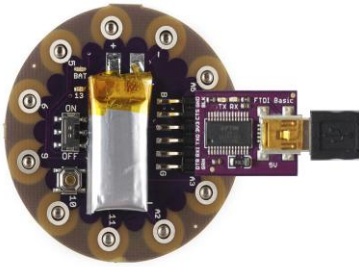
Lily Pad (Arduino) Programming

Andrew Bullen

Getting Started…

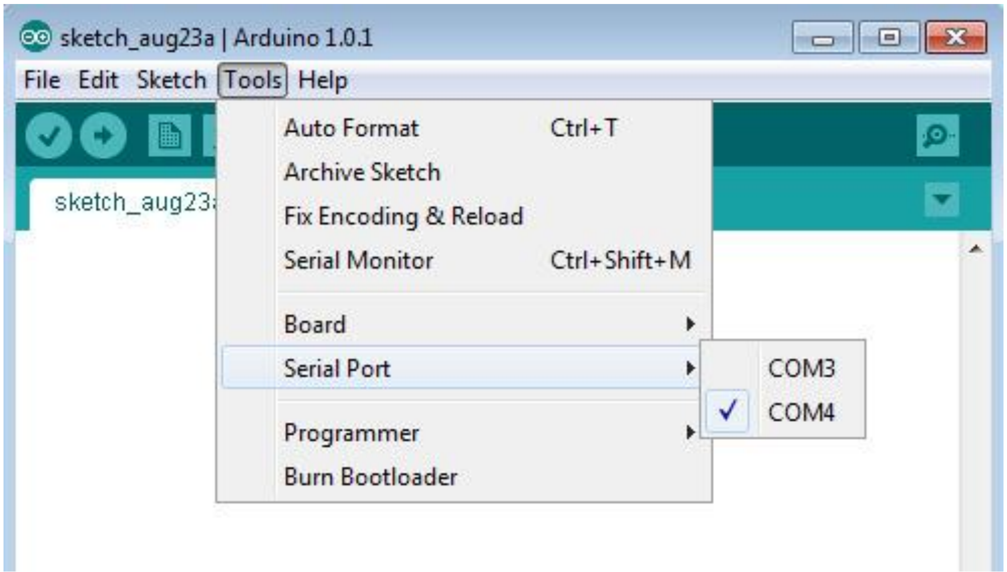
1. Attach your LilyPad Arduino to your computer and start the Arduino software



2. Select the correct serial port

Now you need to select the right serial port so that the Arduino software can talk to your LilyPad.

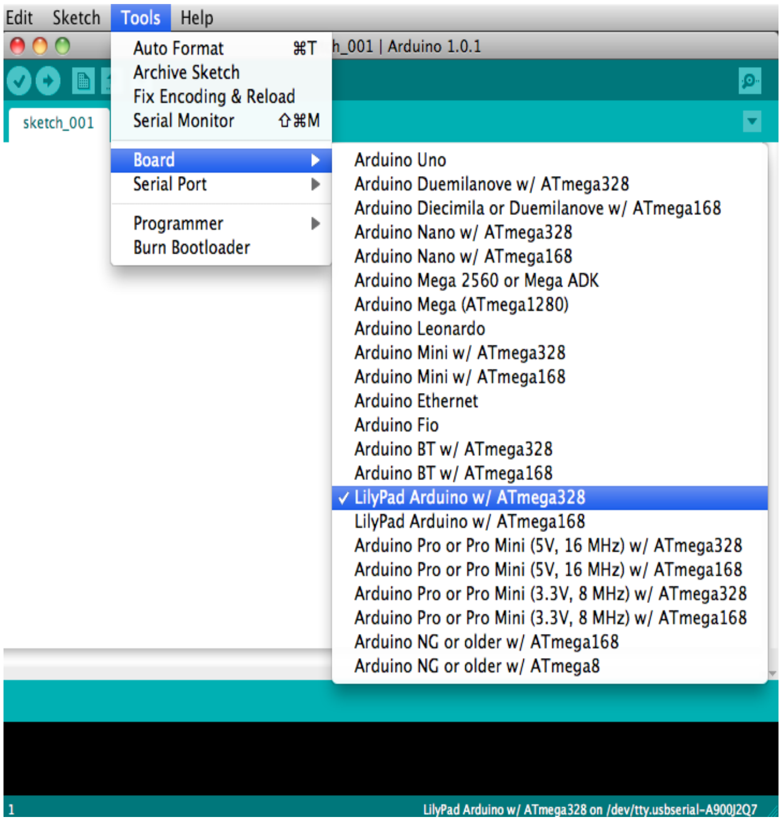
Under the “Tools–>Serial Port” menu, choose the highest numbered COM port.



Note: for either Mac or PC computers, you can also find the correct port by unplugging your board, looking at the menu, and then plugging your board in and looking at the menu again. The item that has appeared is the correct port.

3. Select the appropriate board

Now you need to select the right board, so that the software knows you’re using a LilyPad and not some other Arduino board. Under the “Tools–>Boards” menu, select the entry appropriate for the board you’re using. If you’re using a LilyPad Arduino SimpleSnap, LilyPad Aruduino Simple Board, LilyPad Arduino Main Board, or one of the LilyPad ProtoSnap Development boards, choose “LilyPad Arduino w/ ATmega328″. If you’re using a LilyPad Arduino USB Board, choose “LilyPad Arduino USB”.


Edit
Sketch
Tools
Help
h_001 | Arduino 1.0.1
sketch_001
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
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
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
1

LilyPad Arduino w/ ATmega328 on /dev/tty.usbserial-A900J2Q7

Basic Programming Steps

Programming the LilyPad and executing your program takes place in four stages:

1. Write a program

First, you write your program. The Arduino environment lets you program in the programming language called C, which is what we’ll be using. You must obey the C language format for writing programs.

2. Compile the program

Second, you compile your program, translating the code that you wrote to code that the computer chip on the LilyPad can understand. If you’ve made any mistakes in your code, they will be detected during this phase and you will have to correct them and recompile your code before you move on to the next step.

3. Load the program onto the LilyPad

Third, you load your compiled code onto your LilyPad, moving the code from your computer to the LilyPad.

4. The program executes on the LilyPad

Once it has been loaded, your program executes on your LilyPad. The LilyPad does what you told it to do.

Basic Code Elements

So, that’s the process you go through in writing and executing a program, but how do programs work? Well, all programs execute in an orderly fashion, obeying your written commands (the lines in your program) line by line as they are written. There are a few different types of commands or code elements in most programming languages. To start with, we will use comments, simple statements and conditional statements.

Since we will be using the C programming language, this section will describe what these elements look like in the C language, and we will be using the Arduino Language Reference to get more detailed information on how each code element works.

1. Comments

Comments are pieces of code that are ignored by computers. You can use them to make notes in your code, but they’re only for reference. They don’t affect the behavior of your program. Anything written on a line after two slash characters // is a comment. Comments show up a greyish brown in the Arduino window. Here’s are a couple example comments:

```
// This is a comment. It doesn’t do anything.
// Comments just make code easier for you and other people to understand
```

Anything written between “/*” and “*/” characters is also a comment. Comments that start with “//” can only be on one line, but comments between the “/*” and “*/” characters can take up several lines. These types of comments will show up in that same greyish brown in the Arduino window so that you can always spot them easily. Here’s how we’d rewrite the comments above in this style:

```
/* This is a comment. It doesn’t do anything.
Comments just make code easier for you and other people to understand
*/
```

Throughout the rest of this page, we will use comments as you would normally use them in a program, to explain what pieces of code are doing. Check out the Comment Page of the Arduino Language Reference for more information.

2. Simple statements

A simple statement is a single instruction. It takes up a single line and is *always followed by a semi-colon*. Here are some example statements:

```
int ledPin = 13;        // This statement declares a variable called “ledPin” and sets its initial value to 13.
                        // We’ll use “ledPin” later to control the LED that is on the LilyPad
int switchPin = 2;      // This creates a variable called “switchPin” and sets its initial value to 2
```

```
                            // We’ll use “switchPin” later to read data from a switch attached to petal 2.
int switchValue;            // This creates a variable called “switchValue”, but does not initialize it
delay(1000);                // This tells the LilyPad to do nothing for 1000 milliseconds (1 second).
digitalWrite(ledPin,
HIGH);                      // This sets ledPin (pin 13 on the LilyPad as defined above) to +5 volts

                            // Setting pin 13 to +5 volts/HIGH will turn the LED on the LilyPad on.
digitalWrite(ledPin,
LOW);                       // This sets ledPin to 0 volts

                            // Setting pin 13 to 0 volts/LOW will turn the LED on the LilyPad off.
```

Notice how each statement is followed by a comment that describes what the statement is doing.
To get a more thorough explanation of what each statement is doing, we would turn again to the Arduino Language Reference. The reference contains an entry for every simple statement form in the Arduino language. To understand the “int ledPin = 13;” statement better, see the variable declaration section of the Arduino reference. To understand the “delay(1000);” statement better, see the delay page of the Arduino reference. To understand the “digitalWrite(ledPin, HIGH);” statement better, see the digitalWrite page of the Arduino reference.

