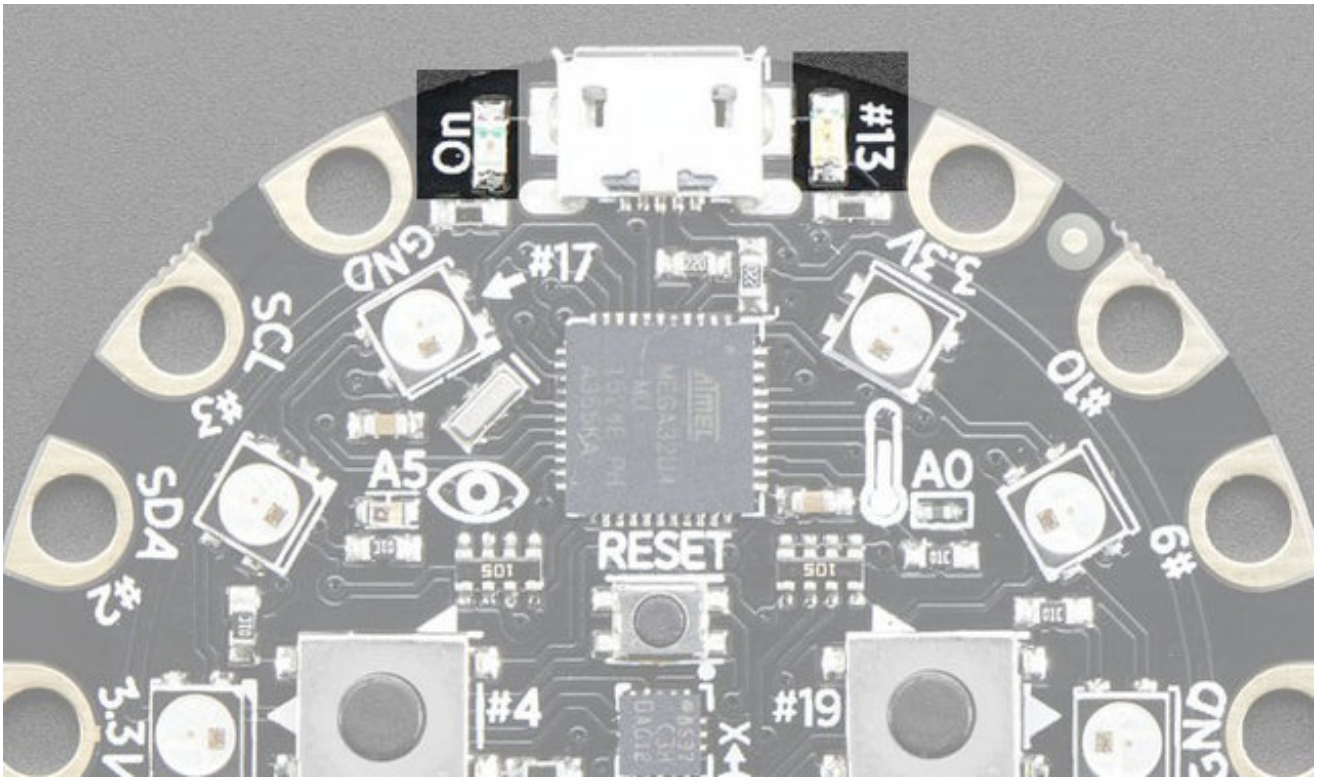


Image from Wikipedia (http://adafru.it/oBB)

Each LED indicates the status of the modem. For example, on this device the one to the right is the **POWER** LED, its lit if there's good power. In the middle is the**ONLINE** LED, which lets you know if the modem was able to connect to the Internet.

Circuit Playground Basic LEDs

Likewise, the Circuit Playground has **two** basic (single-color) LEDs: **ON** and **#13**



They sit to the left and right of the Micro USB jack

ON LED - this LED will shine green whenever the Circuit Playground is powered. Always check this LED if your board is not acting right, if its flickering or off then you should check your power supply

#13 LED - this is the one LED that you can control. The ON lights up automatically no matter what. The **#13** LED, however, is connected to one of the main microcontroller's pins and you can turn it on or off when you start writing code. This LED also pulses to let you know when the *bootloader* is active and ready to receive a program

It's called lucky **#13** because it is connected to pin number 13 on the microcontroller!

Library Reference

You can turn the #13 LED on by calling the procedure`CircuitPlayground.redLED(true)` and turn it off with `CircuitPlayground.redLED(false)`

Reset Button & Bootloader

There's a special mini button labeled **RESET** right below the microcontroller. This is the, well, *reset* button. It will perform a clean restart of the microcontroller or let you start the bootloader by hand.

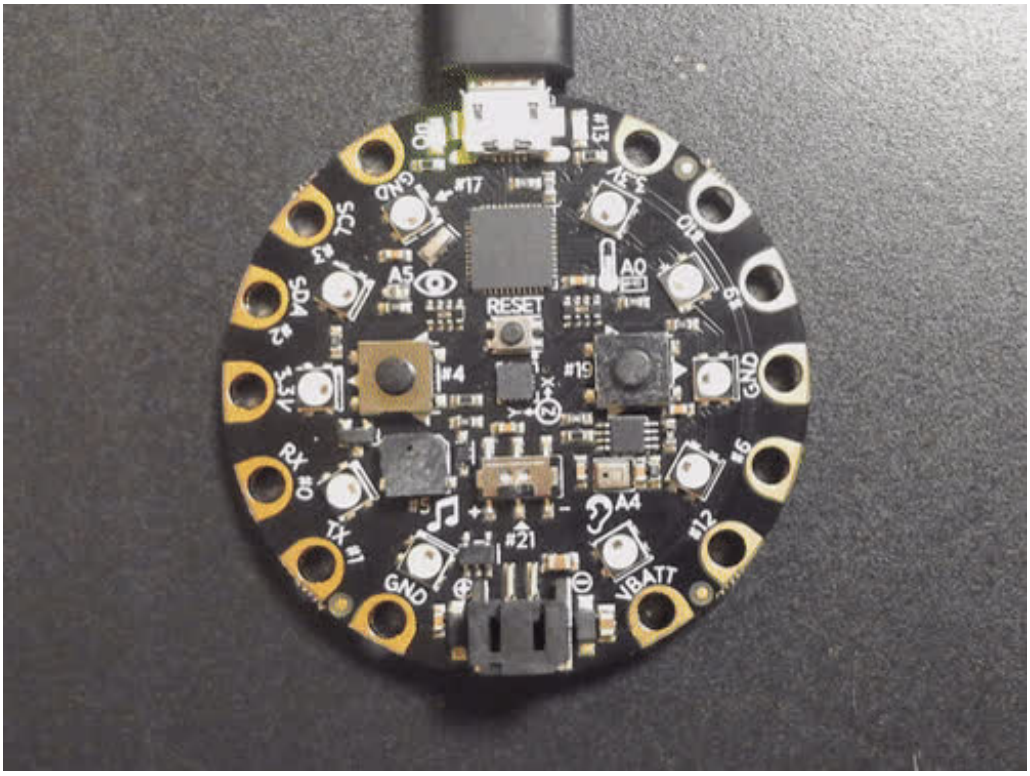
Re-starting/Resetting

To re-start your Circuit Playground (basically, same as turning it off and back on again) - **click the reset button once**. The Circuit Playground will restart itself.

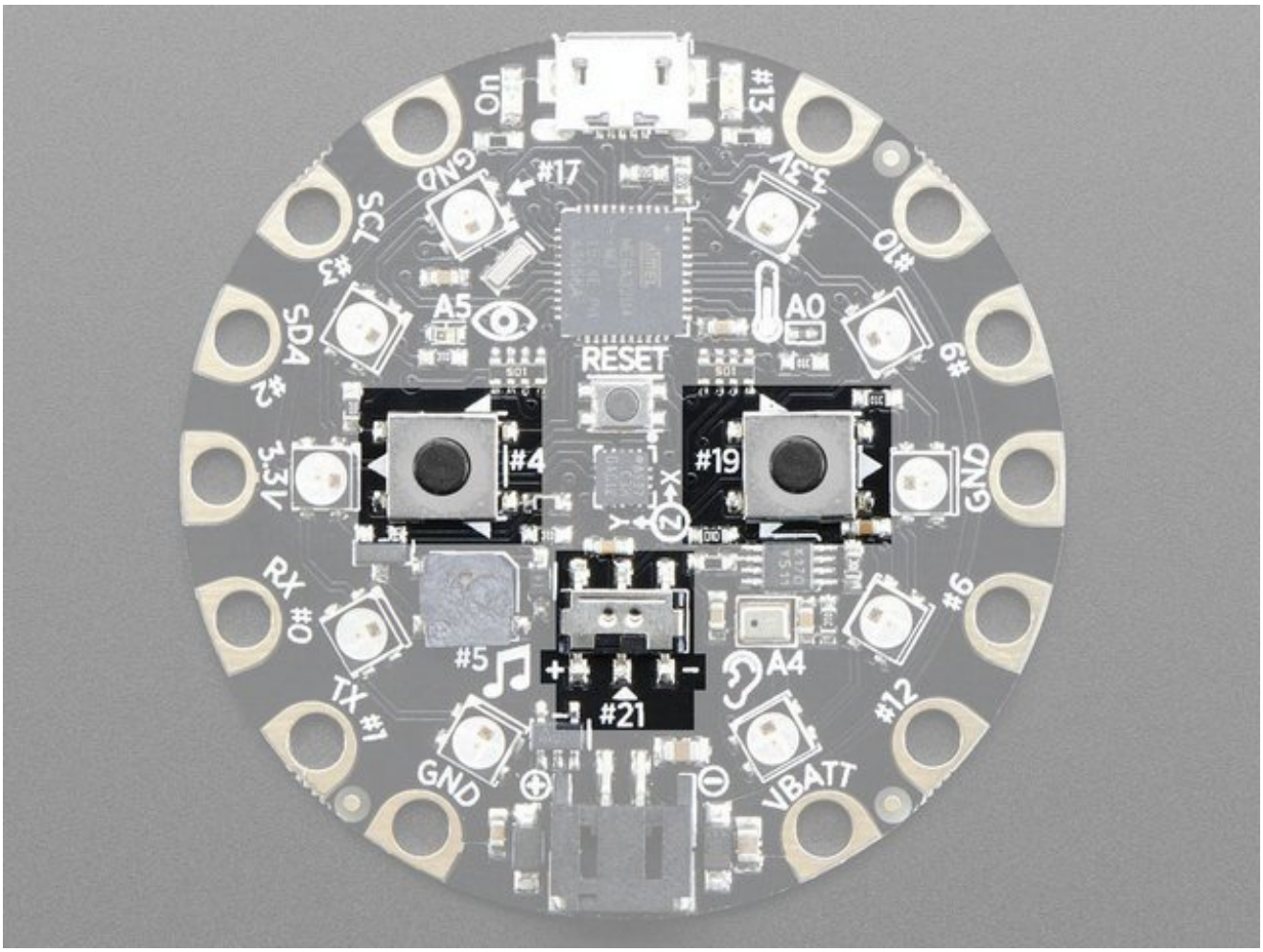
Bootloader Mode

To start the Circuit Playground bootloader - **click the reset button twice**. The bootloader will run for 10 seconds during which time you can upload new code. You shouldn't have to do this unless something got a little confused in the 'Play, but its an essential skill! During bootloader mode, the red #13 LED will breate on and off once a second.

It's good to practice both resetting (single click) and bootloading (double click).Check the Power Up test page for more details. (http://adafru.it/pdH)

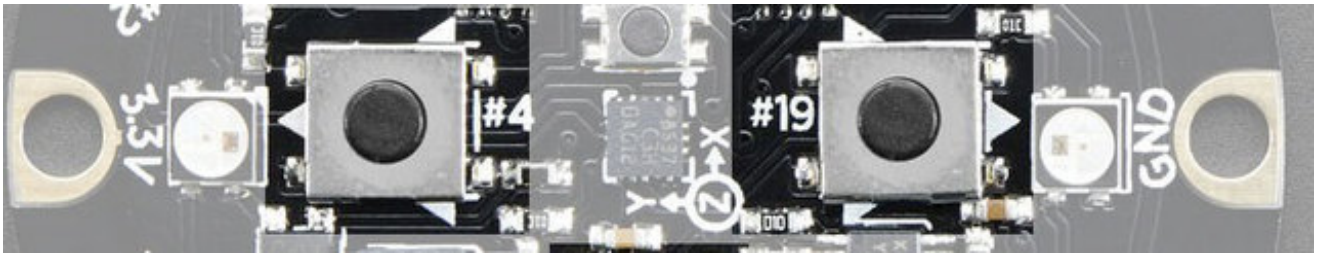

SCL
SDA
3V
RX
TX
GND
RESET
A5
A0
A4
VBATT
GND
3V
#13
#17
#4
#19
#21
#12

Buttons & Slide Switch



Now we're starting to get to the sensors built into the Circuit Playground. These are parts that are normally *not* included on an Arduino-like board. They are yours to play with and use as you travel on the path of learning!

Two Pushbuttons

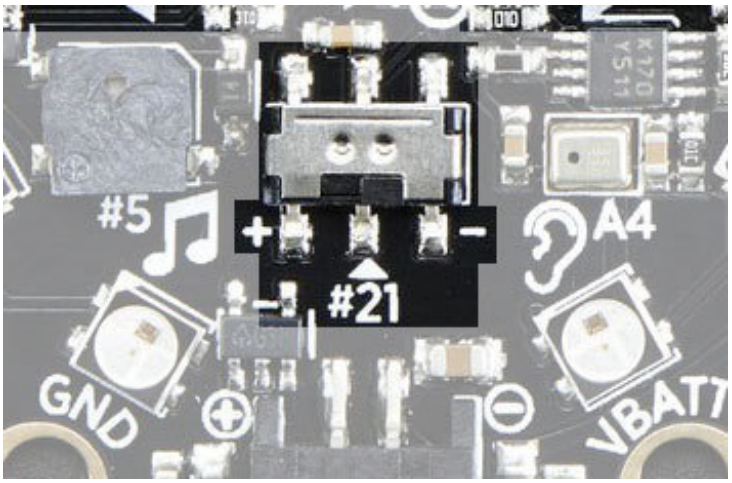


There are two momentary pushbuttons, you can use these in any of your projects to change modes, indicate when to start or stop a sketch, modify behavior, really anything!

- The **Left** button is connected to digital pin **#4**
- The **Right** button is connected to digital pin **#19**

Both have 10KΩ *pull-down* resistors. That means that when the button is *pressed* the value read off those pins will be 'high'. You'll learn more about that later, but its here for your reference

Slide Switch



The slide switch is near the center, right above the JST Battery Jack. There's a little nub you can *slide* left or right. This switch is connected to **digital pin #21**. When switched to the left, the value reading is 'high' (thus the + symbol) and when switched right, the value read is 'low' (- symbol).

Library Reference

slideSwitch

You can call CircuitPlayground.slideSwitch() to tell you which way the switch is flipped. It will return **true** if the switch is to the left (+ side) and **false** if the switch is to the right (- side)

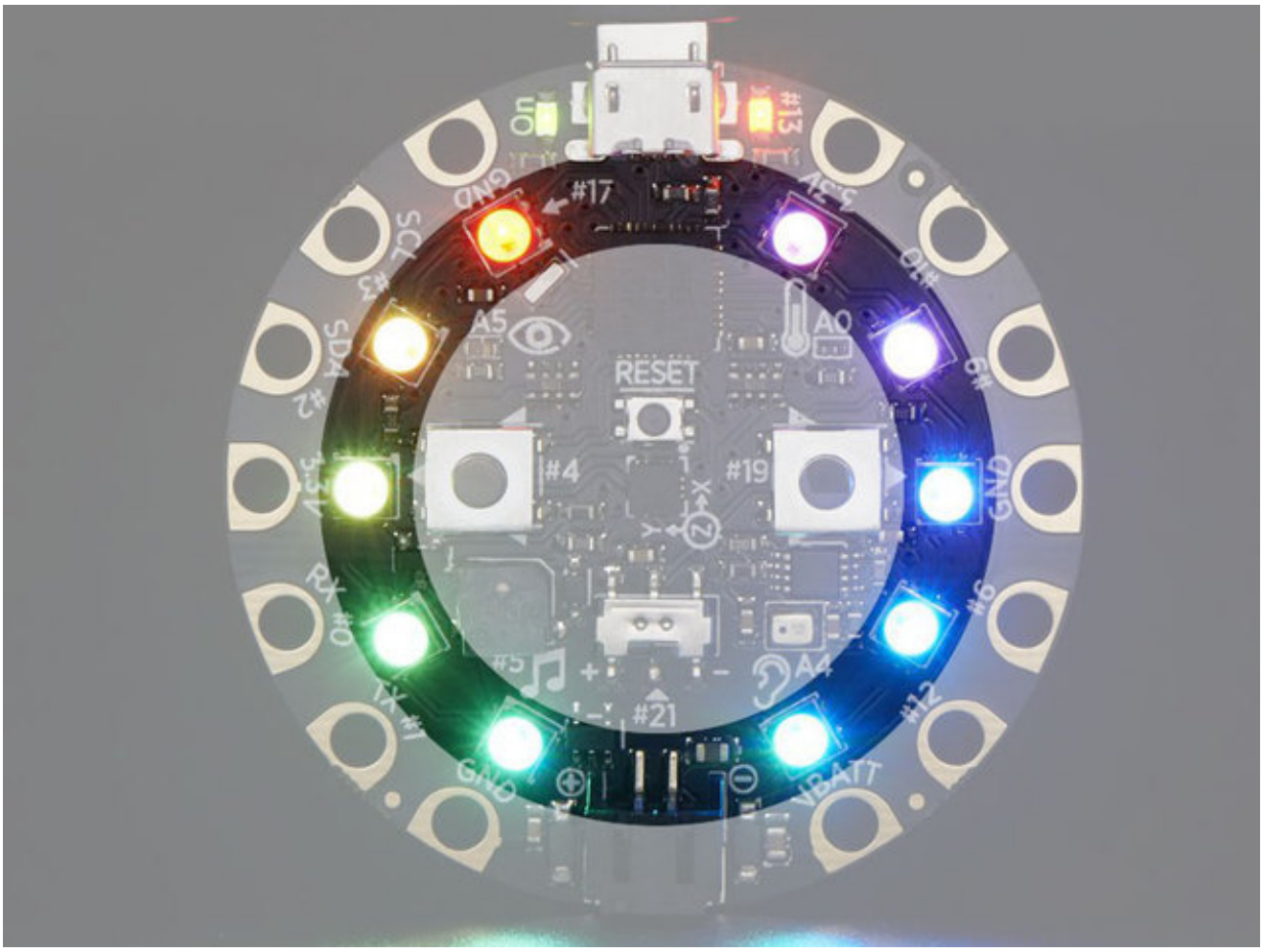
leftButton and rightButton

You can ask if the buttons are pressed by calling CircuitPlayground.leftButton() or CircuitPlayground.rightButton(). They will return **true** if and only if the button in question is being

pressed right when the function is called.

