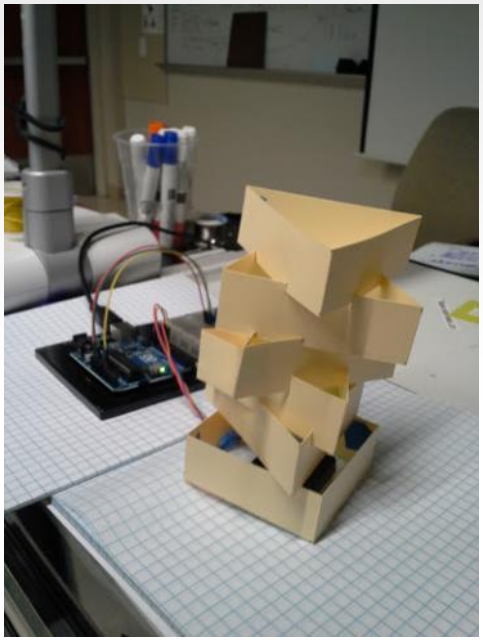
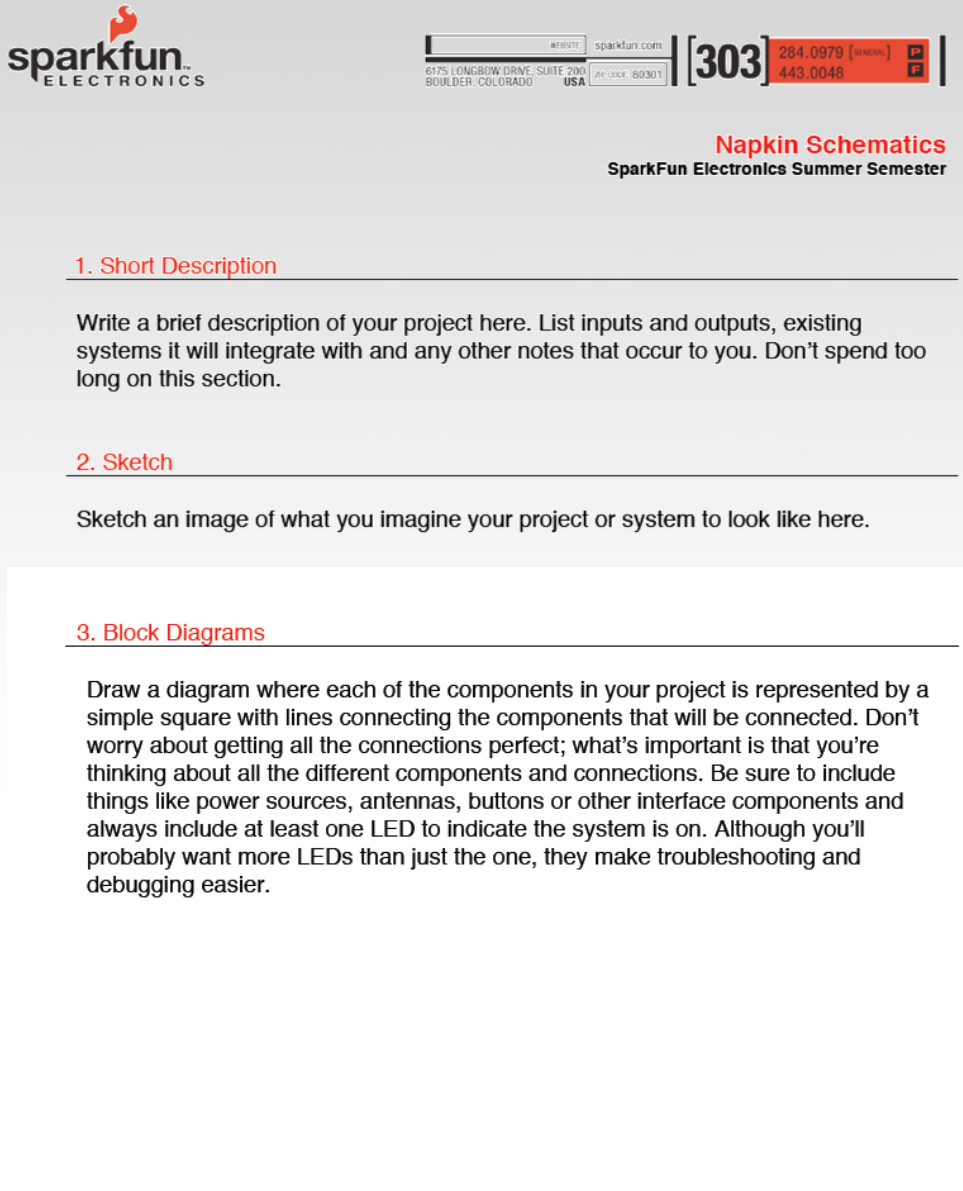
Project: Mood Lamp / Light Sculpture



