LilyPad Development

// Programming the Dev Board:

For setting up Arduino on your own computer:

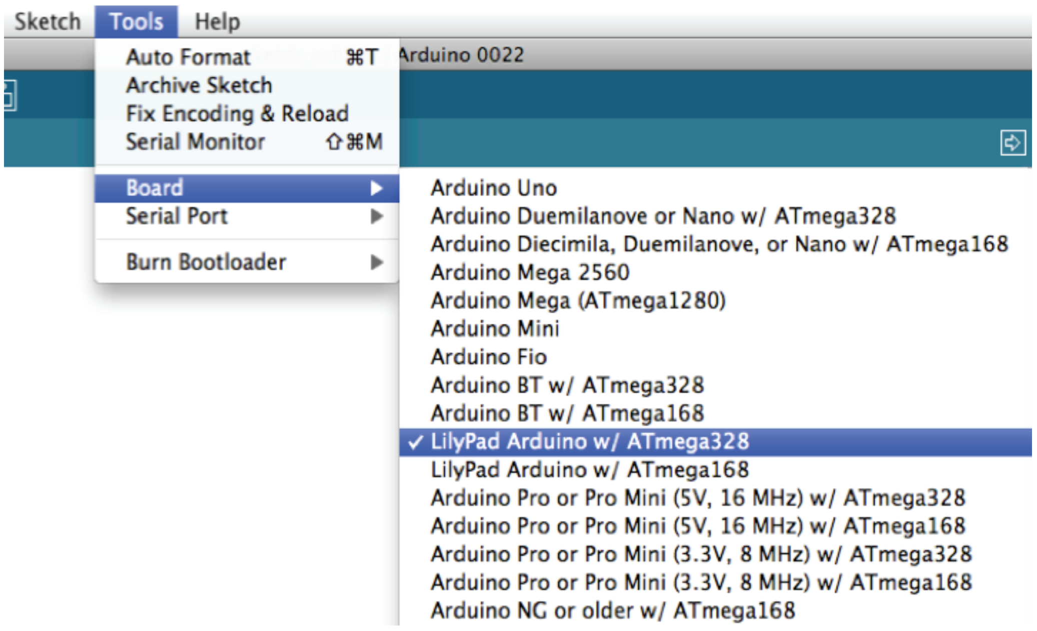
If you are having trouble when you first install Arduino and try to use it on your computer you may have issues with your drivers. Don't worry, you'll only have to fix this once. For detailed instructions on how to fix this go here:

For Windows: http://arduino.cc/en/Guide/Windows
For Mac: http://arduino.cc/en/Guide/MacOSX

For using the LilyPad Dev Board (and other microcontrollers) after setting up Arduino:

When hooking up your hardware (USB connection from computer to arduino) check two different things, the Board Type and the COM port:

Board Type: The LilyPad has the LilyPad Arduino w/ ATmega328 Bootloader (a bootloader is like an operating system) on it. Go to Board under Tools and select LilyPad Arduino w/ ATmega 328.

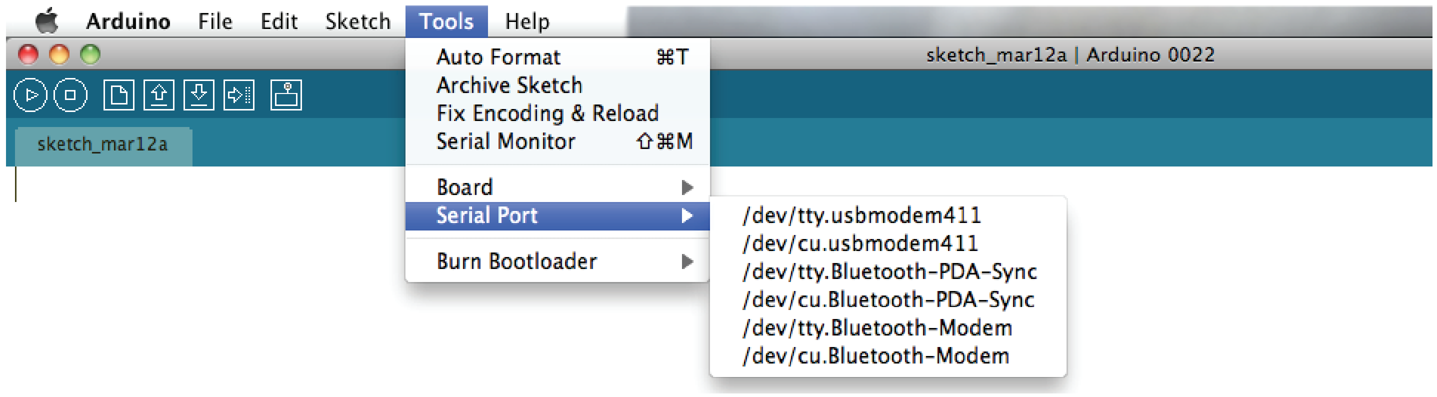

Sketch
Tools
Help
Arduino 0022
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Burn Bootloader
Arduino Uno
Arduino Duemilanove or Nano w/ ATmega328
Arduino Diecimila, Duemilanove, or Nano w/ ATmega168
Arduino Mega 2560
Arduino Mega (ATmega1280)
Arduino Mini
Arduino 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
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

Name:
Date:

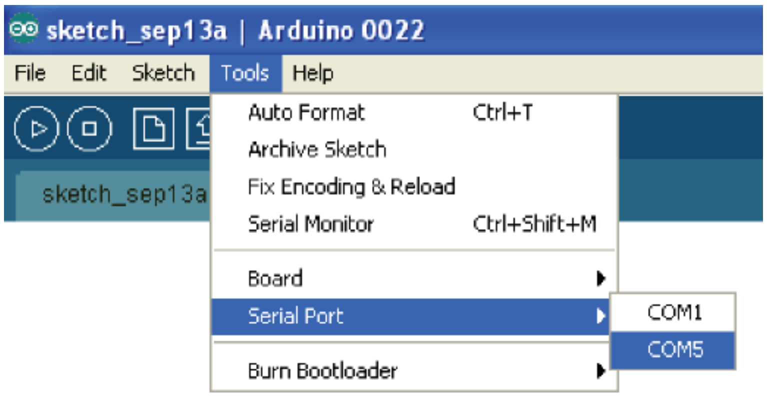
// Programming the Dev Board:

COM port: One easy way to find your COM port is to plug in your microcontroller, check the available COM ports under the COM port menu, exit the COM port menu (simply click anywhere in the coding area), unplug your microcontroller and then check the available COM ports again. The COM port that is missing is the COM port that your hardware is connected to when you plug it back in.

