Reprogramming the Simon

// Demo Overview:

• Arduino Software and Arduino hardware

• Uploading different code onto the Simon

• Disco Mode Code found: www.sparkfun.com/tutorials/203

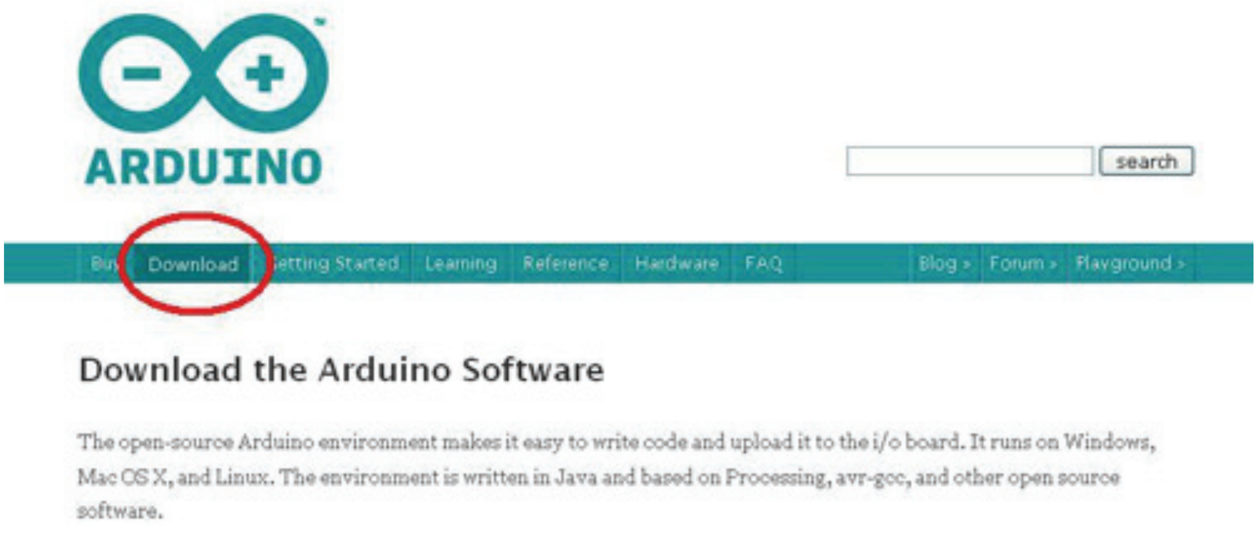
• Adding a photocell and using Disco Mode

// Setting Up:

Downloading the Arduino Environment:

First go the Arduino website: http://www.arduino.cc/

Then click on Download and follow the instructions:



Download the Arduino Software

The open-source Arduino environment makes it easy to write code and upload it to the i/o board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, avr-gcc, and other open source software.

Name:
Date:

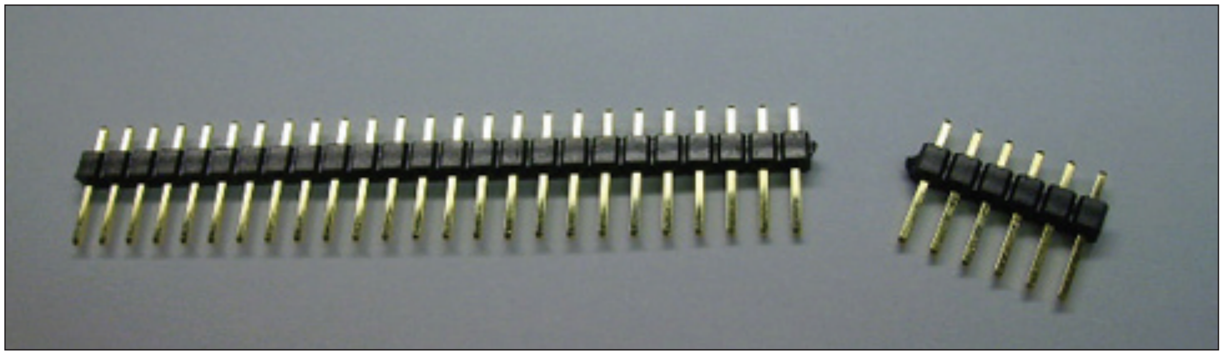
// Hardware:

You will need the following:

- USB cable
- FTDI Breakout Board
- six pins of Break Away Headers

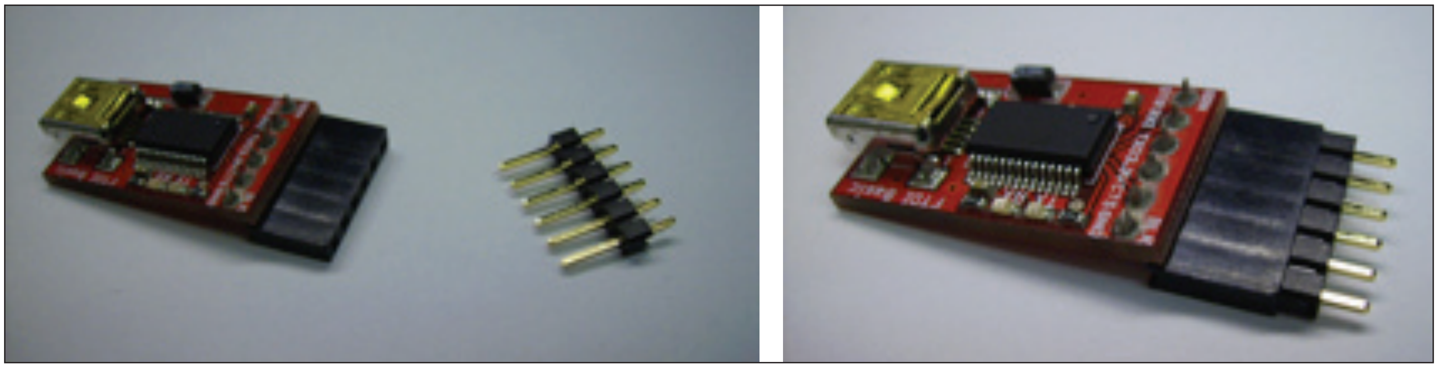
Break Away Headers:

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:

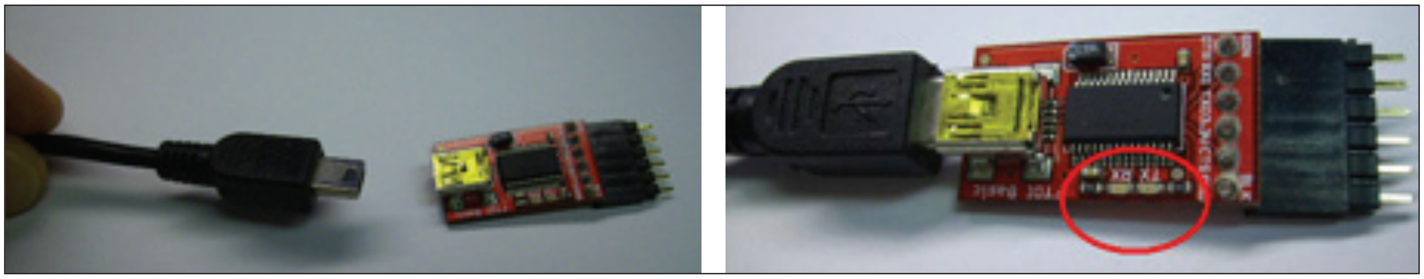


Name:
Date:

// Plugging In:

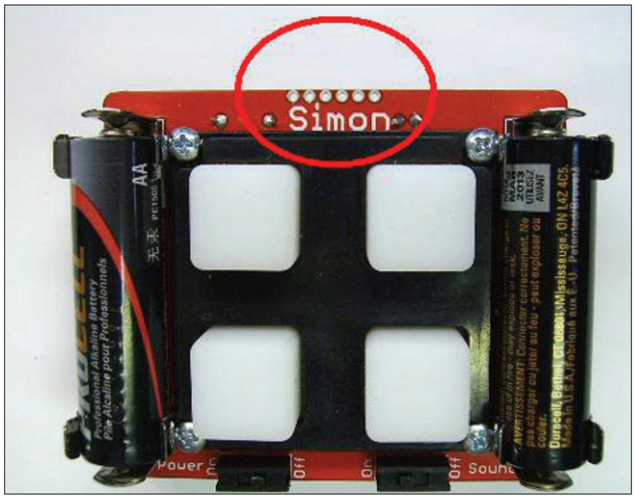
Plugging it all into the computer:

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.

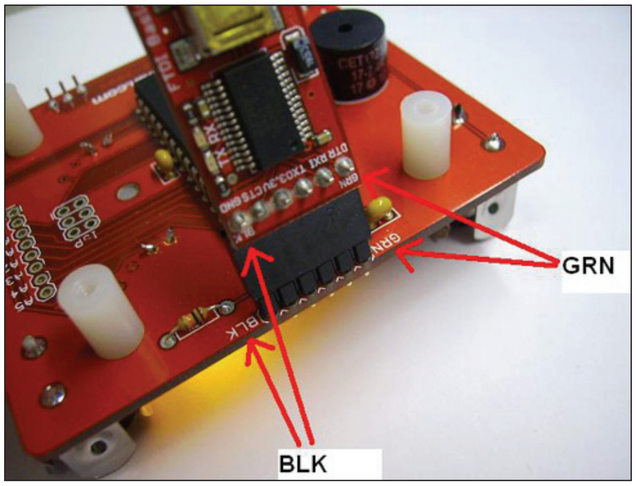


Plugging it all into the Simon:

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



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.



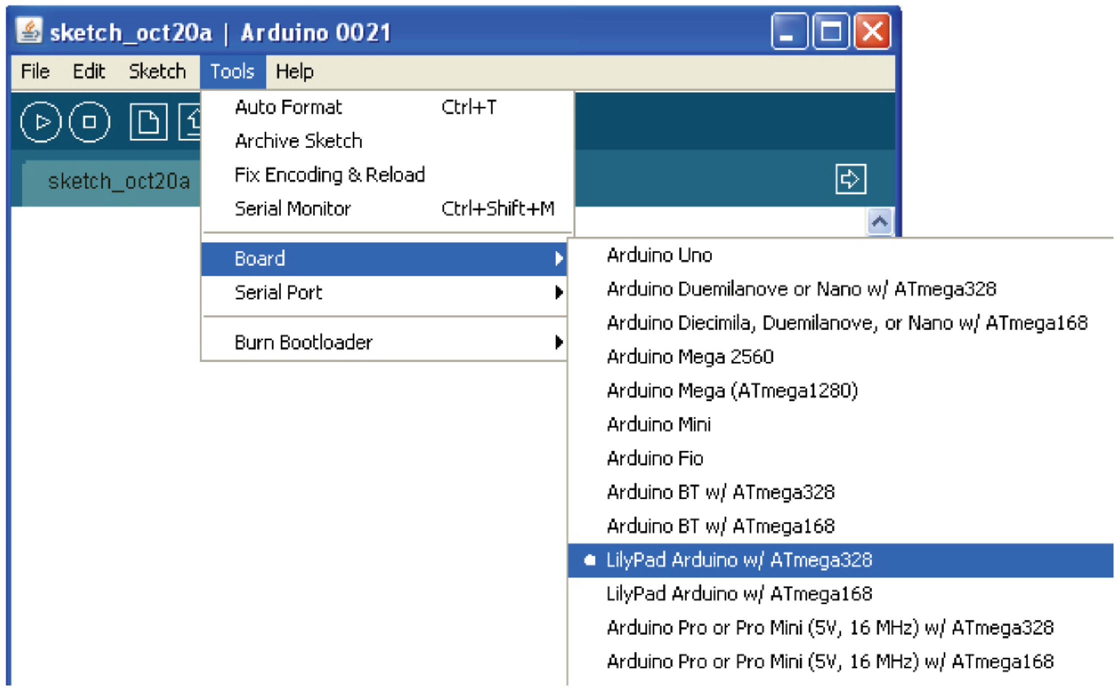
Reprogramming the Simon
Worksheets v. 1.0