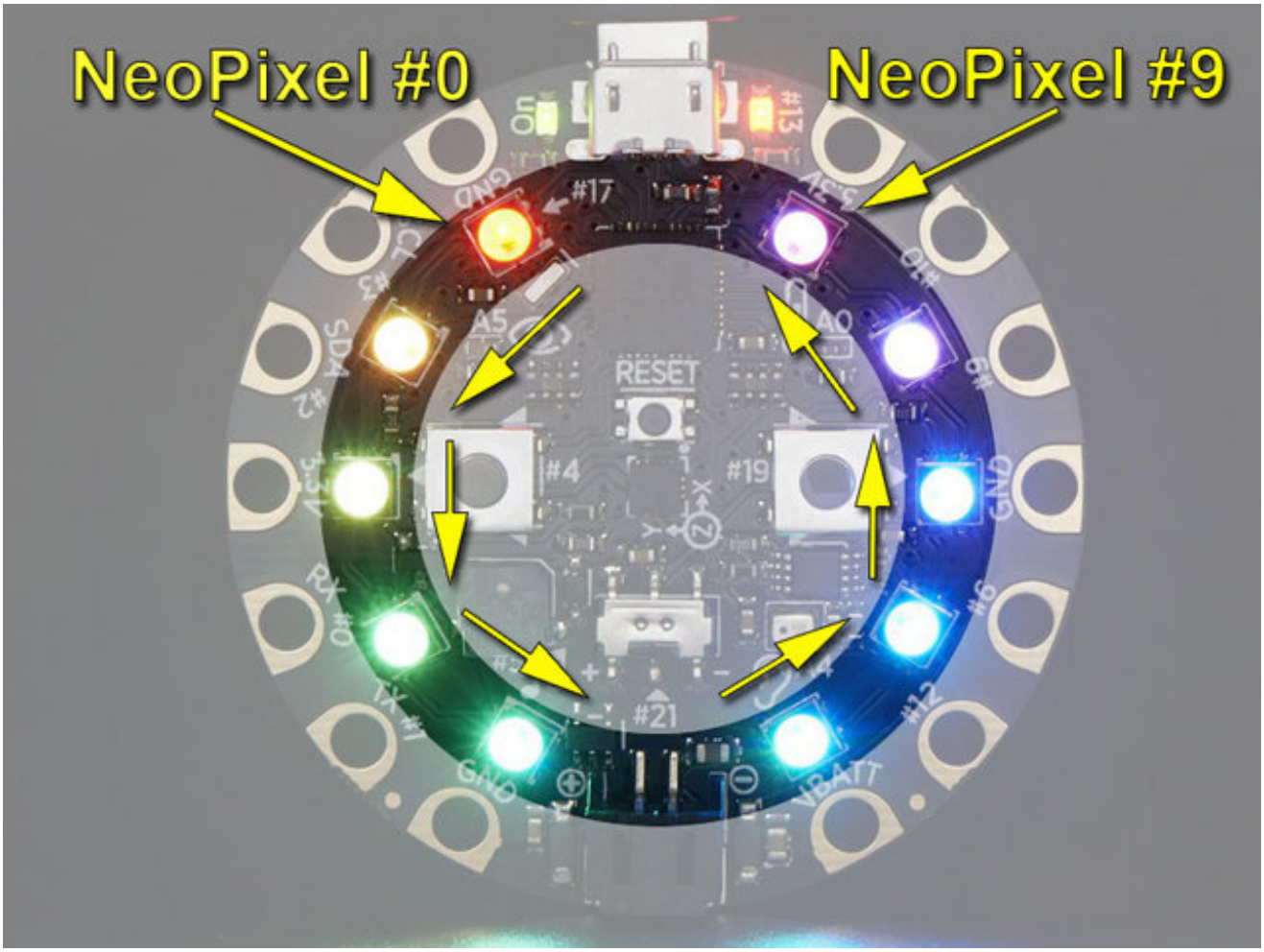
NeoPixels

One of the fanciest accessories on Circuit Playground are ten (10!) full color LEDs called NeoPixels. Each of these ten Pixels contain bright red+green+blue sub-LEDs and can display any of 16,777,216 different colors!



These pixels are controlled by a single data pin,**#17**. Unlike non-smart RGB LEDs, you can set the color and a little chip inside the LED will handle the PWM for you.

Since they are controlled in a*chain*, you will need to tell the Circuit Playground's NeoPixel code which one you want to set the color for. Each NeoPixel is numbered, starting with #0 at the top left and going counter clockwise till you reach #9



Library Reference

setPixelColor

You can set the color for each pixel with the built in NeoPixel library:

CircuitPlayground.setPixelColor(*n*, *red*, *green*, *blue*)

Where *n* is between 0 and 9 and indicates which LED you are setting.*red/green/blue* vary between 0 (off) and 255 (full on)

clearPixels

You can quickly turn *all* the LEDs off with CircuitPlayground.clearPixels()

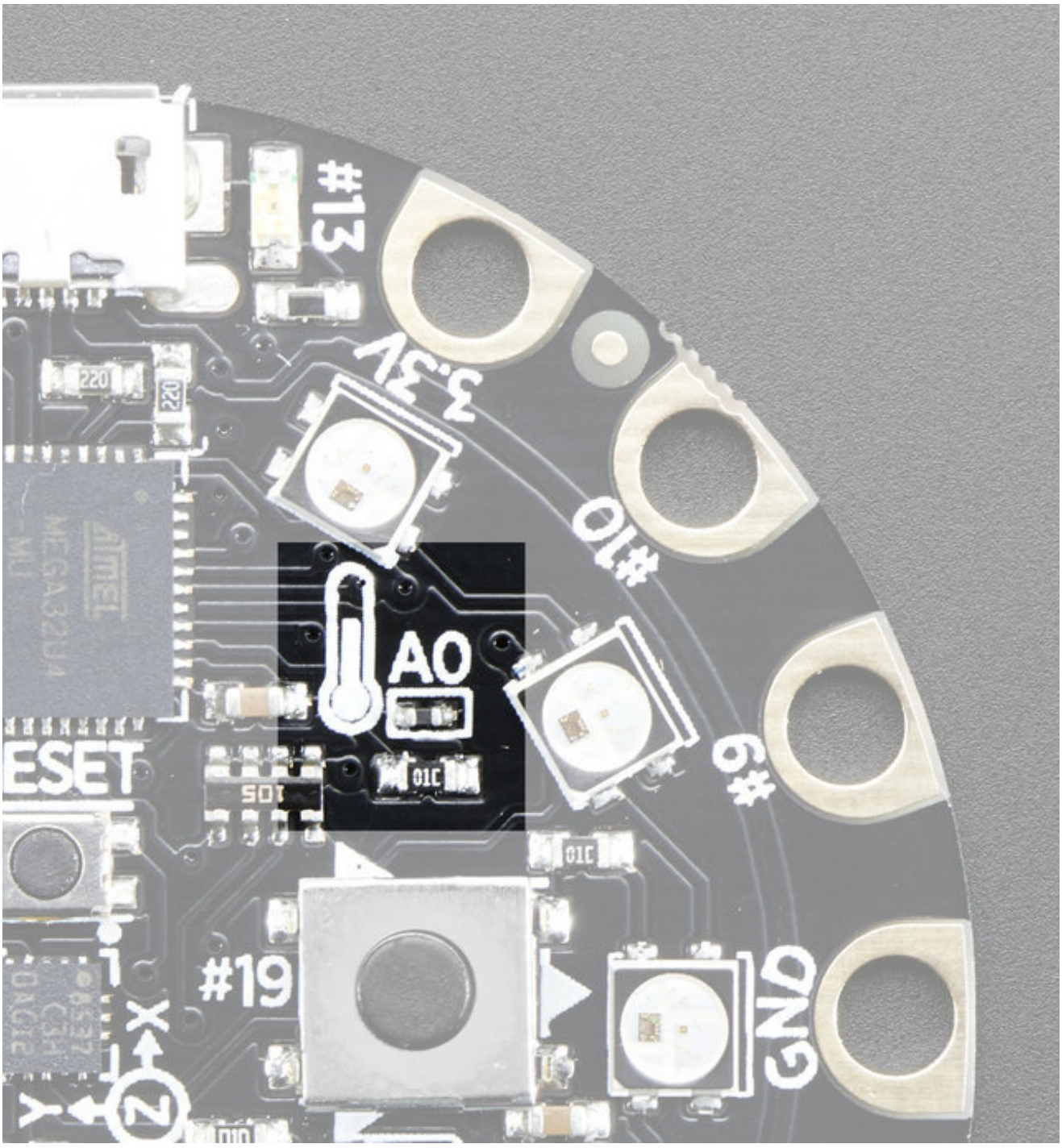
setBrightness

By default the *global* brightness of the LEDs is set to 30 out of 255. This means that the LEDs won't be *insanely* bright when you first use them.

You can change the global brightness at the beginning of the sketch (in setup) by running CircuitPlayground.setBrightness(*b*) where *b* varies from 0 (no LEDs will ever light) to 255 (incredibly bright!)

setBrightness only affects pixel colors set after it is called. That is, if you set an LED color and then call setBrightness, it wont affect that older LED color. Only going forward will the brightness change to be different.

Temperature Sensor



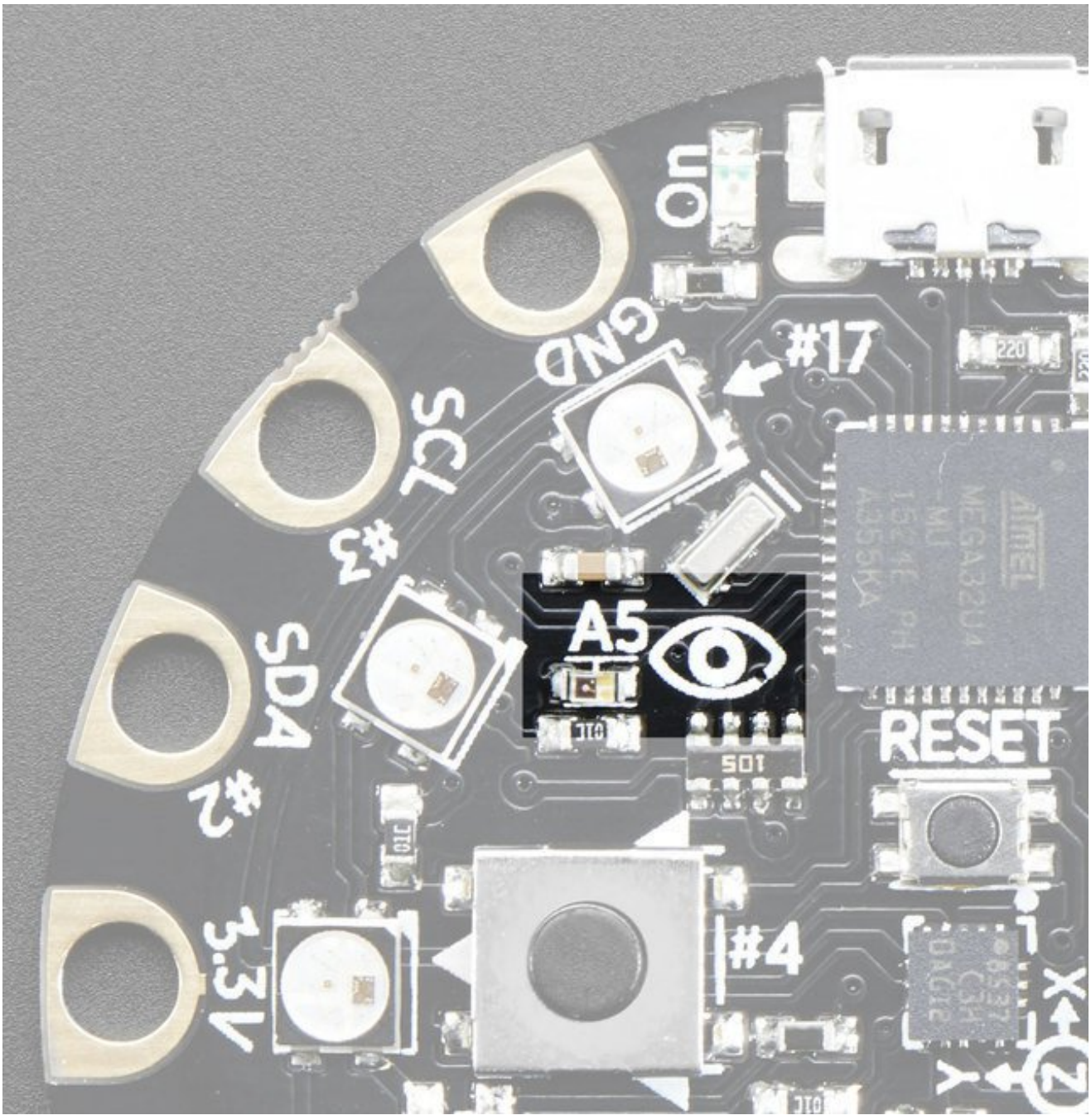
You can sense the temperature using the onboard thermistor. A thermistor is a resistor that changes it's resistance based on the temperature. They are a *little* more complex to use than temperature sensors that spit out Celsius, but they're also much less expensive.

We use the NTC thermistor (Murata NCP15XH103F03RC (http://adafru.it/p3a)) for temperature sensing. While it isn't an all-in-one temperature sensor, with linear output, it's easy to calculate the temperature based on the analog voltage on analog pin **#A0**. There's a 10K 1% resistor connected to it as a pull down. You can read the analog value and calculate the temperature with the beta-constant (http://adafru.it/p3b), or just use the Circuit Playground library

Library Reference

You can retreive the current calculated temperature with`CircuitPlayground.temperature()` which will return you a floating point number in Centigrade. If you need Fahrenheit use `CircuitPlayground.temperatureF()`

Light Sensor



There is an analog light sensor, part number ALS-PT19 (http://adafru.it/n8F), in the top left part of the board. This can be used to detect ambient light, with similar spectral response to the human eye. There is a 10K balancing resistor to convert the current drive to voltage.

This sensor is connect to analog pin **#A5**, you can use analogRead or the library to read the

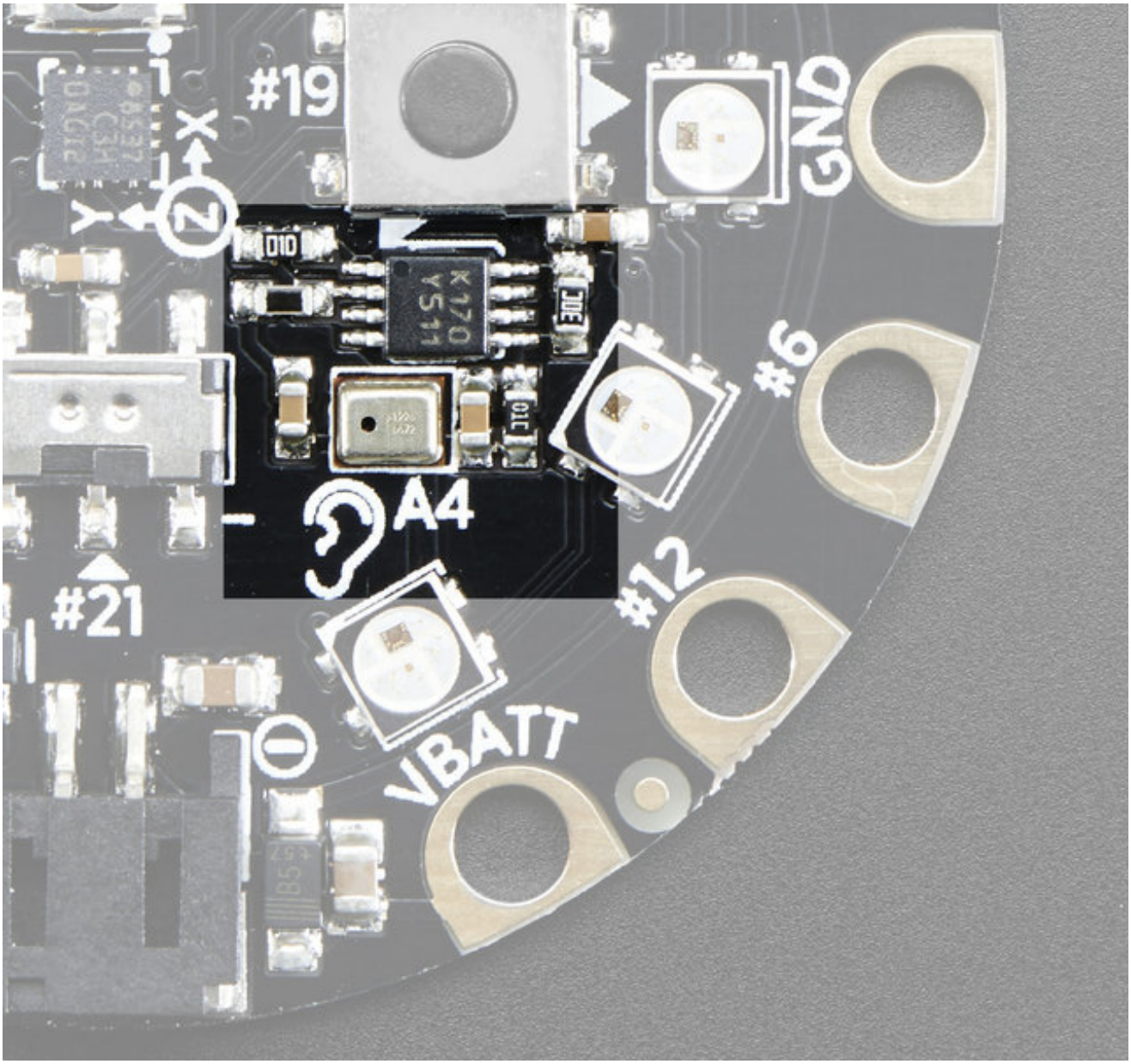
analog value. It will read between 0 and 1023 with higher values corresponding to brighter light levels. The range is approximately 0 Lux to 1500 Lux maximum.