Napkin Schematics

Emphasize the engineering design process with students. We like to skirt the line between formal and informal with a tool called **Napkin Schematics.**



sparkfun ELECTRONICS

6175 LONGBOW DRIVE, SUITE 200 BOULDER, COLORADO USA

sparkfun.com

[303] 284.0979 443.0048

Napkin Schematics

SparkFun Electronics Summer Semester

1. Short Description

Write a brief description of your project here. List inputs and outputs, existing systems it will integrate with and any other notes that occur to you. Don't spend too long on this section.

2. Sketch

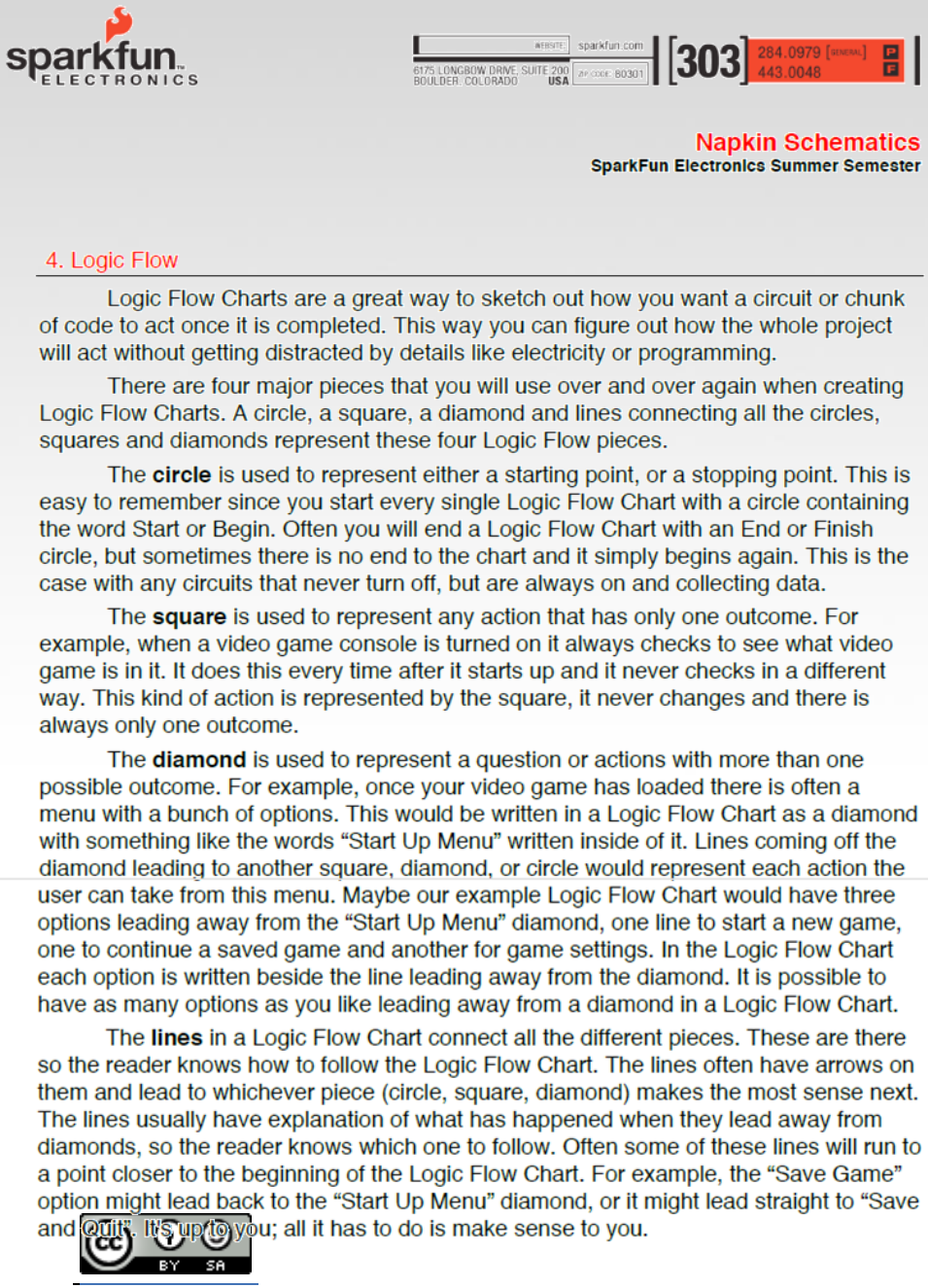
Sketch an image of what you imagine your project or system to look like here.