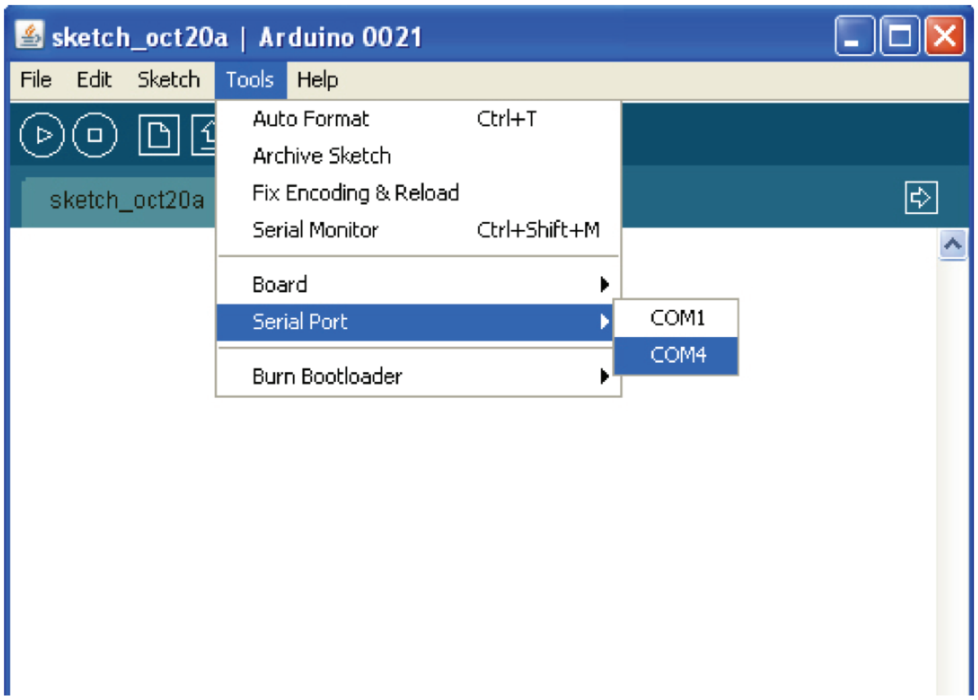
Name:
Date:

// Uploading Code to the Simon:

Before uploading code onto the Simon:

Open the Arduino environment. Choose your board type and serial port.

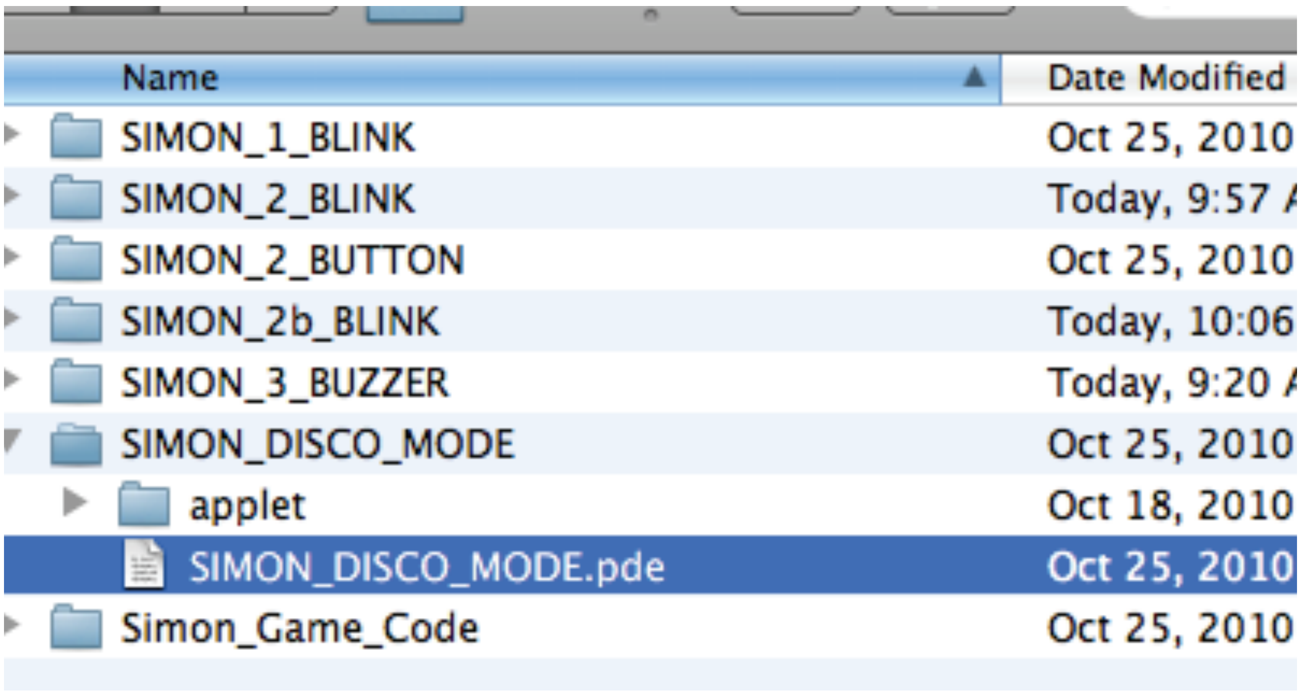

sketch_oct20a | Arduino 0021
File Edit Sketch Tools Help
sketch_oct20a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Burn Bootloader
Arduino Uno
Arduino Duemilanove or Nano w/ ATmega328
Arduino Diecimila, Duemilanove, or Nano w/ ATmega168
Arduino Mega 2560
Arduino Mega (ATmega1280)
Arduino Mini
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
● LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168



Name:
Date:

// Uploading Code to the Simon:

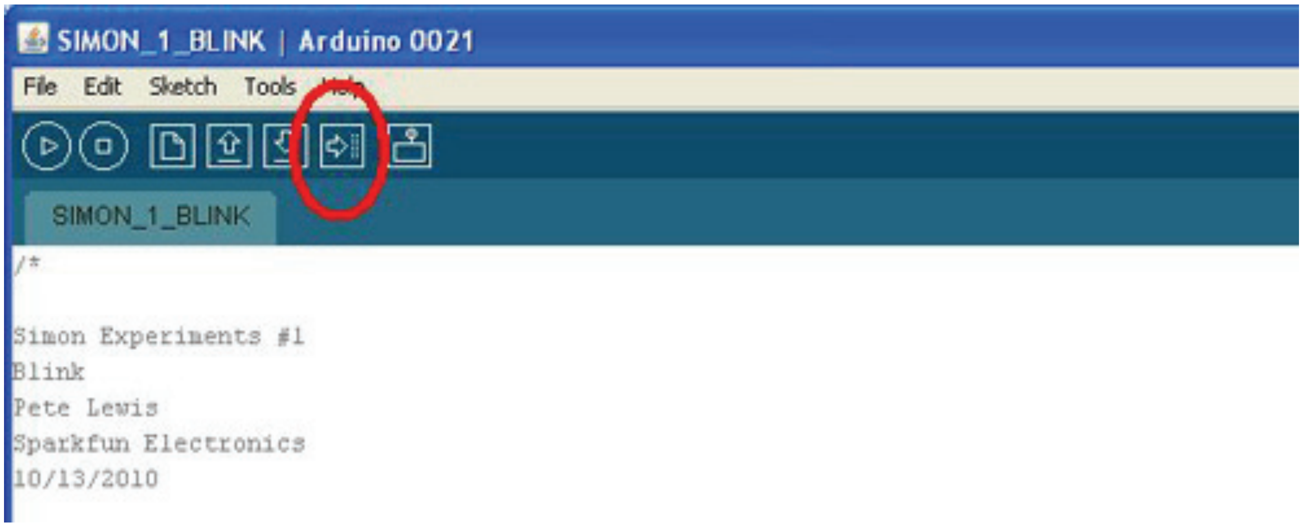
Find the Disco Stu Mode Code (It should be in a folder named SimonProgramming)