1000 Lux will roughly read as 2 Volts (or about 680 as a raw analog reading). A reading of about 300 is common for most indoor light levels. Note that outdoor daylight is 10,000 Lux or even higher, so this sensor is best suited for indoor light levels!

Library Reference

You can request the raw analog reading (0 = no light, up to 1023 = ~1500 Lux) with CircuitPlayground.lightSensor()

Sound Sensor



A thin MEMS microphone (http://adafru.it/p3c) can be used to detect audio levels and even perform basic FFT functions. You can read the analog voltage corresponding to the audio on analog pin **#A4**. Note that this is the raw analog audio waveform! When it's silent there will be a reading of ~330 and when loud the audio will read between 0 and 800 or so. Averaging and smoothing must be done to convert this to sound-pressure-level.

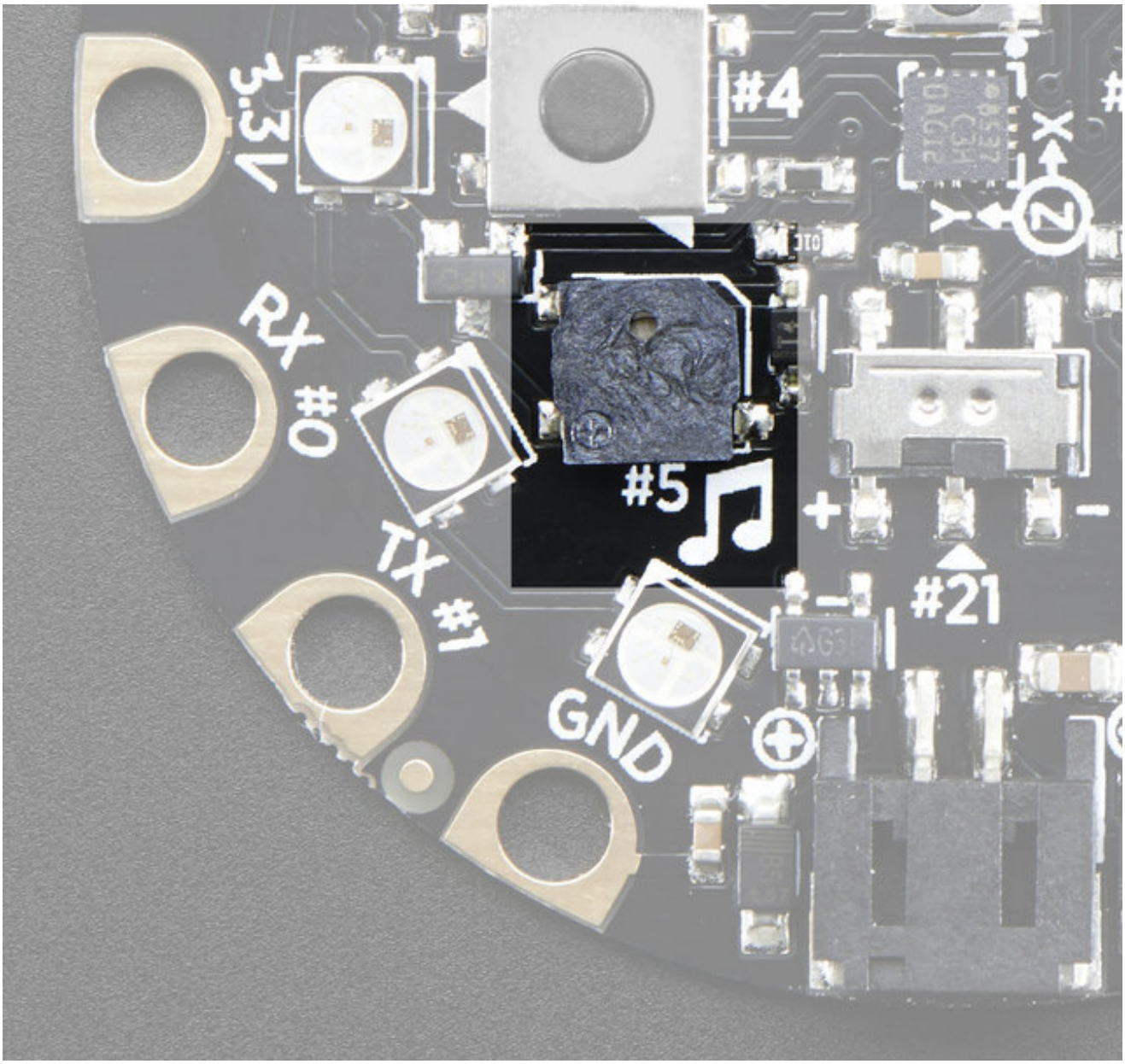
The microphone is sensitive to 100 Hz - 10,000 Hz audio frequencies.See the datasheet

for more details. (http://adafru.it/p3d)

Library Reference

You can read the raw analog value from the amplified microphone with `CircuitPlayground.soundSensor()` this will give you just a value from between 0 and 1023 where the default 'quiet' voltage is ~330. At this time there is no function for automatic sound pressure level or frequency binning.

Mini Speaker



You can make your circuit playground sing with the built in buzzer. This is a miniature magnetic speaker connected to digital pin **#5** with a transistor driver. It is quite small but can beep with conviction! It's not good for playing detailed audio, more for beeping and buzzing and simple bleepy tunes.

The driver circuitry is an on/off transistor driver, so you will only be able to play square waves. It is also not the same loudness over all frequencies but is designed to be the

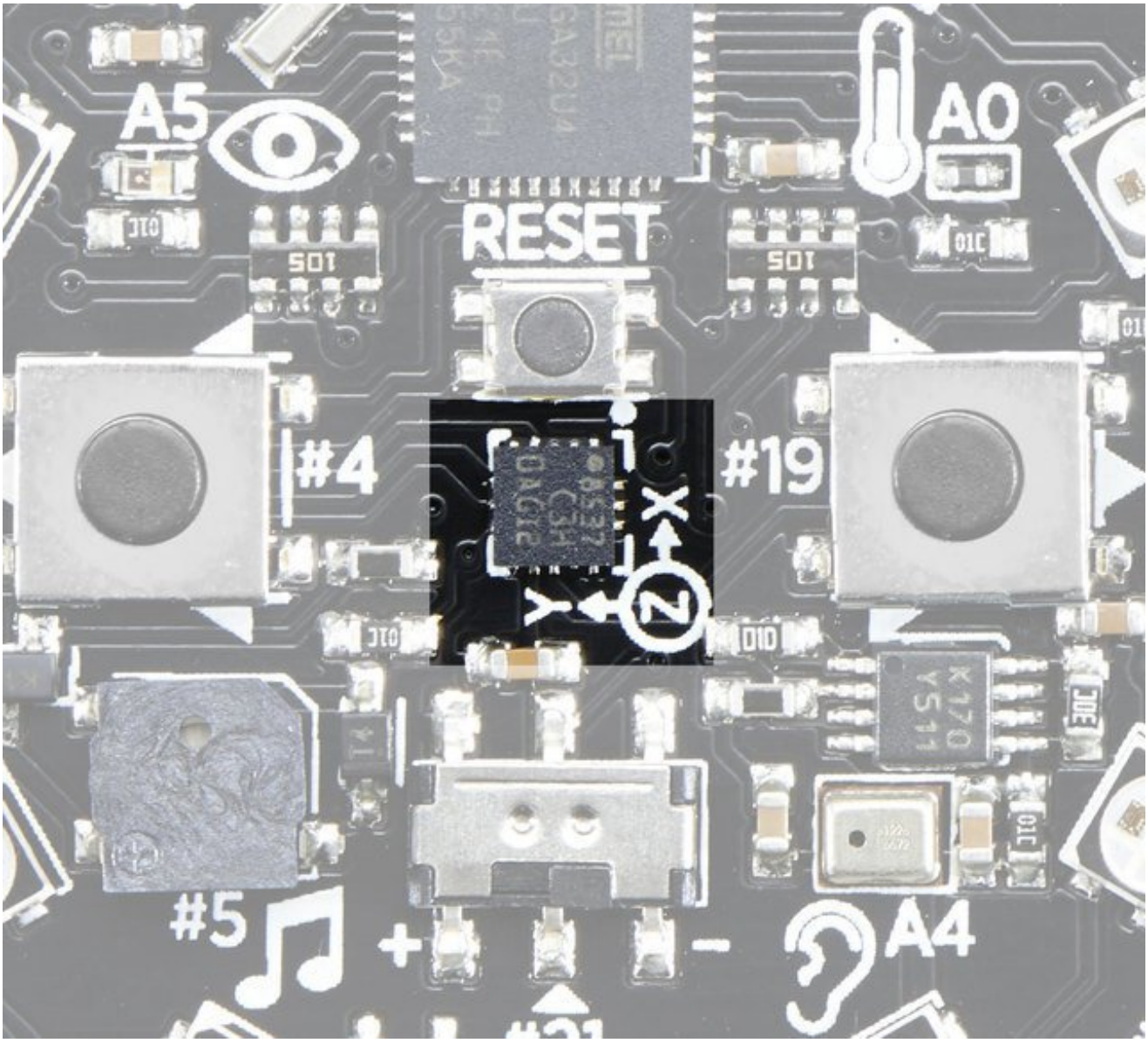
loudest at around 4 KHz. See the datasheet for some more details(http://adafru.it/p3e)

Library Reference

You can play basic square wave beeps/tones with`CircuitPlayground.playTone(`*frequency*, *duration_ms*`)`. Where *frequency* is the frequency in Hertz and *duration* is the time to play the beep for, in millisecods.

That means `CircuitPlayground.playTone(440, 500)` will play 440 Hz (middle A) for 500 milliseconds (half of a second). The call will take the amount of time that it takes to play the tone, so if you have a duration of, say, 5000 milliseconds your Circuit Playground will spend 5 seconds playing a tone and won't be doing anything else.

Accelerometer



There is a powerful MEMS accelerometer in the very center of your Circuit Playground. This part is the LIS3DH, a 3-axis (X, Y and Z) sensing accelerometer. Accelerometers are the sensors in your WiiMote, phone, and other electronic devices that can sense tilt, gravity, motion and 'tap' effects. These sensors used to cost $20 each but now are so common we can include them for beginners!

The sensor can sense ±2*g*, 4*g*, 8*g* (*g* = 9.8 meters/s^2). Of course, at all times on earth you will sense 1 *g* of gravity!

The LIS3DH is connected to the hardware SPI pins (to leave the I2C pins free) and has the chip select (CS) pin on digital pin **#8** and an optional interrupt output on digital pin**#7** (also known as IRQ #4)

Library Reference

Motion in X Y and Z

You can ask the sensor for the amount of*g* detected in X, Y and Z directions. Positive values mean acceleration in the direction of the arrow on the silkscreen. For X that means towards the USB jack. For Y that is to the left. For Z that is straight up pointing towards you when looking at the circuit board.

CircuitPlayground.motionX(), CircuitPlayground.motionY(), CircuitPlayground.motionZ() will each return a floating point value.

Setting Accelerometer Range

You can change the range, a smaller range (±2*g*) will give more precision. Larger ranges (±8*g*) can sense greater forces. You can set it whenever you like using CircuitPlayground.setAccelRange(range) where range can be LIS3DH_RANGE_2_G, LIS3DH_RANGE_4_G or LIS3DH_RANGE_8_G

Tap Detection

You can turn on tap detection by callingCircuitPlayground.setAccelTap(*taps*, *threshhold*)

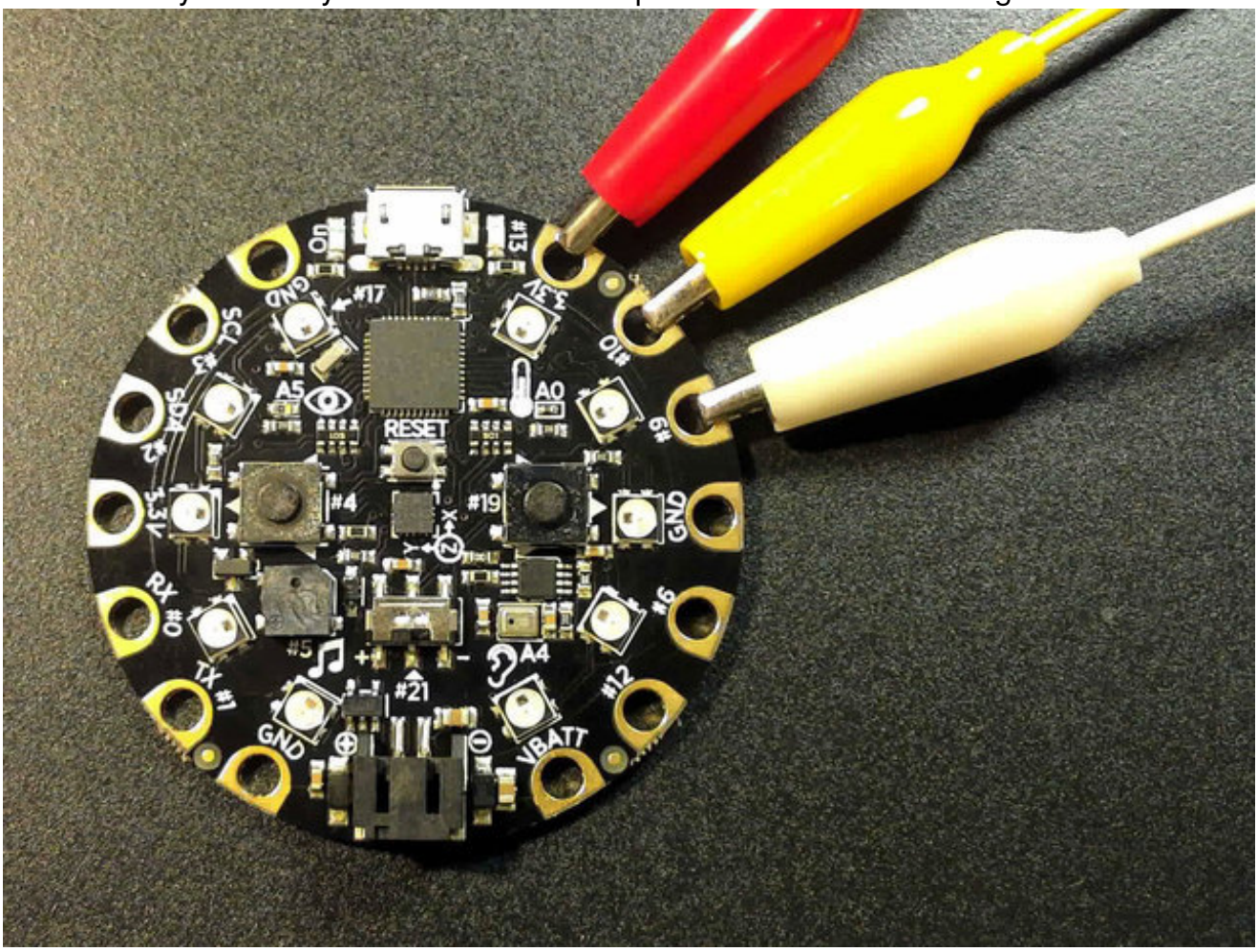
Tap detection can detect single taps or 'double taps' (like a double-click). If*taps* is 1 you will only detect single taps, one at a time. If *taps* is 2, you will be able to detect both single taps and double taps.

You can detect taps in real time by callingCircuitPlayground.getAccelTap(). It will return 0 if no tap is detected, 1 if a single tap is detected, and 2 or 3 if double tap is detected.

The problem with that is you need to query it within about 100 milliseconds of the tapping, which can be difficult to do. If you are comfortable with interrupts you can use the interrupt output to have a function called when a tap is detected: attachInterrupt(digitalPinToInterrupt(7), myfunction, RISING) will call *myfunction* whenever the tap or doubletap occurs.

Alligator Pads & Pinout

Don't feel like you have to understand this part fully! Skim it for now, and consider it a resource for you when you want to take a deeper dive into understanding the hardware!