3. Block Diagrams

Draw a diagram where each of the components in your project is represented by a simple square with lines connecting the components that will be connected. Don't worry about getting all the connections perfect; what's important is that you're thinking about all the different components and connections. Be sure to include things like power sources, antennas, buttons or other interface components and always include at least one LED to indicate the system is on. Although you'll probably want more LEDs than just the one, they make troubleshooting and debugging easier.

Napkin Schematics

Emphasize the engineering design process with students. We like to skirt the line between formal and informal with a tool called **Napkin Schematics.**



sparkfun ELECTRONICS

6175 LONGBOW DRIVE, SUITE 200 BOULDER, COLORADO USA 80301 | sparkfun.com | [303] 284.0979 GENERAL | 443.0048 P F

Napkin Schematics
SparkFun Electronics Summer Semester

4. Logic Flow

Logic Flow Charts are a great way to sketch out how you want a circuit or chunk of code to act once it is completed. This way you can figure out how the whole project will act without getting distracted by details like electricity or programming.

There are four major pieces that you will use over and over again when creating Logic Flow Charts. A circle, a square, a diamond and lines connecting all the circles, squares and diamonds represent these four Logic Flow pieces.

The **circle** is used to represent either a starting point, or a stopping point. This is easy to remember since you start every single Logic Flow Chart with a circle containing the word Start or Begin. Often you will end a Logic Flow Chart with an End or Finish circle, but sometimes there is no end to the chart and it simply begins again. This is the case with any circuits that never turn off, but are always on and collecting data.

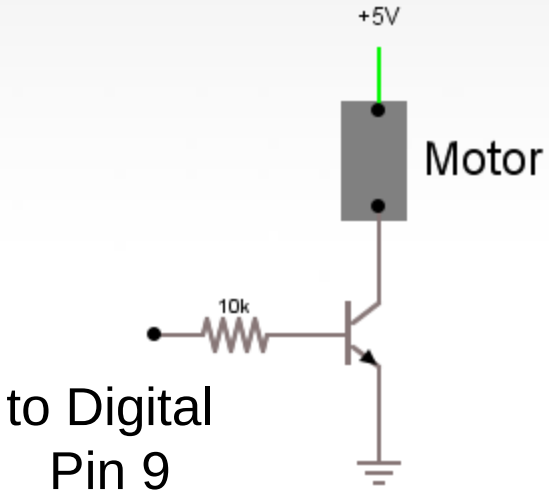
The **square** is used to represent any action that has only one outcome. For example, when a video game console is turned on it always checks to see what video game is in it. It does this every time after it starts up and it never checks in a different way. This kind of action is represented by the square, it never changes and there is always only one outcome.