Name	Date Modified
SIMON_1_BLINK	Oct 25, 2010
SIMON_2_BLINK	Today, 9:57
SIMON_2_BUTTON	Oct 25, 2010
SIMON_2b_BLINK	Today, 10:06
SIMON_3_BUZZER	Today, 9:20
SIMON_DISCO_MODE	Oct 25, 2010
applet	Oct 18, 2010
SIMON_DISCO_MODE.pde	Oct 25, 2010
Simon_Game_Code	Oct 25, 2010

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.

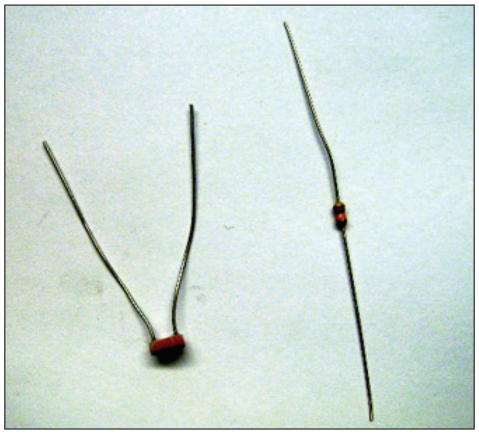


```
/*

Simon Experiments #1
Blink
Pete Lewis
Sparkfun Electronics
10/13/2010
```

Name:
Date:

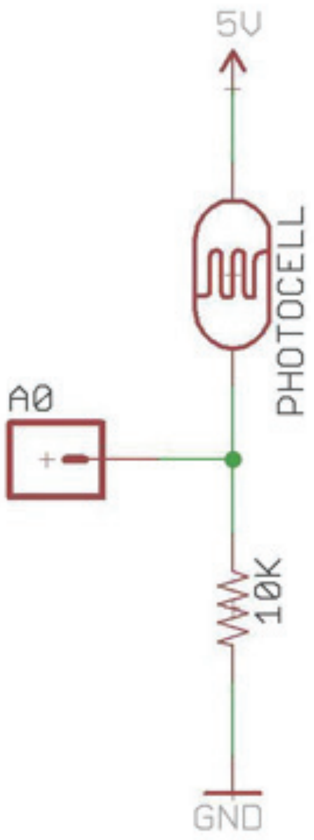
// Adding a Photoresistor to the Simon (Analog Input):

We're ready to add a sensor to your Simon!
You'll need a photoresistor and a plain old 10K Ohm resistor:



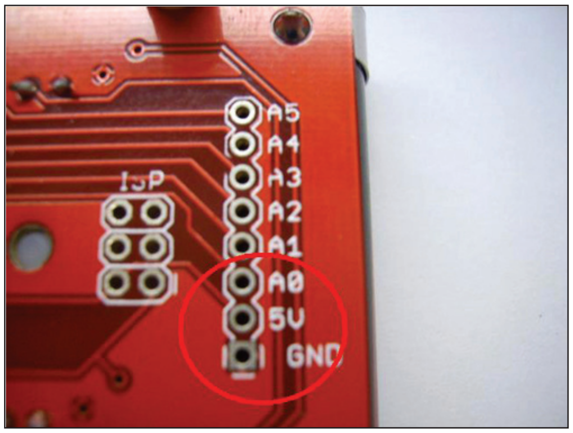
The Schematic:

This schematic shows the three connections we will have to make 5V (or VCC), GND and A0.



The Simon Analog Pins:

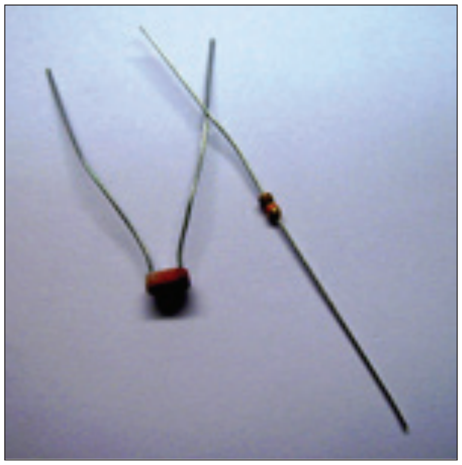
The three holes (5V (or VCC), GND and A0) you will connect to the photoresister circuit.

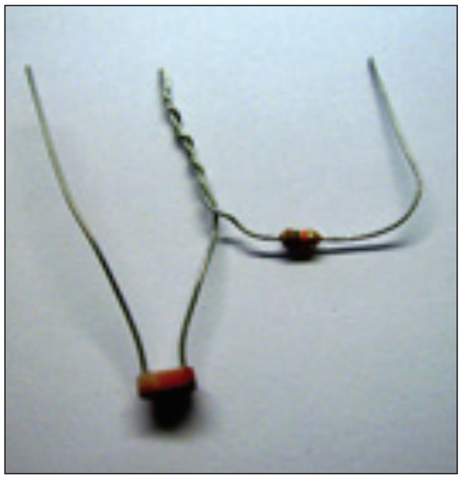


Name:
Date:

// Creating the Circuit:

Make the photoresistor and regular resistor look like the schematic.