Selecting a COM port on a MAC:
(MACs are easier and usually it will be the top serial port.)



Selecting a COM port on a PC:
(This will always be a COM#)



Name:
Date:

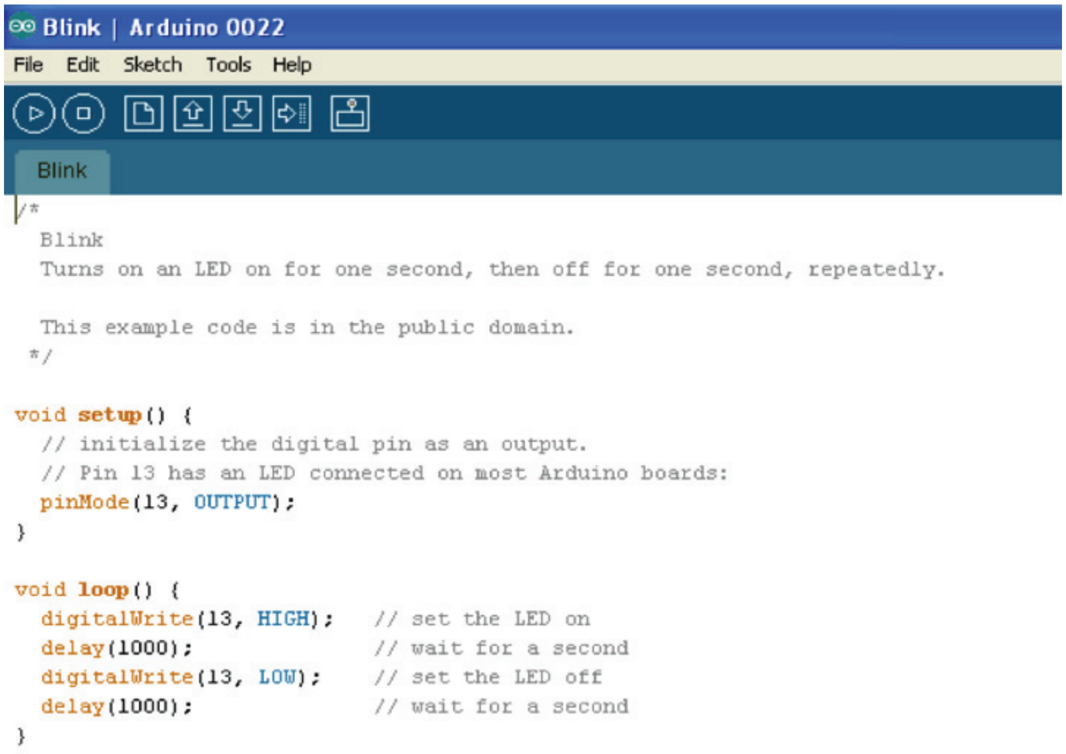
// Programming the Dev Board:

For creating Code to use on the LilyPad Dev:

The Code: You will use four different commands (or lines of code) to perform the four basic functions of a microcontroller. These are “writing” (or outputting) in a digital and analog format and “reading” (or inputting) in a digital and analog format.

Lighting up the LED!

To find the Blink Sketch, which is the basic “blink an LED” example, go to Examples under the File Menu and select Basics. Inside this menu choose the Blink sketch.



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

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

To turn the LED on and off on your LilyPad Dev you can change the lines of code in this sketch or write your own. See below for definitions of the different lines.

To “**write**” to the LED in a digital format use these lines of code highlighted in red:

digitalWrite (5, HIGH); This line turns the LED attached to pin # 5 all the way on.

digitalWrite (6, LOW); This line turns the LED attached to pin # 6 all the way off.

Change the numbers in the two lines above to operate different pins.

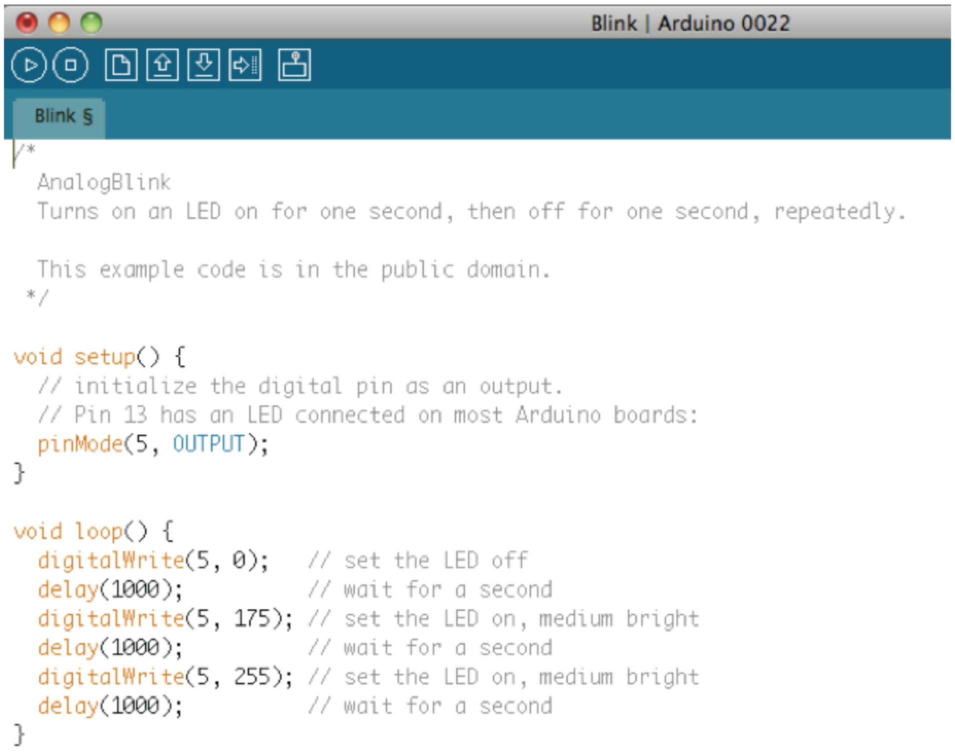
Name:
Date:

// Programming the Dev Board:

To "**write**" to the LED in an analog format use these lines of code written in red:

analogWrite (5, 255); This line turns the LED attached to pin # 5 all the way on, but you can change the second number for a different brightness. For example if you replace the 255 with a 250 the LED will be just a little dimmer.

analogWrite (6, 0); This line turns the LED attached to pin # 6 all the way off, but you can change the second number for a different brightness. For example if you replace the 0 with a 5 the LED will be just a little brighter. 255 is the highest this number will go to.



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

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(5, OUTPUT);
}

void loop() {
  digitalWrite(5, 0);   // set the LED off
  delay(1000);          // wait for a second
  digitalWrite(5, 175); // set the LED on, medium bright
  delay(1000);          // wait for a second
  digitalWrite(5, 255); // set the LED on, medium bright
  delay(1000);          // wait for a second
}
```

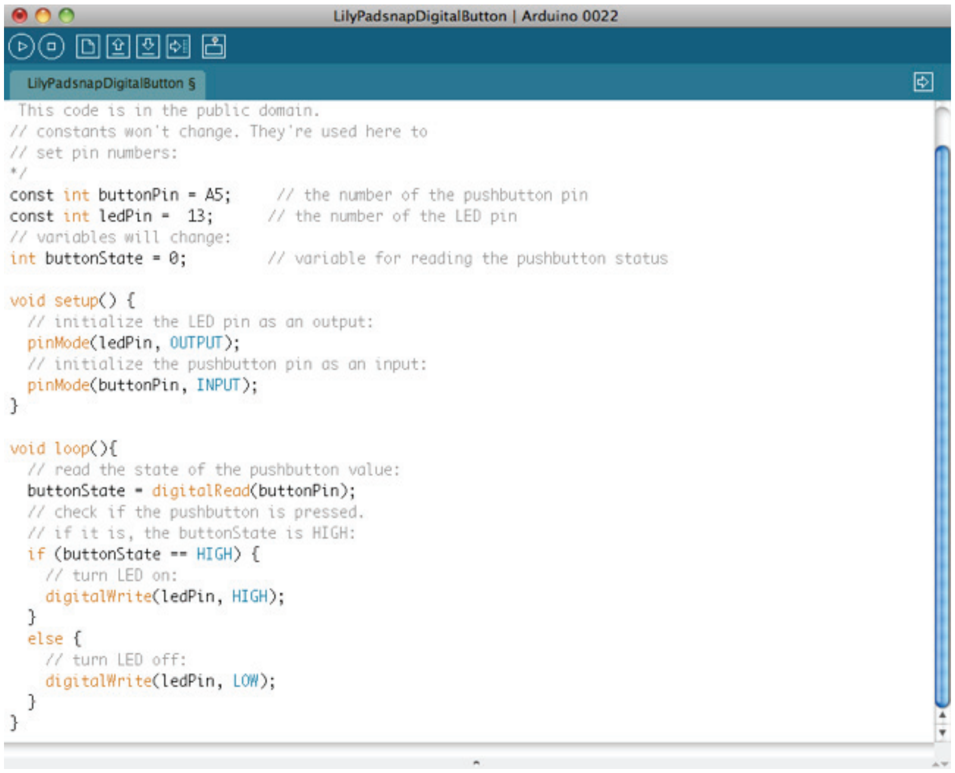
Fun SparkFun Fact: (The LilyPad Dev RGB (Red Green and Blue) LEDs are the opposite of normal LEDs because they have a common anode, meaning they share the same power source. This means that for this LED 255 is all the way off and 0 is all the way on. Don't worry if you're not sure what that means, but it explains why 255 is all the way off and 0 is all the way on.)

Name:
Date:

// Programming the Dev Board:

To “**read**” from the button you will need to use this line of code highlighted in red:

int buttonState = digitalRead (A5); This line takes a reading from pin # A5 which has the button attached to it and assigns it to the variable buttonState. (the ‘int’ is so that the variable is an integer) We could have called the variable anything, but buttonState makes sense because the sensor is a button. When the button is pressed it will read LOW or zero, and when it is not pressed it will read HIGH or 1023 (although depending on the resistance of your circuit this may be a smaller value). This is the opposite of what sensors usually read because buttons are special because they use something called Internal Pullup Resistors. Don’t worry if you’re new to microcontrollers, all it means is that in order to get a reading you need to put the following line (highlighted in red) in your setup function: digitalWrite (A5, HIGH); After the microcontroller performs the digitalRead line you can treat the variable buttonState just like a HIGH or LOW, depending on if the button is pushed or not.


LilyPadsnapDigitalButton | Arduino 0022
LilyPadsnapDigitalButton §

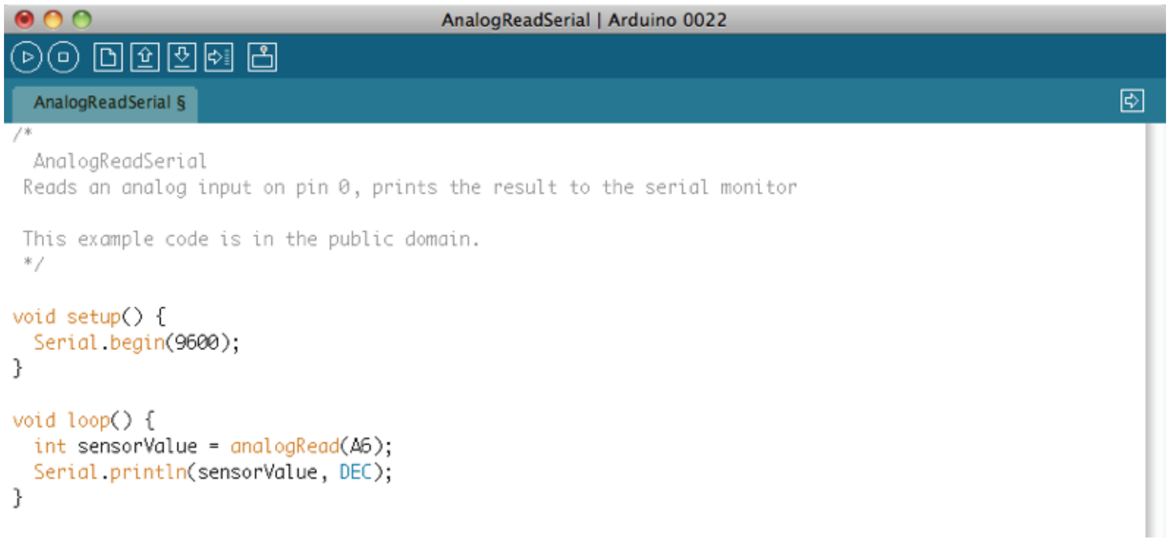
```
 This code is in the public domain.
// constants won't change. They're used here to
// set pin numbers:
*/
const int buttonPin = A5;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin
// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Name:
Date:

// Programming the Dev Board:

To “read” from the photoresistor (light sensor) you will need to use this line of code highlighted in red:

int sensorValue = analogRead (A6); This line takes a reading from a sensor attached to analog pin 0 and assigns it to a variable named sensorValue (we could have named the variable anything, we named it sensorValue in this example because that is what we use in the code). (the ‘int’ is so that the variable is an integer) Because it is analog it will return a number between 0 (no data or off) and 1023 (sensor is overloaded or all the way on). After the microcontroller performs this line you can treat the variable sensorValue just like a number that changes depending on what the sensor reads.



```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A6);
  Serial.println(sensorValue, DEC);
}
```

Name:
Date:

// Programming the Dev Board:

To display microcontroller information on the computer:

To display a reading or variable from the microcontroller to computer using "**Serial Communication**" you will need to use these lines of code highlighted in red:

Serial.begin (9600); This line should go inside your setup function. The number 9600 is your "baud rate". Don't worry if that doesn't make sense right now.

The following lines should go wherever you like in your loop function, they will not affect how the microcontroller acts, they will simply display information in your Serial Communication Window (we'll explain your Serial Communication Window in a moment):

Serial.println ("text message"); This line will display whatever text is inside the quotation marks, in this example it will display the words text message.

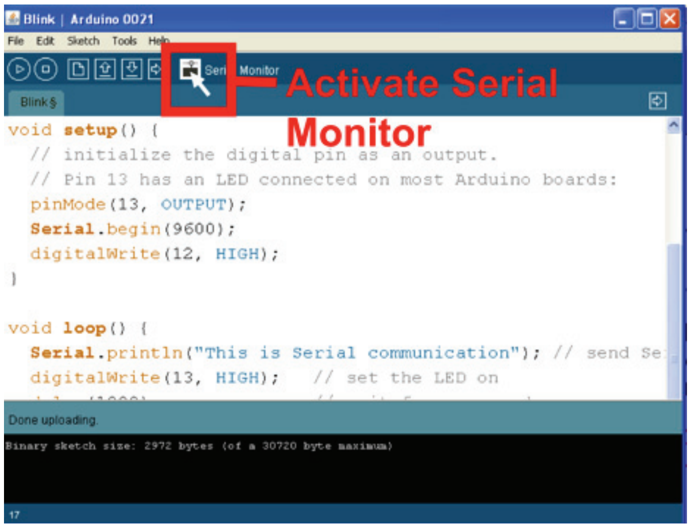
Serial.println (sensorValue, DEC); This line will display the value of a variable, in this case it will display the value of the variable called variableName. Change variableName to display whatever variable you want to see in your Serial Communication window.

Serial.println(analogRead (A1)); This line will display a value between 0 and 1023. This number signifies the reading it is getting from the sensor attached to analog pin # 1. On the ProtoSnap this pin is where your photoresistor is attached.

Once you have put the Serial.begin (9600); line in your setup function and one (or more) of the other Serial lines in your loop function, click the Serial Communication button to activate the monitor.

Name:
Date:

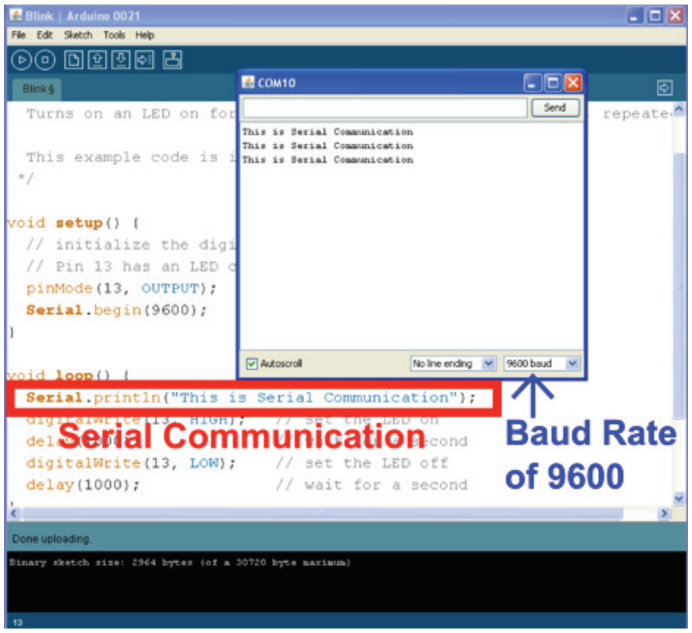
// Programming the Dev Board:



```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  digitalWrite(12, HIGH);
}

void loop() {
  Serial.println("This is Serial communication"); // send Se
  digitalWrite(13, HIGH);   // set the LED on
```

```
Binary sketch size: 2972 bytes (of a 30720 byte maximum)
```

