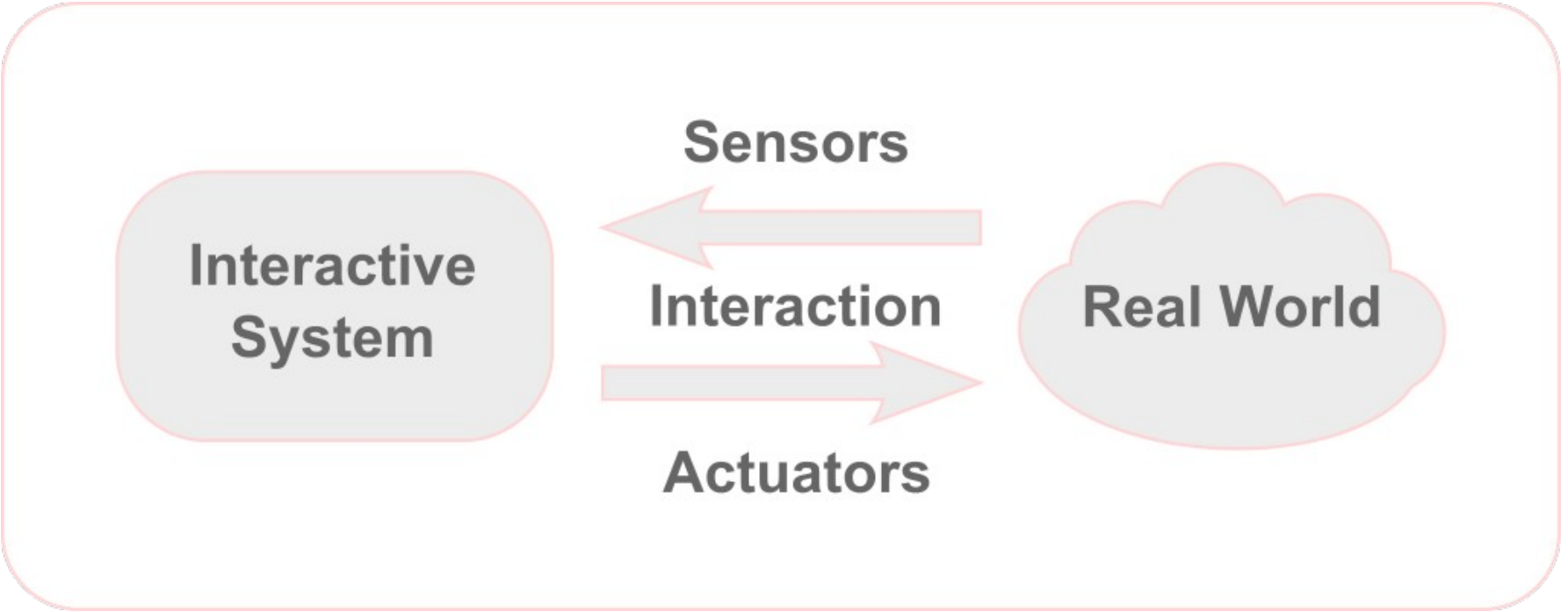
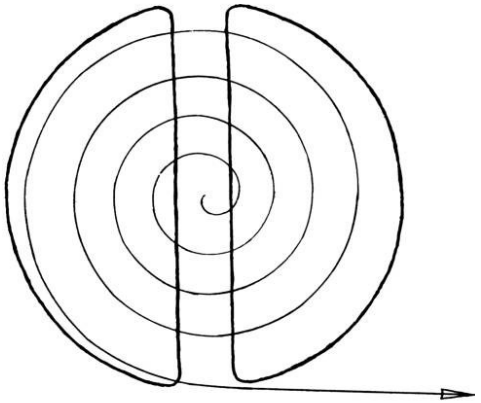
Introduction to Physical Computing

Ralph J. Steinhagen, CERN

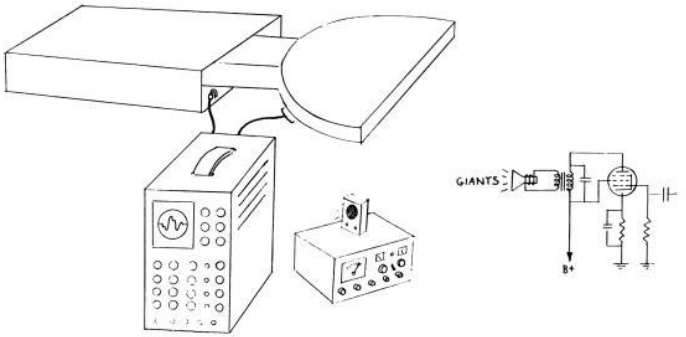


- Excellent online resources:
  - http://www.arduino.cc/
  - https://learn.sparkfun.com/tutorials
- Slide materials and acknowledgements: Linz Craig, Pete Lewis

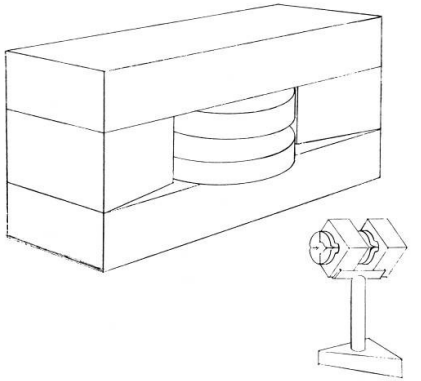
The Accelerator seen by …



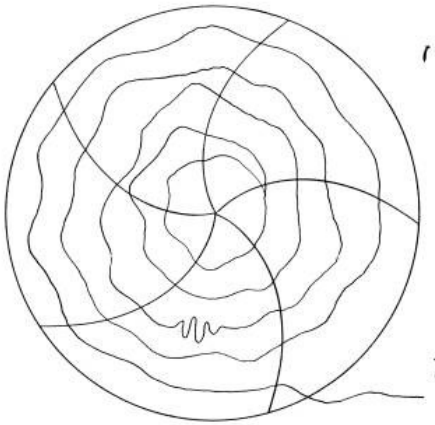
. . . the inventor



. . . the electrical engineer



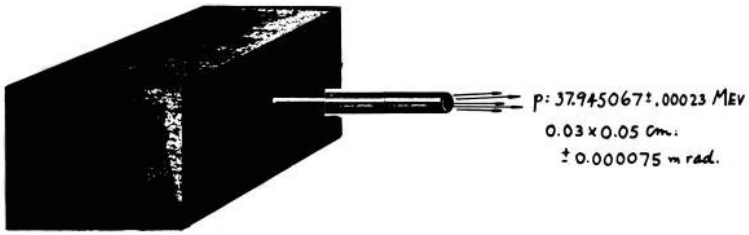
. . . the mechanical engineer



$$r = r_0\left[1+\left(\frac{fr\omega}{c}\right)\cos(3\theta+\delta_0+\delta_1 r)+\left(\frac{fr\omega}{c}\right)^2\cos(5\theta+\delta_2-\delta_5 r^3)+\left(\frac{fr\omega}{c}\right)^2\cos(7\theta+\delta_7-\delta_9 r^3)+\text{------}\right]\times\left\{\frac{e^{7/5}r^2\ln Z}{1+\left(\frac{a}{r}\right)^{7/4}}\right\}$$

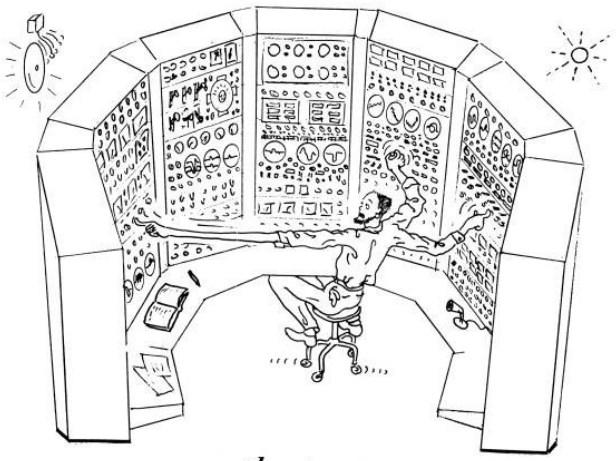
$$\frac{d\phi}{dt}=\left[\sin(\omega t-h\phi)-\sinh\beta-\tfrac{3}{5}fff_1f_2f_3'\right]\frac{eV_0}{2\pi\omega}$$