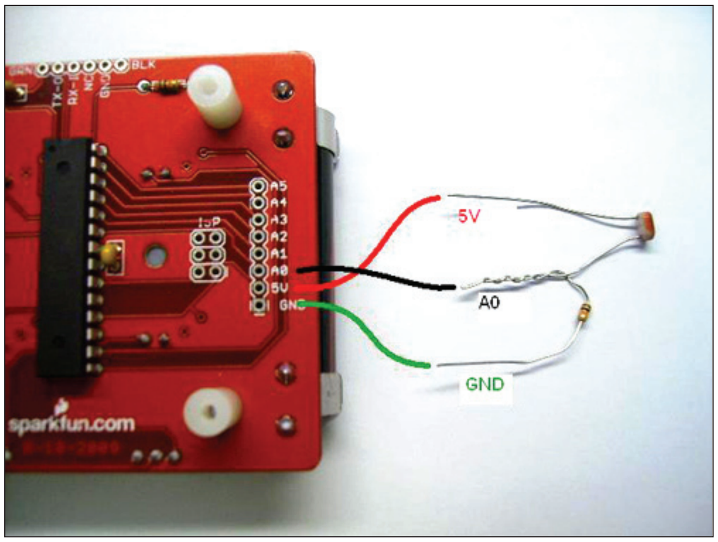




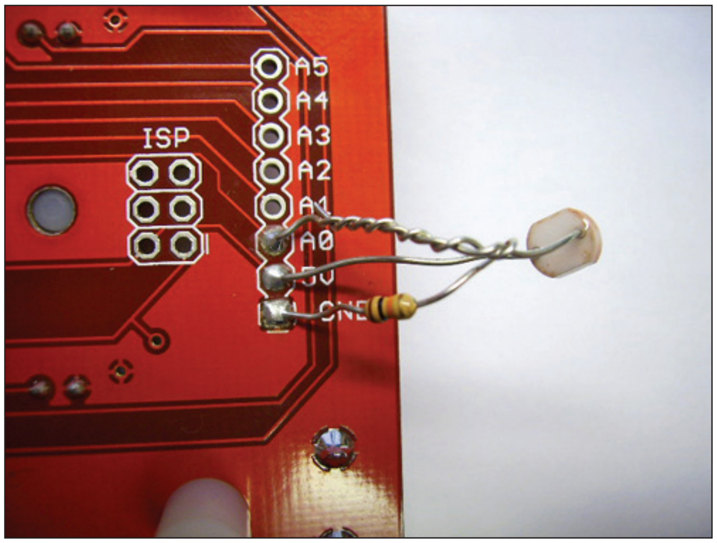
Name:
Date:

// Connecting the Circuit:

Just prior to soldering your photoresistor circuit should look like this.



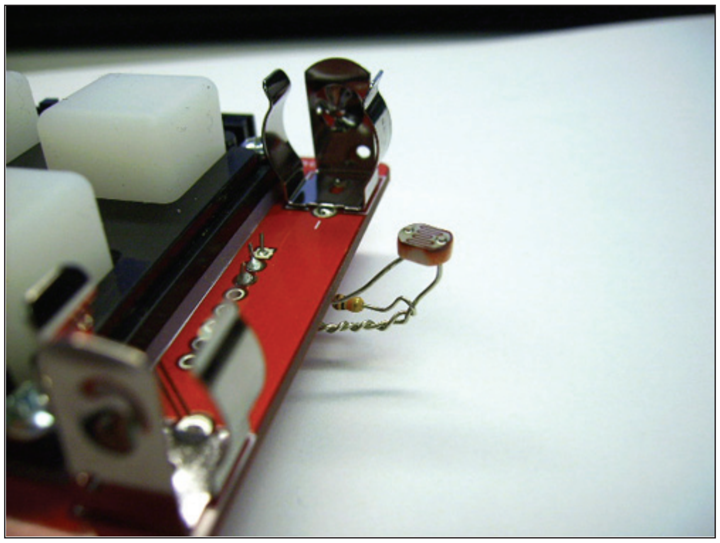
Just prior to soldering your photoresistor circuit should look like this.



Name:
Date:

// Bending the Sensor:

Bend the sensor photoresistor around so it faces your light source.



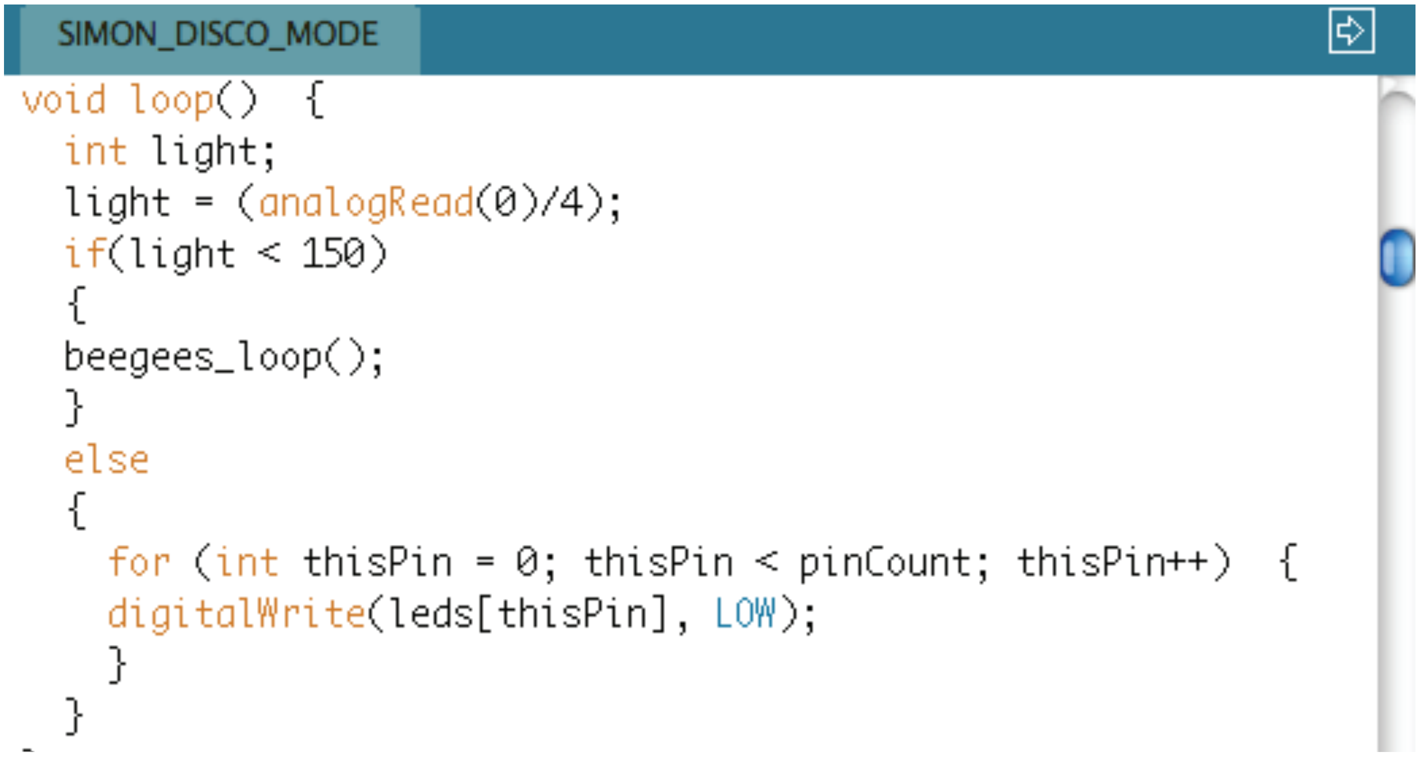
Bend the sensor photoresistor around so it faces your light source.