There are 14 total Alligator Pads on the outside of the Circuit Playground. These are for you when you want to add more sensors or circuitry without the need for soldering. Use any kind of alligator clip (the small ones work best but any size should be able to grip onto the pads)

- 

You can pick up a pack of 12 clips from Adafruit (http://adafru.it/1008)

Capacitive Touch

All 8 non-power pads (e.g. not the GND/3.3V/VBATT) around the Circuit Playground have the ability to act as capacitive touch pads. Each pad has a 1 MΩ resistor between it and digital pin #30. You can toggle this pin in your sketch to control whether the resistor is a pullup or pulldown or floating. Note that this means that all the pads have a 2 MΩ resistance between them, not important for 99% of uses but may be confusing for some cases where you are trying to detect very high resistance values

Library Reference

You can get the capacitance reading from a pin using CircuitPlayground.readCap(*pin*) and just set the *pin* to any of the pad #'s next to the pad (e.g. 0, 1, 2, 3, 6, 9, 10 and 12) The number returned will vary from about 0 to about 1000. Any reading above approximately 50 will correspond to a touch, but it does depend a bit on what material and length is connected to the pin so you will need to adjust your code based on your setup.

Pad Usage (pinout)

We'll use terminology like analog inputs, PWM outputs, interrupts, I2C, UART etc. We'll cover all this later, just know you can use this list for reference!

Left Side

Starting from the Micro USB jack and going clockwise here are what the pins do.

- **3.3V** - this is the *output* from the onboard 3.3V power supply. This is the main power voltage for the 'Play but can also be used to power sensors, GPS or bluetooth modules, etc. You can draw a maximum of 500mA. See the **Battery Jack & Supply** page for more details
- **#10** - this is connected to the microcontroller's pin #10. This pin can also be used for *analog inputs* and is called **A10**. It can also act as a **PWM** output.
- **#9** - this is connected to the microcontroller's pin #9. This pin can also be used for *analog inputs* and is called **A9**. It can also act as a **PWM** output.
- **GND** - shorthand for **Ground** you'll need to this when powering or connecting to external devices.
- **#6** - this is connected to the microcontroller's pin #6. This pin can also be used for *analog inputs* and is called **A7**. It can also act as a **PWM** output.
- **#12** - this is connected to the microcontroller's pin #12. This pin can also be used for *analog inputs* and is called **A11**. It can also act as a **PWM** output.
- **VBATT** - this is the battery voltage *output* from whichever is higher: the JST battery pack or USB. *It cannot be used as a power input!* It is designed to power high-current or high-voltage devices that need more like 5V than 3.3V. If connected to USB, this pin provides 5V. If powered from battery, the output voltage will vary depending on the battery pack.

Right Side

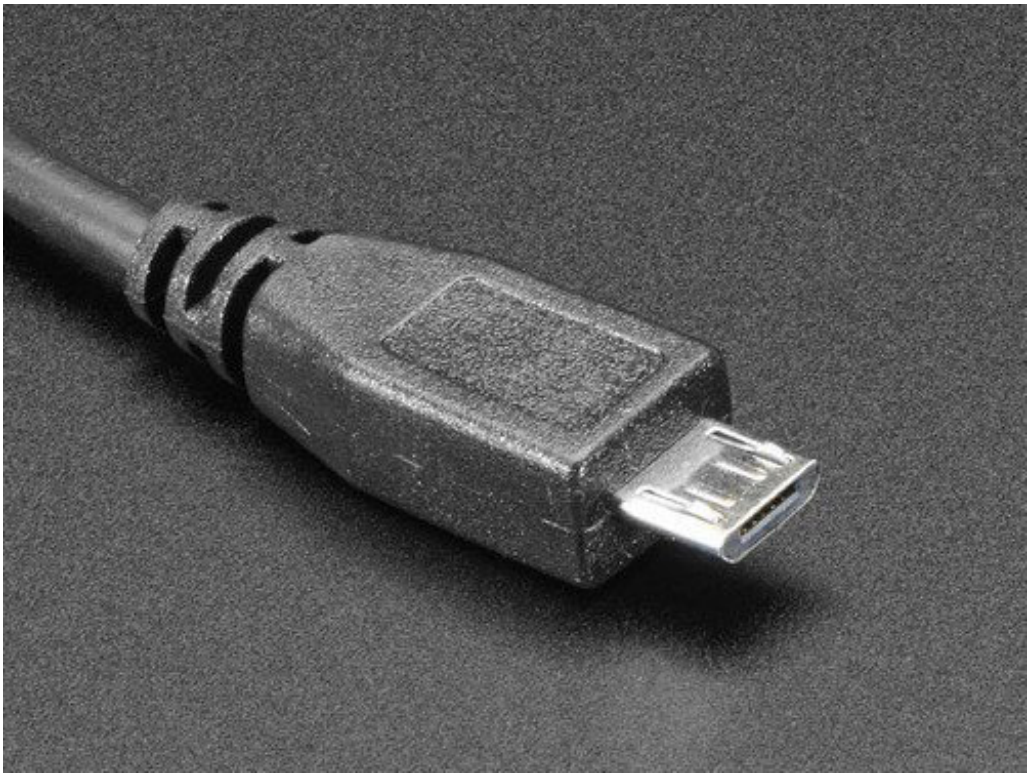
Starting from the Micro USB jack and going counter-clockwise here are what the pins do.

- **GND** - Same as the other **Ground** pins
- **SCL #3** - This is a special-purpose pin. It is connected to the microcontroller's pin #3. This pin can also act as a **PWM** output and an **interrupt** input (INT0). The other special usage is connecting to I2C sensors and devices, as the I2C Clock pin
- **SDA #2** - This is a special-purpose pin. It is connected to the microcontroller's pin #2. This pin can also act as a **PWM** output and an **interrupt** input (INT1). The other special usage is connecting to I2C sensors and devices, as the I2C Data pin
- **3.3V** - Same as the other **3.3V** power supply pins
- **RX #0** - This is a special-purpose pin. It is connected to the microcontroller's pin #0. This pin can also act as an **interrupt** input (INT2). The other special usage is connecting to UART/Serial sensors and devices, as the Data Receive (RX) pin.
- **TX #1** - This is a special-purpose pin. It is connected to the microcontroller's pin #1. This pin can also act as an **interrupt** input (INT3). The other special usage is connecting to UART/Serial sensors and devices, as the Data Transmit (TX) pin.
- **GND** - Same as the other **Ground** pins

Power Up Test

Now we are ready for the moment of truth, it's time to plug your Circuit Playground in and power it up. The easiest way to do this is to plug one end of the USB cable into the 'Play and the other end into a computer. The computer will then power the Circuit Playground.

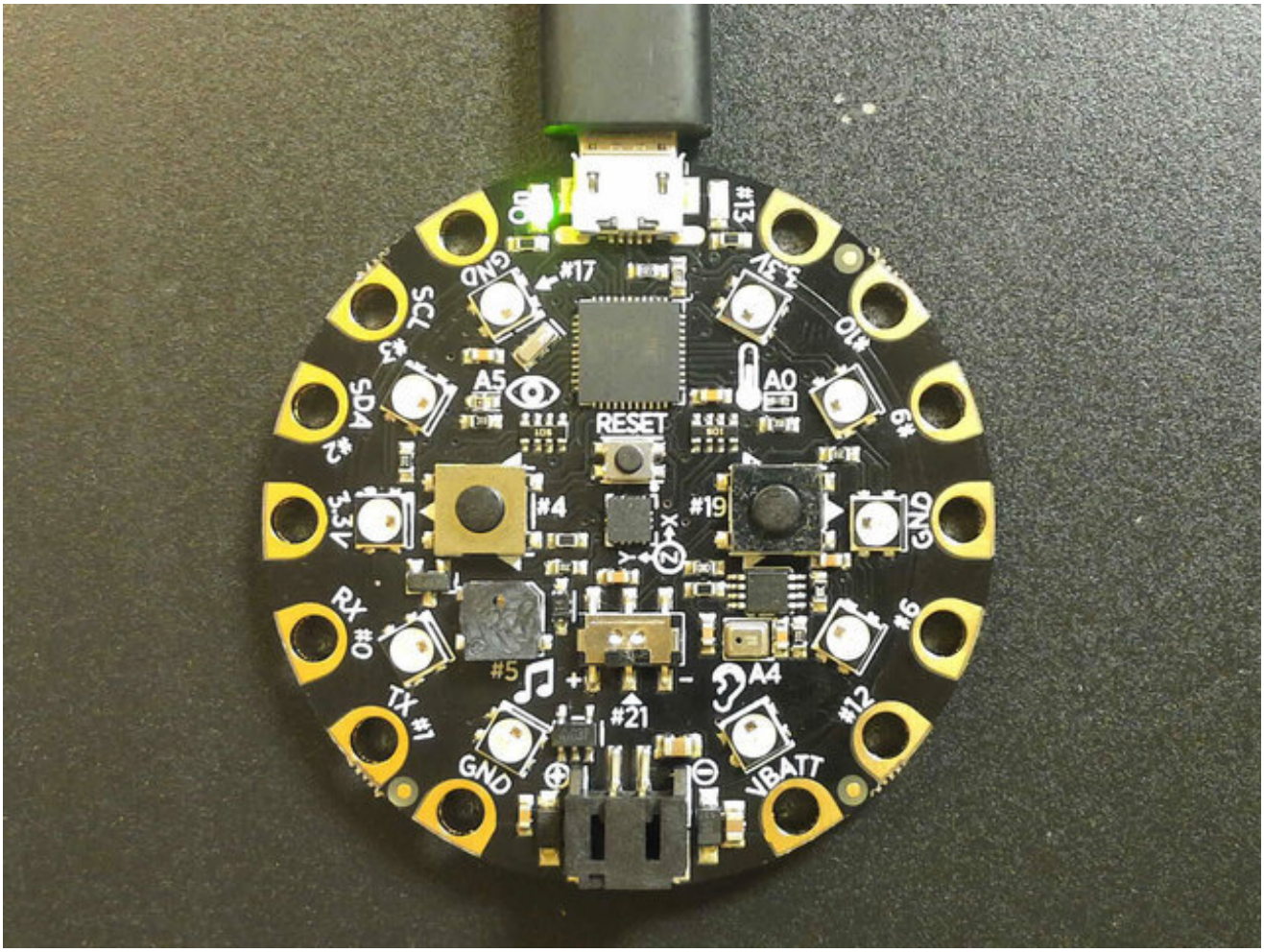
You'll need a USB cable with a slim Micro B-type end:



Make sure that the USB cable is plugged in directly to a computer port. Sometimes monitors or keyboards have a USB port you can plug into. Most of the time this is fine, but I strongly suggest you plug it directly into the computer as that will eliminate any possible problems. Same goes for USB hubs - skip those for now and go direct

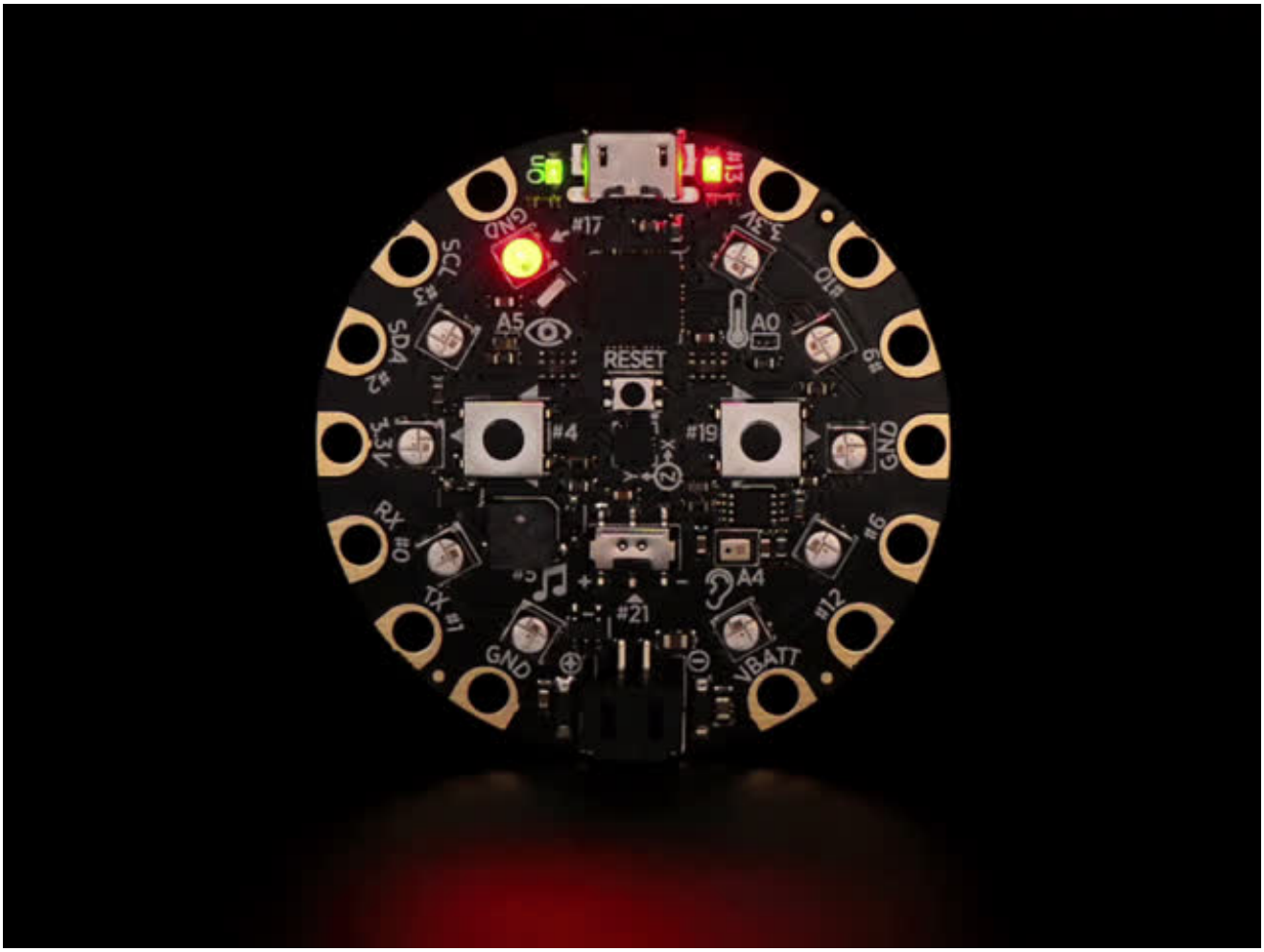
Later on, once you've verified you can power the Circuit Playground and upload sketches no problem, then you can try plugging it into other ports.

OK anyways, so plug int he USB cable and check that your board looks like this:



In particular, make sure the green **ON** LED is lit!

If this is the first time you're plugging in your Circuit Playground fresh from the factory, the red **L** LED and and NeoPixels might also be lit or blinking - this is normal.



If no lights or blinking occurs, double check:

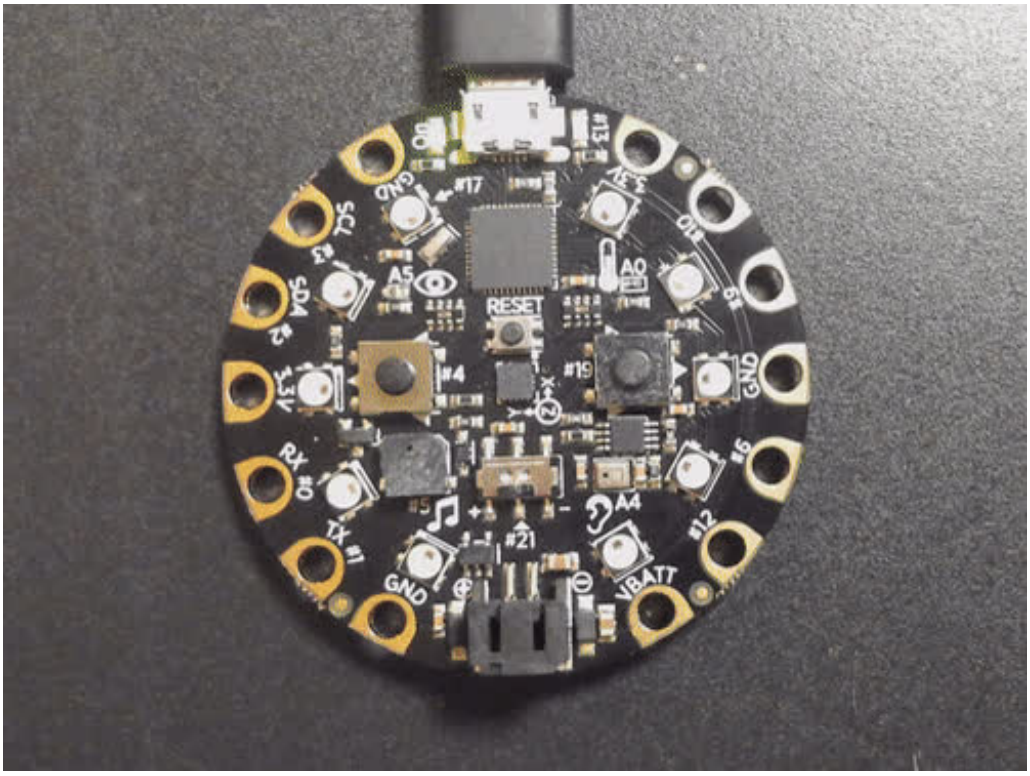
- Is the USB cable plugged into the computer and into the Circuit Playground?
- Try another USB cable
- Check there's nothing metallic touching the Circuit Playground that could be*shorting out* the device
- Is the computer on?
- Try another USB port, USB cable, and computer?

If you still can't get it working, your Circuit Playground may be faulty and need replacement!

Bootloader Reset Test

Next up, you can do a quick**bootloader test** - this will let you know that the Circuit Playground chip has been programmed with a bootloader which is required!

While powered, double click the **Reset button** - you will see the**#13** red LED breathe/pulse on and off. It will pulse about 10 times, once per second



If you get both the green **ON** LED lit and the red **#13** LED to pulse you are well on your way and have passed the power up test!