3. Conditional statements

Conditional statements are statements that consist of a condition and then a series of statements in brackets like this { } that execute when the condition is met. See the Control Structures section of the Arduino Language Reference for more information. Here are some example conditional statements:

```
if (switchValue ==
LOW) {                      // This is the condition followed by the first bracket

digitalWrite(ledPin,        // This tells the LED on the LilyPad to turn on
HIGH);
}                           // Finally, you have the closing bracket
```

Notice how the condition above consists of the word “if” and then a condition in brackets like this ( ). This is the basic form that a conditional statement always takes. To make sense of conditional statements, read through them as though they were regular english. In the case above, you would read it “if the variable switchValue is equal to LOW then turn the LilyPad’s LED on.” Here’s another example:

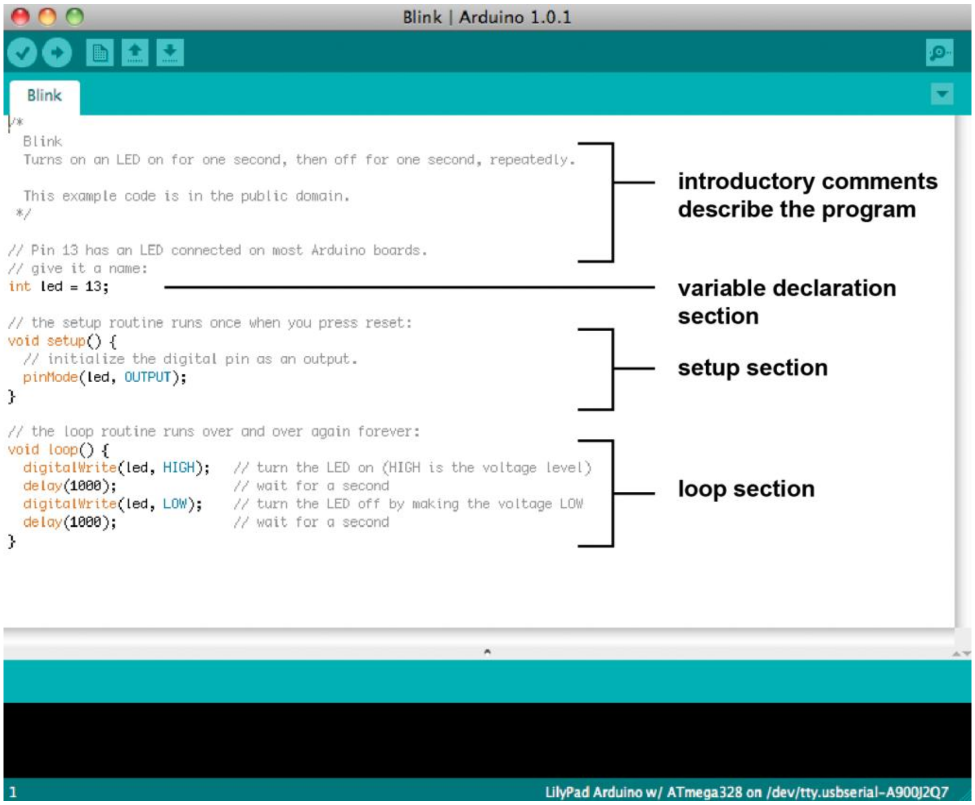
```
while (switchValue == LOW)
{                                   // This line is the condition followed by the first bracket
      digitalWrite(ledPin,
HIGH);                              // This tells the LED on the LilyPad to turn on
      delay(100);                   // This tells the LilyPad to wait for 100 milliseconds.
      digitalWrite(ledPin,
LOW);                               // This tells the LED on the LilyPad to turn off
      delay(100);
      switchValue =                 // This stores a new value into the variable switchValue
digitalRead(switchPin);
}
```

So, this statement would be read like this “as long as the variable switchValue is equal to LOW, do the following: turn the LilyPad’s LED on, then do nothing for 100 milliseconds (1/10th of a second), then turn the LilyPad’s LED off, then do nothing for 100 milliseconds, then store a new value into the variable switchValue”.

Arduino Program Structure

The previous section looked at the different code elements of the Arduino language. Now let’s see how to put these elements together into a complete program.

Each Arduino program has three main parts: a section where you declare variables, a “setup” section and a “loop” section. When your program executes, it will first define your variables, will then execute the setup section once and will then execute the loop section over and over. Here is a full example program in the Arduino window:


Blink | Arduino 1.0.1
Blink
introductory comments describe the program
variable declaration section
setup section
loop section
1
LilyPad Arduino w/ ATmega328 on /dev/tty.usbserial-A900J2Q7

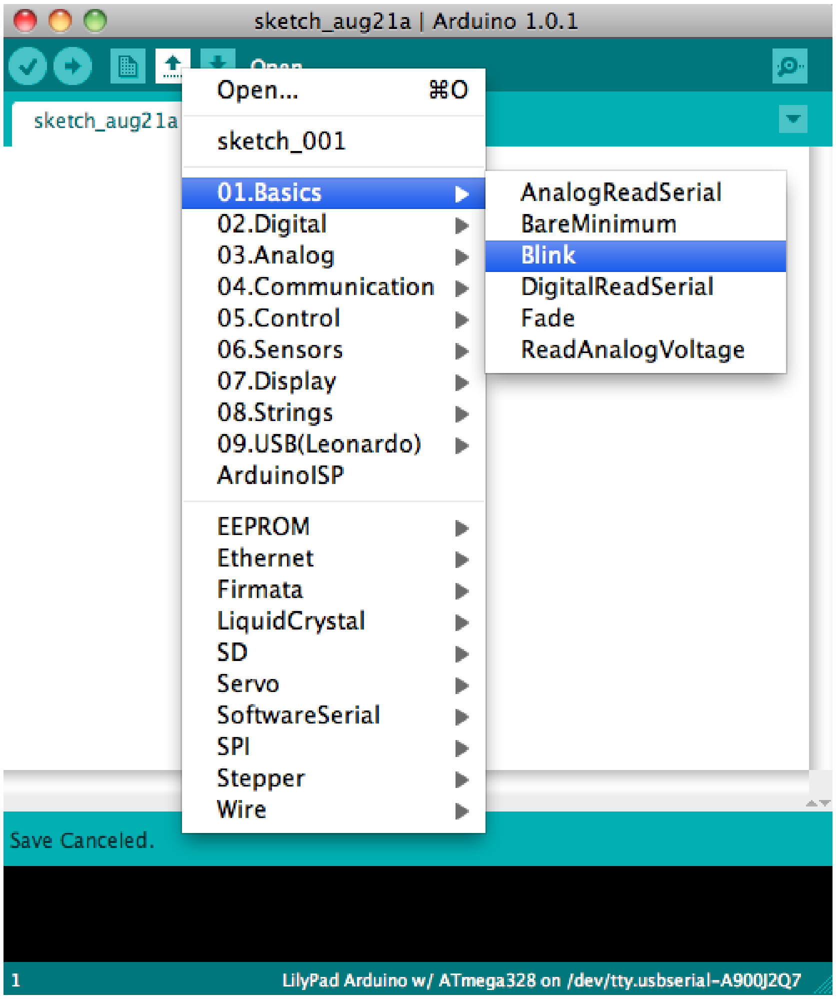
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

First Program

1. Open the Blink sketch

Click on the Open icon (the upward pointing arrow in the Arduino software) and then browse to “01.Basics–>Blink”. You can also access this same example through the “File” menu. Go to “File–>Examples–>Basics–>Blink”. This will open the Blink “sketch”