. . . the theoretical physicist

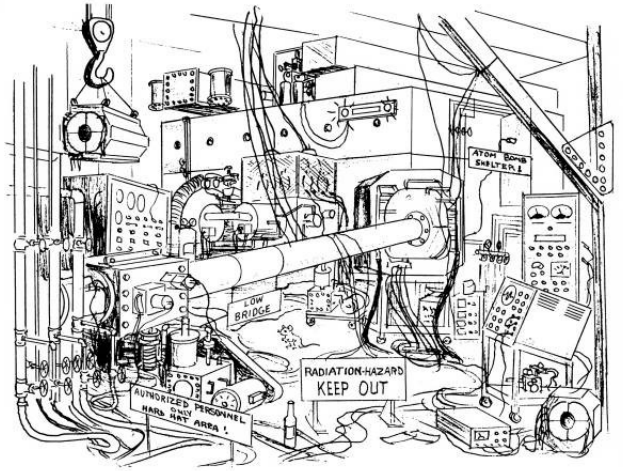


. . . the experimental physicist

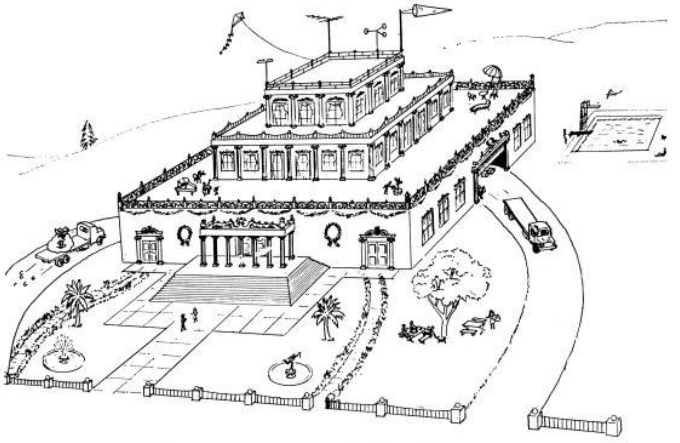
The Accelerator seen by …



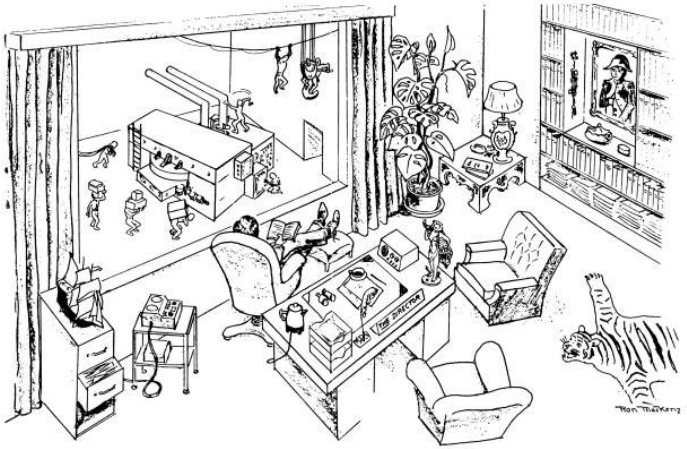
. . . the operator



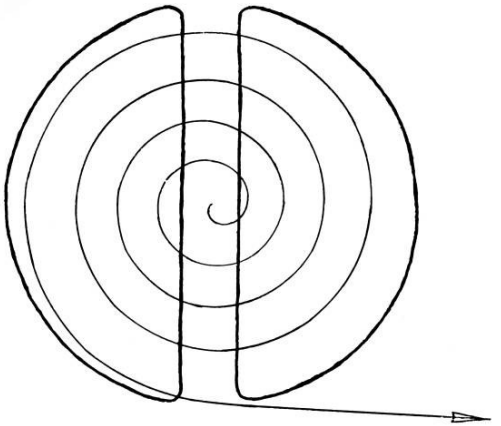
. . . the visitor



. . . the governmental funding agency



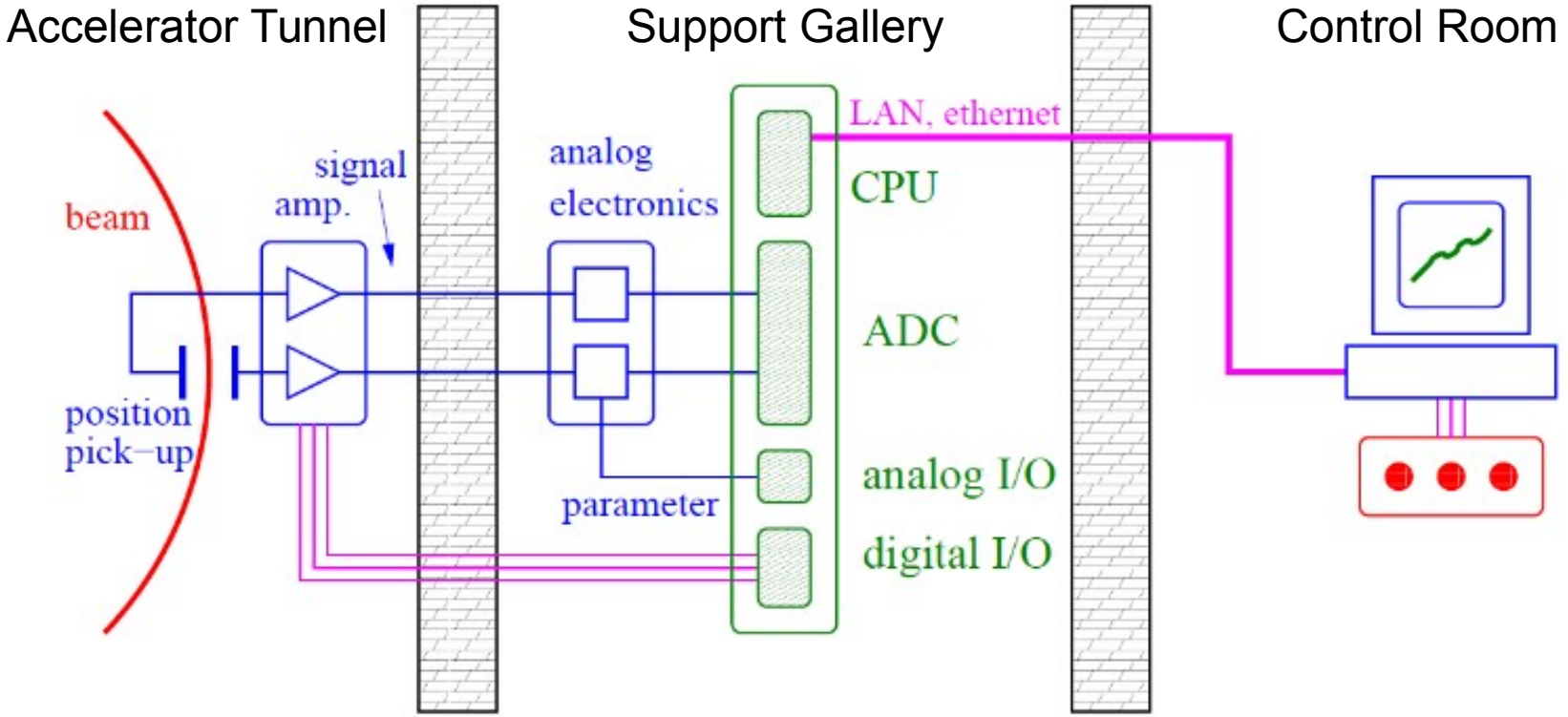
. . . the laboratory director



. . . the student

Cartoons: Dave Judd and Ronn MacKenzie

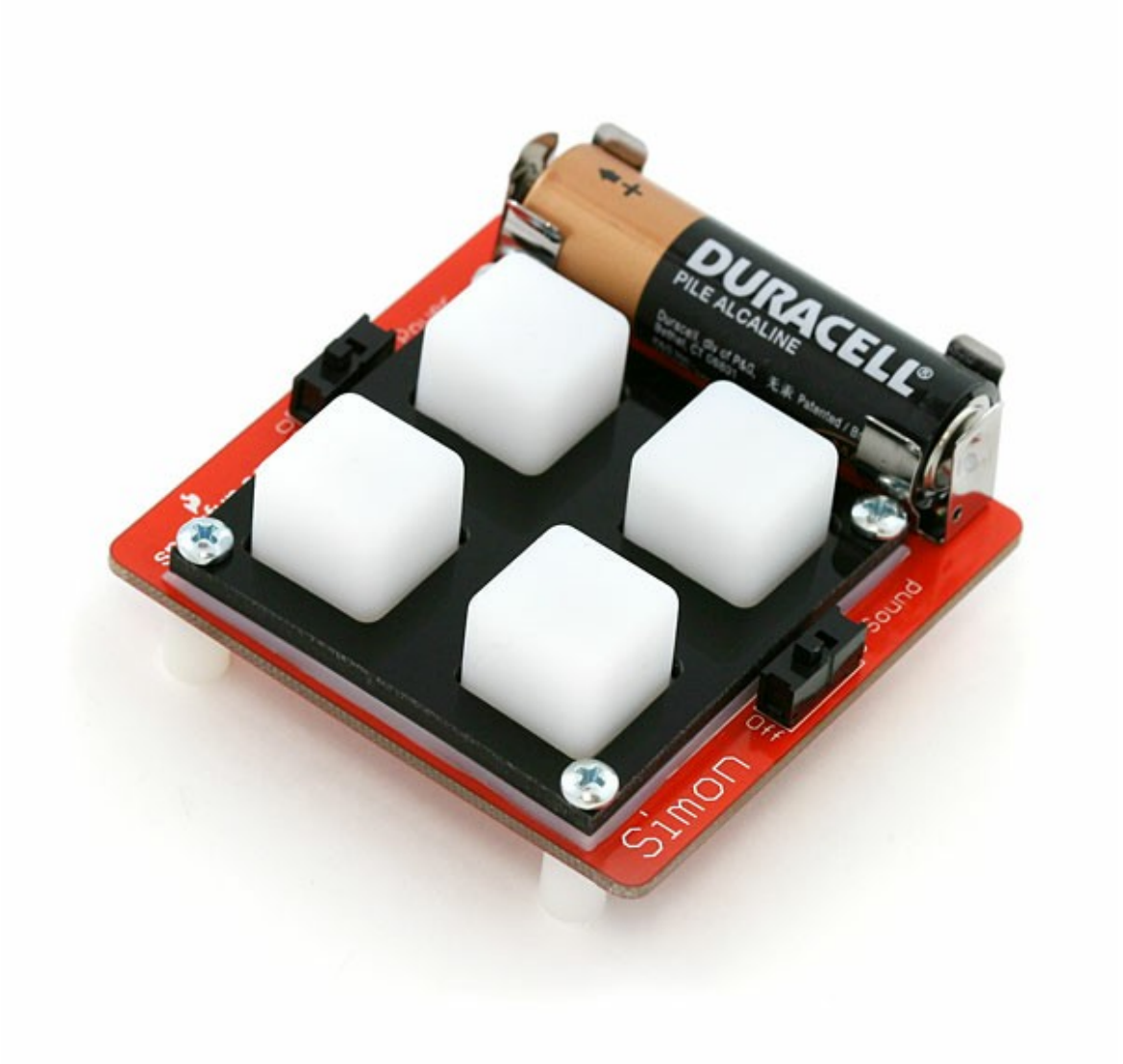
The Accelerator seen by … a Controls Engineer



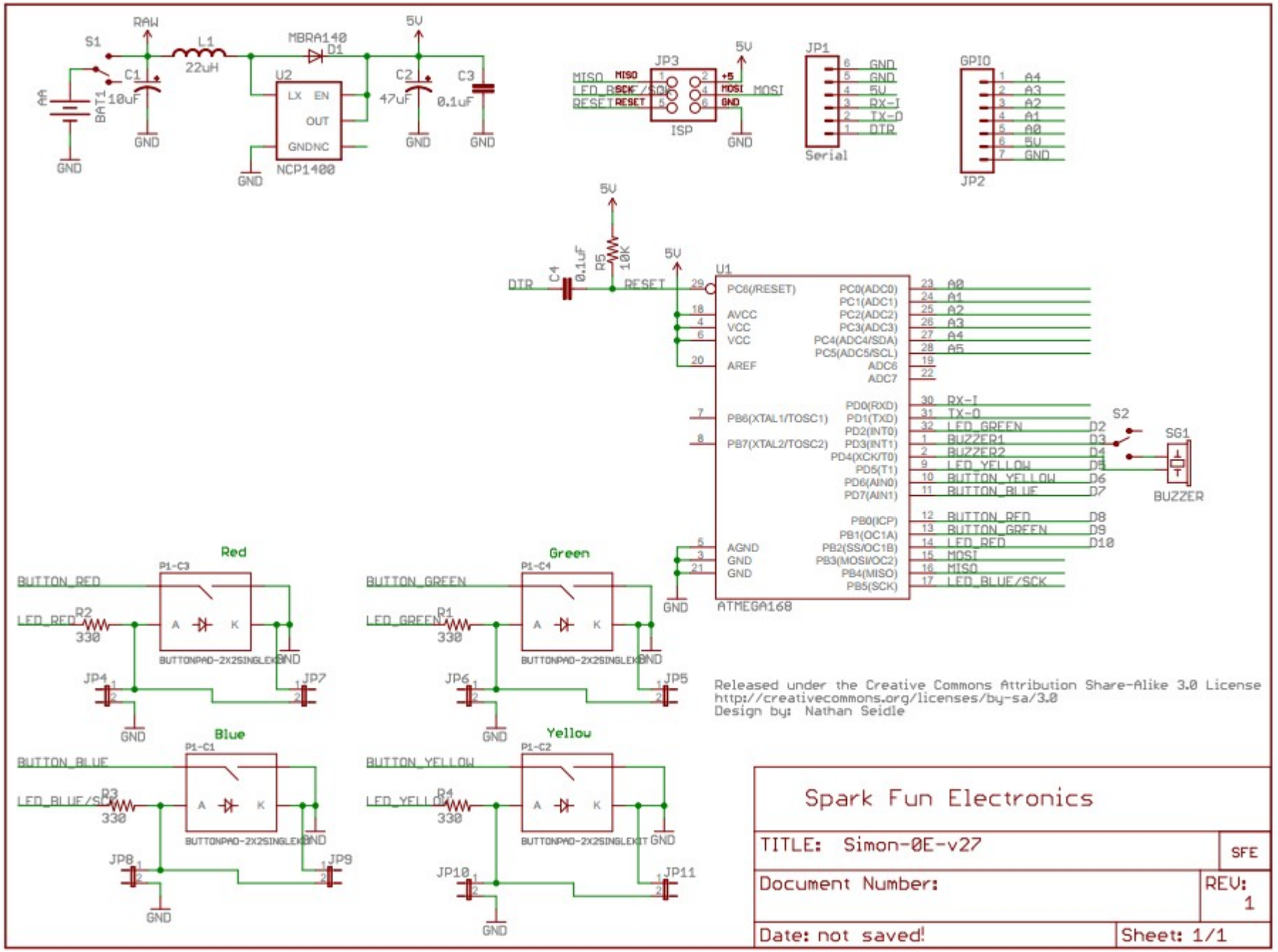
- Accelerator tunnel – mostly only analog signals
  - Pick-ups, HF electronics, signal pre-conditioning (aka. 'analog front-ends')
  - Inaccessible, noisy, radiation environment → need to keep stuff simple and robust
- Support Gallery – transition between analog and digital signals
  - controls racks, signal post-processing, digitisation, fast control (feedback) loops
- Control room – the accelerator brain – mostly digital
  - Supervision, automation, human-machine interfaces, ...

Image courtesy P. Forck

Your Interface between the Analog and Digital World

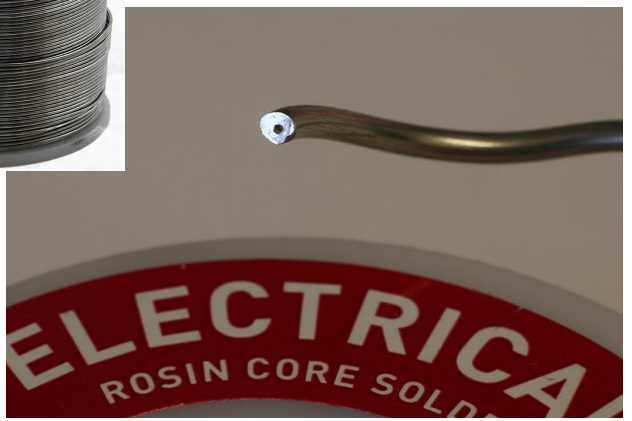


Simon Schematic



Solder & Soldering Ion

- … metal “glue” used to join metal parts
  - made of Sn-Ag-Cu, sometimes (/traditionally) Sn-Pb
    - wash hands before/after eating (which is anyway a sensible thing)
  - contains rosin flux
    - lowers metal surface tension (ie. easier adherence/creeping into gaps)
    - some seem to use extra flux pens to increase the flux



Use this to … → … melt this

Soldering tips

- … come in many shapes
  - often a personal choice but conical fits most applications
  - need to be kept clean

hoof conical chisel

Sponge

- Used to clean off soldering iron tip (often comes with a holder)
- Can be made of various materials
- Should be used every time before soldering a joint!

this needs to be wet!

Soldering – Addition Tools

- Additional tools used to handle components and PCBs

tweezers

clippers

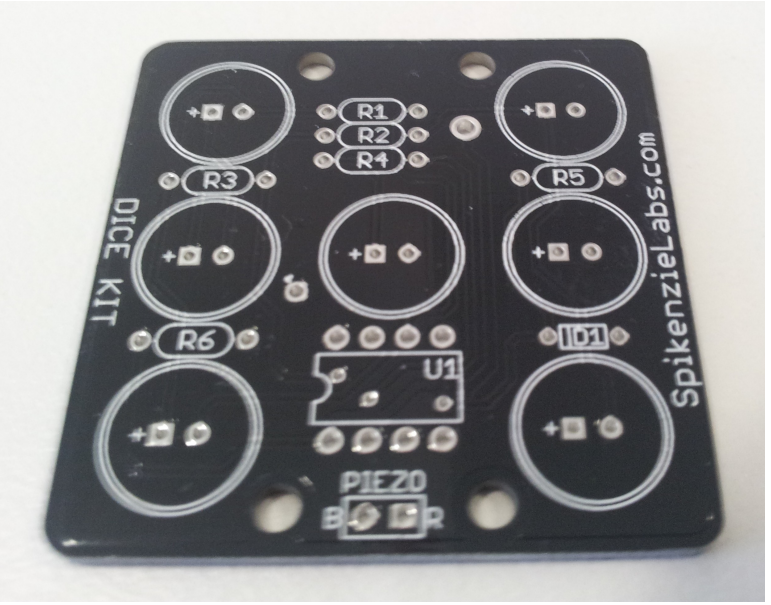
pliers

- Also useful:
  - Semi-fireproof work surface (wooden boards work just fine)
  - Eye-loop (10x): cost-effective option to see small defects
    – more needed when dealing with surface-mount-devices (SMDs)

Printed Circuit Board

- … mostly woven resin-enforced fibre glass and copper layers
  - Commonly one uses 'FR4' – 'flame-resistant epoxy resin' (weak standard)
  - Teflon & ceramics more common for RF applications (more stable properties)

- Has a 'top' and 'bottom', and often many intermediate layers
  - Solder stop (coloured varnish that prevents solder adhering to it)
  - printed silk-screen indicating and aiding component location and orientation