Ok. Remember "baud rate" ? Once the Serial Communication Window is open make sure that the baud rate in your Serial.begin (9600); line matches the baud rate in the lower right hand corner of the Serial Communication Window. See image below.



```
  Turns on an LED on for                          repeate
  
  This example code is i
 */

void setup() {
  // initialize the digi
  // Pin 13 has an LED c
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  Serial.println("This is Serial Communication");
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
```

```
This is Serial Communication
This is Serial Communication
This is Serial Communication
```

```
Binary sketch size: 2964 bytes (of a 30720 byte maximum)
```

Note: Once you have snapped apart your LilyPad Dev Board you will most likely need to change your Pin variables so that they match the pins you sew your components to because some of the pins are not "broken out", meaning they are not available to sew stuff to.

Name:
Date:

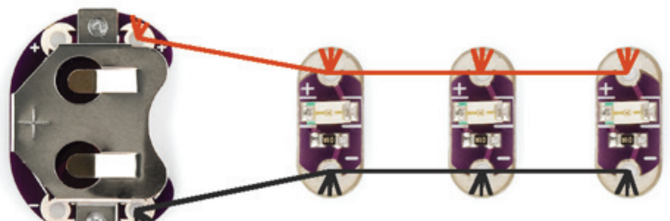
// LilyPad Basics:

Introduction: What are Lilypads?

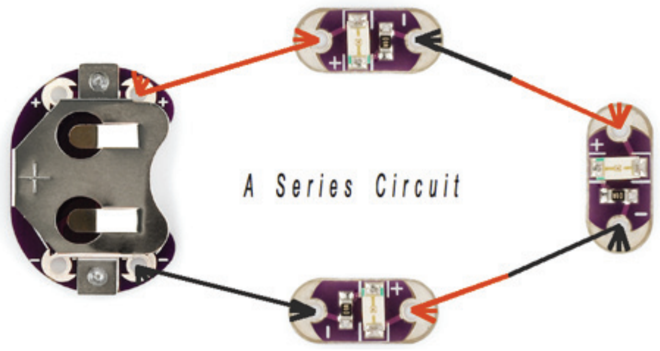
LilyPad is a wearable e-textile technology developed by Leah Buechley and cooperatively designed by Leah and SparkFun. Each LilyPad was creatively designed to have large connecting pads to allow them to be sewn into clothing. Various input, output, power, and sensor boards are available.
They're even washable!

Great, how do we use them:

Utilizing conductive thread we can connect power to different component boards, but to start we need to cover a few basics of circuit design. In building basic circuits, there are two types of configurations: a parallel and series circuit.



A Parallel Circuit



Both have advantages and disadvantages. The largest disadvantage to a series circuit is that you need a high voltage battery to run even a small circuit. Due to this fact, it is recommended that you always use Lilypads in a parallel circuit.

Linking batteries in Series combines the voltages of the batteries to create a power supply with a higher voltage.

Linking batteries in Parallel combines the current of the batteries to create a power supply with a higher Amperage.

Name:
Date:

// LilyPad Basics:

Up Next: Battery life!

Before you start it is a good idea to pick all your components and figure out how long a typical battery will run them. For this we will need to do a little, easy math. The first thing to do is to figure out how much current (I) each component will use. In the parallel circuit above, each LED draws about 20 mA (milli-Amps). Since there are three LEDs we need to account for all of them: 20 mA + 20 mA +20 mA = 60 mA. Then we need to find how how many amps the battery holds. For the coin cell battery, this is 250 mAh (milli-Amp hours). This means the battery will run at 250 milli-Amps for one hour. In order to discover how long your battery will run use this equation:

$$\text{Battery run time} = \frac{\text{mill – Amp hour of the battery}}{\text{total milli – Amps of the circuit}}$$

For the circuit above:

1. Total milli-Amps of the circuit = 20 mA +20 mA +20 mA so... Total milli-Amps of the circuit = 60 mA
2. Milli-Amp hour of the battery = 250 mAh
3. Using these two values in the equation above- Battery Run time = $\frac{\text{250 mAh}}{\text{60 mA}}$ so...
   Battery Run time = 4.167 hours

Now we know about how long one coin cell battery can power 3 LEDs. If the battery run time you come up with is not long enough, there are some other options available to you. The Lilypad line has various components designed to allow you to plug in different types of batteries.

Enough science, let's get to the fun part: LilyPad basics

Do not sew any components in with the battery installed. There is no risk of getting hurt, but you might kill the battery.

There are generally two types of thread, thick and thin. The thick type is better if you are going to have LEDs far away from the battery. The thin type will work in a sewing machine but the components cannot be too far away from the battery.

Any time you make a connection between a component and the thread, make a few loops through the connection hole. The section of metal on the outside of the connection is hole is where the electricity will flow between the thread and the component. Often if there is a short, or break, in your circuit it is because the conductive thread does not always touch this piece of metal on the component. It is possible to sew the thread through the connection without touching this piece of metal and the thread, so make sure to double check your connections.

The thread will melt similar to nylon. It is good to knot the end of your thread at a component and melt the extra, however be careful you do not burn your project.