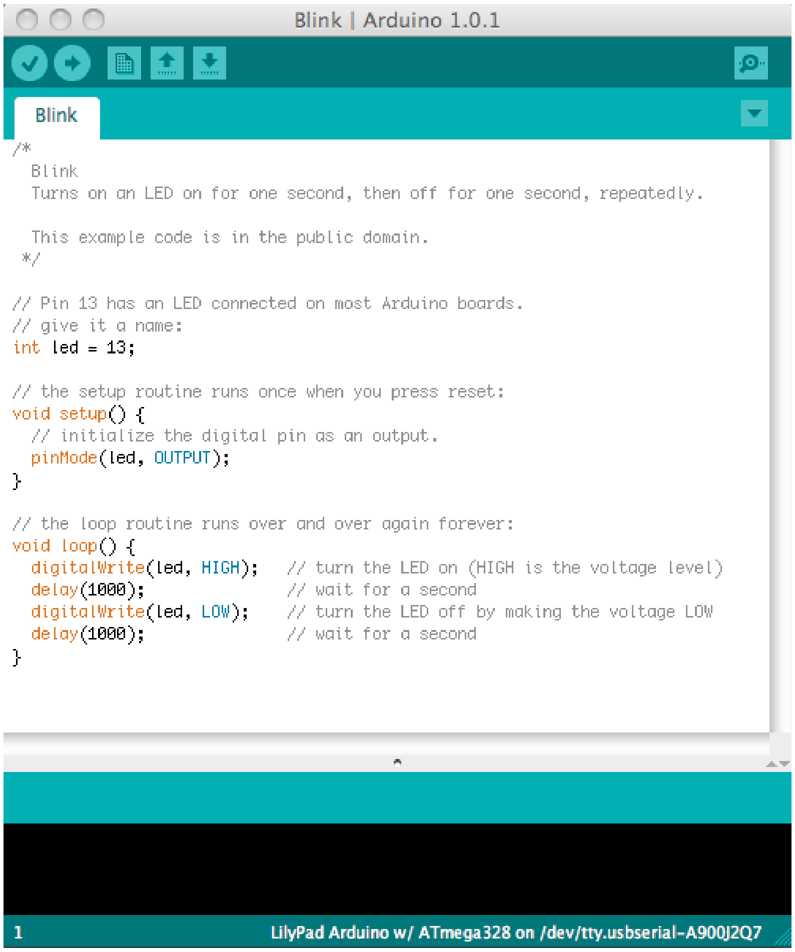
A little note about “sketches”: programs in Arduino are called sketches. All of your programs will be saved in your Arduino “sketchbook”.



Save Canceled.

1 LilyPad Arduino w/ ATmega328 on /dev/tty.usbserial-A900J2Q7

Your window should now look like the one below, with the Blink sketch loaded into the top half of the screen under a tab labled “Blink”.

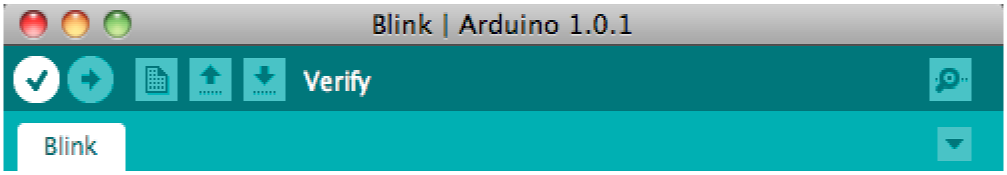


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

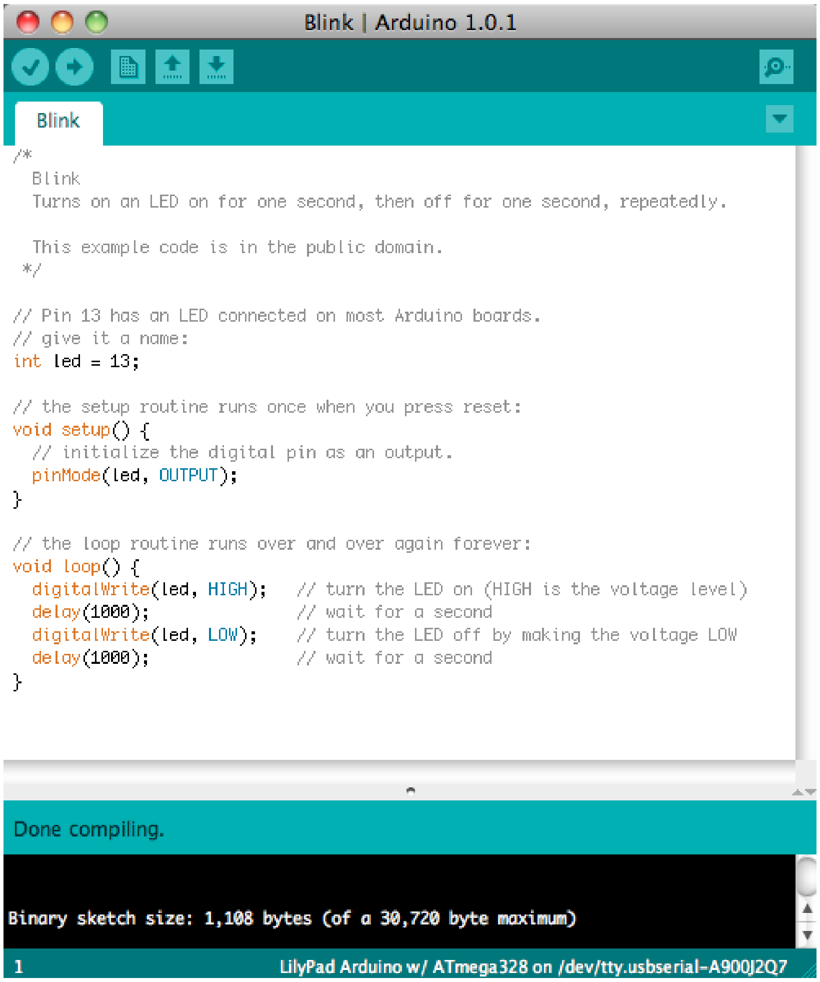
```
1                LilyPad Arduino w/ ATmega328 on /dev/tty.usbserial-A900J2Q7
```

2. Compile the Blink sketch.

Click on the Verify/Compile icon (the check button in upper left corner of the Arduino window). When you hover your mouse over the icon, text saying “Verify” will appear at the top of the Arduino window.



The compiling process will take a few seconds. Once it’s done the Arduino window will tell you it’s “Done compiling” and will tell you the size of your sketch (the amount of memory it takes up). Your window should look something like this:



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

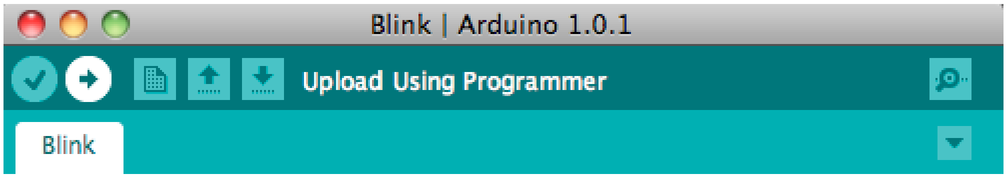
Done compiling.

```
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
```

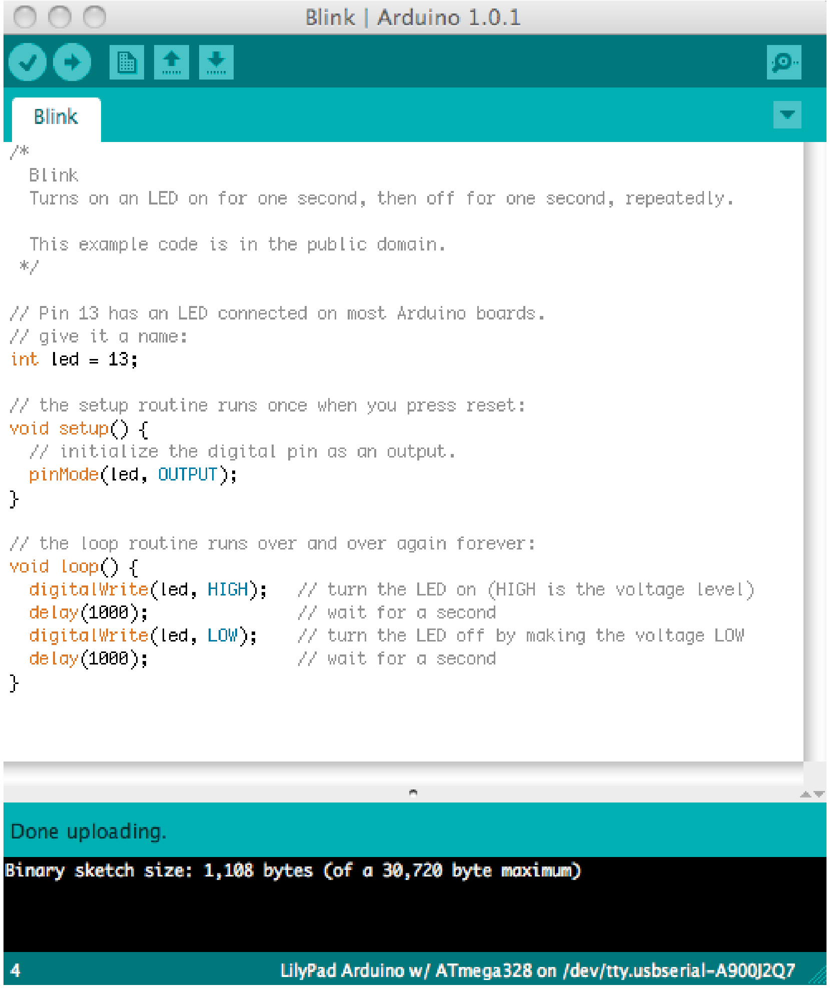
1 LilyPad Arduino w/ ATmega328 on /dev/tty.usbserial-A900J2Q7

3. Load the sketch onto your LilyPad

Click on the Upload button in the software (the right pointing arrow in the Arduino window). When you move your mouse over this icon, text saying “Upload” (or “Upload Using Programmer”) will appear at the top of the Arduino window.