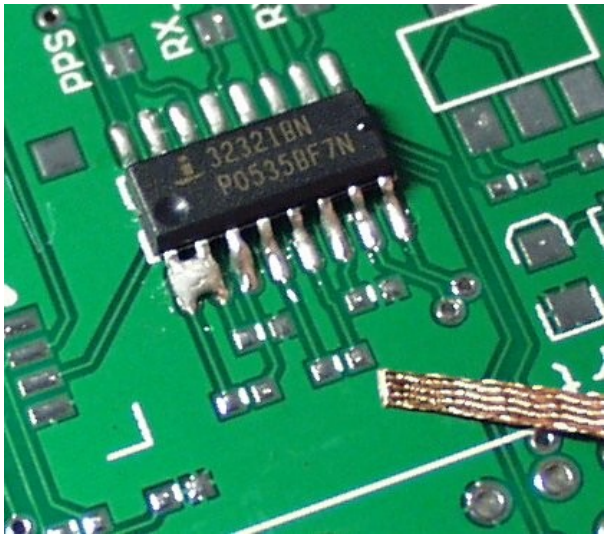


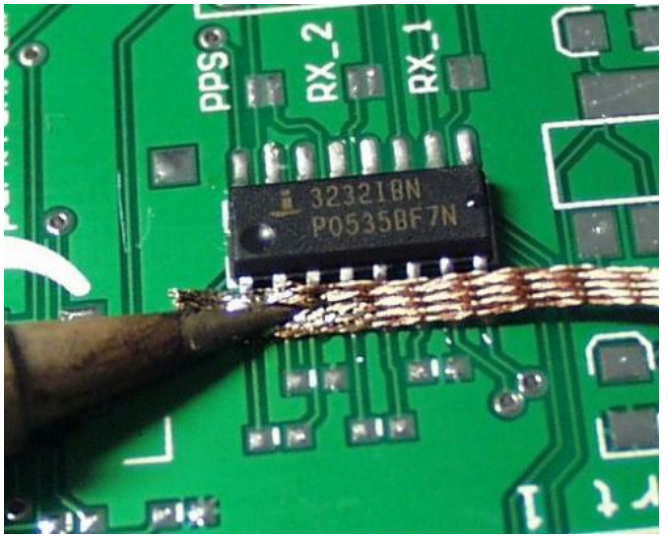
Trouble Shooting I/II – Multimeter

- … you can solder without!
- However, a great aid for SMD work to check/identify components values, short-circuits and general debugging (15 ↔ 60 AUD)

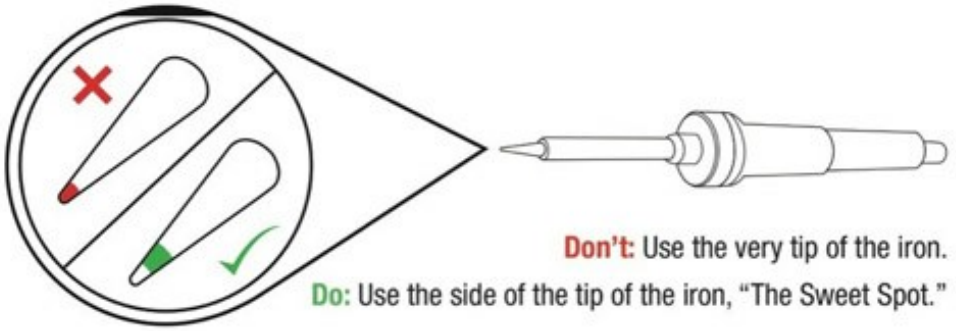
Trouble Shooting II/II – Soldering Wick

- Used to remove solder
  - (small braided wire have higher surface tension than component legs)
- Careful, this can get hot





Soldering – The Essentials



Don't: Use the very tip of the iron.

Do: Use the side of the tip of the iron, "The Sweet Spot."

Do: Touch the iron to the component leg and metal ring at the same time.

Do: While continuing to hold the iron in contact with the leg and metal ring, feed solder into the joint.

Don't: Glob the solder straight onto the iron and try to apply the solder with the iron.

Do: Use a sponge to clean your iron whenever black oxidization builds up on the tip.

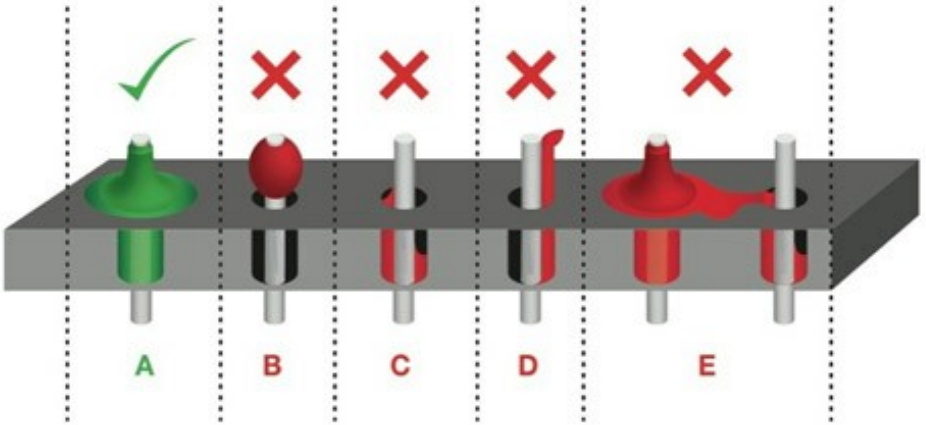
A Solder flows around the leg and fills the hole - forming a volcano-shaped mound of solder.

B Error: Solder balls up on the leg, not connecting the leg to the metal ring.
Solution: Add flux, then touch up with iron.

C Error: Bad Connection (i.e. it doesn't look like a volcano)
Solution: Flux then add solder.

D Error: Bad Connection...and ugly...oh so ugly.
Solution: Flux then add solder.

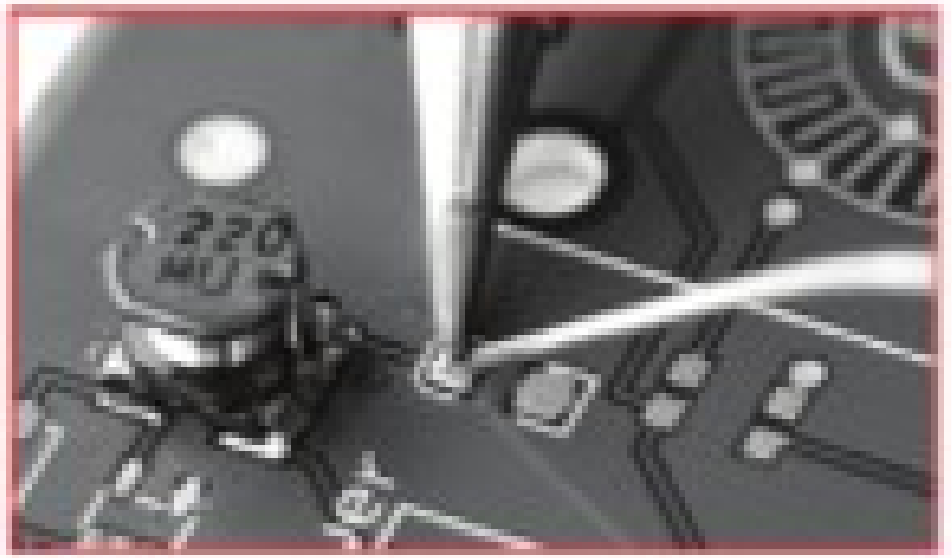
E Error: Too much solder connecting adjacent legs (aka a solder jumper).
Solution: Wick off excess solder.



Some Simple Soldering Rules

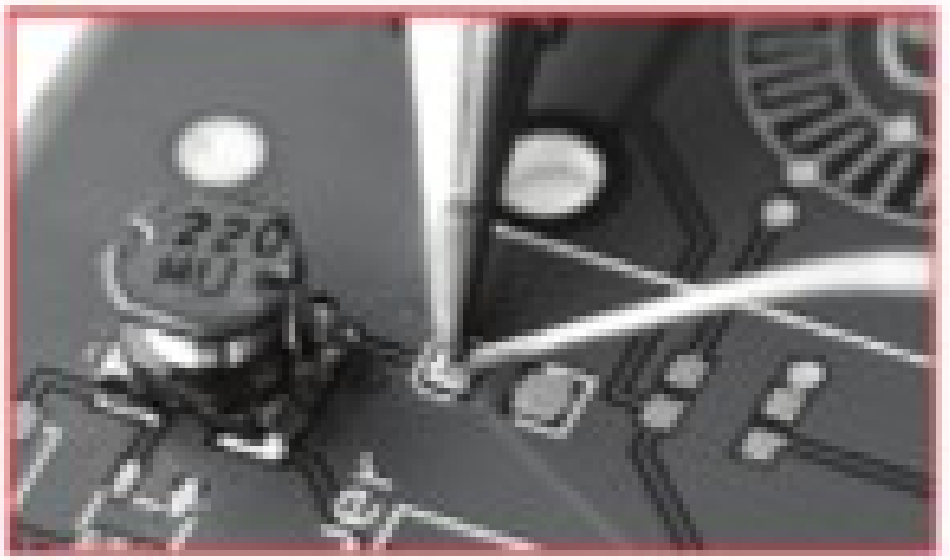
- Soldering is like gluing but with hot metal
  - Pick-up the soldering ion at the right end
  - Metal parts conduct heat – watch what you are touching
- Keep your soldering tip clean → use your wet/brass sponge
- Always solder first one pin → adjust/check polarity → solder other(s)
- Start with the lowest profile component
- Apply heat to the pad and pin and move the solder to the pin (not the ion)
- Check (at appropriate intervals) whether your joints are 'short-free' i.e. not only at the end → this helps to isolate potential problems early

SMD – Perfect Solder Joint I/IV



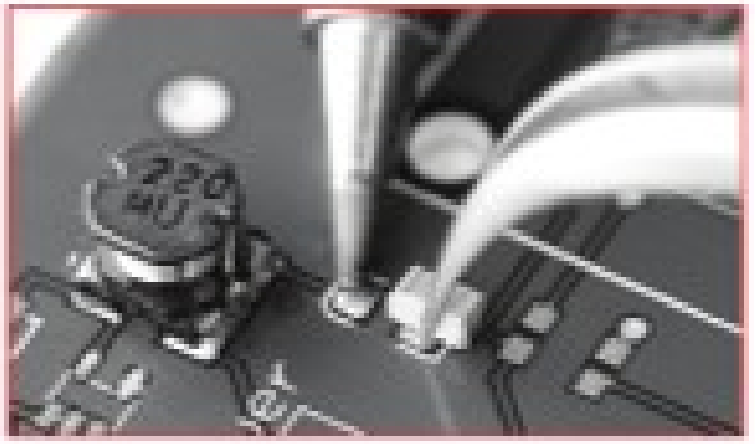
- Hold the iron touching one of the pads for a count of one
- Make sure to touch the side of the iron tip to the pad, not the actual tip of the iron

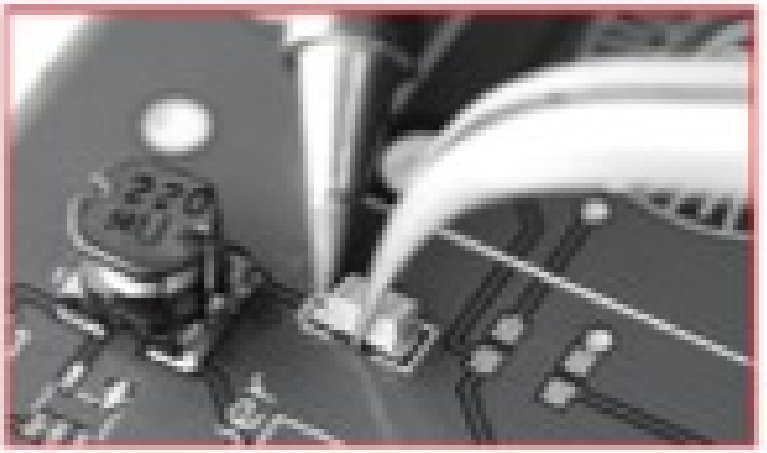
SMD – Perfect Solder Joint II/IV



- Feed a tiny bit of solder onto the pad while continuing to hold the iron to the pad
- Feed enough solder onto the pad to cover the pad, but no more

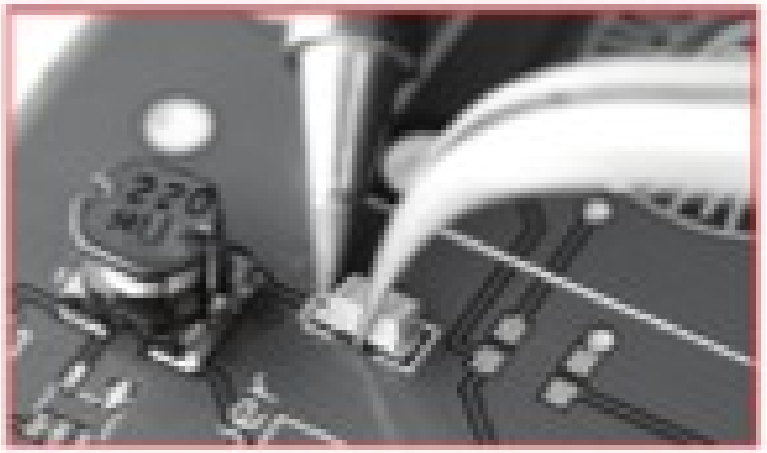
SMD – Perfect Solder Joint III/IV





- Continue to hold the soldering iron on the pad and pick up your component using the tweezers with your other hand
- Make sure the tweezers are making contact only with the sides of the component that do not have leads, otherwise you may get your tweezers in the solder

SMD – Perfect Solder Joint III/IV



- Continue to hold the soldering iron on the pad so the solder stays liquid
- Slide the component into the solder from the side while keeping the bottom of the component flat on the PCB
- Make sure you slide the component far enough into the pad so that the pad on the other side of the component is at least half exposed