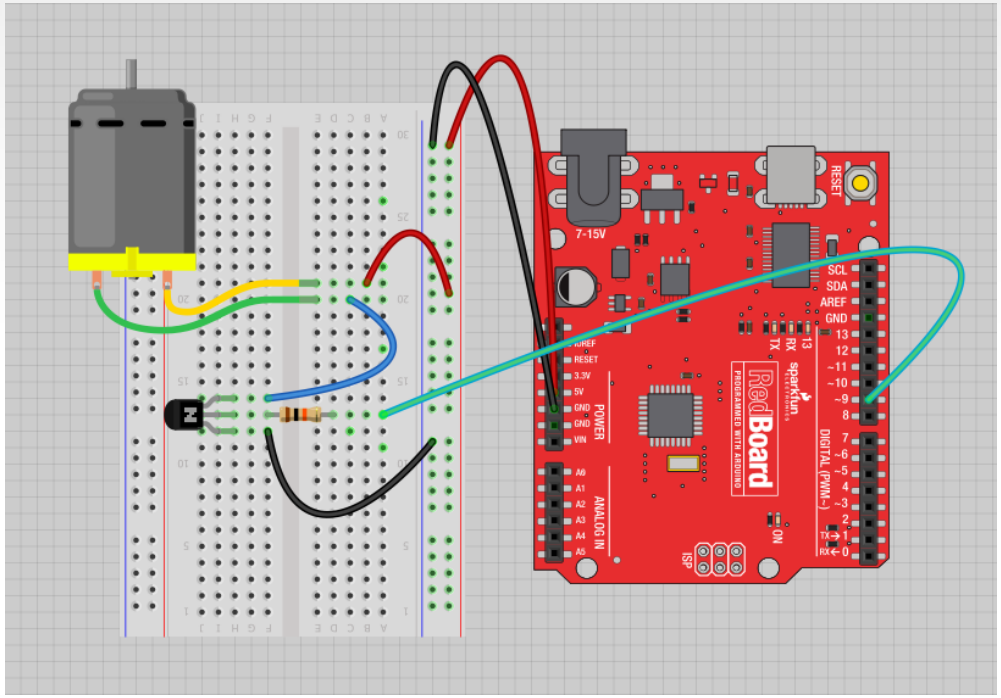
The **diamond** is used to represent a question or actions with more than one possible outcome. For example, once your video game has loaded there is often a menu with a bunch of options. This would be written in a Logic Flow Chart as a diamond with something like the words “Start Up Menu” written inside of it. Lines coming off the diamond leading to another square, diamond, or circle would represent each action the user can take from this menu. Maybe our example Logic Flow Chart would have three options leading away from the “Start Up Menu” diamond, one line to start a new game, one to continue a saved game and another for game settings. In the Logic Flow Chart each option is written beside the line leading away from the diamond. It is possible to have as many options as you like leading away from a diamond in a Logic Flow Chart.

The **lines** in a Logic Flow Chart connect all the different pieces. These are there so the reader knows how to follow the Logic Flow Chart. The lines often have arrows on them and lead to whichever piece (circle, square, diamond) makes the most sense next. The lines usually have explanation of what has happened when they lead away from diamonds, so the reader knows which one to follow. Often some of these lines will run to a point closer to the beginning of the Logic Flow Chart. For example, the “Save Game” option might lead back to the “Start Up Menu” diamond, or it might lead straight to “Save and Quit”. It’s up to you; all it has to do is make sense to you.

Driving Motors or other High Current Loads

NPN Transistor (Common Emitter "Amplifier" Circuit)





Input

Input is any signal entering an electrical system.

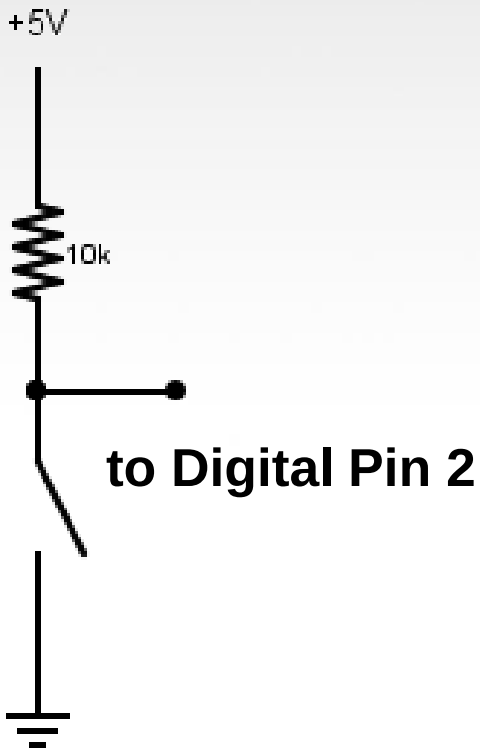
- Both digital and analog sensors are forms of input
- Input can also take many other forms: Keyboards, a mouse, infrared sensors, biometric sensors, or just plain voltage from a circuit