If all goes well, your window should look something like the one below when you’re done uploading. It will tell you that the upload is finished and give you the size of your sketch again.

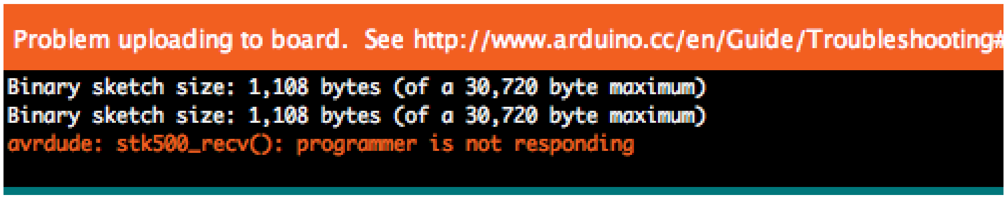


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Done uploading.

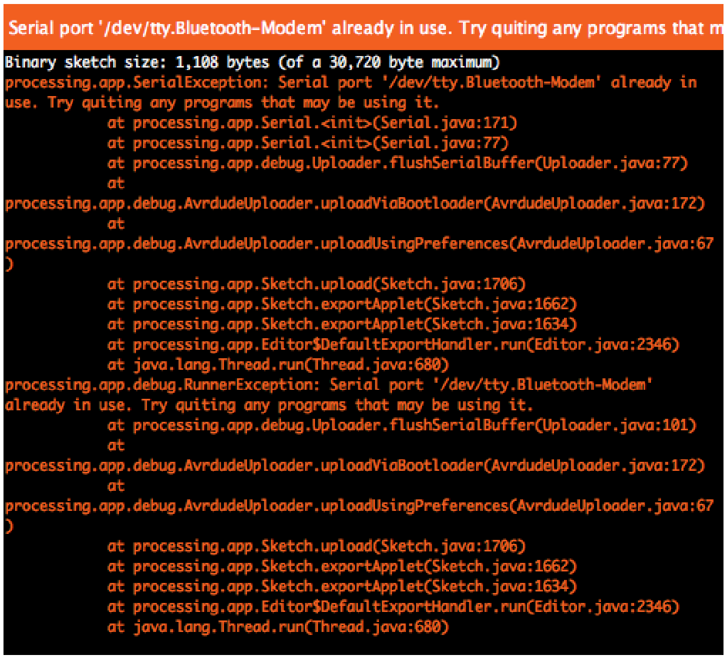
```
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
```

Here are a few errors that you might encounter. These will appear in the bottom half of your screen.



Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#

```
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
avrdude: stk500_recv(): programmer is not responding
```

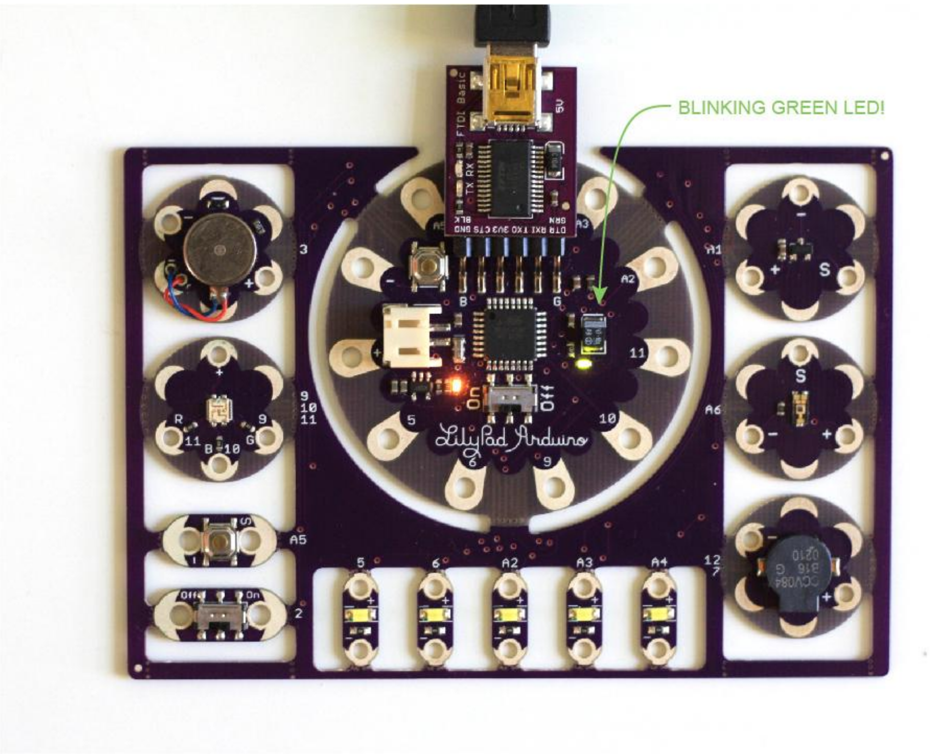


```
Serial port '/dev/tty.Bluetooth-Modem' already in use. Try quiting any programs that m
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
processing.app.SerialException: Serial port '/dev/tty.Bluetooth-Modem' already in
use. Try quiting any programs that may be using it.
	at processing.app.Serial.<init>(Serial.java:171)
	at processing.app.Serial.<init>(Serial.java:77)
	at processing.app.debug.Uploader.flushSerialBuffer(Uploader.java:77)
	at
processing.app.debug.AvrdudeUploader.uploadViaBootloader(AvrdudeUploader.java:172)
	at
processing.app.debug.AvrdudeUploader.uploadUsingPreferences(AvrdudeUploader.java:67
)
	at processing.app.Sketch.upload(Sketch.java:1706)
	at processing.app.Sketch.exportApplet(Sketch.java:1662)
	at processing.app.Sketch.exportApplet(Sketch.java:1634)
	at processing.app.Editor$DefaultExportHandler.run(Editor.java:2346)
	at java.lang.Thread.run(Thread.java:680)
processing.app.debug.RunnerException: Serial port '/dev/tty.Bluetooth-Modem'
already in use. Try quiting any programs that may be using it.
	at processing.app.debug.Uploader.flushSerialBuffer(Uploader.java:101)
	at
processing.app.debug.AvrdudeUploader.uploadViaBootloader(AvrdudeUploader.java:172)
	at
processing.app.debug.AvrdudeUploader.uploadUsingPreferences(AvrdudeUploader.java:67
)
	at processing.app.Sketch.upload(Sketch.java:1706)
	at processing.app.Sketch.exportApplet(Sketch.java:1662)
	at processing.app.Sketch.exportApplet(Sketch.java:1634)
	at processing.app.Editor$DefaultExportHandler.run(Editor.java:2346)
	at java.lang.Thread.run(Thread.java:680)
```

You didn’t select the correct serial port. Or, maybe your LilyPad isn’t actually attached to your computer. Double check your connections and see the setup page for instructions on selecting the serial port.

Other errors? Check out ladyada’s fabulous help page and the Arduino Troubleshooting guide.

4. Hey, it blinks!



5. Play with modifying the code to get different behavior

- Can you get the LED to blink faster or slower?
- Can you get the it to blink in a heartbeat pattern?
- How about getting it to flicker like a candle?