Name:
Date:

// 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.



```
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.

Name:
Date:

// Digital Input and Digital and Analog Output:

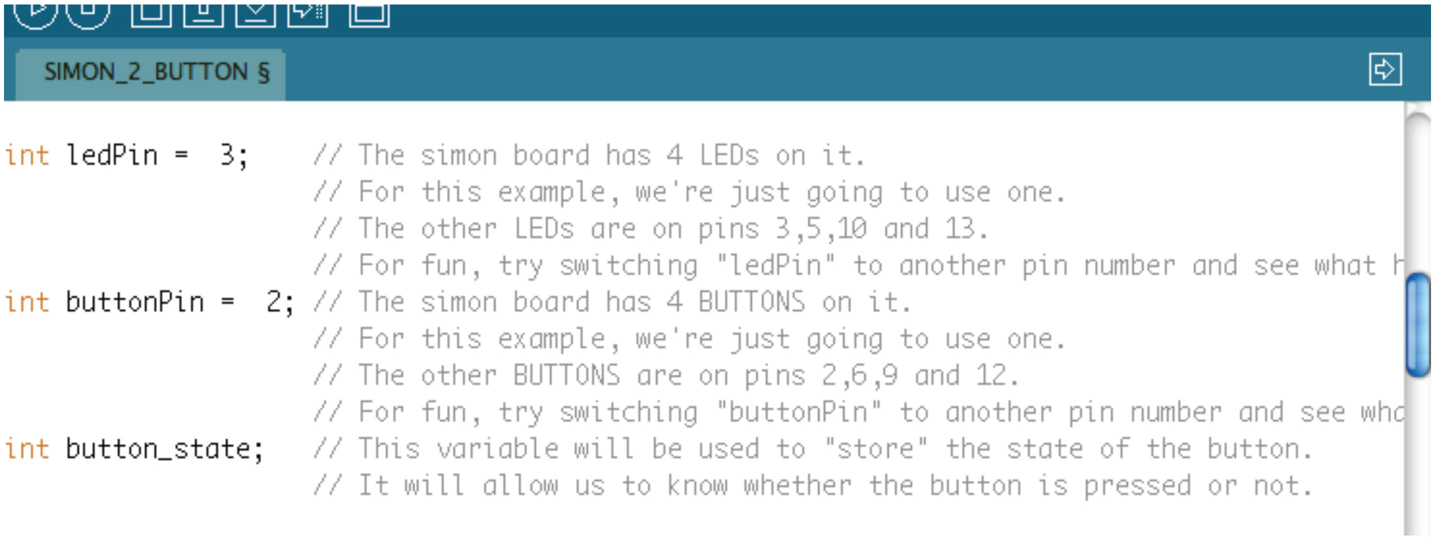
We will use the following sketches to discuss each of these topics, so feel free to upload the first sketch to your Simon:

- 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.

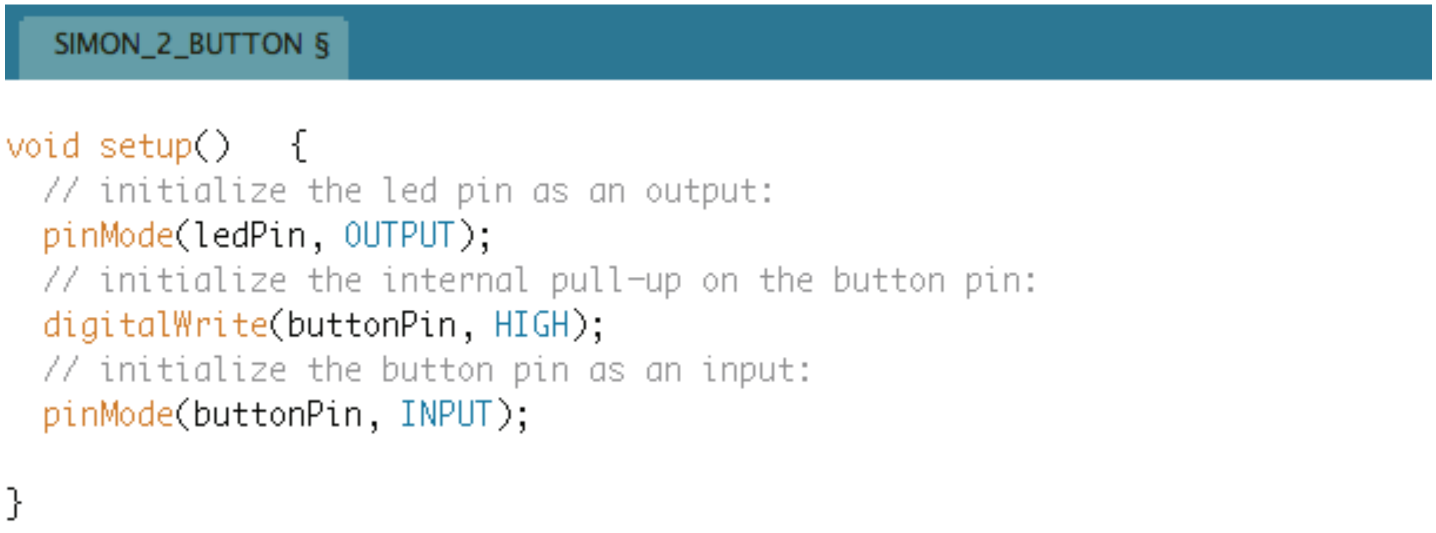


```
int ledPin =  3;    // The simon board has 4 LEDs on it.
                    // For this example, we're just going to use one.
                    // The other LEDs are on pins 3,5,10 and 13.
                    // For fun, try switching "ledPin" to another pin number and see what h
int buttonPin =  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.