SMD – Perfect Solder Joint III/IV

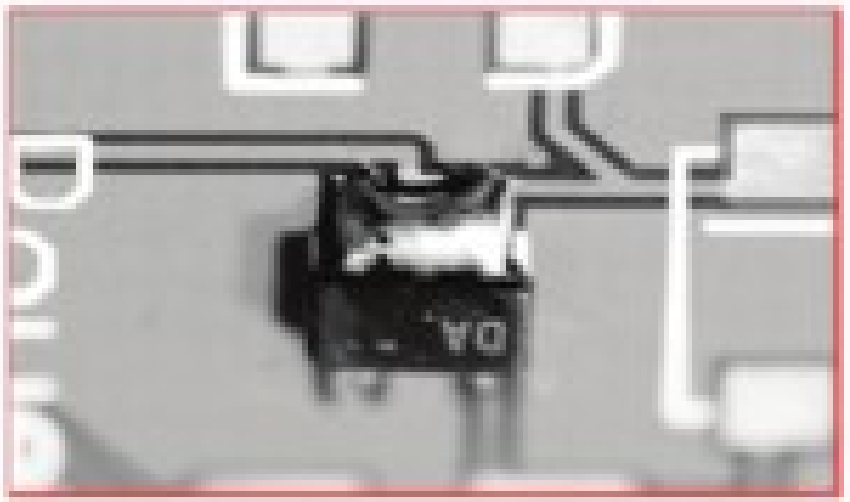
- Continue to hold the component on the pad with your tweezers and remove the soldering iron, wait for the solder to solidify and then remove your tweezers

SMD – Perfect Solder Joint III/IV

– Oops –

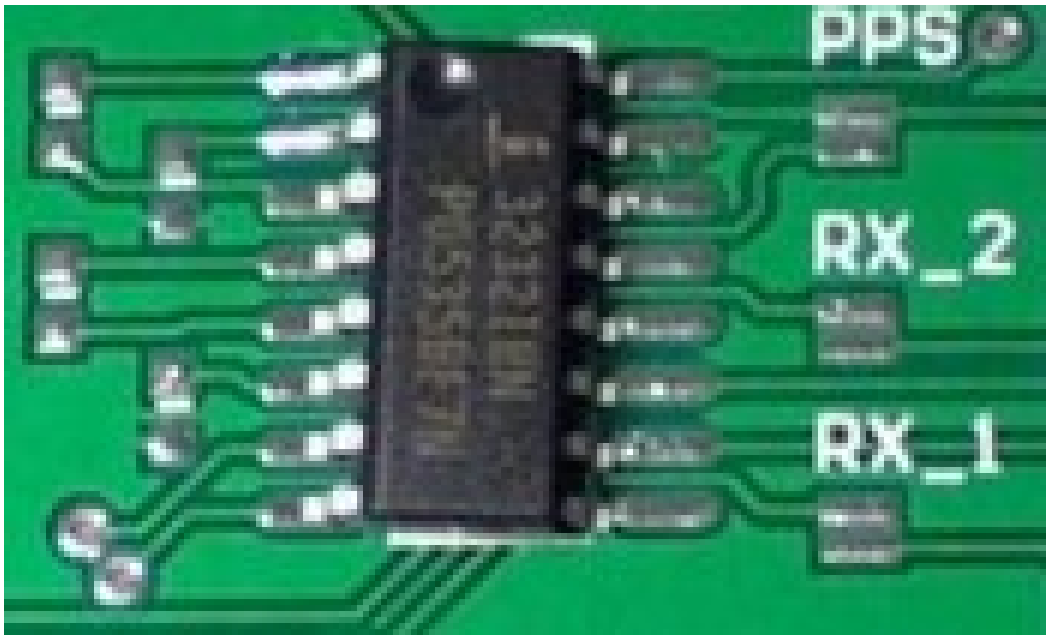
- If you don’t slide the component in from the side you may wind up with a component that does not touch pads on both sides
- If this happens just reflow the solder on the first join and replace the component

How to Use Solder Wick



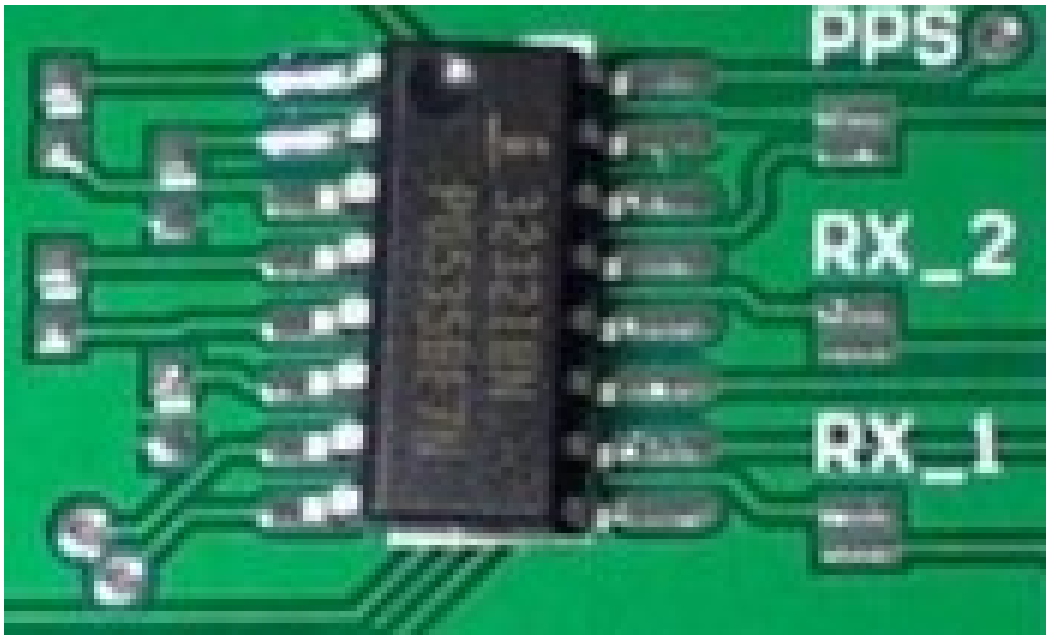
- Place the solder wick on top of the solder you want to remove
- Sandwich the solder wick between the solder and your iron until the heat sucks the solder into the wick
- The solder will flow into your wick and the wick will turn silver with solder
- This may take a while, be patient and apply constant contact with the iron
- Try rolling the soldering iron along the wick if you're really having trouble
- You might also want to try cleaning your iron if it isn't working

ICs are Tricky!



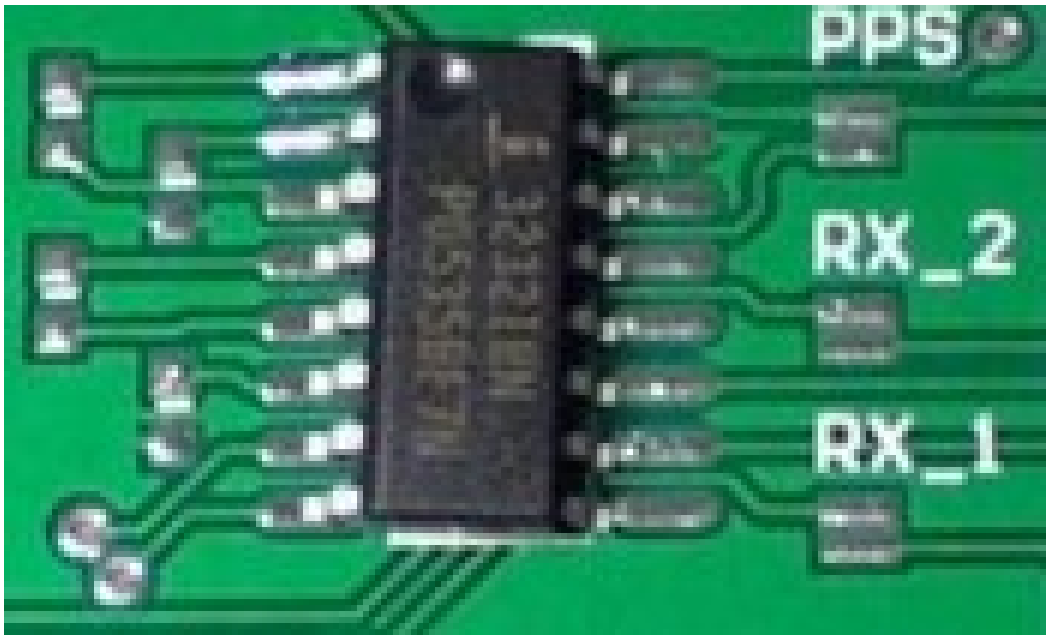
- Tack down a corner lead of your IC onto the corner pad
- Do this just like your other components
- Don’t worry if you accidentally solder two leads (you can clean it up with wick later)

ICs are Tricky!



- The most important thing is to make sure that the IC is flat and all the leads line up with the pads

ICs are Tricky!

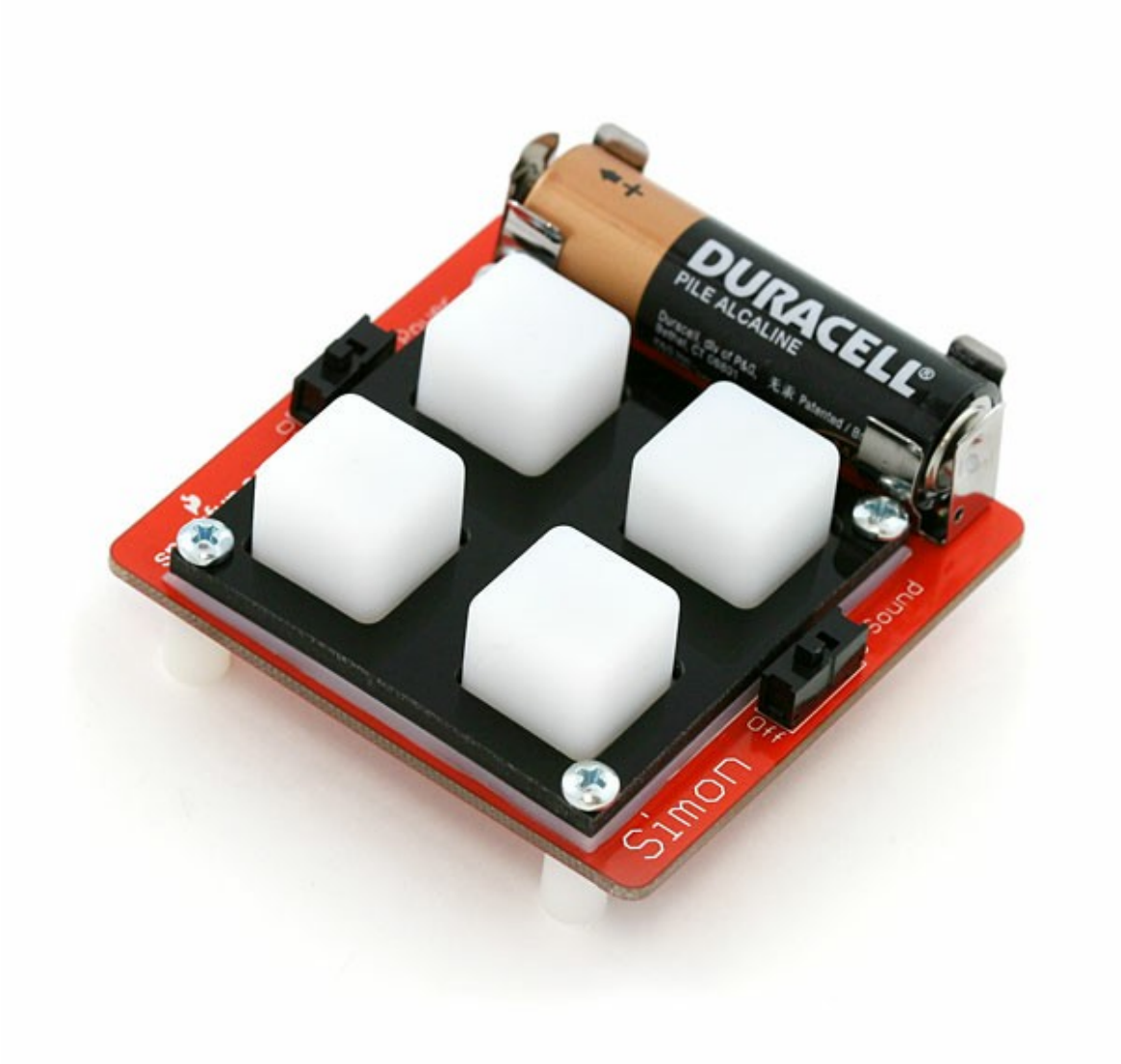


- To keep the IC in place, next solder the lead on the bottom right corner (or the corner diagonal from your first solder joint)
- After that put solder on the rest of the leads and pads
- Don’t worry if you wind up with a lot of jumpers, you can clean them up later

Cleaning Up an IC

- Some people will create jumpers on the IC leads on purpose and clean them up with wick later
- You can also apply flux to the jumper and if there is not too much solder, push the iron tip between the IC leads. This will melt the jumper that is connecting the two leads
- You can also use the solder's tendency to be drawn towards heat to push or pull the solder along the IC leads to a corner lead and then use wick to remove the excess solder
- To do this you will need to touch at least two leads at once with your soldering iron, constantly clean your iron and use a lot of flux

Let's get started



Tweaking your Simon

- First go the Arduino website: http://www.arduino.cc/
  - Then click on Download and follow the instructions

- Uploading different code onto the Simon
  - Disco Mode Code found: www.sparkfun.com/tutorials/203
  - Adding a photocell and using Disco Mode

Hardware

You will need the following:

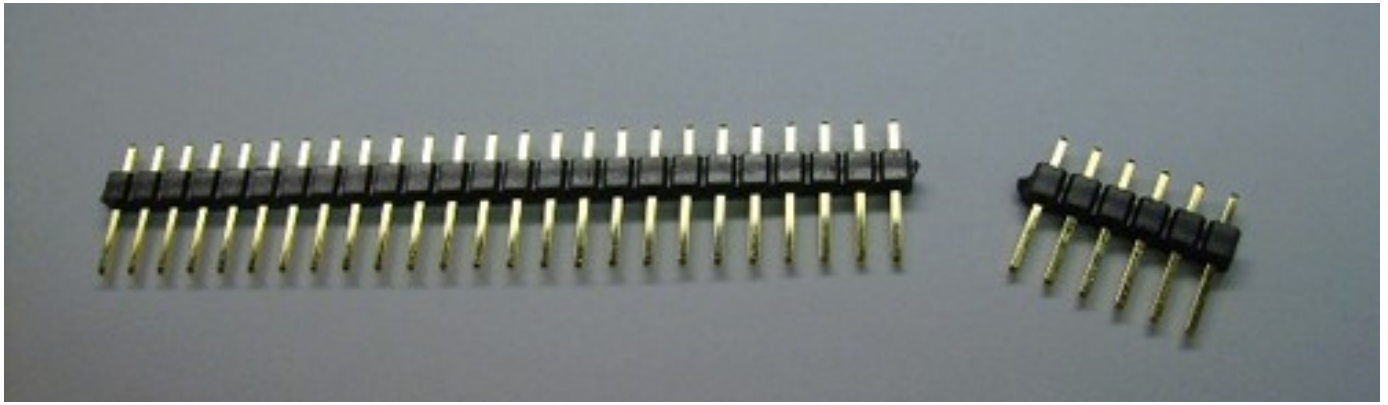
A USB cable, an FTDI Breakout Board,

and six pins of Break Away Headers

Break Away Headers

Headers come in strips of forty

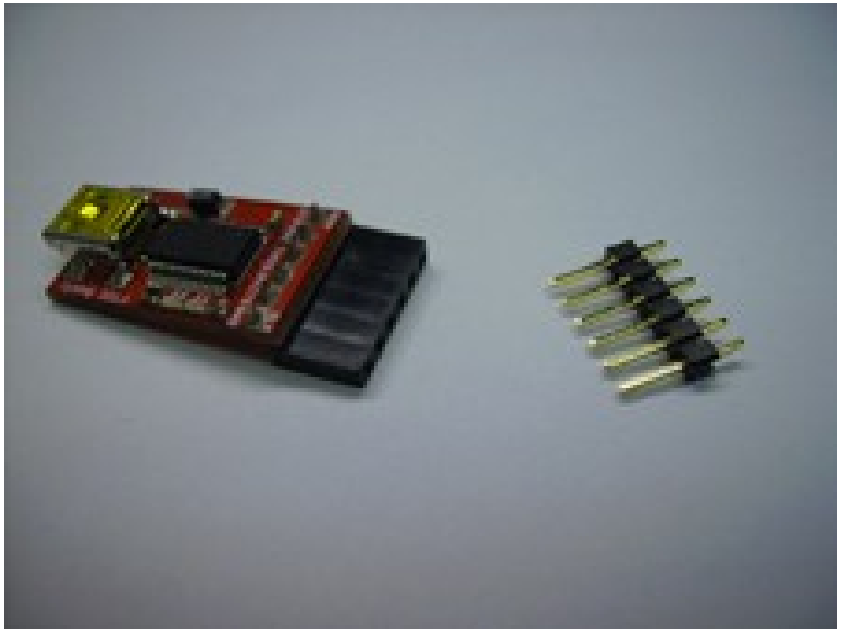
We just need six for the FTDI Breakout Board so clip off six of the forty by cutting through the seventh pin



FTDI and Headers

Attach the FTDI Basic Breakout Board and Headers

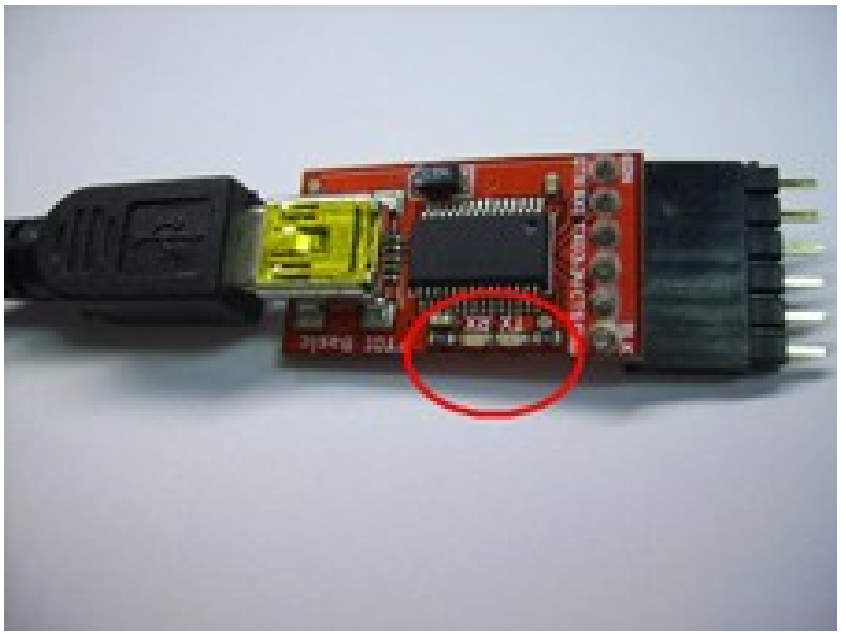
Make sure you plug the long side of the headers into the FTDI Breakout Board:



Plugging it All into the Computer

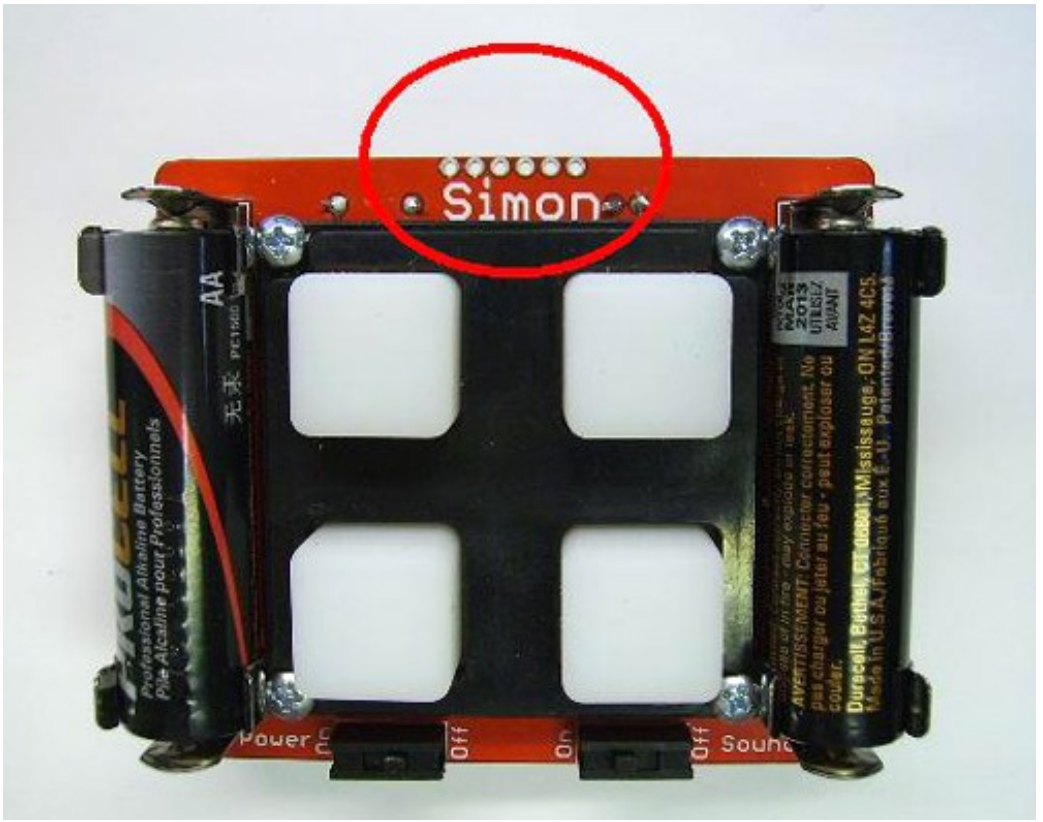
Plug the USB cord into the computer and then into the FTDI Board

The TX and RX LEDs will blink as you plug the FTDI in, this means the FTDI is talking to your computer



Plugging it All into the Simon:

Plug the Headers on the FTDI Board into the Programming Port on the Simon

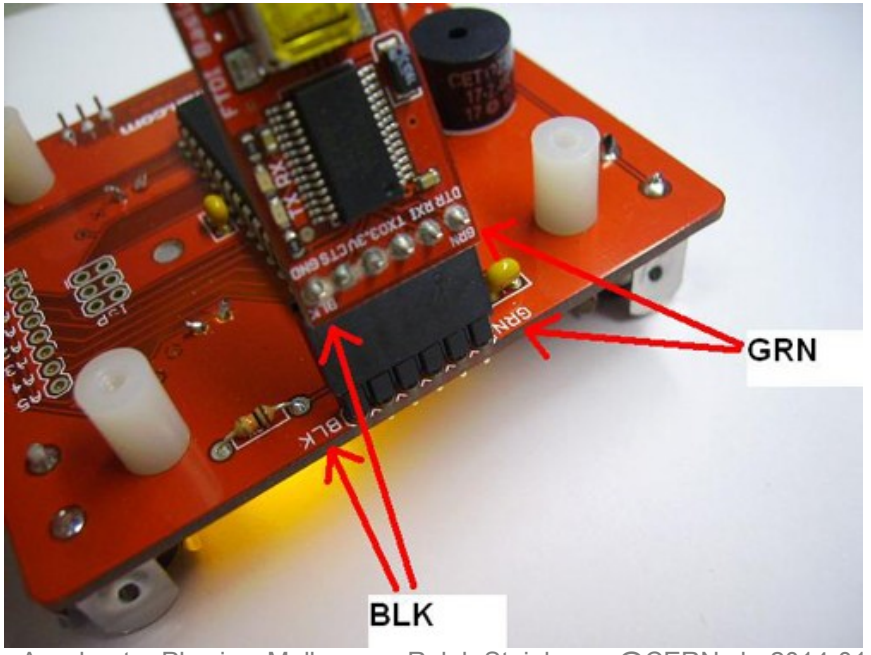


Plugging it All into the Simon

Orient the FTDI properly,

GRN goes to GRN and BLK goes to BLK

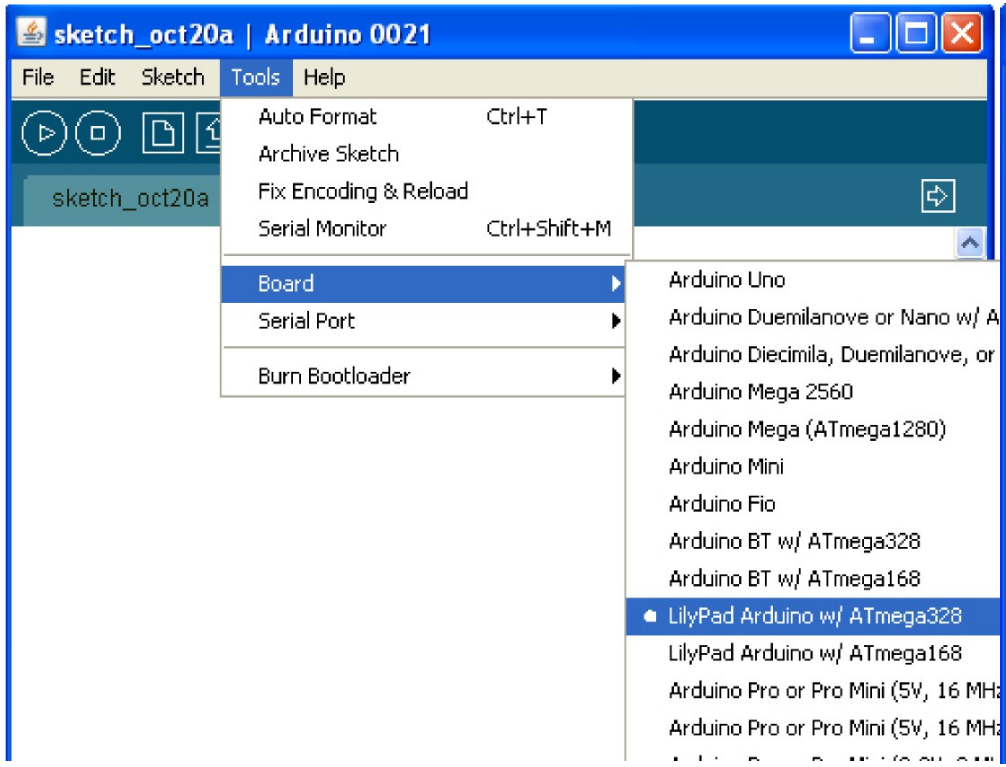
Also make sure to hold the FTDI at a 45 degree angle so the Headers make contact with all the ports