Download Software

To get you all set up, start by installing the**Arduino IDE Software**

This is the free application you'll use to write programs and talk to your Arduino or compatible. Did we mention it is free? How awesome is that?

You can download Arduino from

https://www.arduino.cc/en/Main/Software (http://adafru.it/fvm)

There's a lot of other companies and groups that may try to get you to download the Arduino software but it could have viruses or malware. Only download from arduino.cc !

When you visit the Arduino site you'll see a section like this:

The Arduino software is under constant revision. As of this writing, the version available is 1.6.9 but you may have a more recent version. Just grab whatever is the most recent

Windows

Download and install with the**Installer.** The Zip file (non-admin install) is not recommended unless you cannot run the installer

Mac

Download and drag the Application out of the compressed folder.

Linux

Available for 32-bit or 64-bit Linux, once you download you will need to manually decompress and install

Raspberry Pi and other ARM-based Linux

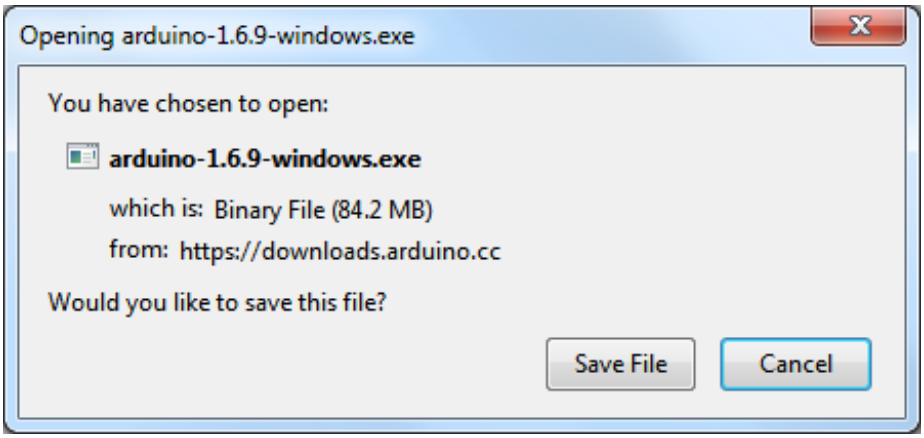
There's a new version you can use that is compiled for ARM processors! It works on the Raspberry Pi and will likely work on any other ARM core Linux

Install Software (Windows)

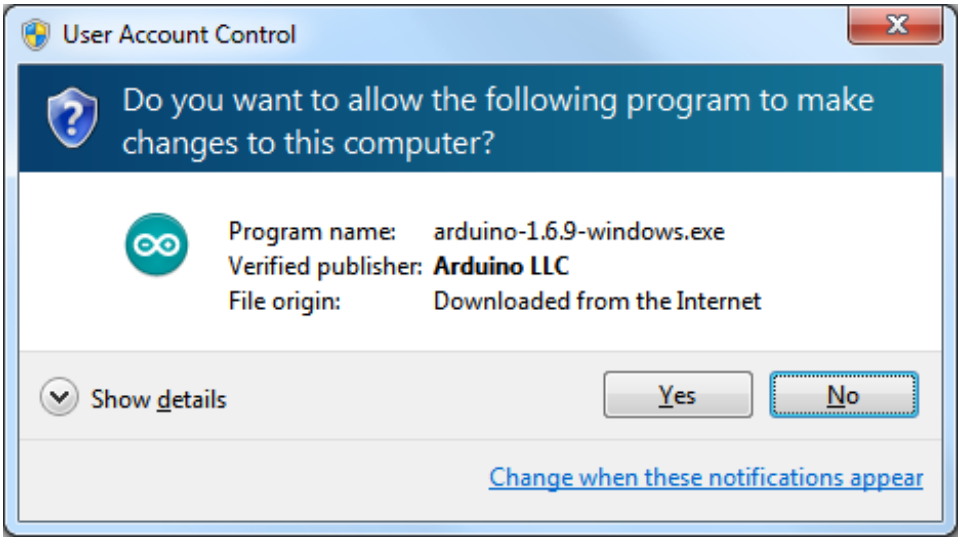
Installing Arduino

Visit **arduino.cc** to download the latest version of Arduino

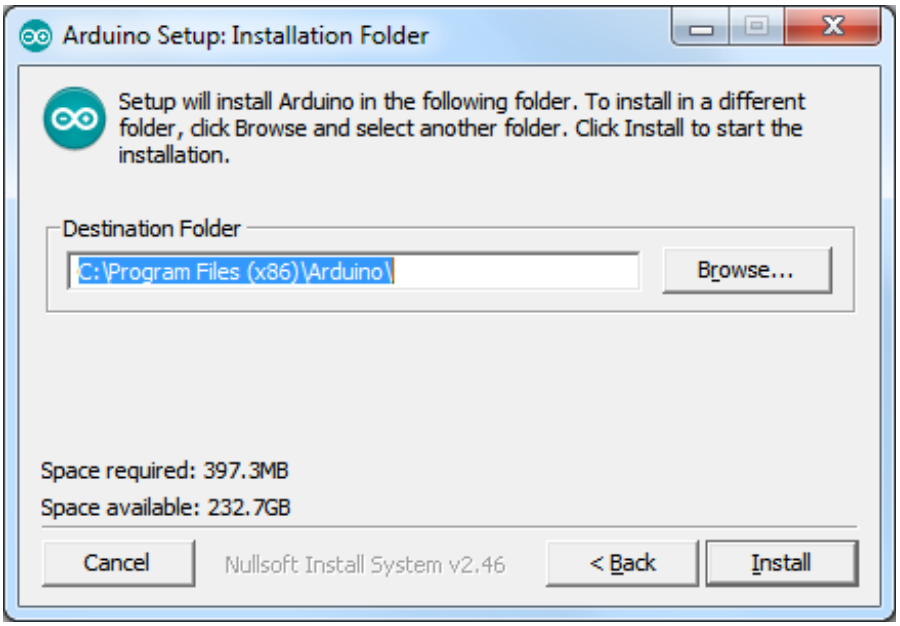
Click on the **Windows Installer** link to download the installer, then double click to launch it



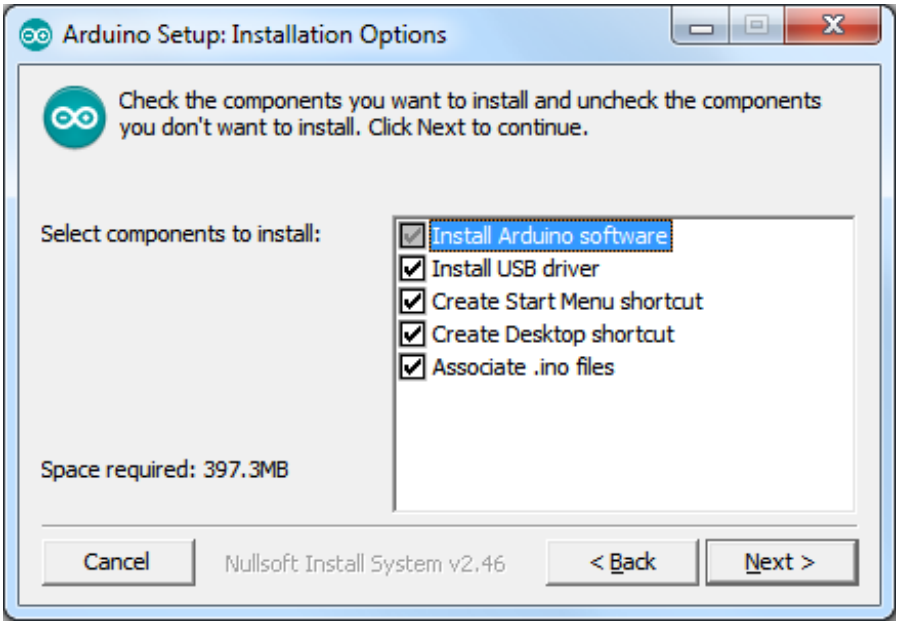
You may get a warning asking if you're sure you want to run the installer. It's ok, click **YES**



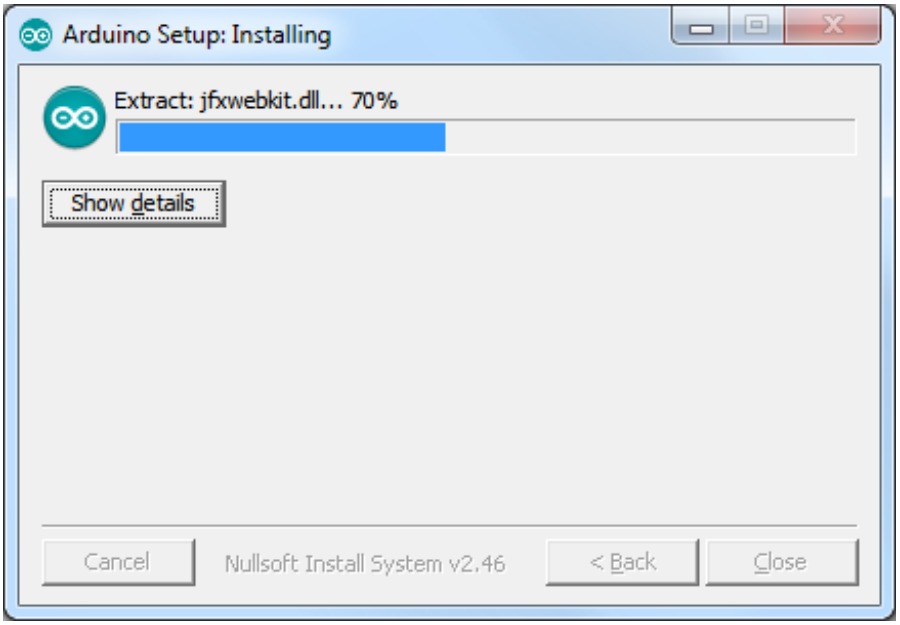
There is an open source license to click through. Install in the default location



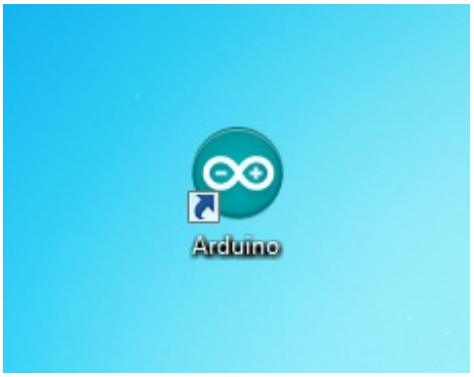
You can use the default setup installation options



Finally it will take a minute or two to install



When done you'll have the software installed



Circuit Playground Driver

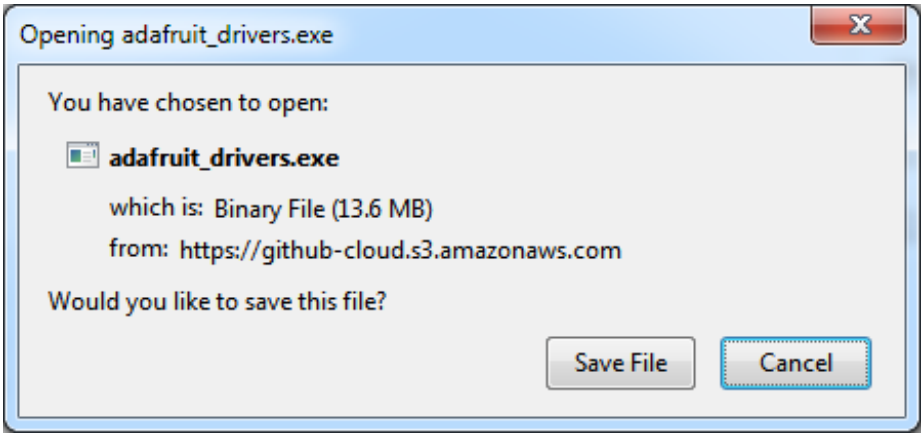
You will need to install seperate drivers for the Circuit Playground board

For all Adafruit compatibles, we have an *all in one* installer that will install all of the Adafruit board drivers

Click below to download our Driver Installer

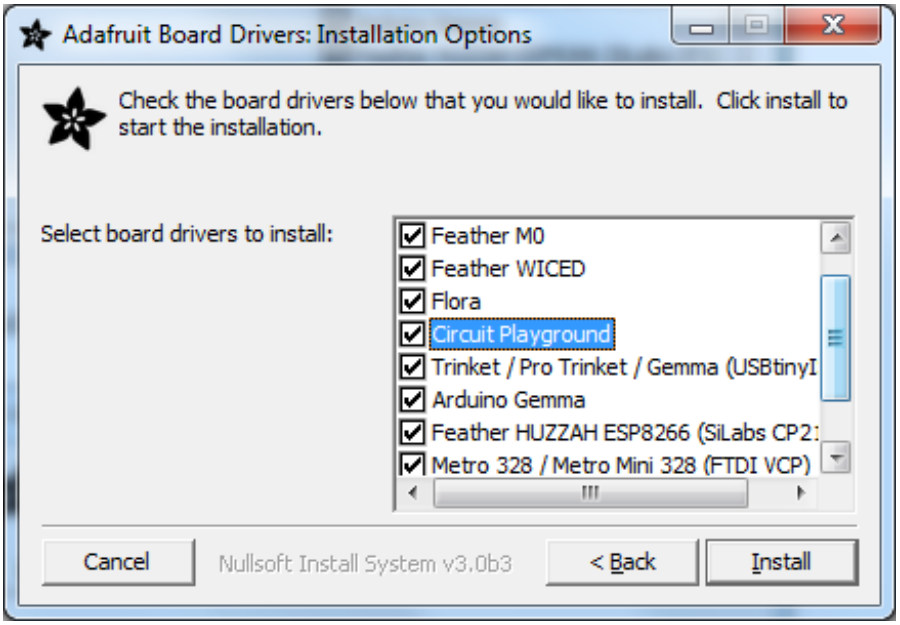
[Download Adafruit Boards Windows Driver Installer]
http://adafru.it/mai

Download and run the installer

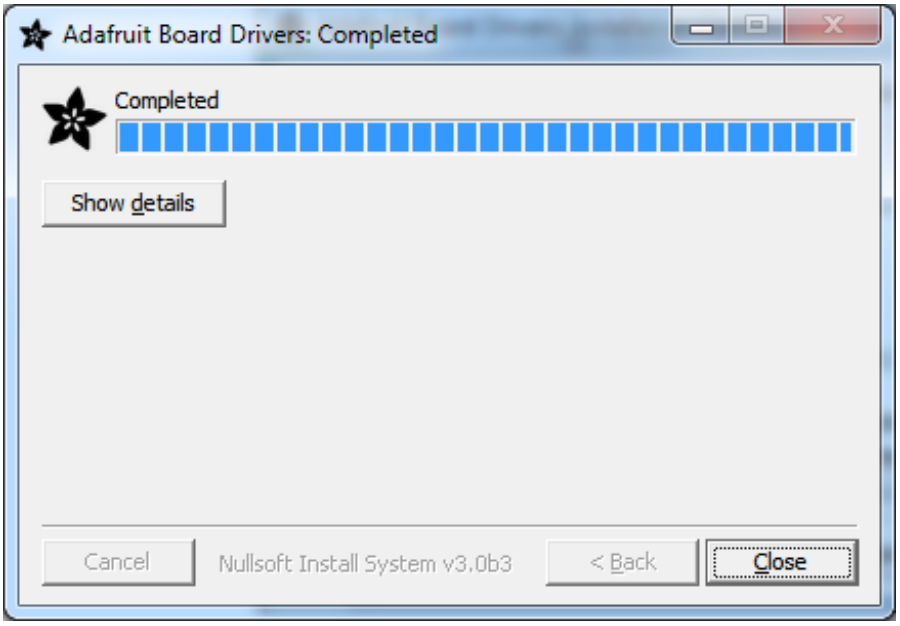


Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license

Select which drivers you want to install (we suggest selecting all of them so you never have to worry about installing drivers when you start to explore other Arduino-compatibles)



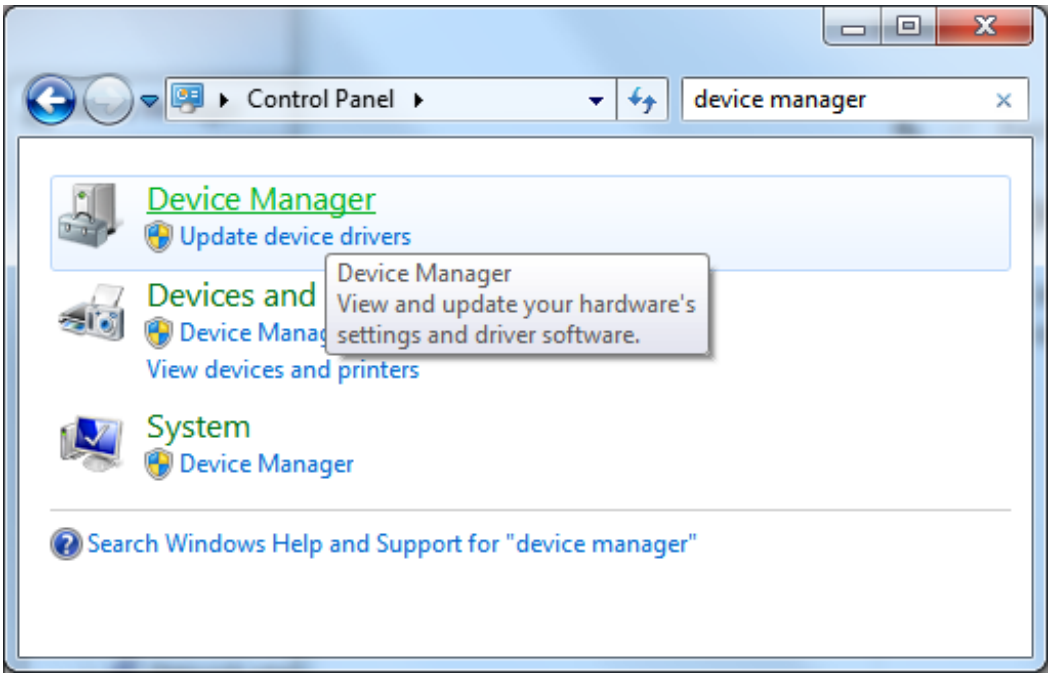
Click **Install** to do the installin'