If you have two threads that need to cross, there must be some fabric between the two otherwise the circuit will not work.

There are other conductive e-textile materials such as buttons, fabric, yarn, paint and velcro. There are many different ways to use the materials as switches, sensors and circuits, you just need to experiment.

Name:
Date:

// Sewing the Circuits:

Not everyone has a background that involves sewing, so here are some basics on sewing to get you started.

Threading the needle:

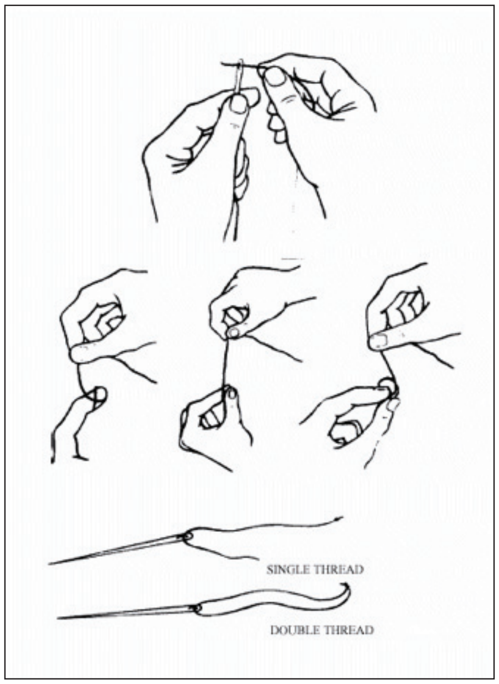


Photo from CIEhub.info.

Threading the needle- at the best of times, threading a needle is a frustrating task, particularly if your eyesight is less than perfect, or your hands aren't entirely steady. Conductive thread is a little bit more difficult than traditional thread, because it's frequently thicker, and frays more easily. Cutting a new end on the thread right before threading will remove any frays, and running the end of the thread through quilting beeswax will help hold it together and stiffen it, making it easier to thread. Follow the steps below. When you're working with e-textiles, the decision to use single or double thread (shown at the bottom of the picture) is a little bit more complicated than it would be with traditional textiles. The thread you're working with has a resistance, usually between a few ohms per foot and several hundred ohms per foot, depending on how thick the thread is and what the metal content is. The higher the resistance is, the more limited the power will be that reaches your components. That means that if you have 3 feet of thread between your battery and a light in your circuit, and the thread is very high resistance, the light might not even come on. More thread can decrease the resistance by providing more conductive material- that means that if your thread is 100 ohms per foot, and there's a foot of thread between the battery and the light, a single thread is going to add 100 ohms of resistance and a double thread is going to add only 50 ohms of resistance. On the whole, it's nice to keep resistance to a minimum- where your thread and fabric allow it, going ahead and doubling the thread is an easy way to do that. Still, there are some threads that are too thick to conveniently double, or low enough resistance that it's not necessary, so try to weigh the resistance against the bulk and make the decision that is practical for your project.

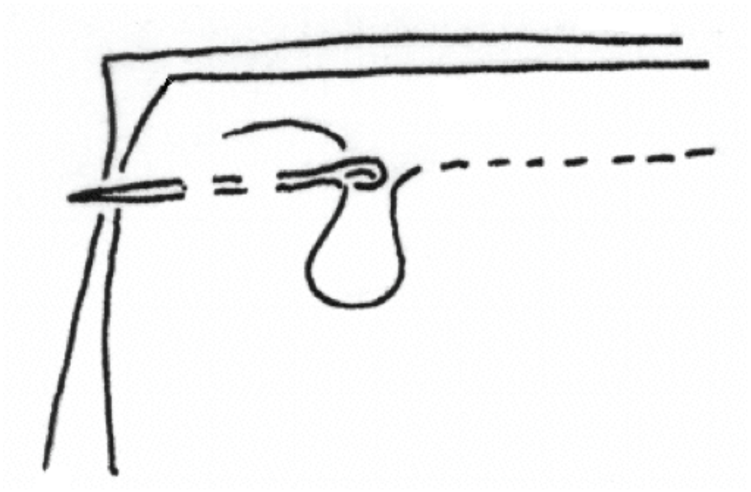
Name:
Date:

// Sewing the Circuits:

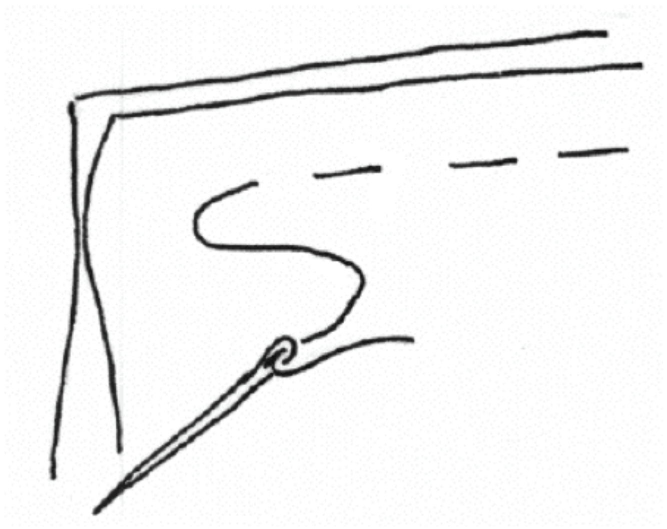
Stitching:

There are a few basic stitches that can be really handy.

A Running Stitch is a very simple, fast, and easy stitch, great for connecting components where it's unimportant whether your thread is visible.