Next Program: Sensing (Switches)

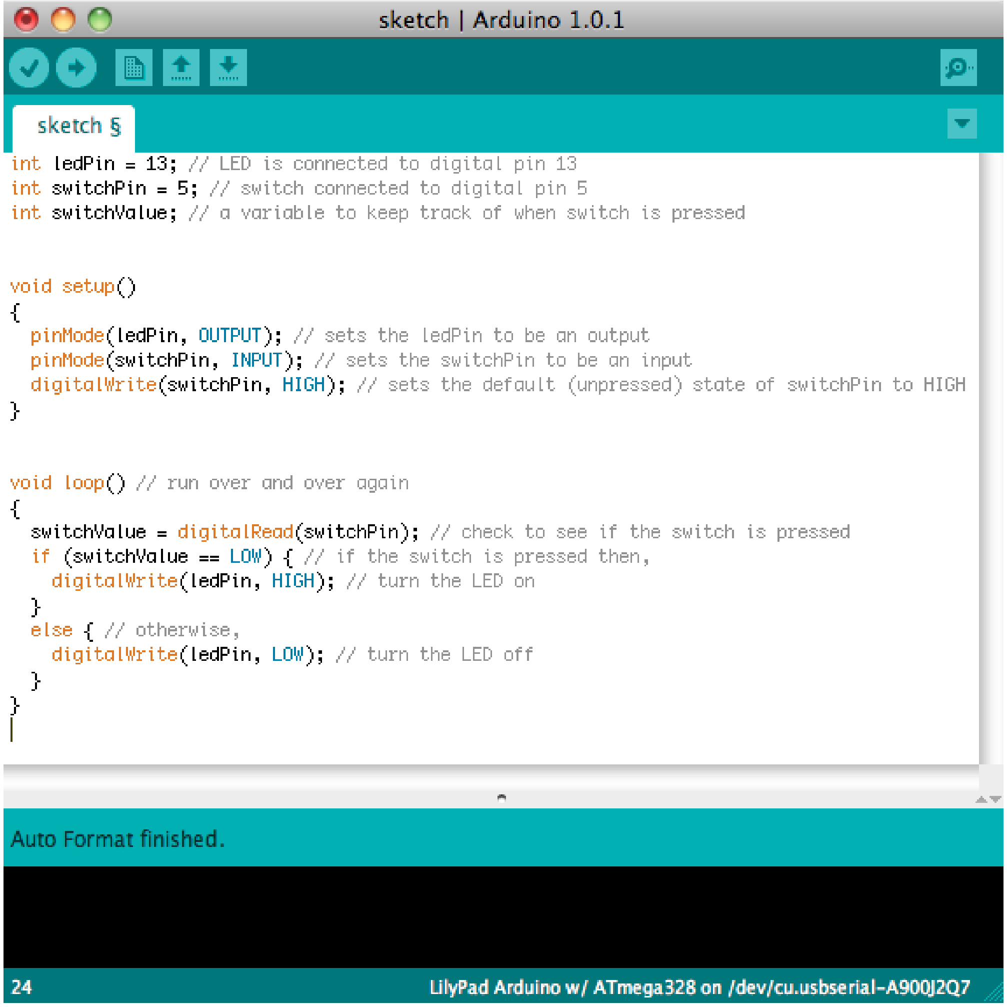
1. Copy this sample code into an Arduino window

(Switch Code)

```
int ledPin = 13;                // LED is connected to digital pin 13
int switchPin = 2;              // switch connected to digital pin 2
int switchValue;                // a variable to keep track of when switch
                                is pressed

void setup()
{
      pinMode(ledPin,           // sets the ledPin to be an output
OUTPUT);
      pinMode(switchPin,        // sets the switchPin to be an input
INPUT);

digitalWrite(switchPin,         // sets the default (unpressed) state of
HIGH);                          switchPin to HIGH
}

void loop()                     // run over and over again
{
      switchValue =             // check to see if the switch is pressed
digitalRead(switchPin);
      if (switchValue ==        // if the switch is pressed then,
LOW) {
                                // turn the LED on
digitalWrite(ledPin, HIGH);
      }
      else {                    // otherwise,
                                // turn the LED off
digitalWrite(ledPin, LOW);
      }
}
```

2. Format the Code

Under the Tools menu, select Auto Format. After you do this, align all of your comments (the statements in grey-brown following "//" on each line) so that they are in readable columns on the right hand side of the screen. This will help you read through the code. Here's what my Arduino window looked like after I formatted everything:



```
int ledPin = 13; // LED is connected to digital pin 13
int switchPin = 5; // switch connected to digital pin 5
int switchValue; // a variable to keep track of when switch is pressed


void setup()
{
  pinMode(ledPin, OUTPUT); // sets the ledPin to be an output
  pinMode(switchPin, INPUT); // sets the switchPin to be an input
  digitalWrite(switchPin, HIGH); // sets the default (unpressed) state of switchPin to HIGH
}


void loop() // run over and over again
{
  switchValue = digitalRead(switchPin); // check to see if the switch is pressed
  if (switchValue == LOW) { // if the switch is pressed then,
    digitalWrite(ledPin, HIGH); // turn the LED on
  }
  else { // otherwise,
    digitalWrite(ledPin, LOW); // turn the LED off
  }
}
```

Auto Format finished.

24 LilyPad Arduino w/ ATmega328 on /dev/cu.usbserial-A900J2Q7

Read through the code to get a sense of what it is doing. The comments at the end of each line should help you understand what's happening. Note that in the code we are listening for a LOW signal on the switchPin. We turn on the LED when the switchPin is attached to ground. As was mentioned earlier, when we put the two alligator clips together this is exactly what is happening: the switchPin is attached to ground via the clips. So, let's test it out in the real world…

3. Load the code onto the LilyPad

Compile the code and load it onto the LilyPad. Do this by hitting the upload button in the Arduino window (that's the right pointing arrow at the top of the Arduino window).

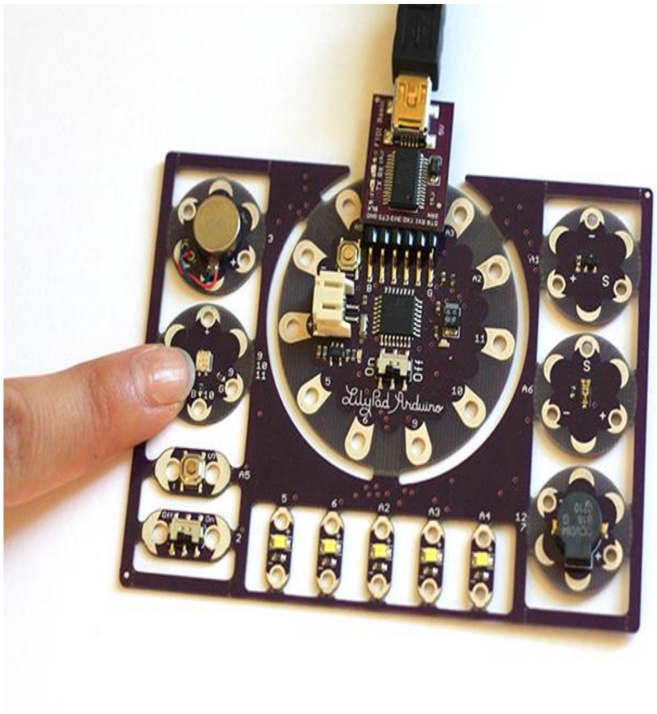
4. See what happens when you close the switch!

If you're using the LilyPad Proto Snap Development Board, turn on the pre-wired switch. The green light (next to pin 11) should turn on. Trying changing the code so you can use the button on pin A5 to turn on the green light.

5. Play with modifying the code to get different behavior

- Can you get the LED to turn on when the switch is open and off when the switch is closed? (Basically swapping the behavior of the sample code.)
- Can you get the LED to blink quickly while the switch is closed and turn off when the switch is open?
- Something a little more challenging… can you get the LED to toggle on and off with each press of the switch? That is, the first time you press the switch, the LED turns on, the second time you press the switch it turns off, and so on?