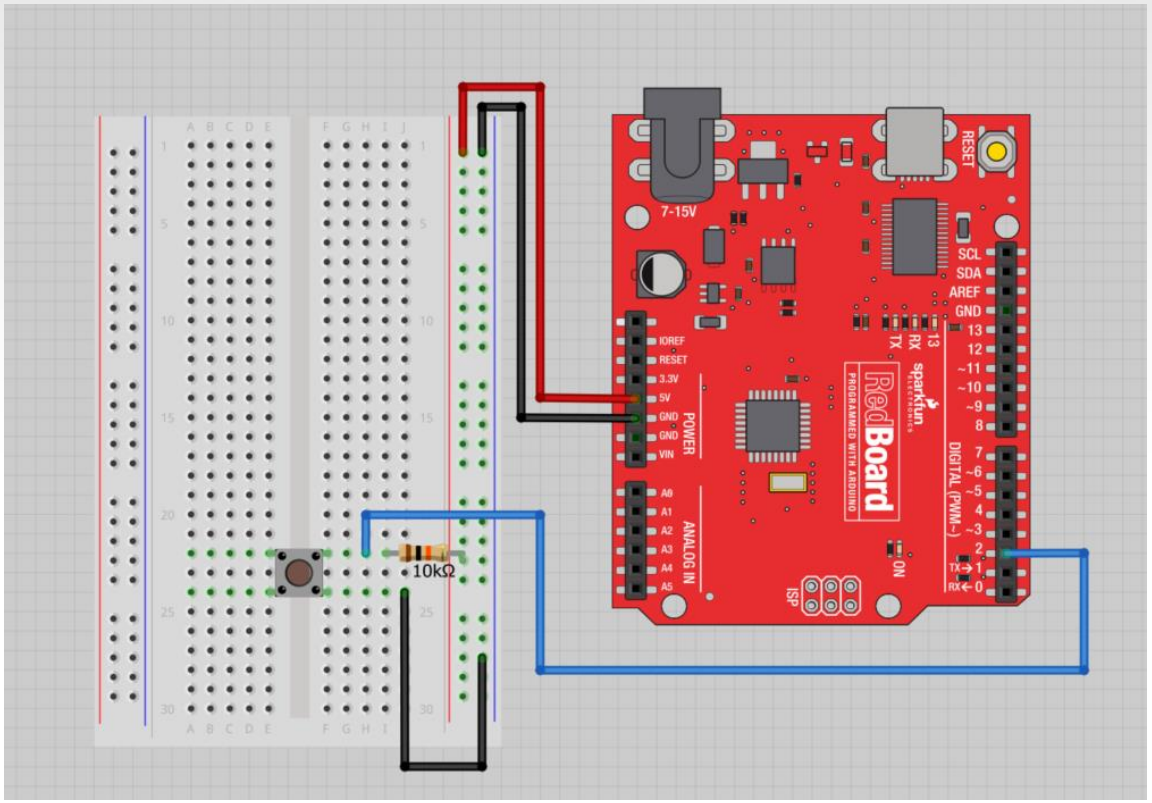
Project #4 – Digital Input

In Arduino, open up:

File → Examples → 02.Digital → Button

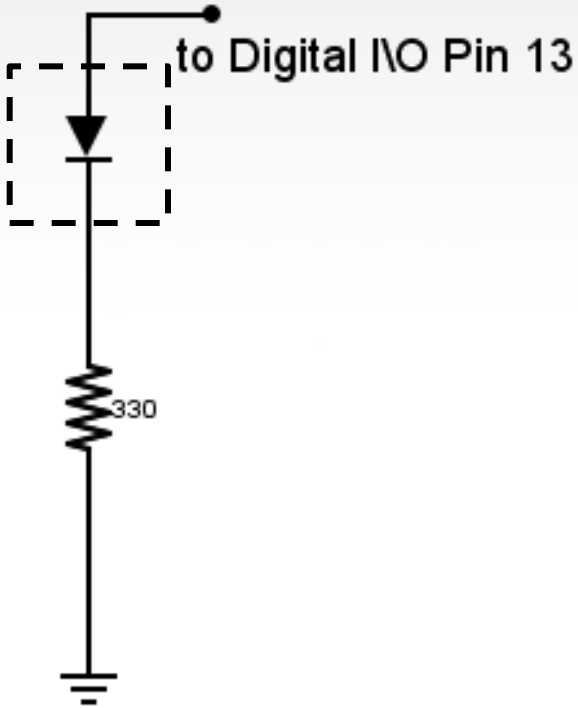
Digital Sensors (a.k.a. Switches)
Pull-up Resistor (circuit)