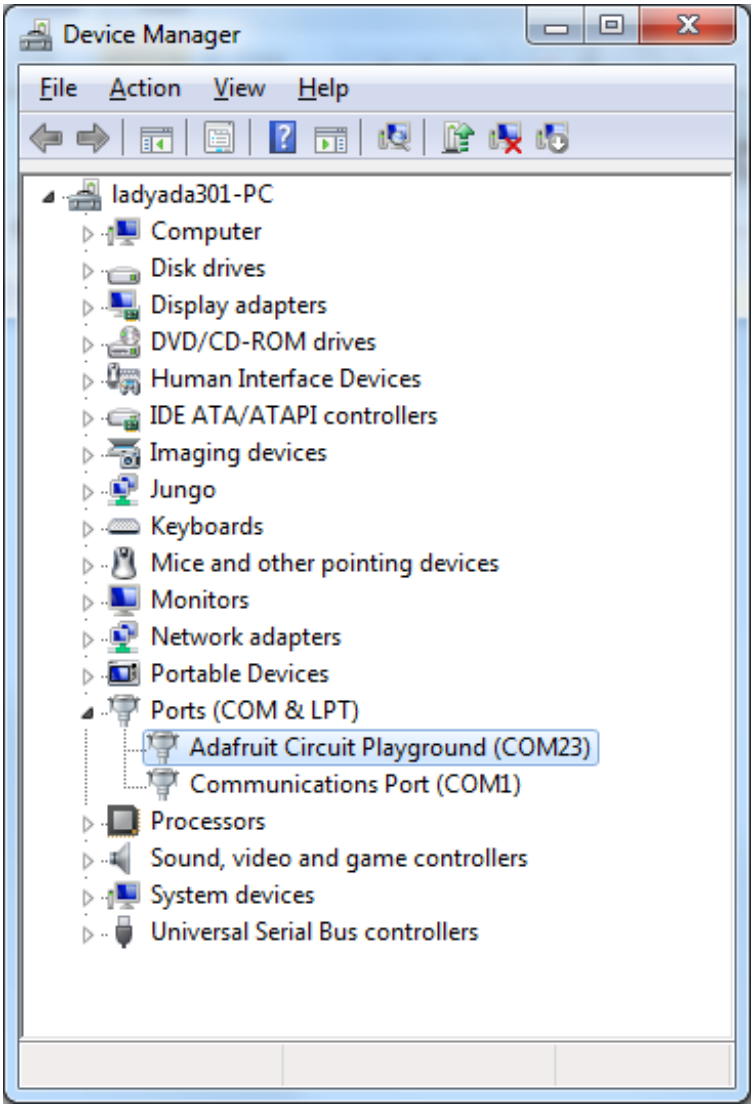
You should not need to restart your computer but it's not a*bad* idea!

Find your Serial (COM) Port

To verify your Arduino driver installed properly, plug it into USB and open up the**Device Manager**. You can find the Device Manager in the**Control Panel** (search for Device Manager)



When you open the Device Manager, find the section called**Ports** and expand it:



You'll see an icon next to some text that says**Adafruit Circuit Playground (COMxx)** where **xx** is a number

The COM number may vary but it should be something like**COM3** or **COM4**. The COM stands for "communication", and each one has a unique number, known as the COM Port number. In this case the COM Port number is COM18.

You can unplug your 'Play to see the COM port device disappear and re-appear when plugged in.

If you **don't** see the Circuit Playground show up, check:

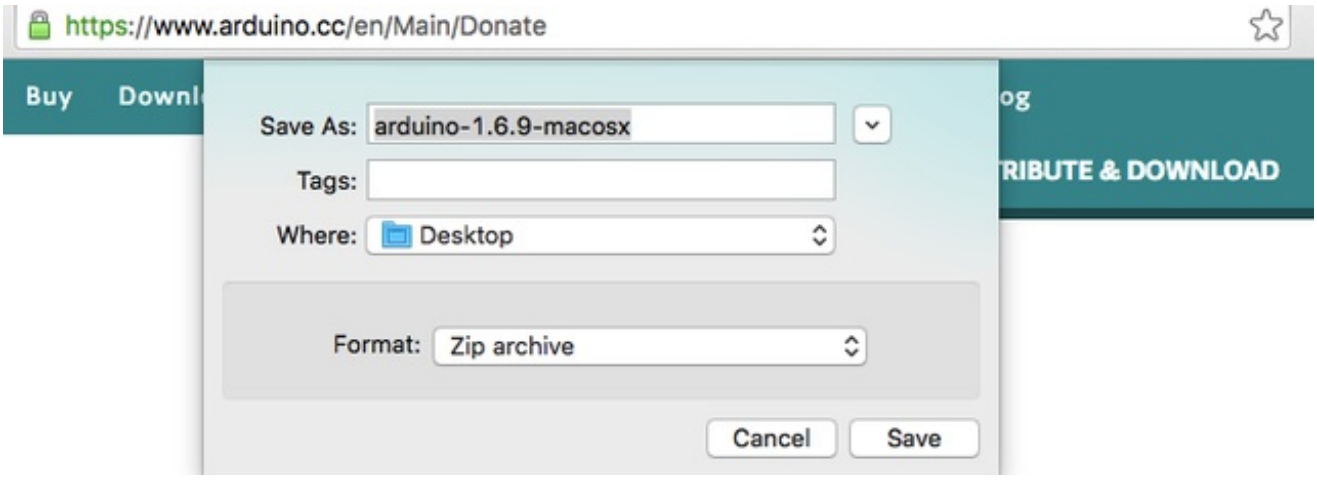
- Is your cable a data cable or charge only? Try another USB cable
- Try another USB port!
- Verify you installed the drivers, you can always try installing them again (never hurts) and rebooting

Install Software (Mac OS X)

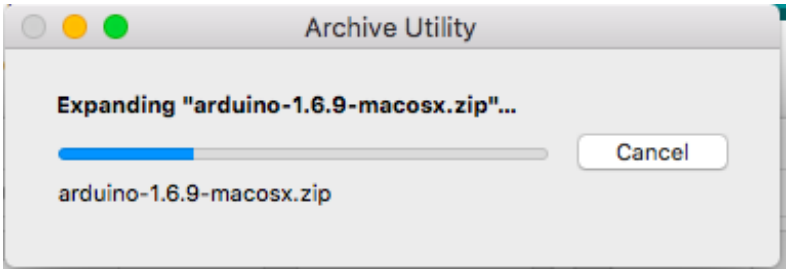
Installing Arduino

Visit **arduino.cc** to download the latest version of Arduino

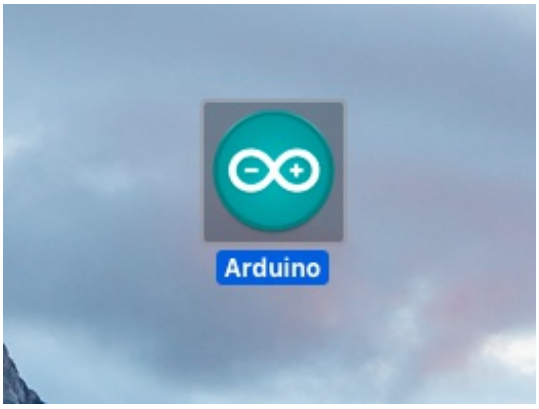
Click on the **Mac OS X Installer** link to download the installer



Then double click to expand/launch it



it will automatically give you the **Arduino app** the teal icon:

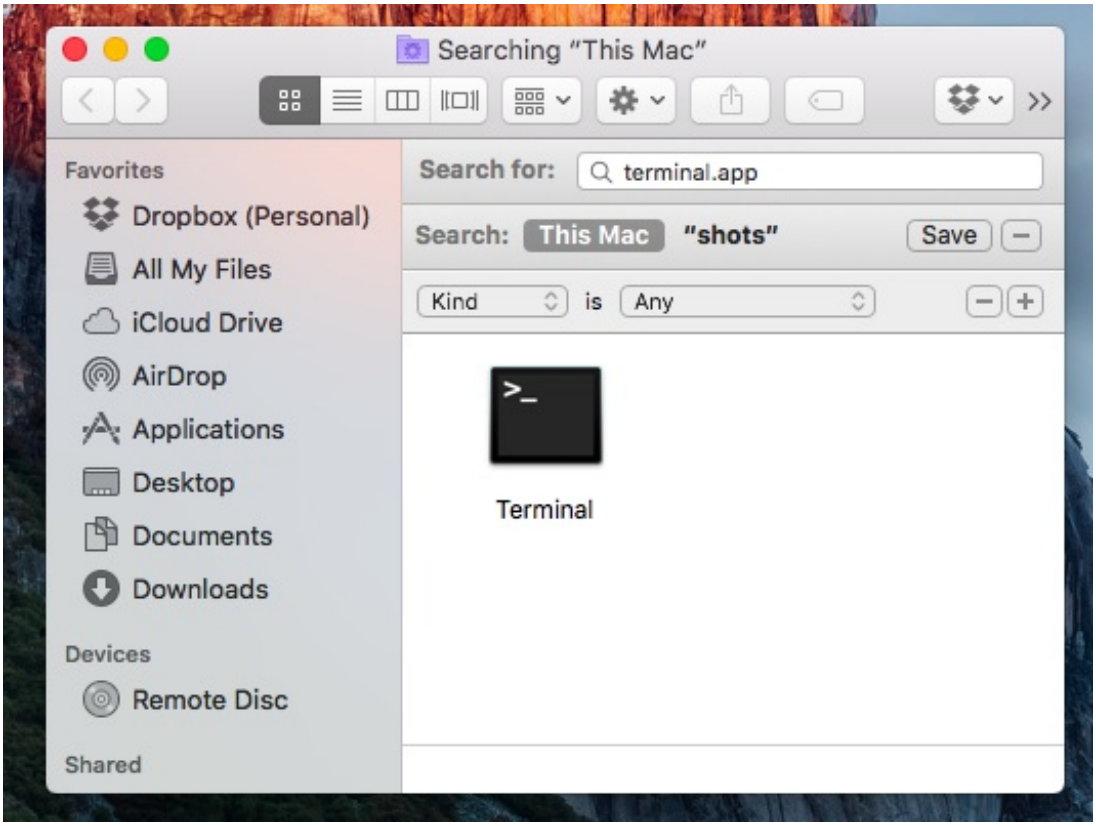


No Drivers To Install!

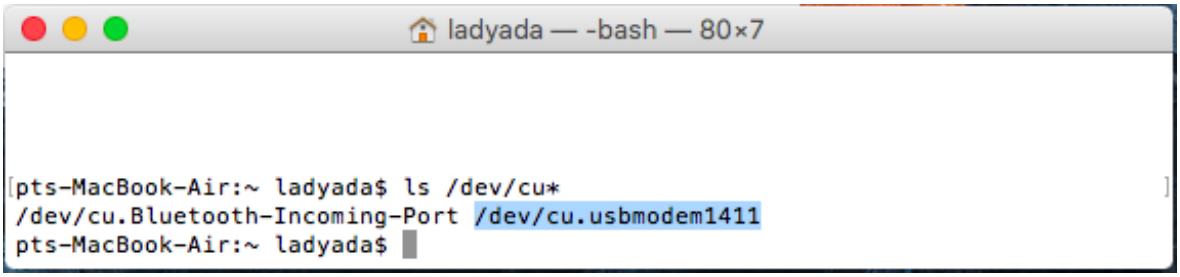
Thanks to Mac OS X having the drivers built in, you won't need to install any other software!

Find your Serial Port Device

You can use **Terminal** to help find and verify your Arduino. First, launch the **Terminal** app - you can Command-F to Find the **Terminal.app** program:



Double click to launch it. At the text prompt, type in **ls /dev/cu*** (note the first letter is a lower-case L)



Make sure you see a line with the text/**dev/cu.usbmodem*xxxx*** or /**dev/cu.usbserial-*xxxxx*** where the xxx's can be anything. This indicates that the driver installed properly and that the Circuit Playground was found.

You can close the Terminal program now

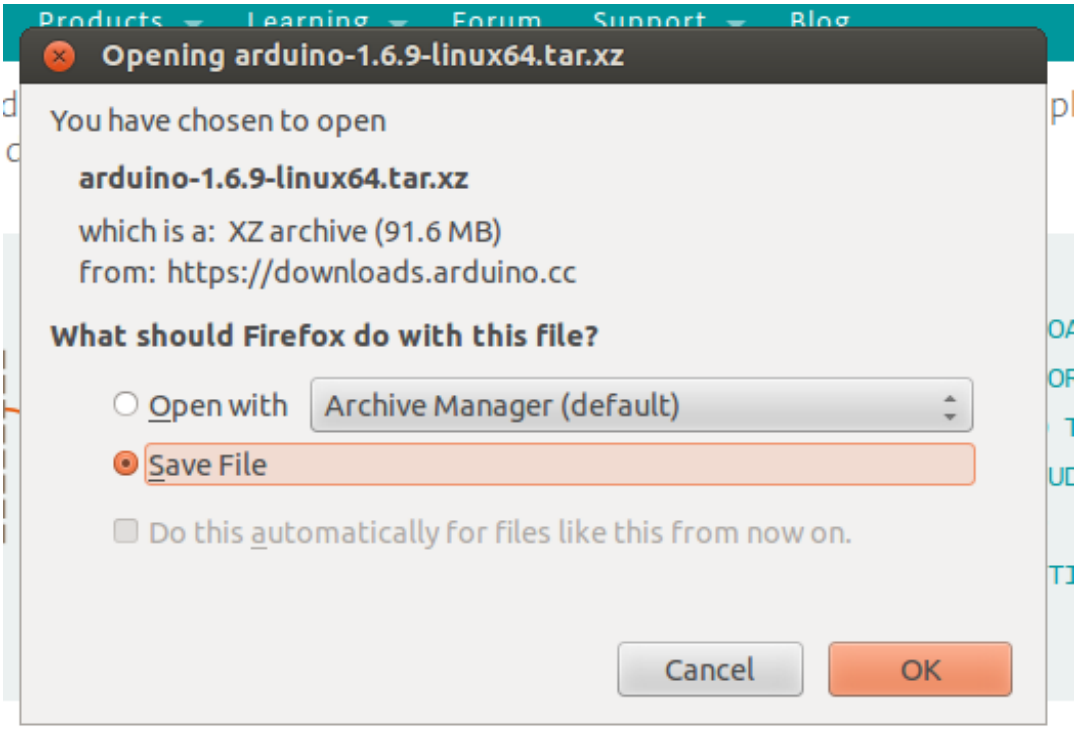
Install Software (Linux)

Installing Arduino

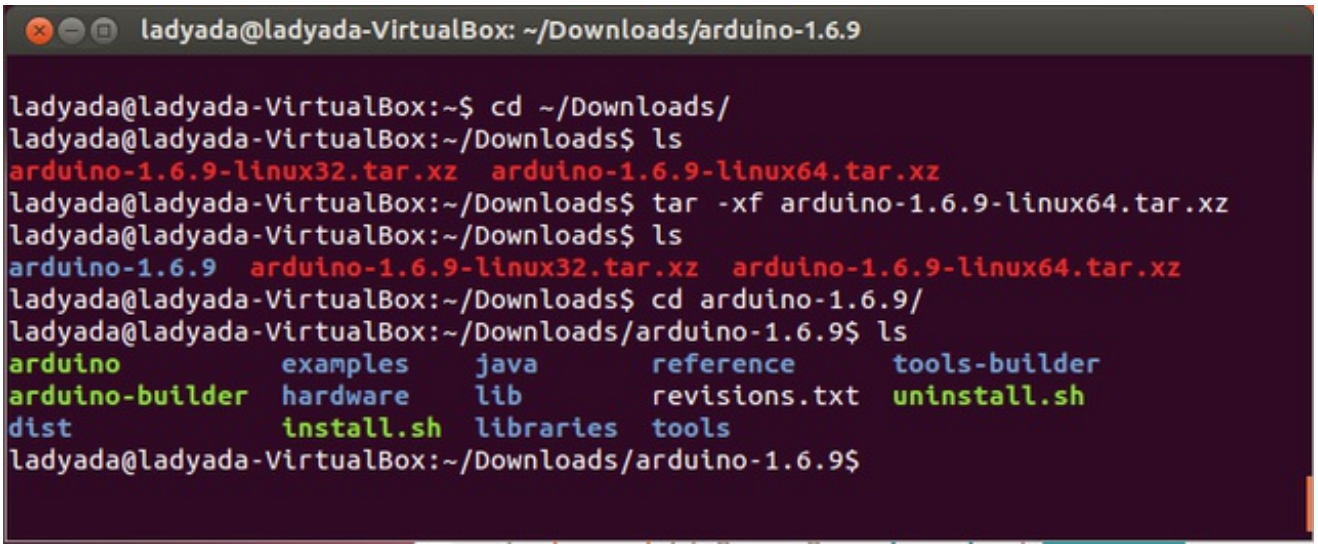
Visit **arduino.cc** to download the latest version of Arduino

Do not use yum or apt-get to install Arduino! You will get an ancient version, always download the latest from arduino.cc!