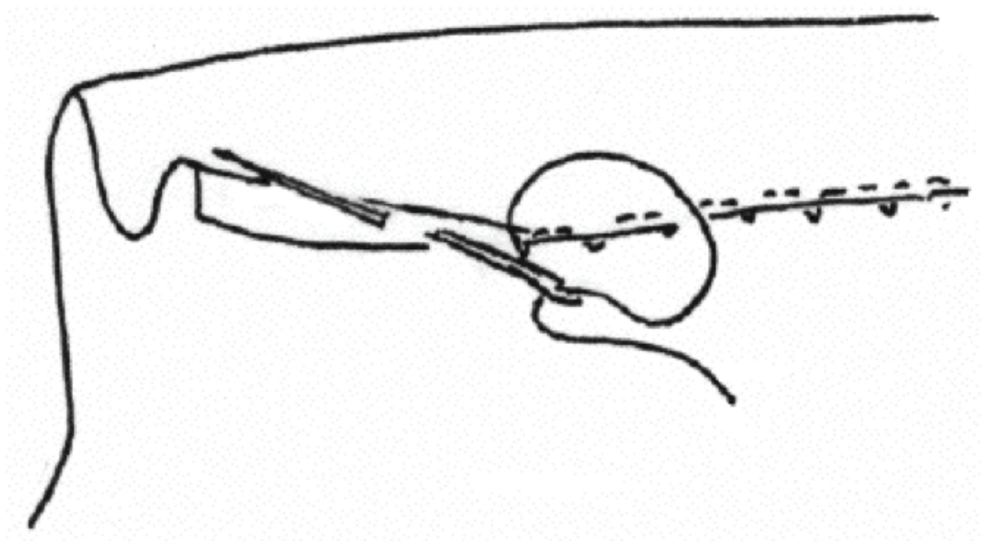
A Basting Stitch is a very wide stitch, best used where you need to cover a lot of ground quickly, on last-minute projects, and when you're confident that the stitching won't be seen or won't have to last long. Basting stitches give you amazing speed payoff, but very little durability, so use them wisely.



Name:
Date:

// Sewing the Circuits:

A Blind Stitch is a great way to conceal your stitching in a project where you don't want the conductive thread to clash with the face of the project. If you're working with two layers of fabric, you can put several solid stitches in the back layer for every one stitch that barely nicks through the top layer of fabric. With only one layer, use a large basting stitch on the back of the fabric, then nick through to the front in a very small stitch. Done well, this technique will make your stitching invisible from the front, which is nice in projects where the dark gray of conductive thread won't blend well.



Stitching pictures above by www.ia470.com

Ohm's Law:

Ohm's Law is a simple equation that explains the relationship between current, voltage and resistance.
It is important to understand because the amount of resistance (how hard it is for electrical current to pass through a material) in your thread and other conductive materials will affect the amount of current and voltage available to your sensors and Micro-Controller Input pins.

Here is the equation:

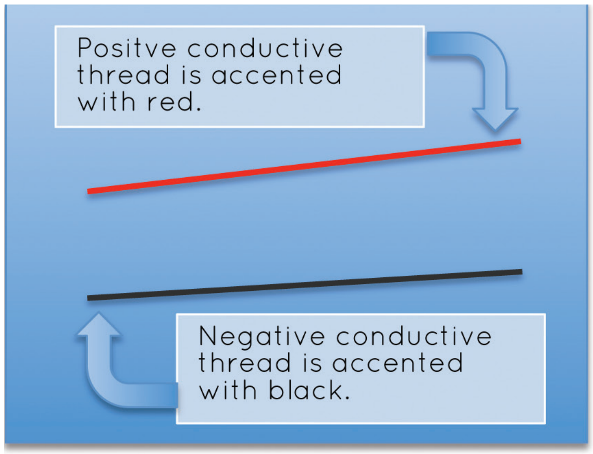
$V = I * R$ or $I = V / R$ or $R = V / I$

In this equation V is Voltage, I is Current and R is Resistance. This means that the less resistance your circuits (which include the conductive thread) provide the more electricity there will be available to be used by your components. You can also use the amount of resistance that your materials provide as a sensor. One example of this is detailed in the next section More Concepts with LilyPad and Conductive Materials.

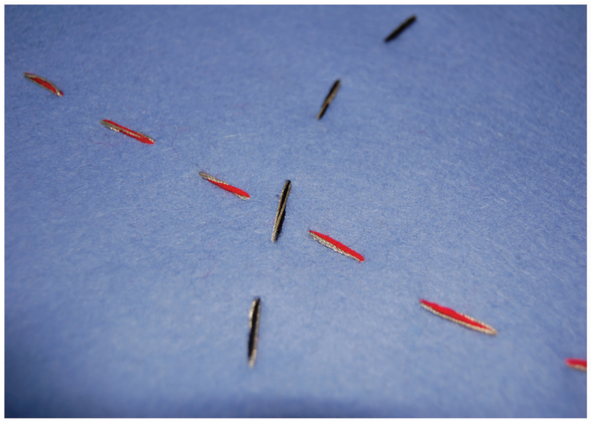
It also means that you can double up your thread by stitching through the same area multiple times in order to decrease resistance. Doubling up your thread decreases resistance because the electricity has more material it can flow through. In order to decrease the amount of resistance you will need to double up all the areas in a circuit where you stitched conductive thread, it will not work if you fail to double up even one section of the circuit.

Name:
Date:

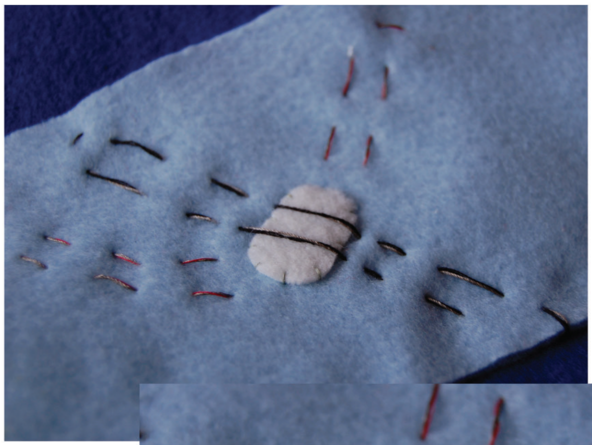
// How to Bridge Your Leads:


Positve conductive thread is accented with red.
Negative conductive thread is accented with black.