Color (LEDs)



1. Attach the LilyPad to your computer and start the Arduino software

2. Copy this sample code into an Arduino window

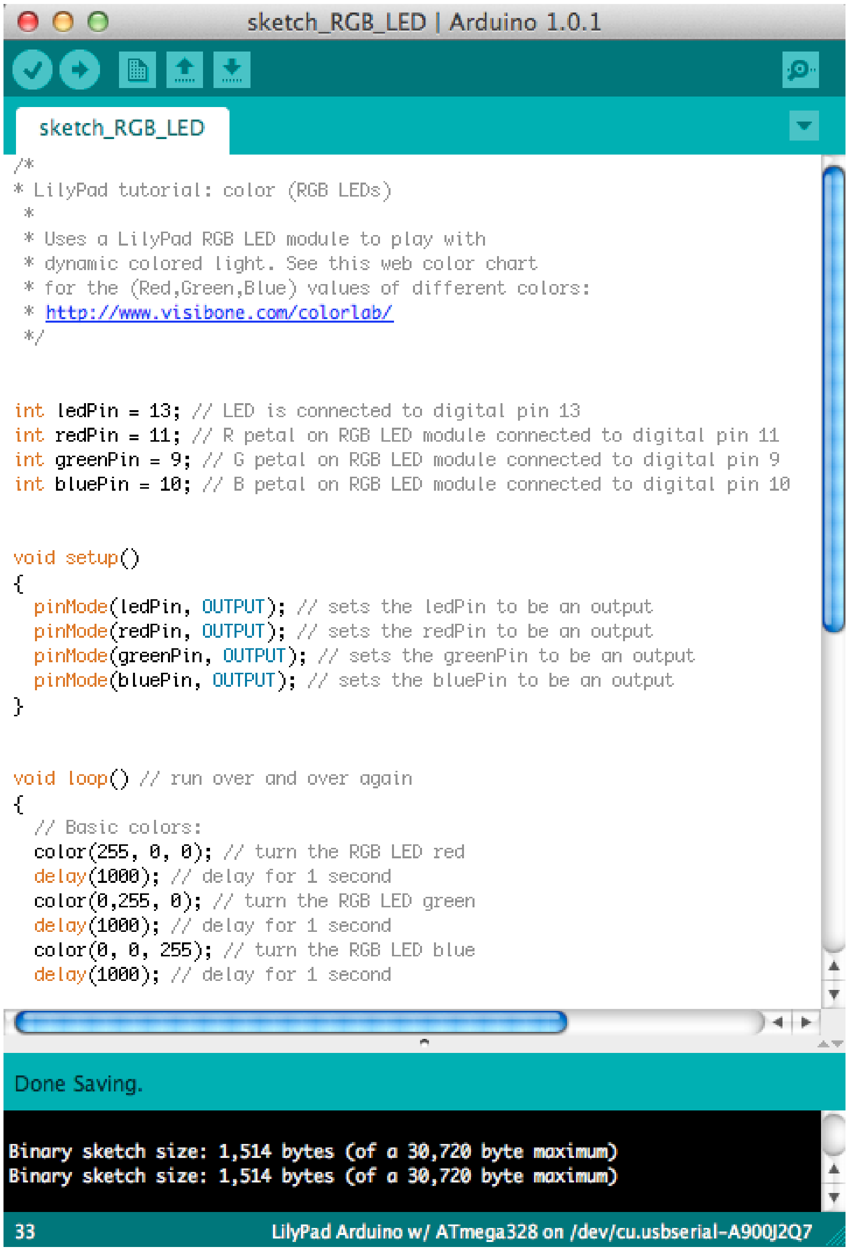
(RGB LED example code)

```
/*
*LilyPad tutorial: color (RGB LEDs)
*
*Uses a LilyPad RGB LED module to play with
*dynamic colored light. See this web color
chart
*for the (Red, Green, Blue) values of
different colors:
*http://www.visibone.com/colorlab/
*/

int ledPin = 13;                // LED is connected to digital pin 13
                                // R petal on RGB LED module connected to
int redPin = 11;                digital pin 11
                                // G petal on RGB LED module connected to
int greenPin = 9;               digital pin 9
                                // B petal on RGB LED module connected to
int bluePin = 10;               digital pin 10

void setup()
```

```
{
        pinMode(ledPin,                 // sets the ledPin to be an output
OUTPUT);
        pinMode(redPin,                 // sets the redPin to be an output
OUTPUT);
        pinMode(greenPin,               // sets the greenPin to be an output
OUTPUT);
        pinMode(bluePin,                // sets the bluePin to be an input
OUTPUT);
}

void loop()                      // run over and over again
{
         //Basic colors:
         color(255,0,0);        //turn the RGB LED red
         delay(1000);           //delay for 1 second
         color(0,255,0);        //turn the RGB LED green
         delay(1000);           //delay for 1 second
         color(0,0,255);        //turn the RGB LED blue
         delay(1000);           //delay for 1 second

         //Example blended
colors:
         color(255,255,0);      //turn the RGB LED yellow
         delay(1000);           //delay for 1 second
         color(255,255,255);    //turn the RGB LED white
         delay(1000);           //delay for 1 second
         color(128,0,255);      //turn the RGB LED purple
         delay(1000);           //delay for 1 second
         color(0,0,0);          //turn the RGB LED off
         delay(1000);           //delay for 1 second
}

void color (unsigned char red, unsigned char green,    //the color generating
unsigned char blue)                                     function
{
         analogWrite(redPin,
255-red);
         analogWrite(bluePin,
255-blue);
         analogWrite(greenPin,
255-green);
}
```

3. Format the code

Under the Tools menu, select Auto Format. After you do this, align all of your comments (the statements in grey-brown following "//" on each line) so that they are in readable columns on the right hand side of the screen. This will help you read through the code. Here's what my Arduino window looked like after I formatted everything:



```
/*
* LilyPad tutorial: color (RGB LEDs)
 *
 * Uses a LilyPad RGB LED module to play with
 * dynamic colored light. See this web color chart
 * for the (Red,Green,Blue) values of different colors:
 * http://www.visibone.com/colorlab/
 */


int ledPin = 13; // LED is connected to digital pin 13
int redPin = 11; // R petal on RGB LED module connected to digital pin 11
int greenPin = 9; // G petal on RGB LED module connected to digital pin 9
int bluePin = 10; // B petal on RGB LED module connected to digital pin 10


void setup()
{
  pinMode(ledPin, OUTPUT); // sets the ledPin to be an output
  pinMode(redPin, OUTPUT); // sets the redPin to be an output
  pinMode(greenPin, OUTPUT); // sets the greenPin to be an output
  pinMode(bluePin, OUTPUT); // sets the bluePin to be an output
}


void loop() // run over and over again
{
  // Basic colors:
  color(255, 0, 0); // turn the RGB LED red
  delay(1000); // delay for 1 second
  color(0,255, 0); // turn the RGB LED green
  delay(1000); // delay for 1 second
  color(0, 0, 255); // turn the RGB LED blue
  delay(1000); // delay for 1 second
```

Done Saving.

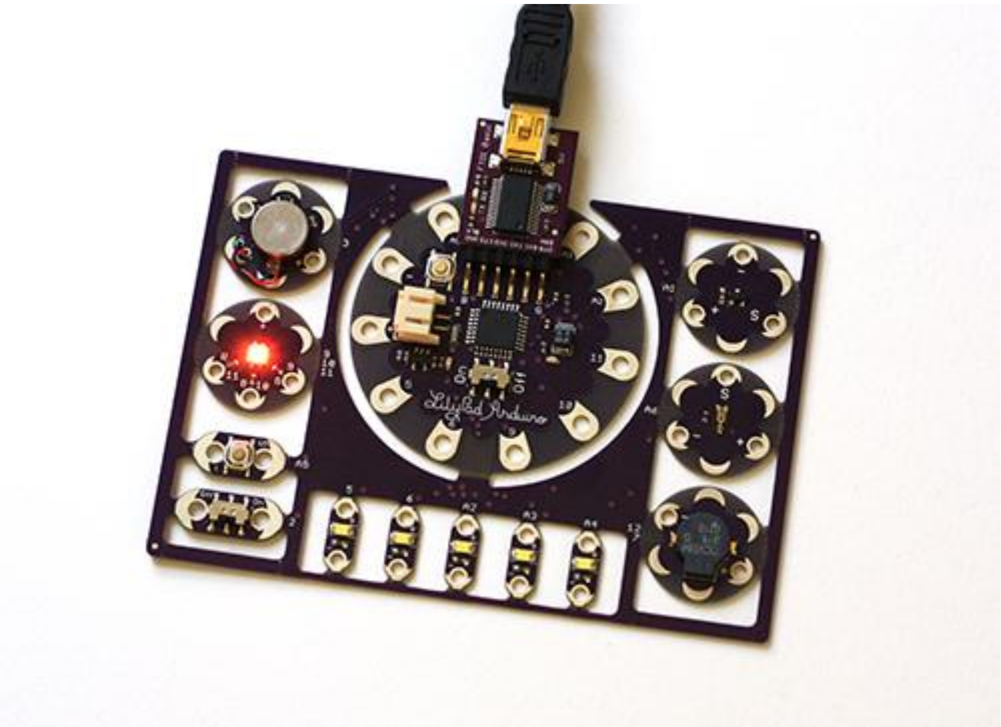
```
Binary sketch size: 1,514 bytes (of a 30,720 byte maximum)
Binary sketch size: 1,514 bytes (of a 30,720 byte maximum)
```

33 LilyPad Arduino w/ ATmega328 on /dev/cu.usbserial-A900J2Q7

4. Load the code onto the LilyPad

Compile the code and load it onto the LilyPad. Do this by hitting the upload button in the Arduino window (that’s the right pointing arrow at the top of the Arduino window).