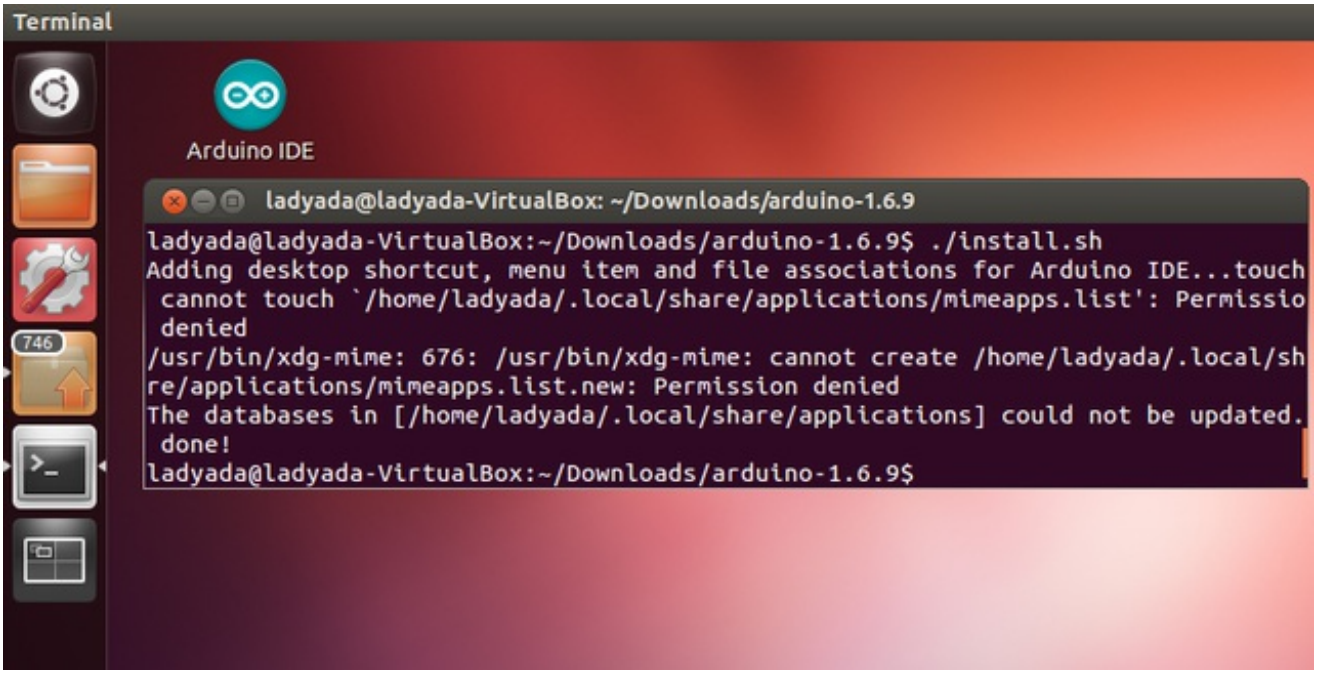
Click on the matching **Linux Installer** link (32 bit, 64 bit or ARM) to download the installer - save the file to your Downloads folder



From within your Terminal program, **cd** to the Downloads directory, and untar the package with **tar xf arduino*.xz** then **cd** into the **arduino-n.n.n** folder created:



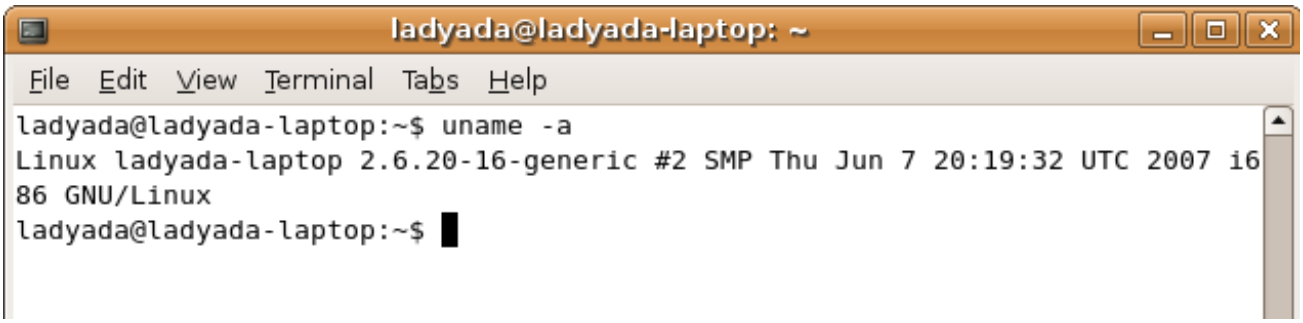
Run **./install.sh** to install the software. I've got an old Ubuntu install so I got warnings, but it did create that desktop icon for me!



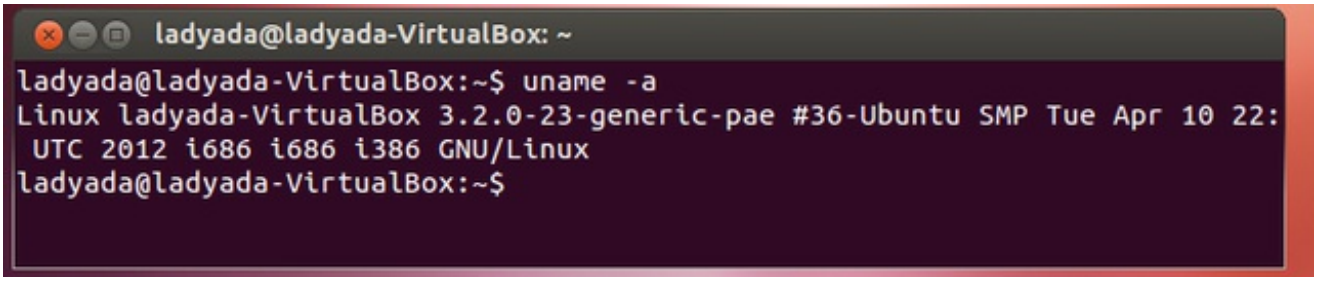
Driver Installation

Linux doesn't have any drivers to install, assuming you're running a v2.6 kernel or higher, which is almost certainly true. These instructions assume you're running Ubuntu. Each Linux distribution is different, but the instructions should be basic enough to follow for other distros.

You can verify your kernel version by running **uname -a** in a terminal window, note that this kernel is version **2.6.20**



And this one is **3.2.0-23**



Some older Linux distributions used to install **brltty** (braille device) which will conflict with the serial port. You **must uninstall brltty if it is installed!** Do so by running sudo apt-get remove brltty or equivalent In a terminal window. If it says it's not installed then thats OK. If you're not running a Debian-derived installation use whatever tool is necessary to verify that you don't have **brltty** running

Verify your Serial Port

Plug in the Circuit Playground, verify that the green LED is lit, and type **ls /dev/ttyUSB*** into a terminal window, you should see a device file called something like ttyUSB0

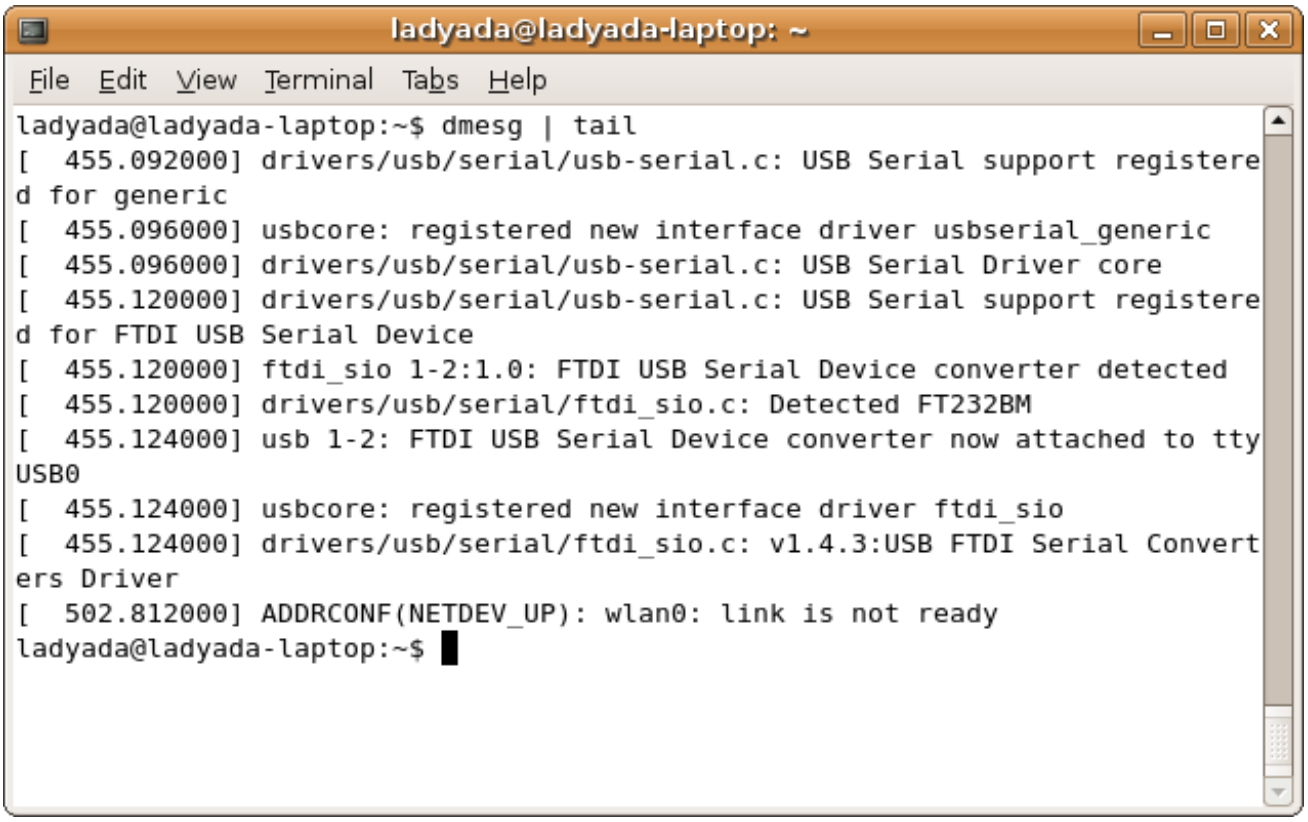
If you can't seem to find it, use **dmesg | tail** right after plugging in the Circuit Playground and look for hints on where it may put the device file. For example here is says **Serial Device converter now attached to ttyUSB0**

If you see something like this

```
[ 1900.712000] ftdi_sio 2-10:1.0: FTDI USB Serial Device converter detected
[ 1900.712000] drivers/usb/serial/ftdi_sio.c: Detected FT232BM
[ 1900.712000] usb 2-10: FTDI USB Serial Device converter now attached to ttyUSB0
```

```
[ 1901.868000] usb 2-10: usbfs: interface 0 claimed by ftdi_sio while 'brltty' sets config #1
[ 1901.872000] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[ 1901.872000] ftdi_sio 2-10:1.0: device disconnected
```

That means you have not uninstalled **brltty** and you should try again.



udev Rules

To use Adafruit's boards with most modern Linux distributions you'll want to ensure a few udev rules are applied. These rules apply special configuration for USB devices like Adafruit's boards to workaround or fix common issues. Specifically these rules allow Trinket and other boards to be programmed by the Arduino IDE that's run as a normal non-root user. The rules also fix an issue with ModemManager hanging on to /dev/ttyACM devices when using Circuit Playground, Flora or Bluefruit Micro boards.

To install the rules you'll want download them and copy to the location of udev rules on your system. For most Linux systems like Ubuntu, etc. udev rules are stored in **/etc/udev/rules.d/** (check your distribution's documentation / help forums if you don't see this folder exists). Run the following commands:

```
wget https://github.com/adafruit/Trinket_Arduino_Linux/raw/master/99-adafruit-boards.rules
sudo cp 99-adafruit-boards.rules /etc/udev/rules.d/
```

Note that you might need to change the rule if you're using a distribution other than Ubuntu/Debian. In particular the first rule in the file applies the 'dialout' group to Trinket, etc. boards so they can be programmed without running the Arduino IDE as root. Some distributions use a different group for this instead of 'dialout'. Check your distribution docs or help forums to see what group should apply to devices that can be accessed by non-root users.

Next you'll need to reload udev's rules so that they are properly applied. You can restart your machine, or run a command like the following:

```
sudo reload udev
```

If the command above fails, try instead running:

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

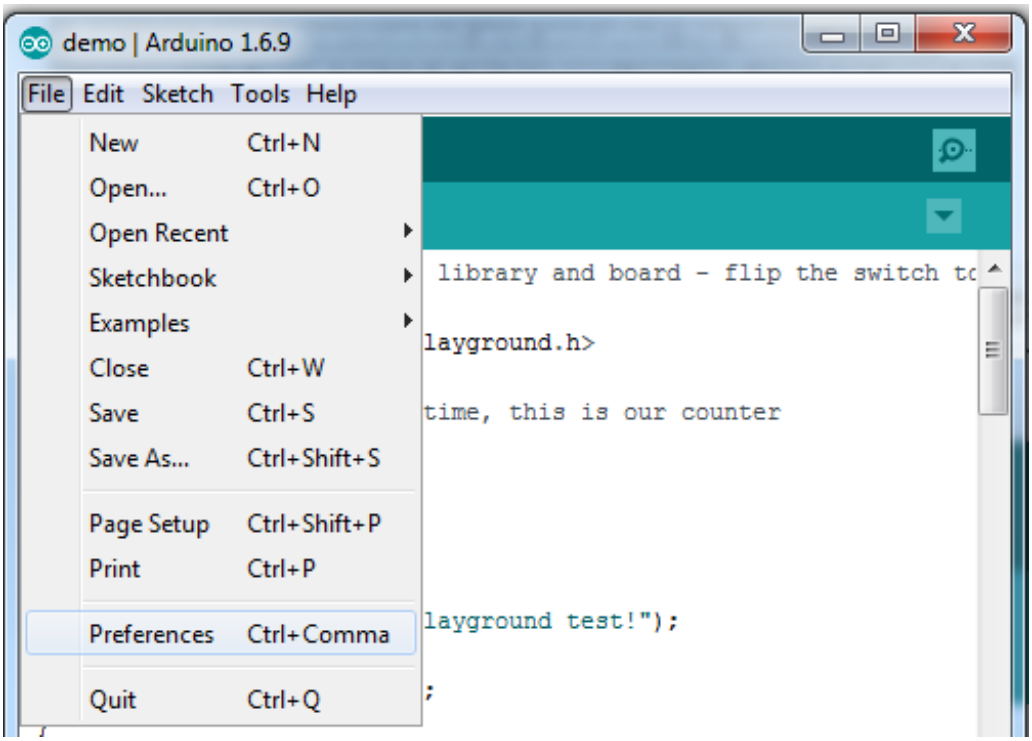
And if that still fails reboot your system as it will ensure udev picks up the new configuration.

Add Circuit Playground to Arduino

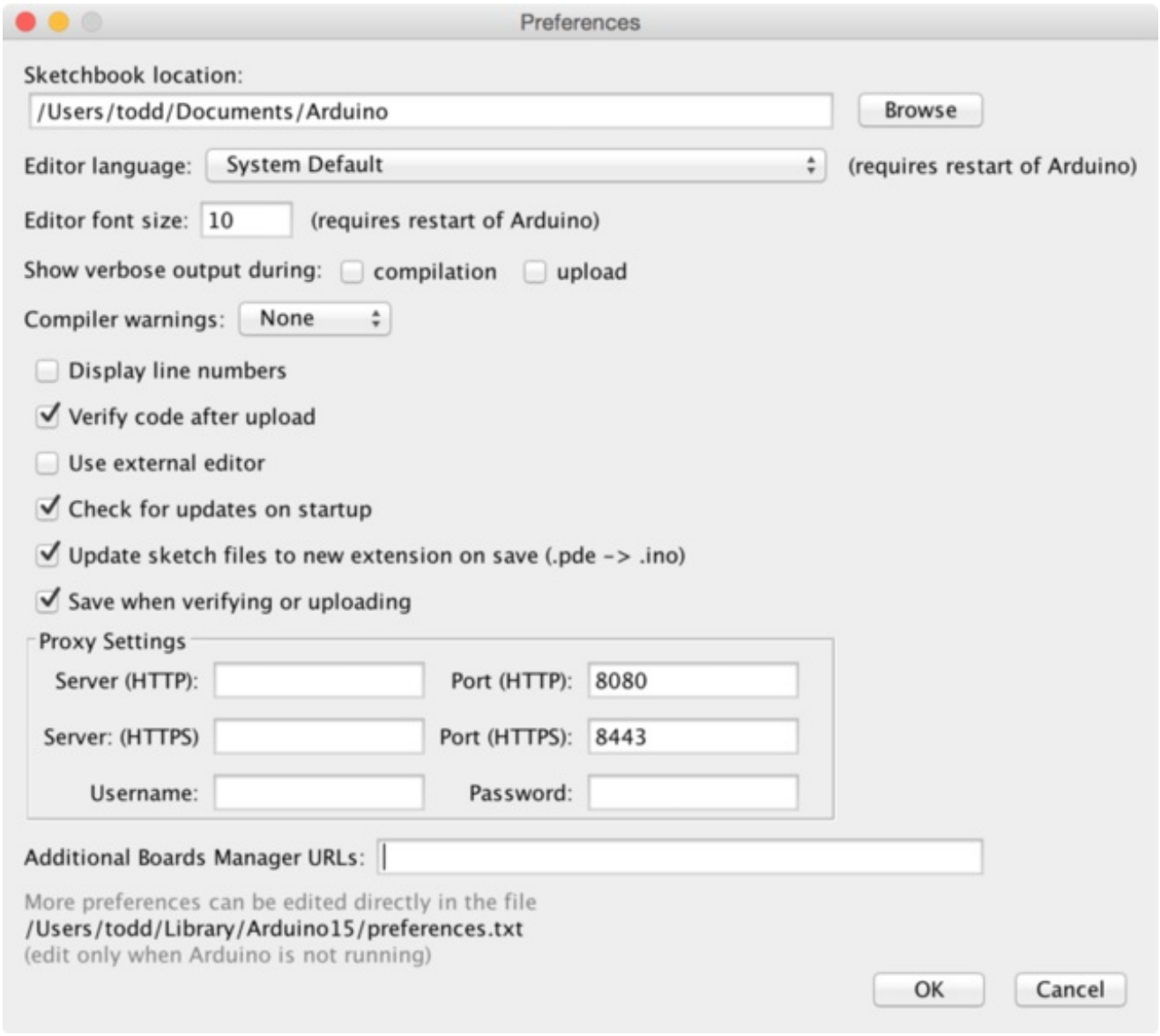
Circuit Playground is not a default option available in Arduino, we need to add software support. This happens once, and will automatically update when we make improvements and upgrades

Open up Preferences Menu

After you have downloaded and installed the latest version (remember it**must** be 1.6.4 or higher), you will need to start the IDE and navigate to the **Preferences** menu. You can access it from the **File** menu in *Windows* or *Linux*, or the **Arduino** menu on *OS X*.