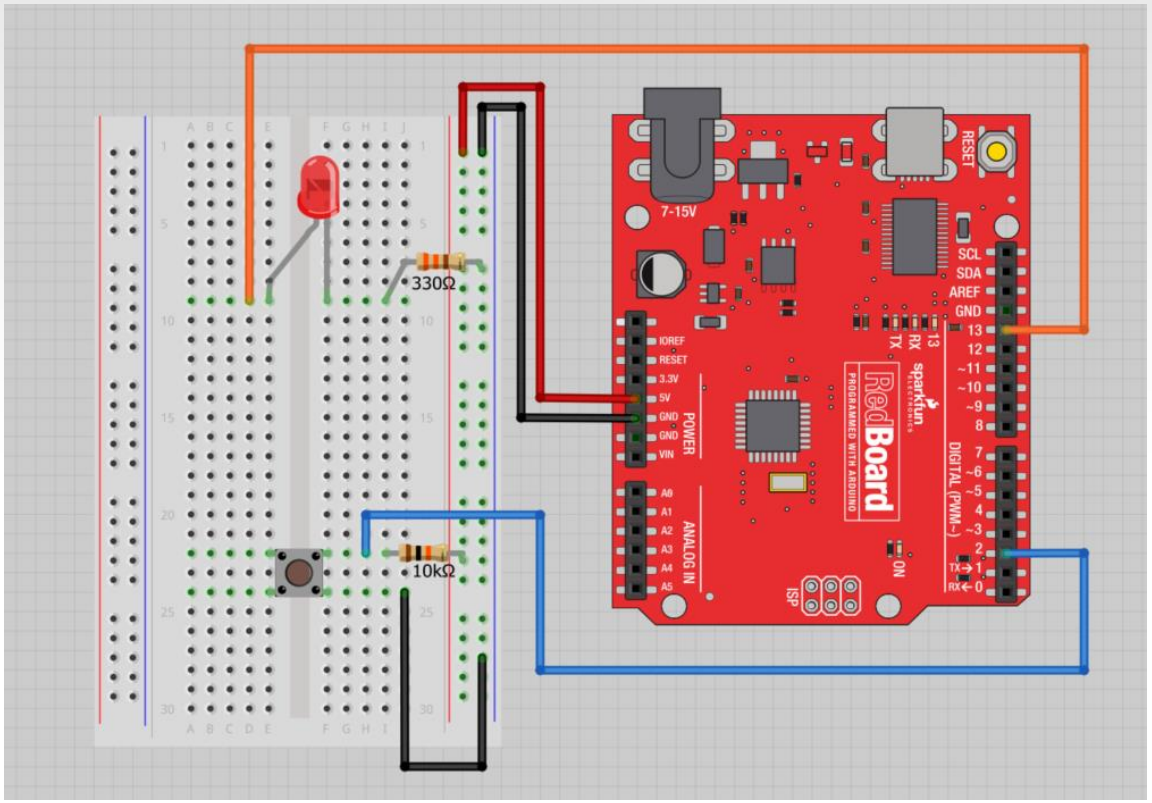




Digital Sensors (a.k.a. Switches)
Add an indicator LED to Pin 13

This is just like our 1st circuit!





Digital Input

- Connect digital input to your Arduino using Pins # 0 – 13 (Although pins # 0 & 1 are also used for programming)
- Digital Input needs a `pinMode` command:

```
pinMode (pinNumber, INPUT);
```