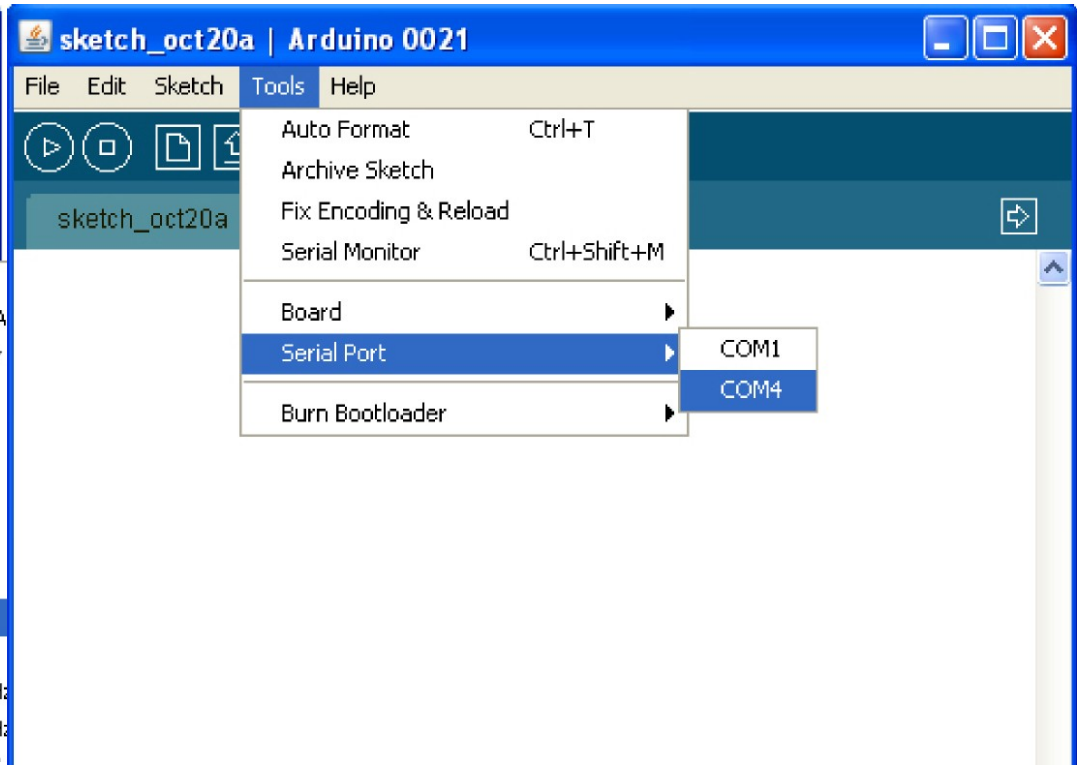


Before uploading code onto the Simon

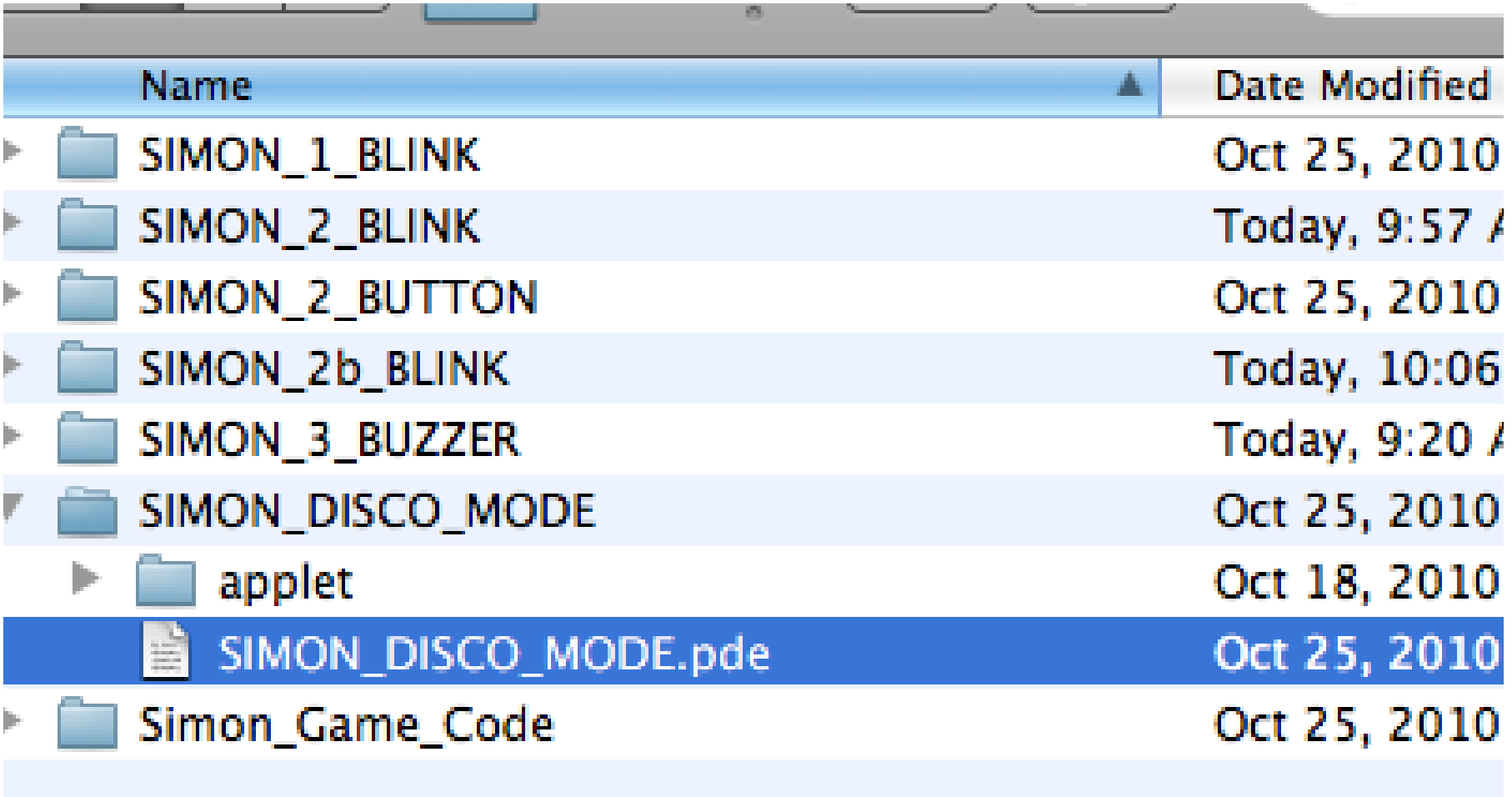
Open the Arduino Environment

Choose your board type and serial port





Find the Disco Mode Code
(It should be in a folder named SimonProgramming)



Uploading Code onto the Simon

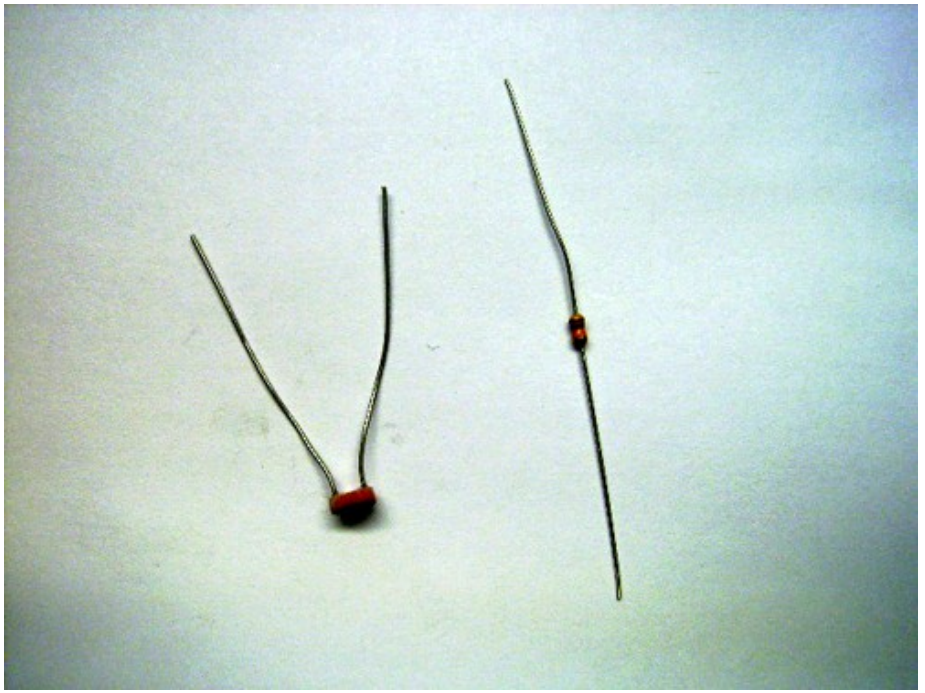
We're finally ready to upload code

Open the Simon_Disco_Mode sketch in the Arduino Environment and click Upload

Adding a Photoresistor – Analog Input

We're ready to add a sensor to your Simon!

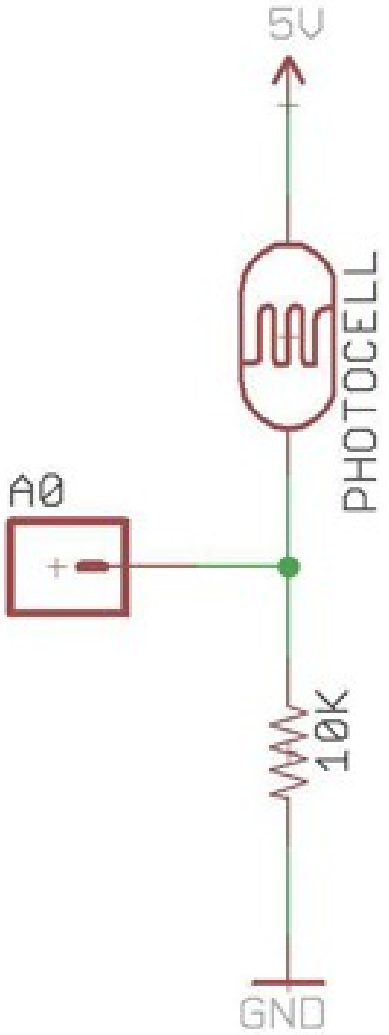
You'll need a photoresistor and a plain old 10K Ohm resistor:



Here's the Schematic

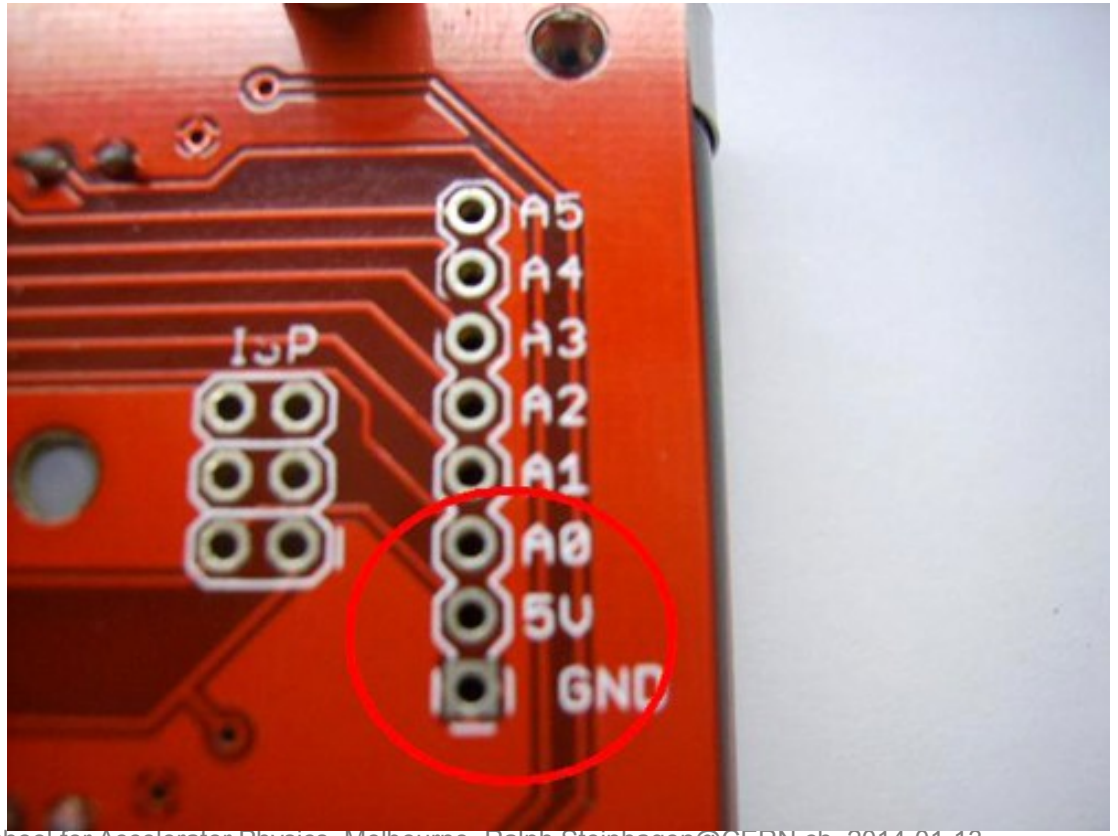
This schematic shows
the three connections
we will have to make

5V (or VCC), GND and A0



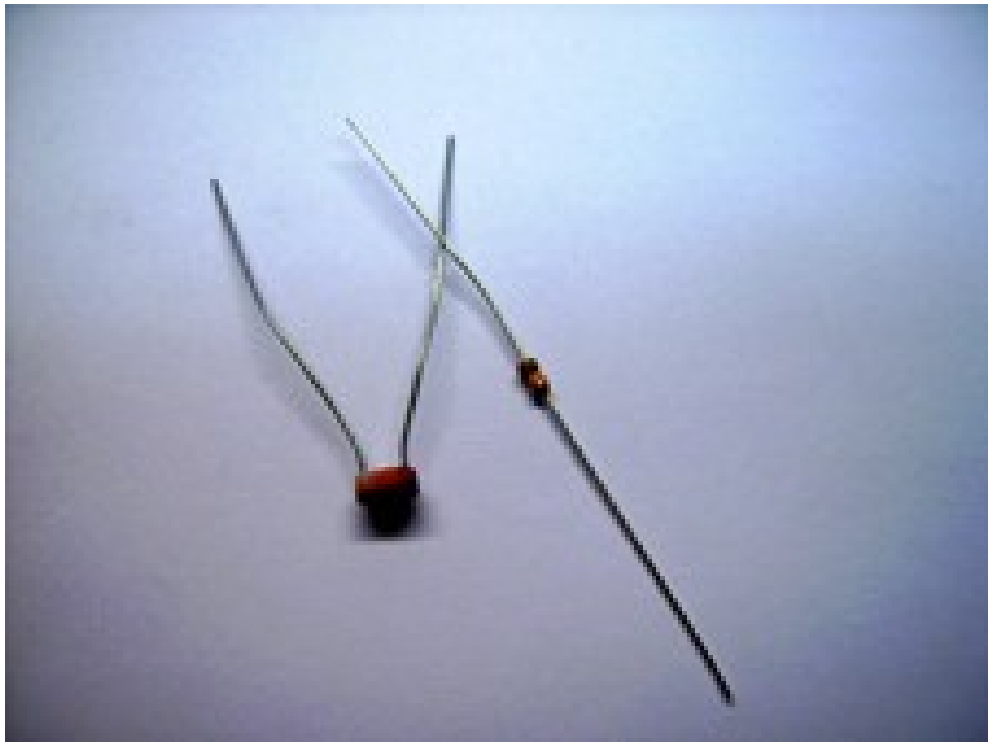
The Simon Analog Pins

The three holes (5V (or VCC), GND and A0) you will connect to the photoresistor circuit