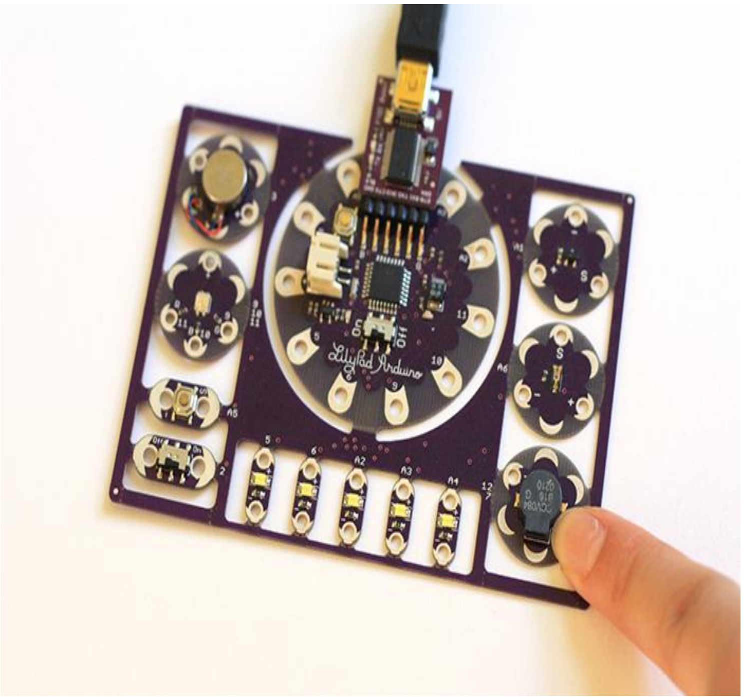
5. See what happens!



6. Play with modifying the code to get different behavior

- Can you get the RGB LED to change color when you press a switch?
- Can you write a function that slowly changes the RGB LED's color from red to blue?

Sound



1. Copy this sample code into an Arduino window

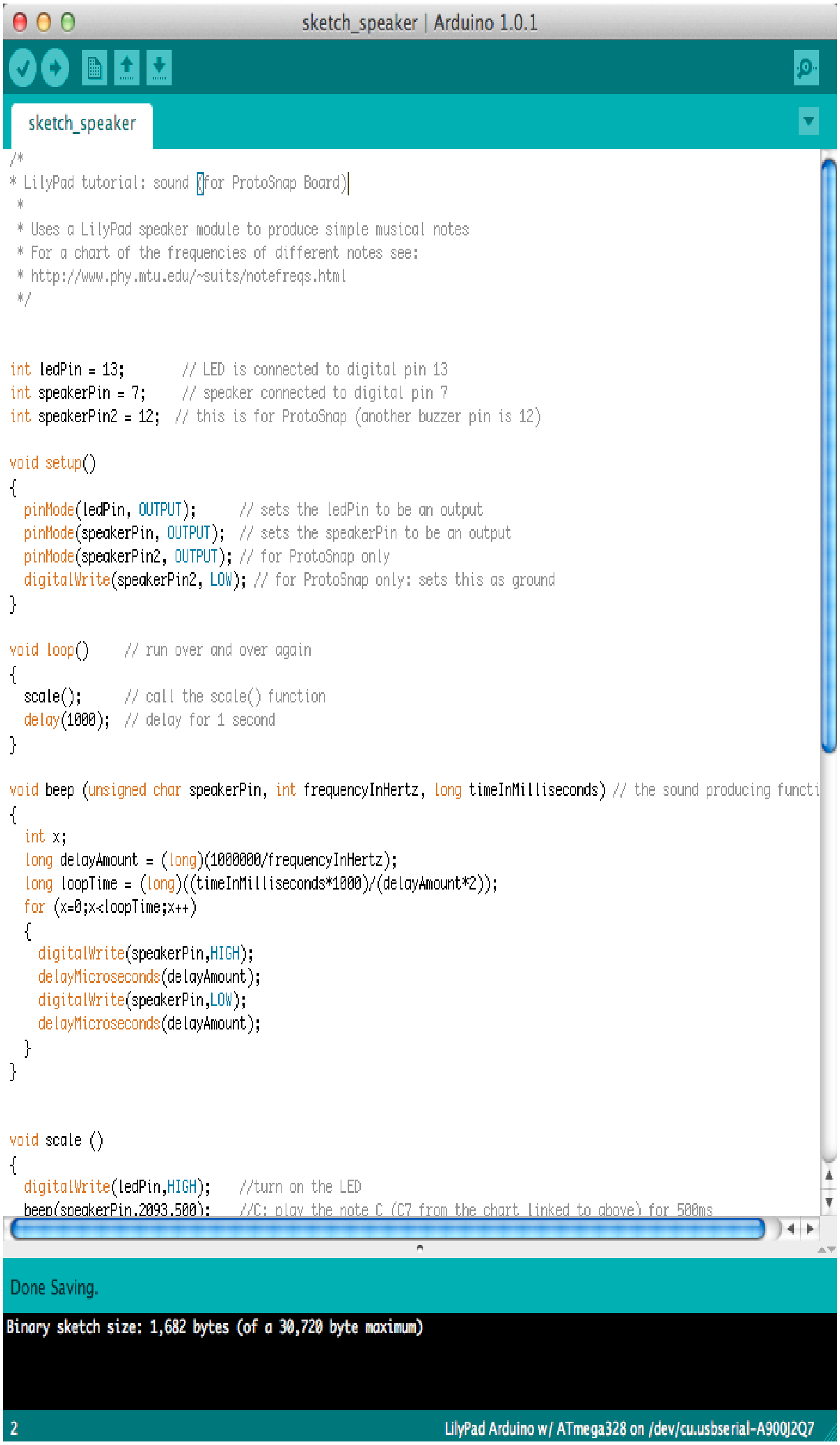
Copy and paste the code into an empty Arduino window.

speaker example code (ProtoSnap)

```
/*
*LilyPad tutorial: sound (for ProtoSnap Board)
*
*Uses a LilyPad speaker module to produce
simple musical notes
*For a chart of the frequencies of different
notes see:
* http://www.phy.mtu.edu/~suits/notefreqs.html
*/

int ledPin = 13;                // LED is connected to digital pin 13
int speakerPin = 7;             // speaker connected to digital pin 7
                                // this is for ProtoSnap only (another buzzer
int speakerPin2 = 12;           pin is 12, we'll ground this)


void setup()
{
        pinMode(ledPin,
OUTPUT);                        // sets the ledPin to be an output
        pinMode(speakerPin,
OUTPUT);                        // sets the speakerPin to be an output
        pinMode(speakerPin2,
OUTPUT);                        // For ProtoSnap only

                                // For ProtoSnap only: sets this as ground
digitalWrite(speakerPin2,
LOW);
}

void loop()                       // run over and over again
{
         scale();               //call the scale() function
         delay(1000);           //delay for 1 second
}

void beep (unsigned char speakerPin, int                //the sound producing
frequencyinHertz, long timeInMilliseconds)              function
{
         int x;
         long
delayAmount=(long)(1000000/frequencyInHertz);
         long loopTime =
(long)((timeInMilliseconds*1000)/(delayAmount*2));
         for (x=0;x<loopTime;x++)
         {
            digitalWrite(speakerPin,HIGH);
            delayMicroseconds(delayAmount);
            digitalWrite(speakerPin,LOW);
            delayMicroseconds(delayAmount);
          }
 }

void scale()
 {

  digitalWrite(ledPin,HIGH);    //turn on the LED

  beep(speakerPin,2093,500);    //C: play the note C (C7 from the chart linked
                                to above) for 500ms

  beep(speakerPin,2349,500);    //D

  beep(speakerPin,2637,500);    //E

  beep(speakerPin,2793,500);    //F

  beep(speakerPin,3136,500);    //G

  beep(speakerPin,3520,500);    //A

  beep(speakerPin,3951,500);    //B

  beep(speakerPin,4186,500);    //C

  digitalWrite(ledPin,LOW);     //turn off the LED
 }
```