A dialog will pop up just like the one shown below.



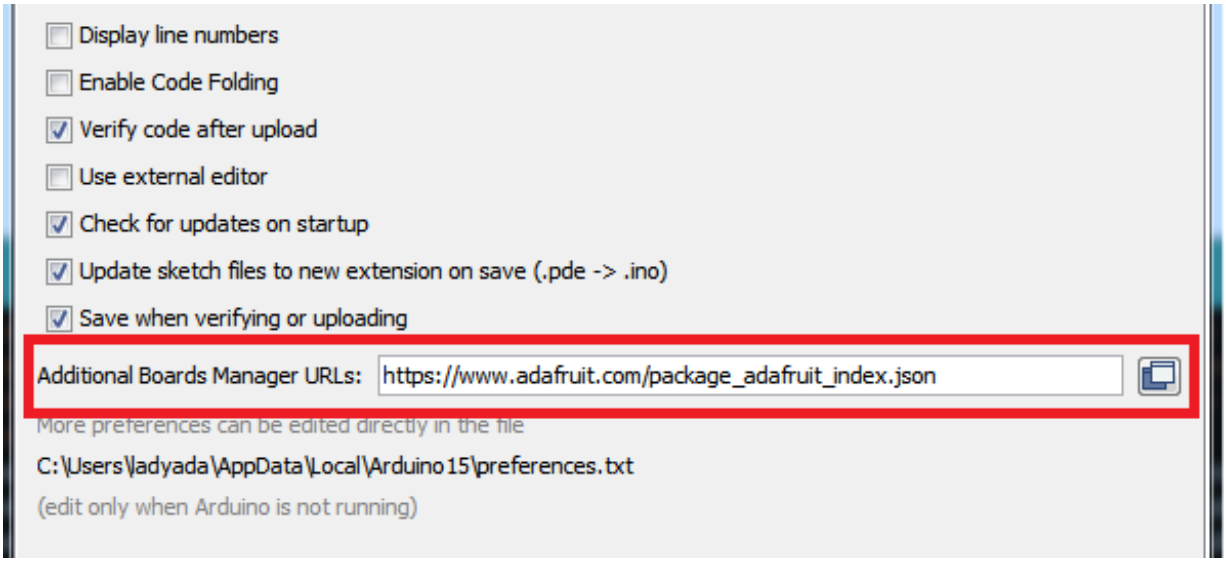
We will be adding a URL to the new**Additional Boards Manager URLs** option. New Adafruit boards and updates to existing boards will automatically be picked up by the Board Manager each time it is opened. The URL points to index files that the Board Manager uses to build the list of available & installed boards.

Add the Adafruit Board Support package!

paste

https://adafruit.github.io/arduino-board-index/package_adafruit_index.json

Into the "Additional Board Managers URLS" box

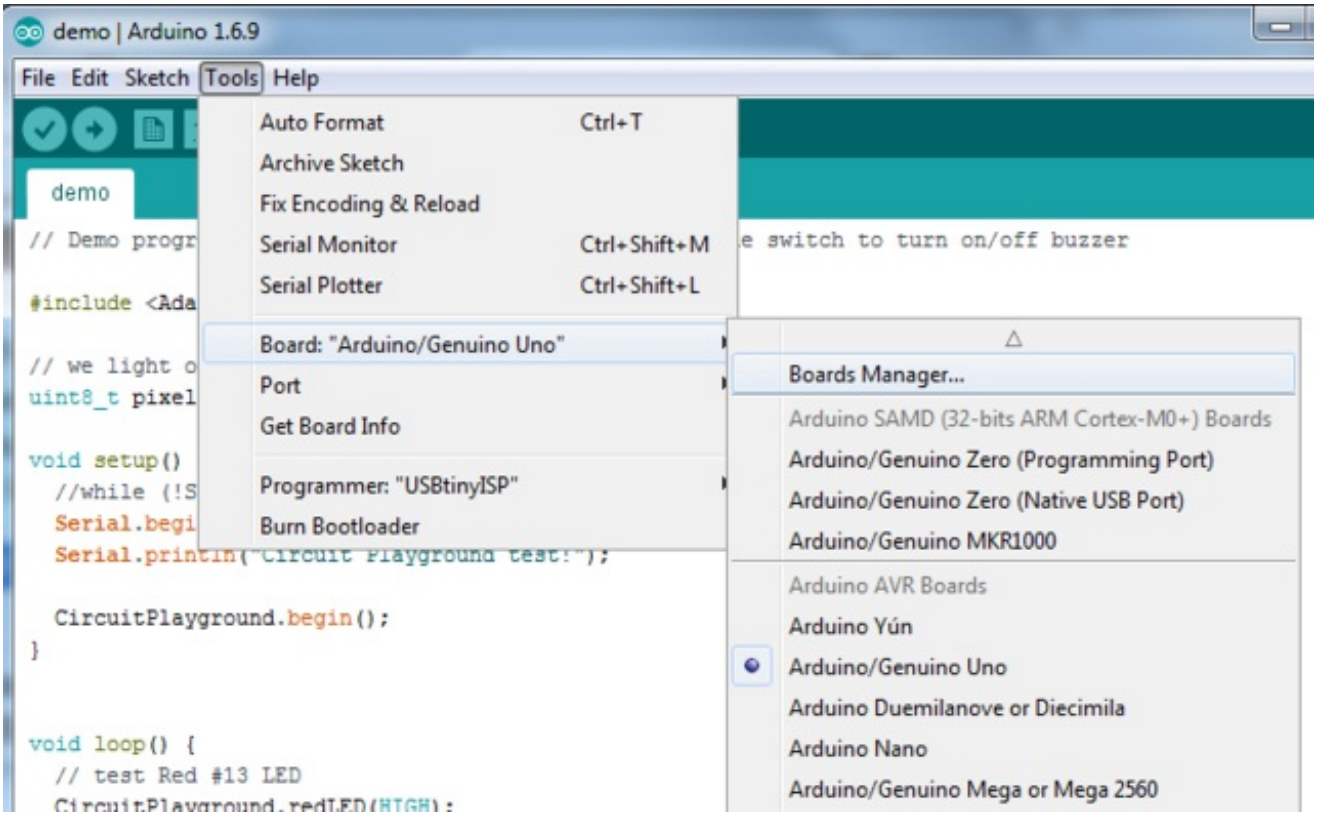


Click **OK** to save the new preference settings. Next we will look at installing boards with the Board Manager.

Manage Board Support

Adding the link to the Adafruit board support package does not actually install anything, it only tells the Arduino IDE where to find the software.

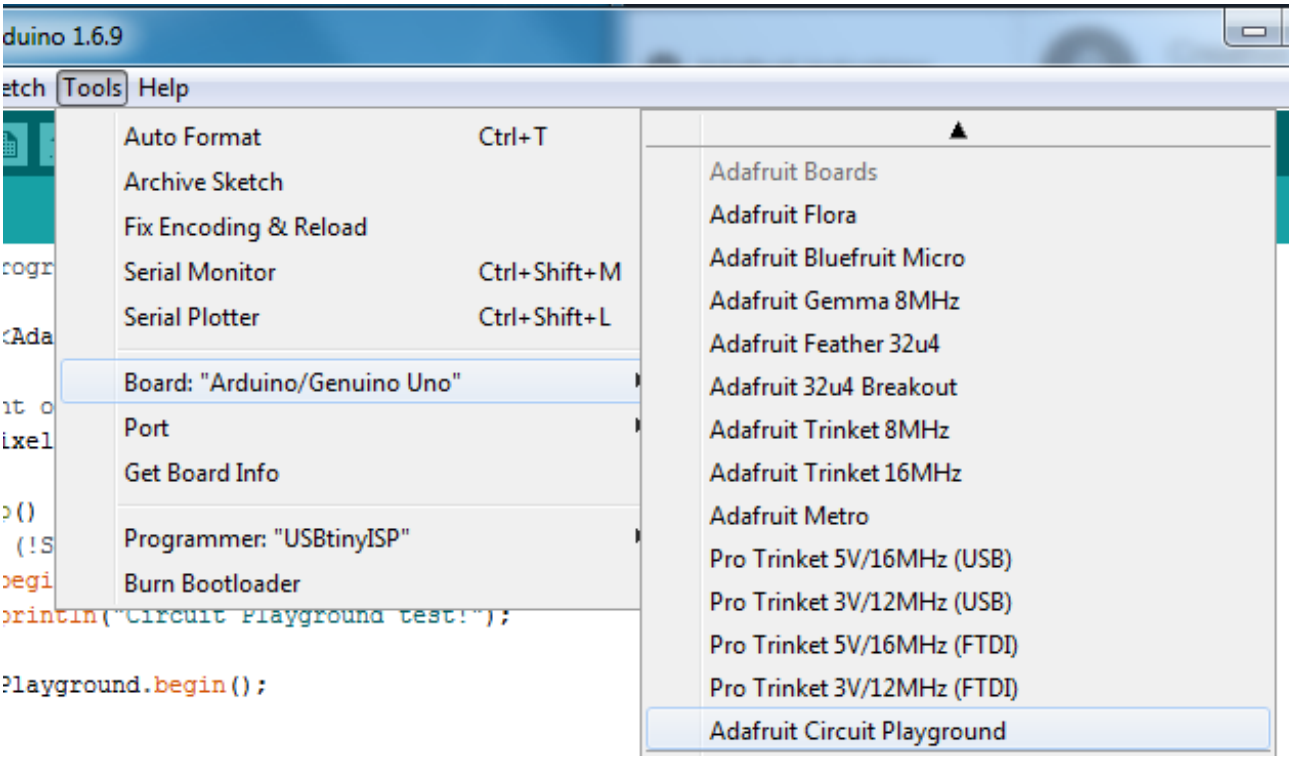
Now that you have added the appropriate URLs to the Arduino IDE preferences, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.



Once the Board Manager opens, type **Adafruit** into the search box and locate the **Adafruit AVR Boards** item. Click on it and then click **Install**

Next, quit and reopen the Arduino IDE to ensure that all of the boards are properly installed.

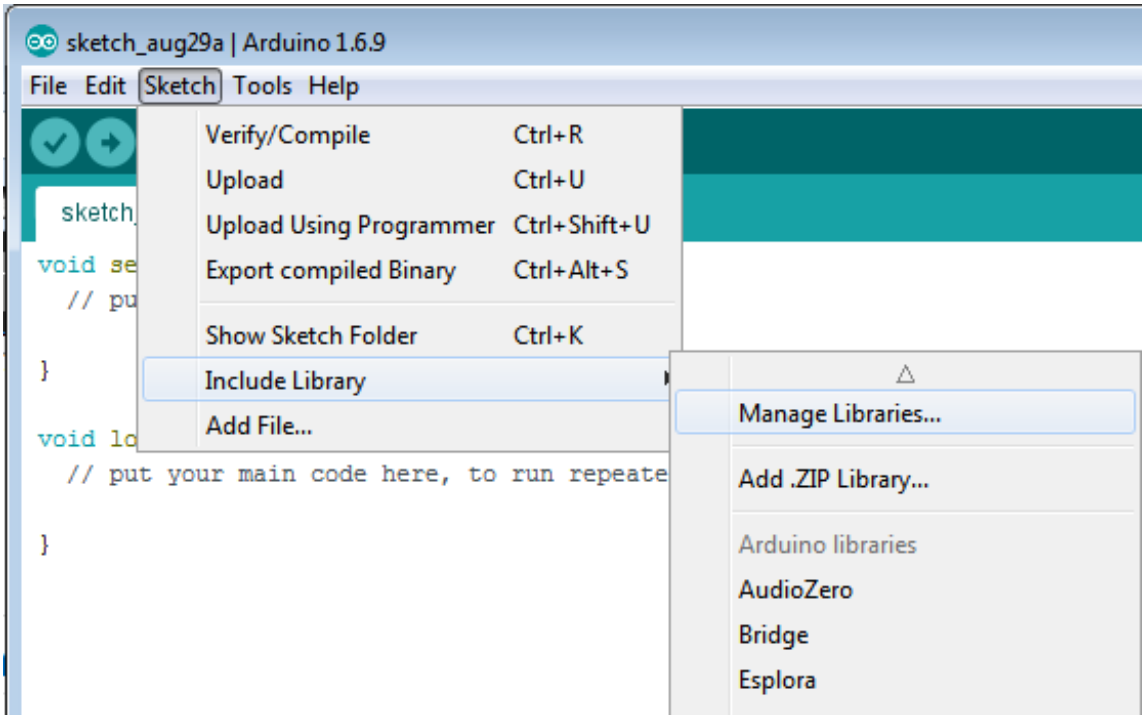
You should now be able to see **Circuit Playground** listed in the **Tools->Board** menu in the *Adafruit Boards* section.



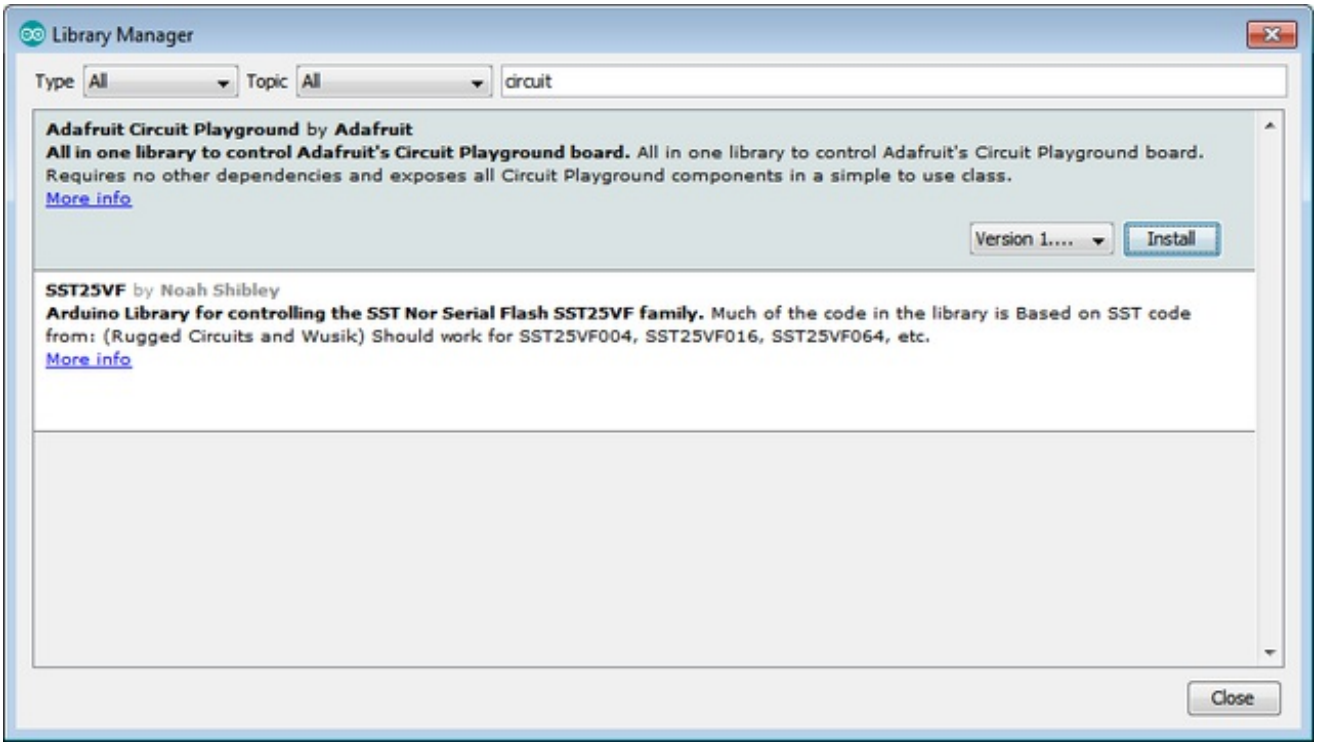
Add Circuit Playground Library

Lastly, we will add the library (collection of software) that gives you all the Circuit Playground goodies.

First, load up the **Library Manager** by going to the **Sketch -> Include Library -> Manage Libraries** menu



Then in the Search box in the top right corner, start typing**Circuit Playground** and you'll pretty quickly end up with the **Adafruit Circuit Playground Library** showing up:



Click that **Install** button!


Library Manager
Type All
Topic All
circuit pl
Adafruit Circuit Playground by Adafruit
All in one library to control Adafruit's Circuit Playground board. All in one library to control Adafruit's Circuit Playground board.
Requires no other dependencies and exposes all Circuit Playground components in a simple to use class.
More info
Version 1....
Install

Library Reference

Thanks to caternuson for this reference while we work on this page(http://adafru.it/rb2)