*Make sure to use ALL CAPS for **INPUT***

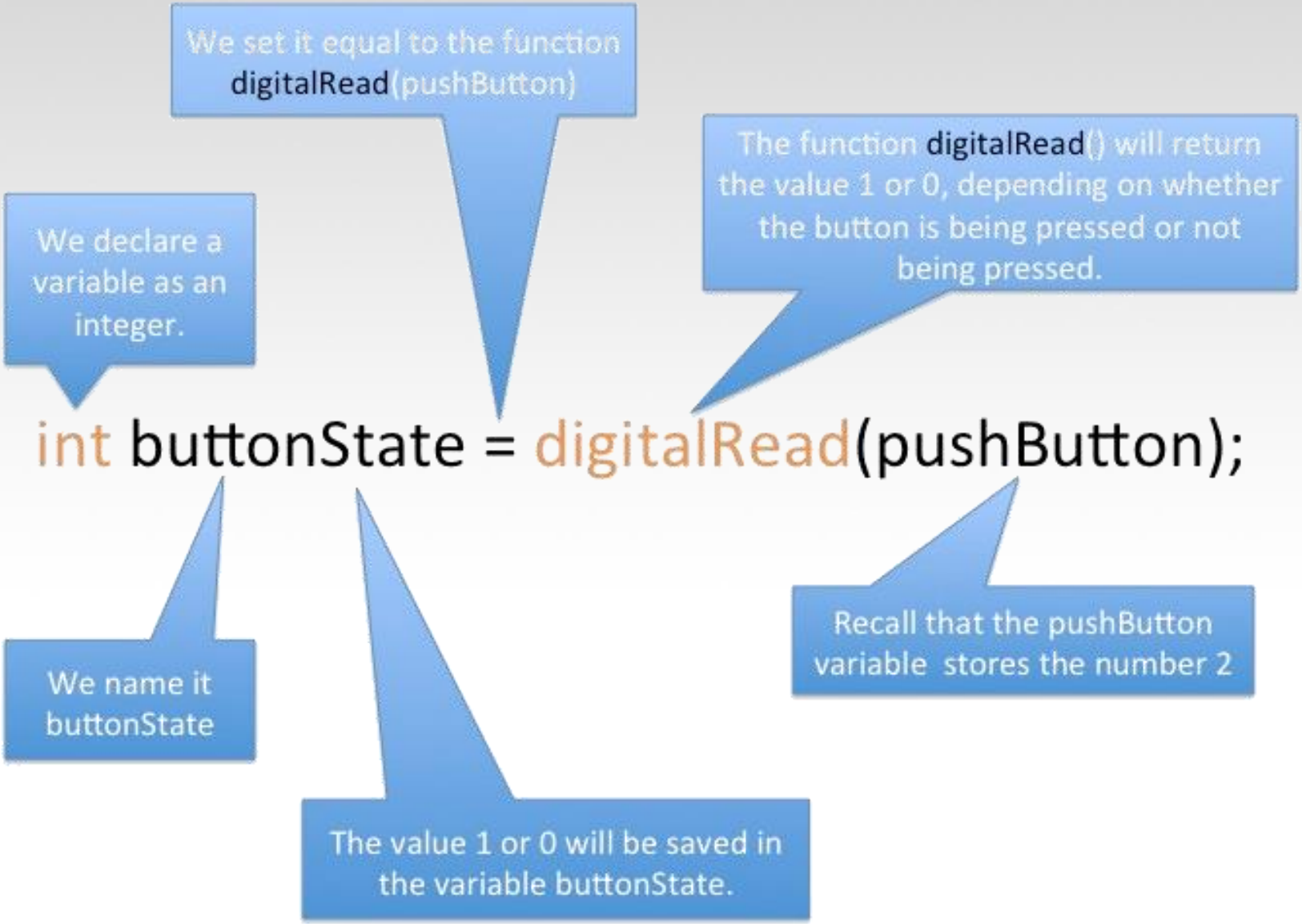
- To get a digital reading:

```
int buttonState = digitalRead (pinNumber);
```

- Digital Input values are only **HIGH** (On) or **LOW** (Off)