2. Format the code

Under the Tools menu, select Auto Format. After you do this, align all of your comments (the statements in grey-brown following “//” on each line) so that they are in readable columns on the right hand side of the screen. This will help you read through the code. Here’s what my Arduino window looked like after I formatted everything:



```
/*
* LilyPad tutorial: sound (for ProtoSnap Board)
 *
 * Uses a LilyPad speaker module to produce simple musical notes
 * For a chart of the frequencies of different notes see:
 * http://www.phy.mtu.edu/~suits/notefreqs.html
 */


int ledPin = 13;        // LED is connected to digital pin 13
int speakerPin = 7;     // speaker connected to digital pin 7
int speakerPin2 = 12;  // this is for ProtoSnap (another buzzer pin is 12)

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the ledPin to be an output
  pinMode(speakerPin, OUTPUT);  // sets the speakerPin to be an output
  pinMode(speakerPin2, OUTPUT); // for ProtoSnap only
  digitalWrite(speakerPin2, LOW); // for ProtoSnap only: sets this as ground
}

void loop()     // run over and over again
{
  scale();      // call the scale() function
  delay(1000);  // delay for 1 second
}

void beep (unsigned char speakerPin, int frequencyInHertz, long timeInMilliseconds) // the sound producing functi
{
  int x;
  long delayAmount = (long)(1000000/frequencyInHertz);
  long loopTime = (long)((timeInMilliseconds*1000)/(delayAmount*2));
  for (x=0;x<loopTime;x++)
  {
    digitalWrite(speakerPin,HIGH);
    delayMicroseconds(delayAmount);
    digitalWrite(speakerPin,LOW);
    delayMicroseconds(delayAmount);
  }
}


void scale ()
{
  digitalWrite(ledPin,HIGH);    //turn on the LED
  beep(speakerPin,2093,500);    //C: play the note C (C7 from the chart linked to above) for 500ms
```

Done Saving.

```
Binary sketch size: 1,682 bytes (of a 30,720 byte maximum)
```

2   LilyPad Arduino w/ ATmega328 on /dev/cu.usbserial-A900J2Q7

3. Load the code onto the LilyPad

Compile the code and load it onto the LilyPad.

4. Listen!

Your LilyPad speaker should be singing out a “do re me fa so la ti do” scale. The LED should be on while the scale is playing. If you’re not getting sound, double check your alligator clip connections.

5. Making sense of things: sound

So, a little bit about sound…Sound is air vibration. Different pitches are produced with different frequencies of vibration. Here’s a good little introduction to sound: http://www.fearofphysics.com/Sound/dist.html.

Note	Frequency (Hz)	Wavelength (cm)
C_0	16.35	2100.
$C^{\#}_0/D^{b}_0$	17.32	1990.
D_0	18.35	1870.
$D^{\#}_0/E^{b}_0$	19.45	1770.
E_0	20.60	1670.
F_0	21.83	1580.
$F^{\#}_0/G^{b}_0$	23.12	1490.
G_0	24.50	1400.
$G^{\#}_0/A^{b}_0$	25.96	1320.
A_0	27.50	1250.
$A^{\#}_0/B^{b}_0$	29.14	1180.
B_0	30.87	1110.
C_1	32.70	1050.
$C^{\#}_1/D^{b}_1$	34.65	996.
D_1	36.71	940.
$D^{\#}_1/E^{b}_1$	38.89	887.
E_1	41.20	837.
F_1	43.65	790.
$F^{\#}_1/G^{b}_1$	46.25	746.
G_1	49.00	704.
$G^{\#}_1/A^{b}_1$	51.91	665.
A_1	55.00	627.
$A^{\#}_1/B^{b}_1$	58.27	592.
B_1	61.74	559.
C_2	65.41	527.
$C^{\#}_2/D^{b}_2$	69.30	498.
D_2	73.42	470.

$D^{\#}_{2}/E^{b}_{2}$	77.78	444.
E_{2}	82.41	419.
F_{2}	87.31	395.
$F^{\#}_{2}/G^{b}_{2}$	92.50	373.
G_{2}	98.00	352.
$G^{\#}_{2}/A^{b}_{2}$	103.83	332.
A_{2}	110.00	314.
$A^{\#}_{2}/B^{b}_{2}$	116.54	296.
B_{2}	123.47	279.
C_{3}	130.81	264.
$C^{\#}_{3}/D^{b}_{3}$	138.59	249.
D_{3}	146.83	235.
$D^{\#}_{3}/E^{b}_{3}$	155.56	222.
E_{3}	164.81	209.
F_{3}	174.61	198.
$F^{\#}_{3}/G^{b}_{3}$	185.00	186.
G_{3}	196.00	176.
$G^{\#}_{3}/A^{b}_{3}$	207.65	166.
A_{3}	220.00	157.
$A^{\#}_{3}/B^{b}_{3}$	233.08	148.
B_{3}	246.94	140.
C_{4}	261.63	132.
$C^{\#}_{4}/D^{b}_{4}$	277.18	124.
D_{4}	293.66	117.
$D^{\#}_{4}/E^{b}_{4}$	311.13	111.
E_{4}	329.63	105.
F_{4}	349.23	98.8
$F^{\#}_{4}/G^{b}_{4}$	369.99	93.2
G_{4}	392.00	88.0
$G^{\#}_{4}/A^{b}_{4}$	415.30	83.1
A_{4}	440.00	78.4
$A^{\#}_{4}/B^{b}_{4}$	466.16	74.0
B_{4}	493.88	69.9
C_{5}	523.25	65.9
$C^{\#}_{5}/D^{b}_{5}$	554.37	62.2
D_{5}	587.33	58.7
$D^{\#}_{5}/E^{b}_{5}$	622.25	55.4
E_{5}	659.26	52.3

F_5	698.46	49.4
$F^{\#}_5/G^{b}_5$	739.99	46.6
G_5	783.99	44.0
$G^{\#}_5/A^{b}_5$	830.61	41.5
A_5	880.00	39.2
$A^{\#}_5/B^{b}_5$	932.33	37.0
B_5	987.77	34.9
C_6	1046.50	33.0
$C^{\#}_6/D^{b}_6$	1108.73	31.1
D_6	1174.66	29.4
$D^{\#}_6/E^{b}_6$	1244.51	27.7
E_6	1318.51	26.2
F_6	1396.91	24.7
$F^{\#}_6/G^{b}_6$	1479.98	23.3
G_6	1567.98	22.0
$G^{\#}_6/A^{b}_6$	1661.22	20.8
A_6	1760.00	19.6
$A^{\#}_6/B^{b}_6$	1864.66	18.5
B_6	1975.53	17.5
C_7	2093.00	16.5
$C^{\#}_7/D^{b}_7$	2217.46	15.6
D_7	2349.32	14.7
$D^{\#}_7/E^{b}_7$	2489.02	13.9
E_7	2637.02	13.1
F_7	2793.83	12.3
$F^{\#}_7/G^{b}_7$	2959.96	11.7
G_7	3135.96	11.0
$G^{\#}_7/A^{b}_7$	3322.44	10.4
A_7	3520.00	9.8
$A^{\#}_7/B^{b}_7$	3729.31	9.3
B_7	3951.07	8.7
C_8	4186.01	8.2
$C^{\#}_8/D^{b}_8$	4434.92	7.8
D_8	4698.64	7.3
$D^{\#}_8/E^{b}_8$	4978.03	6.9

The LilyPad speaker (or “buzzer”) module works the same way any speaker does. It contains a material or device that moves in response to electricity. It is in one position when there is no voltage across the + and – petals on the speaker (when both petals are attached to – or set LOW), and another position when there is voltage across the petals (when one petal is attached to +5V or set HIGH and the other is attached to -). To use this device to create air vibrations, we need to get it to move between those two positions quickly, hundreds to thousands of times per second. How quickly we flip between the two positions (the frequency of the movement) will determine the pitch of the sound.

6. Making sense of things: the code

Now, let's look at the code. The beep() function is the main sound producing chunk of code. Notice how the main part of the function is a loop where we set the speakerPin HIGH for a moment and then LOW for a moment. This is how we are producing sound. The pitch of the sound is determined by the "frequencyInHertz" variable and the duration of the sound is determined by the "timeInMilliseconds" variable.

The scale() function gives an example of how to use the beep() function to produce specific musical notes. I used the frequency chart from http://www.phy.mtu.edu/~suits/notefreqs.html to find the frequencies for a basic musical scale. You can use that same chart to build your own musical libraries.

7. Play with modifying the code to get different behavior

- Can you get the LilyPad to play a tune when you press a switch?
- Can you write a function that plays twinkle twinkle little star?
- Can you write a function that creates a siren like sound (with smooth transitions between notes)?