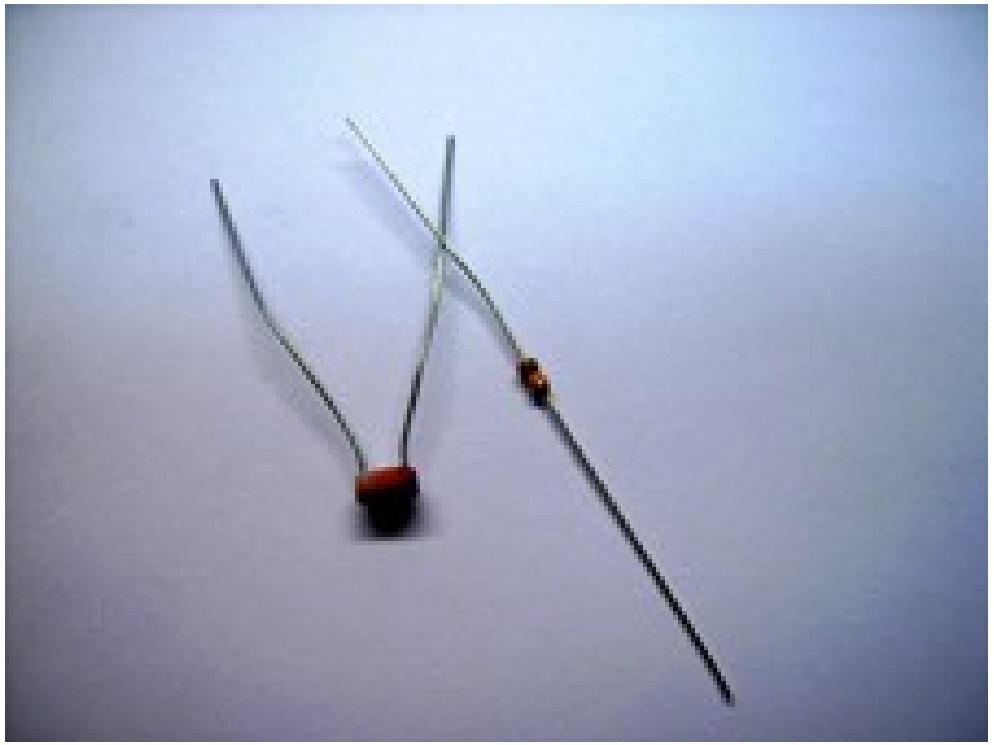
Creating the Circuit

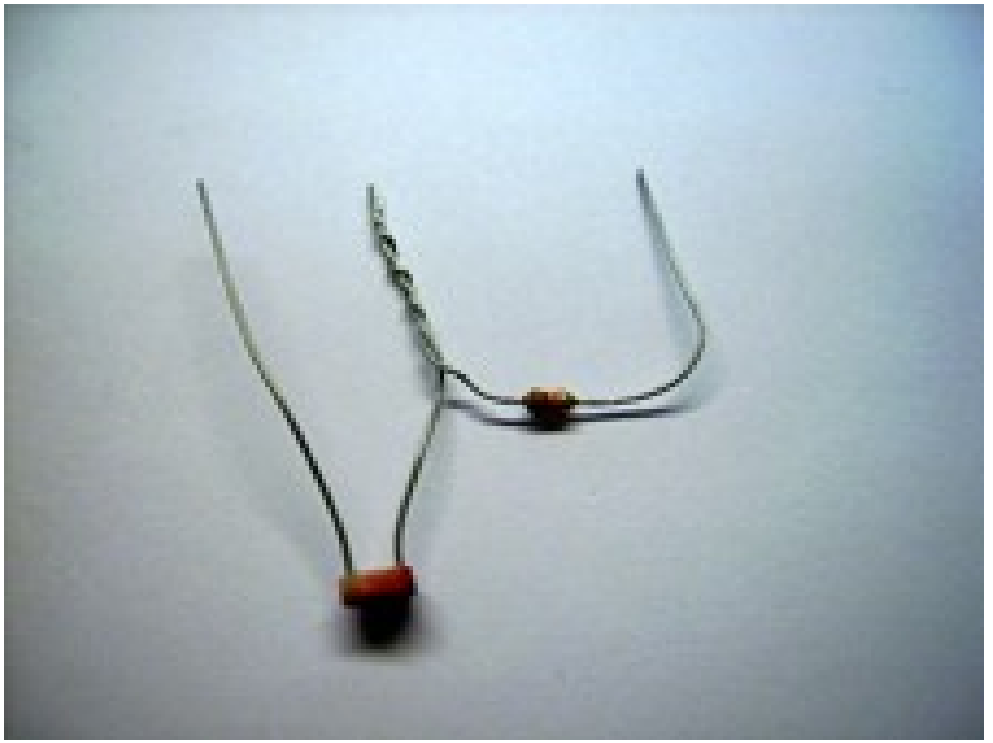
Make the photoresistor and regular resistor look like the schematic



Creating the Circuit

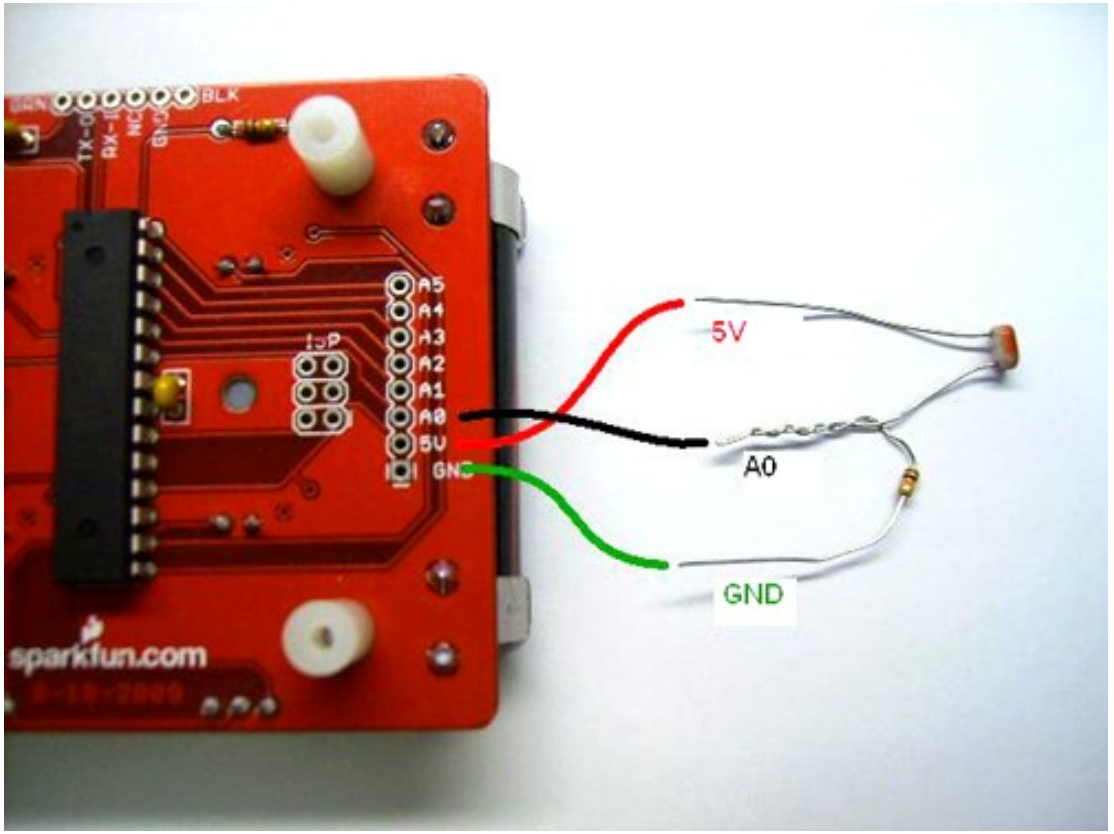
Make the photoresistor and regular resistor look like the schematic