Name:
Date:

// Digital Input and Digital and Analog Output:



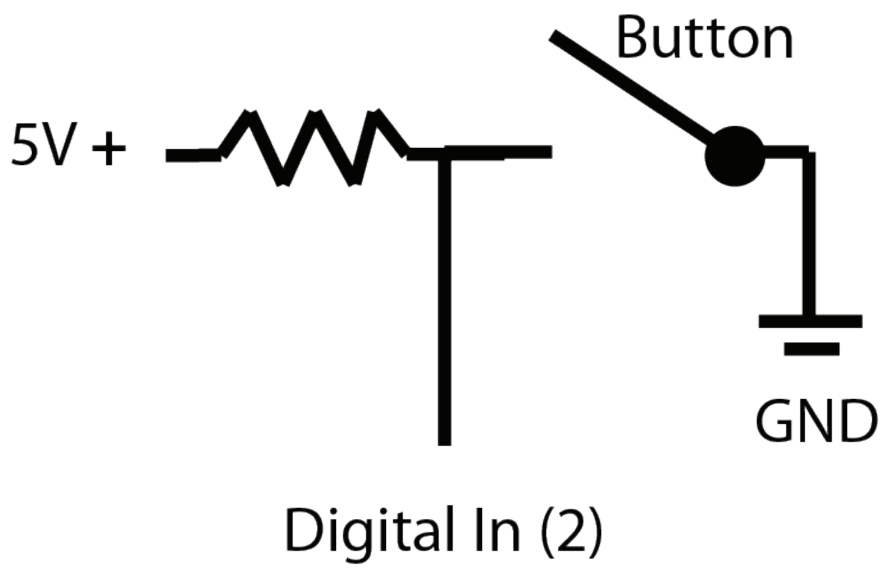
```
void setup()   {
  // initialize the led pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the internal pull-up on the button pin:
  digitalWrite(buttonPin, HIGH);
  // initialize the button pin as an input:
  pinMode(buttonPin, INPUT);

}
```

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.

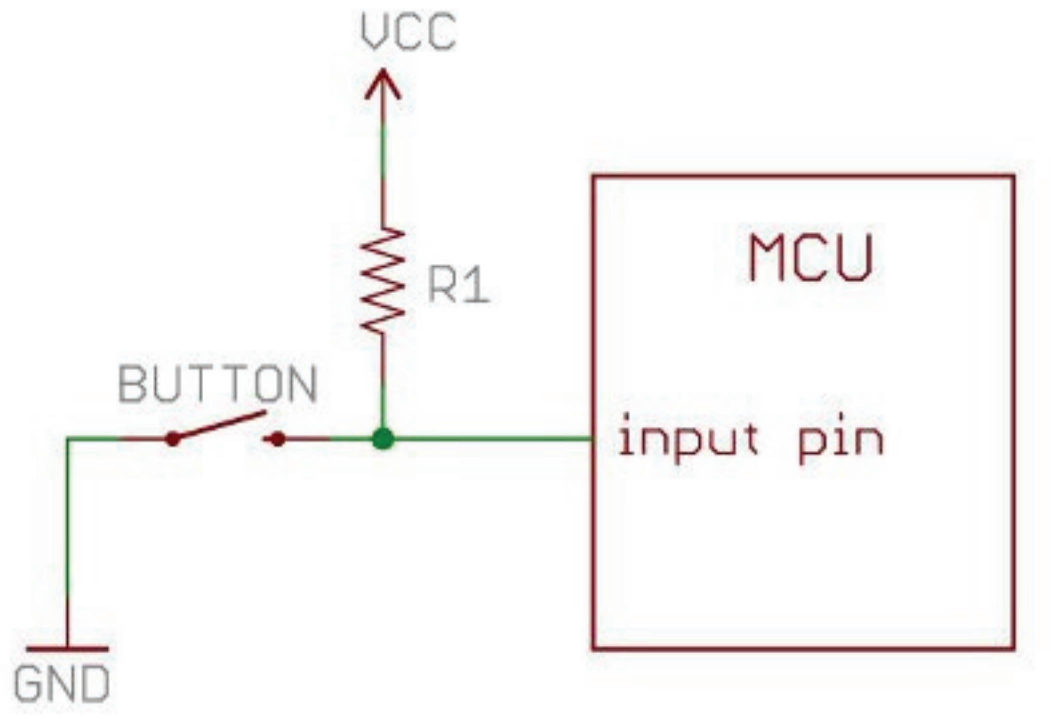
Name:
Date:

// Pull Up Resistors:

A regular pull up resistor looks like this:

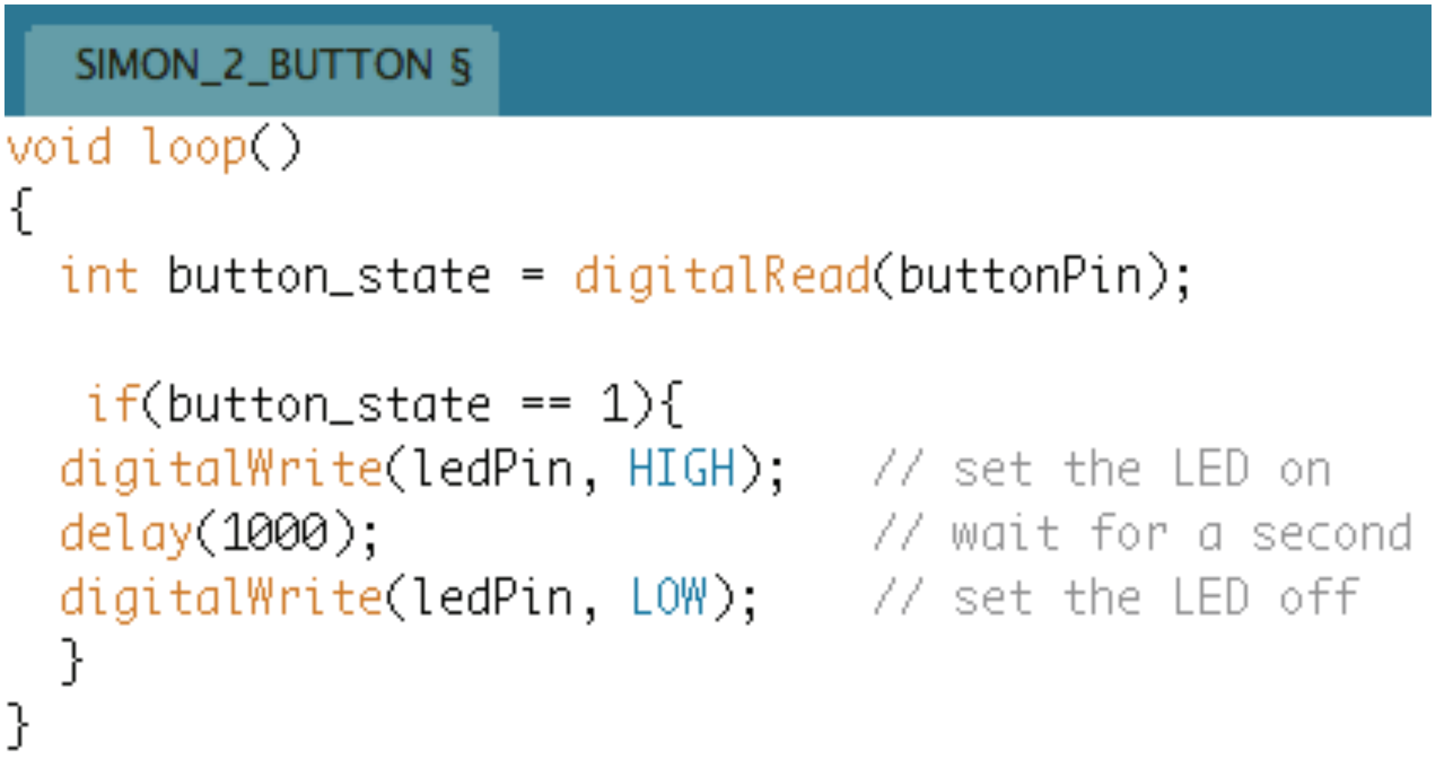


But the internal pull up resistor in the arduino moves the power source from the separate power line onto the input pin line:



Name:
Date:

// Pull Up Resistors:



```
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?

Reprogramming the Simon
Worksheets v. 1.0

Name:
Date:

// 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.

```
// 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.

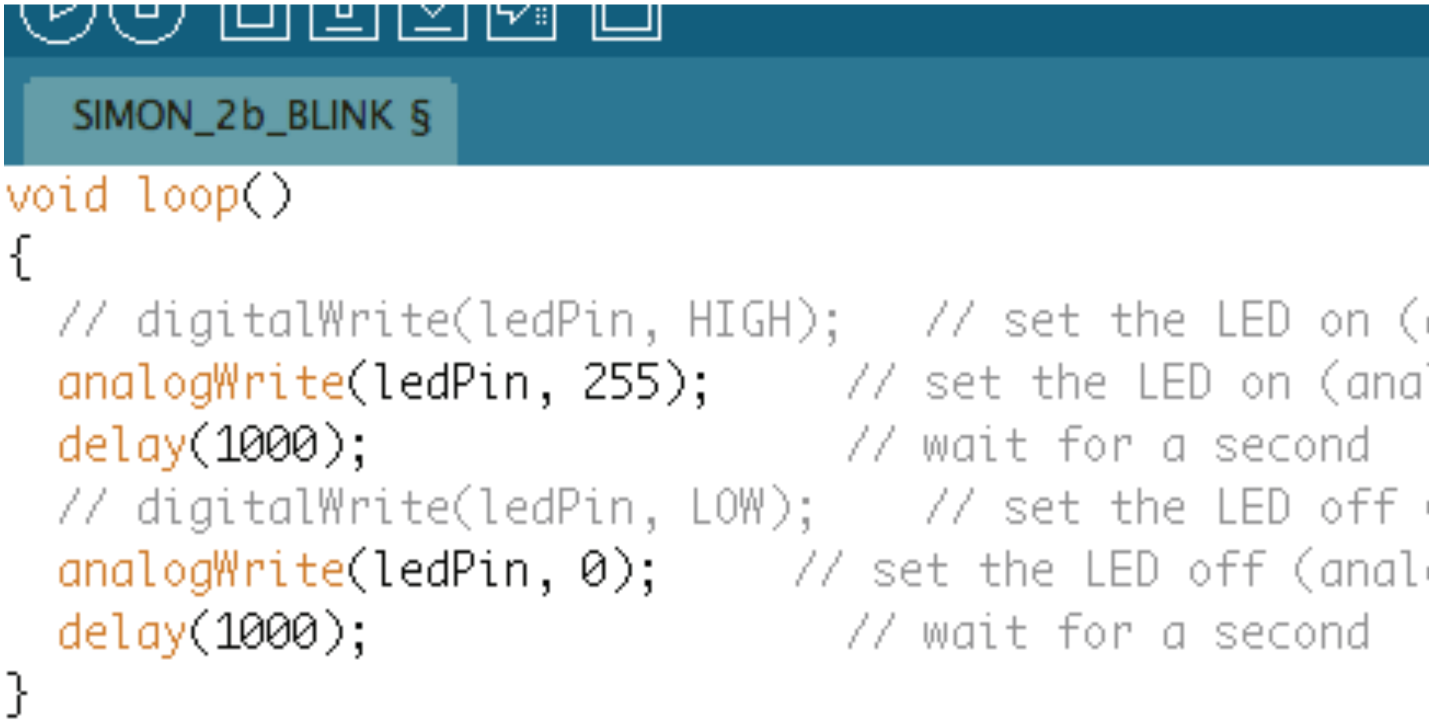
```
void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on (digit
  // analogWrite(ledPin, 255);     // set the LED on (an
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off (digi
  // analogWrite(ledPin, 0);     // set the LED off (anal
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

Name:
Date:

// 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.

Name:
Date:

// The Next Step?

You have used the Simon to cover the four basic concepts of 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.