Digital Sensors

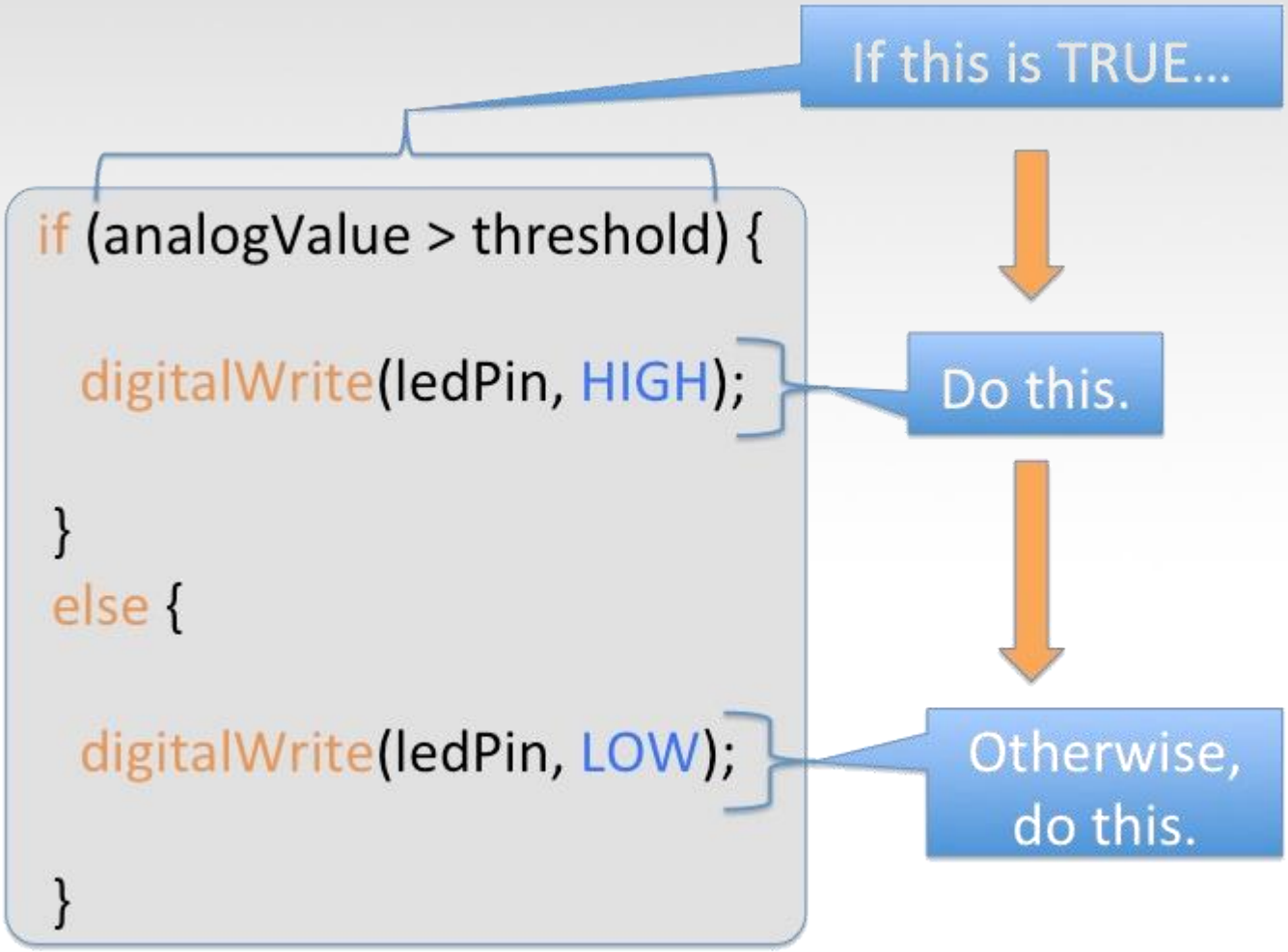
- Digital sensors are more straight forward than Analog
- No matter what the sensor there are only two settings: On and Off
- Signal is always either HIGH (On) or LOW (Off)
- Voltage signal for HIGH will be a little less than 5V on your Uno
- Voltage signal for LOW will be 0V on most systems



http://opensourcehardwarejunkies.com/tutorial-03-digitalread-and-

Programming: Conditional Statements

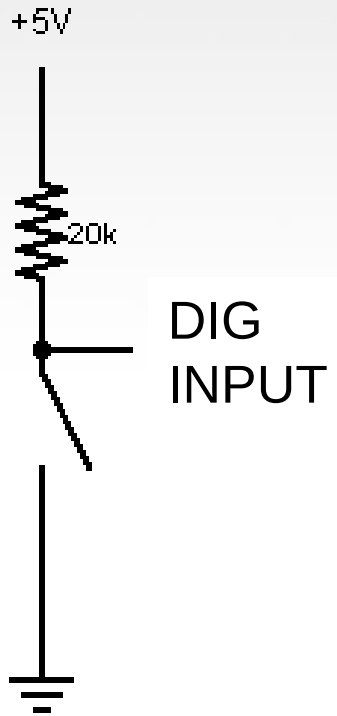
`if()`



Programming: Conditional Statements

if()

```
void loop()
{
  int buttonState = digitalRead(5);
  if(buttonState == LOW)
  {  // do something
  }
  else
  {  // do something else
  }
}
```



Boolean Operators

<Boolean>	Description
( ) == ( )	is equal?
( ) != ( )	is not equal?
( ) > ( )	greater than
( ) >= ( )	greater than or equal
( ) < ( )	less than
( ) <= ( )	less than or equal