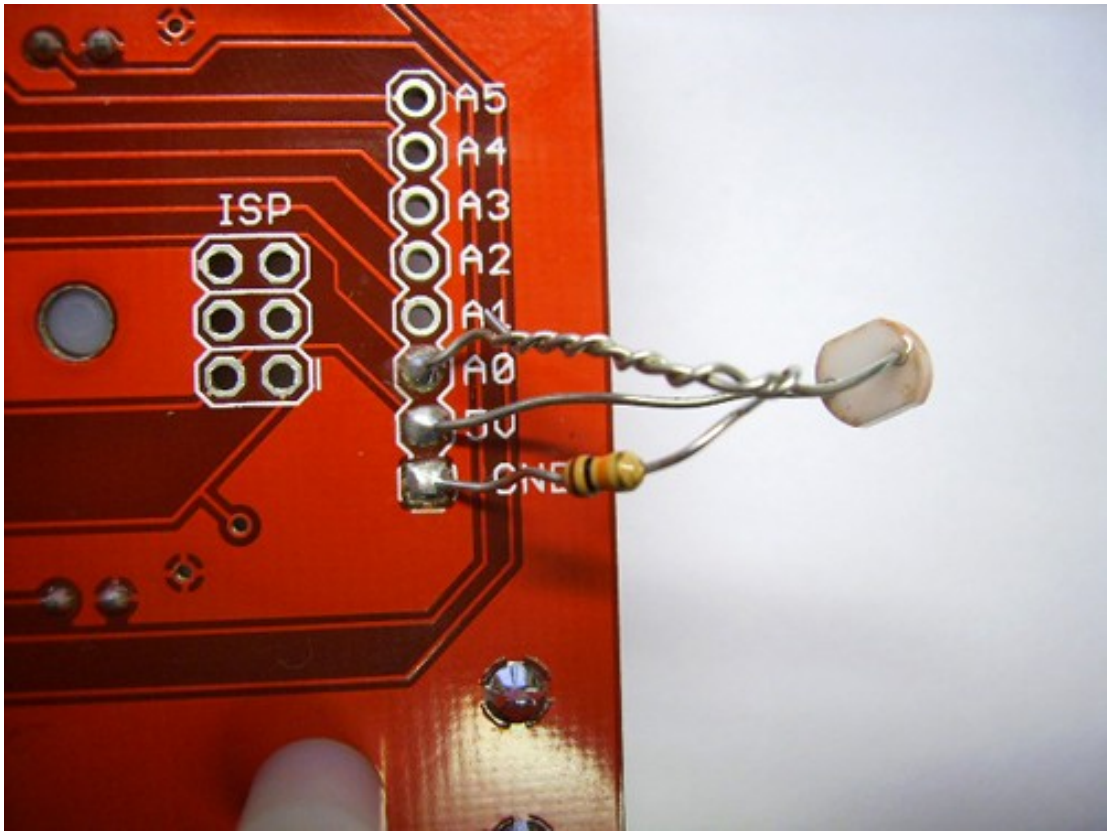
Connecting the Circuit

Just prior to soldering your photoresistor circuit should look like this



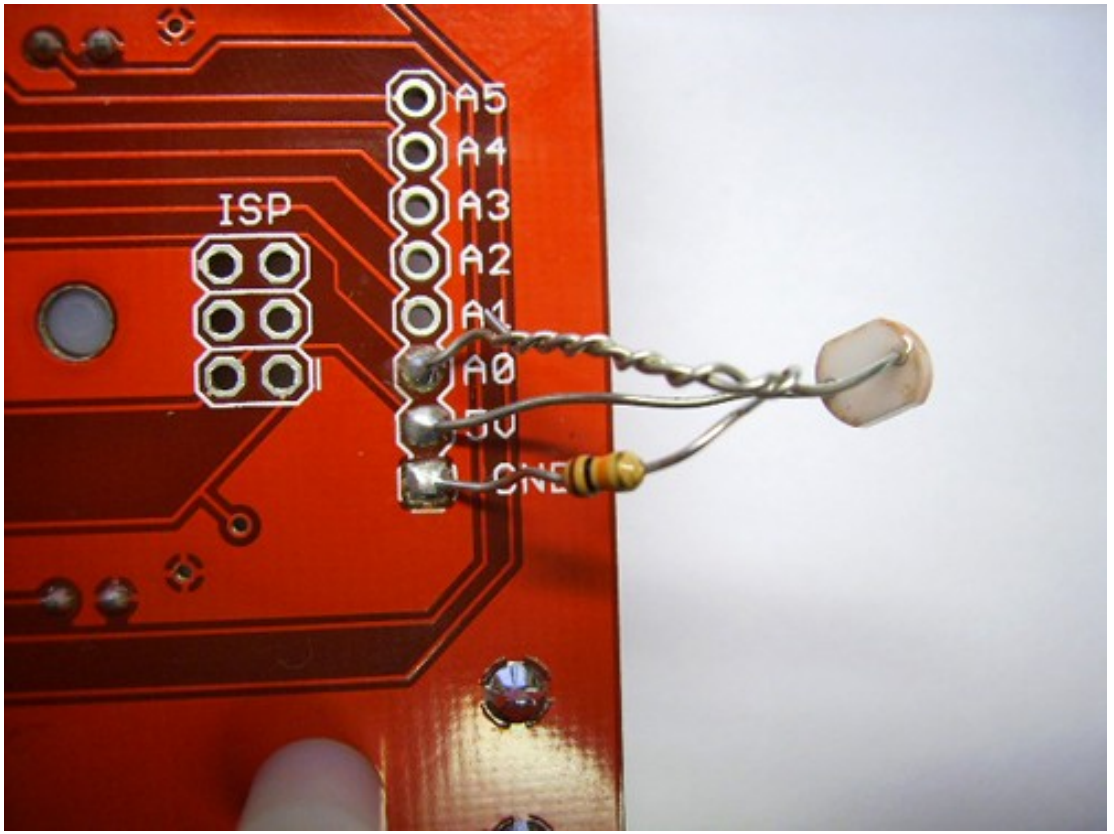
Connecting the Circuit

Just after soldering your photoresistor circuit should look something like this



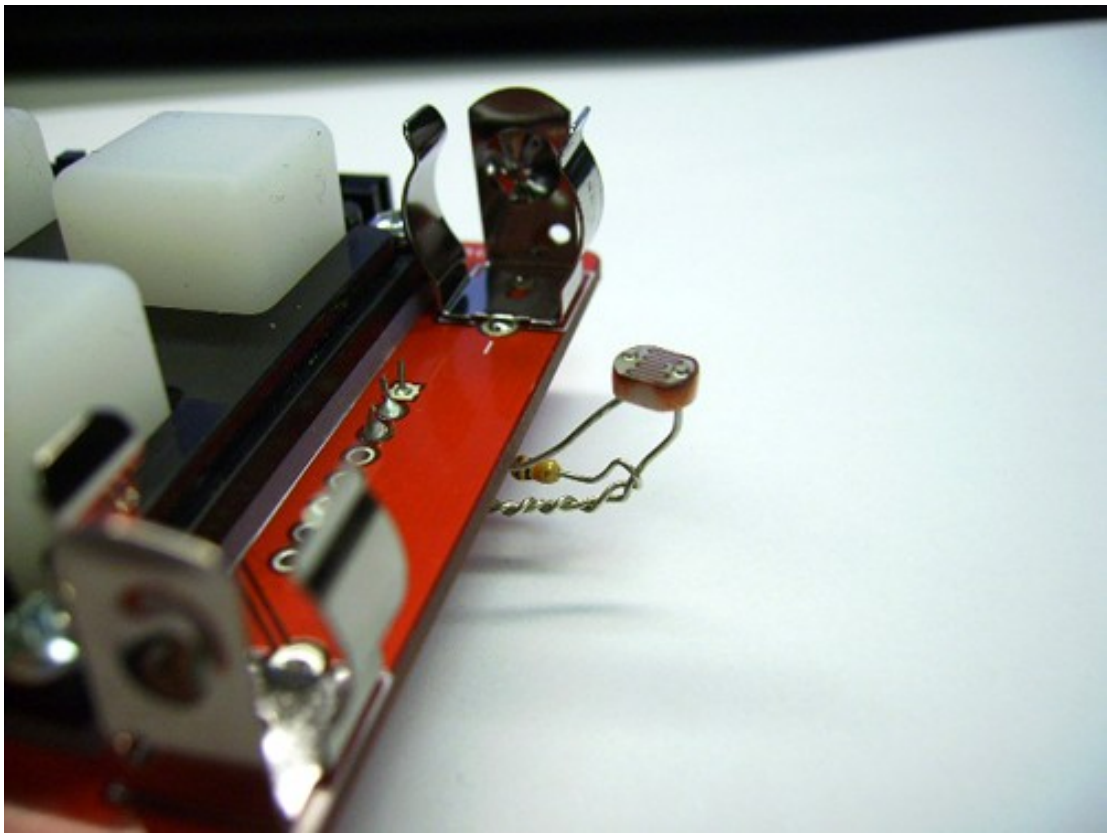
Connecting the Circuit

Just after soldering your photoresistor circuit should look something like this



Bending the Sensor

Bend the photoresistor around so it faces your light source



Insert Your Batteries and Bask in the Glory of Disco Mode

Analog Input

Now let’s talk about how you write code to use your Analog Input.

We’ll talk about variables and setup later, for now let’s talk about the Loop function and how you get Analog Input into your code and use it for an interface.

SIMON_DISCO_MODE

```
void loop()  {
  int light;
  light = (analogRead(0)/4);
  if(light < 150)
  {
  beegees_loop();
  }
  else
  {
    for (int thisPin = 0; thisPin < pinCount; thisPin++)  {
    digitalWrite(leds[thisPin], LOW);
    }
  }
```

This second line in the Loop function is where the variable “light” is set to whatever the analog pin # 0 reads divided by 4.

This variable is then used in the rest of the code to decide if the Simon should play music or turn the LEDs off.

Now for Digital Input, Digital Output & Analog Output

We will use the following sketches to discuss each of these topics, so feel free to upload the first sketch to your Simon:

Digital Input: Simon_2_BUTTON

Digital Output: Simon_2_BLINK

Analog Output: Simon_2_BLINK (lines commented)

Digital Input

Upload the Simon_2_BUTTON sketch to your Simon.

Feel free to change the ledPin variable to any of the other LEDs if you want a different color, just remember to change your button variable as well.

SIMON_2_BUTTON §

```
int ledPin =  3;    // The simon board has 4 LEDs on it.
                    // For this example, we're just going to use one.
                    // The other LEDs are on pins 3,5,10 and 13.
                    // For fun, try switching "ledPin" to another pin number and see what h
int buttonPin =  2; // The simon board has 4 BUTTONS on it.
                    // For this example, we're just going to use one.
                    // The other BUTTONS are on pins 2,6,9 and 12.
                    // For fun, try switching "buttonPin" to another pin number and see wha
int button_state;   // This variable will be used to "store" the state of the button.
                    // It will allow us to know whether the button is pressed or not.
```

This pinMode line sets the ledPin to Output.

First Arduino needs to declare variables to keep track of three things: The LED this sketch will light up, the button it is waiting for you to press and whether the button has been pressed or not.

SIMON_2_BUTTON §

```
void setup()   {
  // initialize the led pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the internal pull-up on the button pin:
  digitalWrite(buttonPin, HIGH);
  // initialize the button pin as an input:
  pinMode(buttonPin, INPUT);

}
```

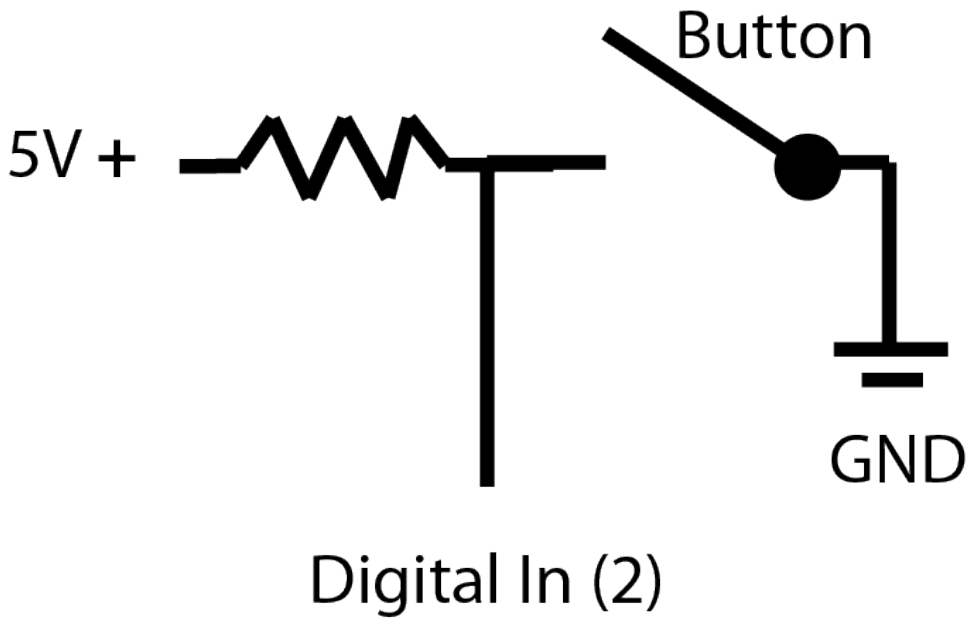
The first pinMode line sets the ledPin to Output.

The second digitalWrite line is necessary to use the Internal Pull-up Resistor (more on this next slide).

The third pinMode line sets the buttonPin to Input.

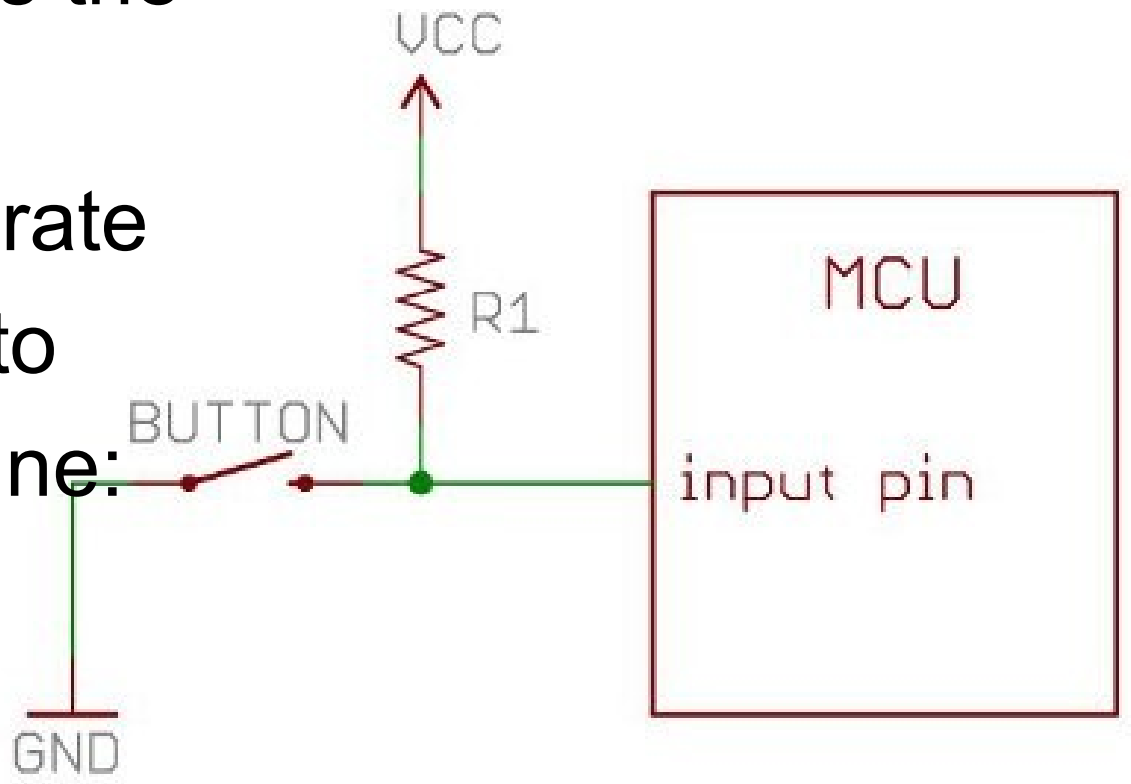
Pull Up Resistors Explained

A regular pull up resistor looks like this:



Pull Up Resistors Explained

But the internal pull up resistor in the arduino moves the power source from the separate power line onto the input pin line:



SIMON_2_BUTTON §

```
void loop()
{
  int button_state = digitalRead(buttonPin);

   if(button_state == 1){
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  }
}
```

The first line inside the loop function sets the button_state variable equal to whatever the button is set to.

The “If” statement turns the LED on for a second if the button is pressed. Change the number 1 in the first “If” line to a zero. What happens?

Digital Output

Upload the Simon_2_BLINK sketch to your Simon.

Feel free to change the ledPin variable to any of the other LEDs if you want a different color.

Also- change the number on the delay lines to see what they do.

SIMON_2_BLINK

```
// The setup() funtion runs once, when the sketch starts


void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}
```

This pinMode line sets the ledPin to Output.

Otherwise we would not get enough voltage from the board to properly light up the LED.

Take this line of code out to see what will happen.

```
void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on (digit
  // analogWrite(ledPin, 255);     // set the LED on (ana
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off (digi
  // analogWrite(ledPin, 0);      // set the LED off (ana
  delay(1000);                  // wait for a second
}
```

digitalWrite (pinNumber , VALUE) ;

value can be HIGH or LOW, just don't forget CAPS!

delay (1000) ;

delay pauses everything for as long as you like in microseconds, so 1000 is one full second

Analog Output

Upload the Simon_2b_BLINK sketch to your Simon.

Feel free to change the ledPin variable to any of the other LEDs if you want a different color.

Also- change the number on the delay lines to see what they do.

SIMON_2b_BLINK §

```
void loop()
{
  // digitalWrite(ledPin, HIGH);   // set the LED on (
  analogWrite(ledPin, 255);      // set the LED on (ana
  delay(1000);                   // wait for a second
  // digitalWrite(ledPin, LOW);    // set the LED off
  analogWrite(ledPin, 0);       // set the LED off (anal
  delay(1000);                   // wait for a second
}
```

analogWrite (pinNumber , VALUE) ;

value can be any number in between zero and 255.
Zero is off and 255 is the brightest the LED can get.
Try a bunch of different numbers to see their effect.

The next steps?

- You have used the Simon to cover the four basic concepts of physical computing and micro-controller programming:
  - Analog Input, Digital Output, Analog Output, and Digital Input.

- Try playing around with existing code from the Open Source community.
  - Upload Arduino sketches as they are posted to make sure they work,
  - then start changing lines of code you feel familiar with.

Some Project Ideas

- Make your own 'dice experiment'
  - generate random numbers using e.g. the ADC input noise
  - you may use the serial interface to get a faster transfer to your computer
- Make a synchrotron-light alarm
  - Use the photo-resistor to detect the synchrotron light levels
  - e.g. raise an (audible, e-mail) alarm once the light is above a given threshold
- Make a game measuring peoples reaction time
  - Random delay → light up LED → measure time until given button is pressed
- Let your imaging flow and think of something yourself...

Reference and Acknowledgements

- The kit was designed and produced by: http://www.sparkfun.com
- Great resource to dive more into electronics (many plots taken from there): https://learn.sparkfun.com/tutorials
- On the power of building things yourself:
  M. I. Norton, D. Mochon, D. Ariely, “The 'IKEA Effect': When Labor Leads to Love”, Havard Business School, Working Paper 11-091, 2001
  http://www.hbs.edu/faculty/Publication%20Files/11-091.pdf
- Wikipedia is your friend (other resource of plots and diagrams): http://en.wikipedia.org/wiki/Book:An_introduction_to_electronics
- Check-out: Open-Source and Open-Hardware https://www.sparkfun.com/news/1229