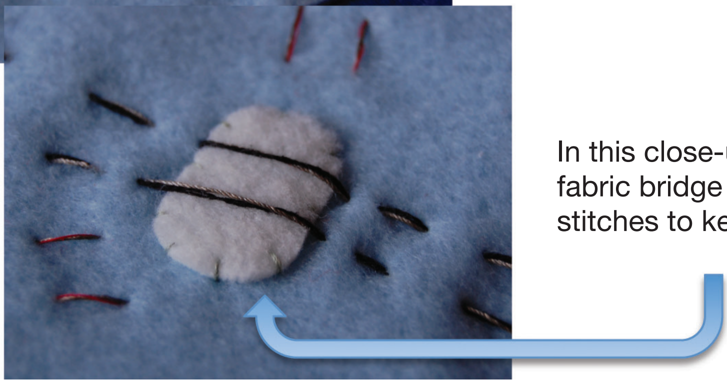
When making e-textile projects, you'll often have leads going every which way. As your projects get more complicated, you'll inevitably run into the problem of crossing leads. As we know, crossed leads that touch each other short circuits. This is bad. Here is a smattering of solutions, elegant and otherwise.



If you're using thick fabric, you can simply make sure that when your negative and positive threads cross, one crosses on the top and one crosses on the bottom – sort of a ships-crossing-in-the-night thing.



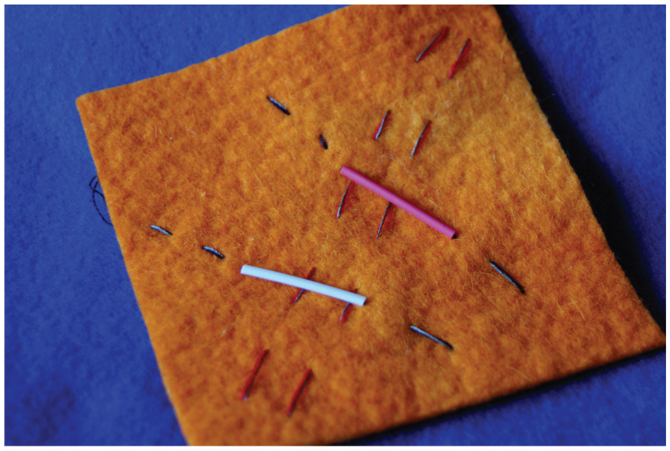
Now, that's a fine option if there is a minimum of movement in your fabric.However, if your piece might fold over on itself, it's best to sew a fabric bridge between your positive and negative threads.



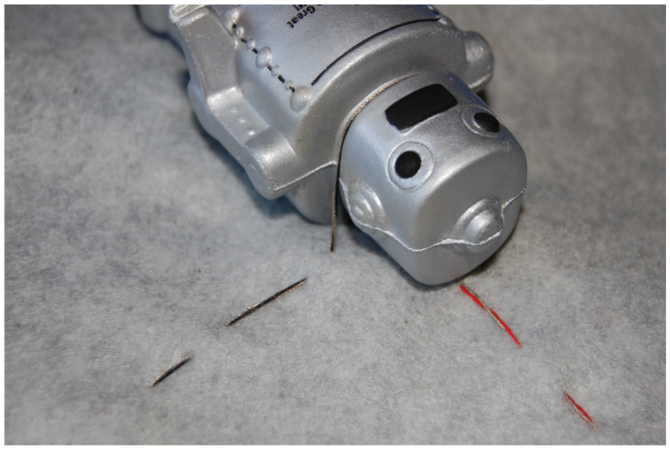
In this close-up, you can see that the fabric bridge is secured with a few stitches to keep it from sliding around.

Name:
Date:

// How to Bridge Your Leads:



Heat-shrink tubing (something you probably already have around if you're doing electronics) is handy for making bridges. Just cut the length you need and stick your needle and thread through it as if it's a big bead. You can put multiple threads through on tube if you need to.



If you think about it, just about anything can be used as a bridge in a pinch, like this robot toy. It's sort of lumpy, though.

Really, the possibilities are limitless. Just consider your fabrics and their uses. Then bridge accordingly.

Name:
Date:

// More Concepts with LilyPad and Conductive Materials:

Making flower button circuits:

These instructions can be used to create a variety of switches- not just the one I've made here. Feel free to change it up and make a button that works with your project, such as a cat button with conductive thread whiskers you can stroke or a guitar button with strummable strings!

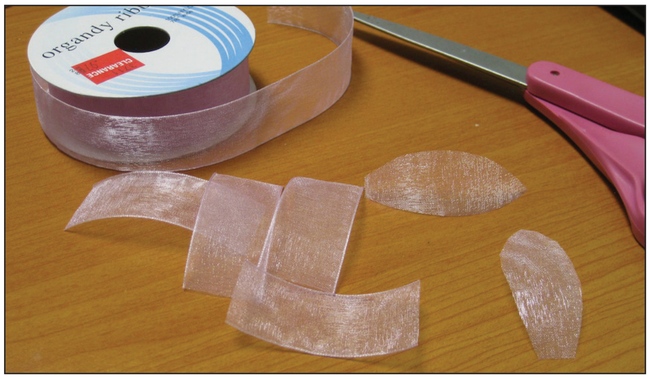
The type of switch you're making is called a 'momentary' button or switch. What that means is that you're only momentarily making a connection. When you bring the thread in contact with the fabric, you are creating a path for electricity that lasts only as long as the thread and fabric are in contact. When you take your hand away, the thread will spring back to its original position, and you will no longer have a closed circuit. On a practical level, this means that you can sew the switch into a circuit and use it to turn something on. If the switch is placed, for instance, between a power supply and an LED, the light will turn on while you are activating the switch, and off again as soon as you stop pressing the thread and fabric together.

Materials:

This list is to create the switch pictured; the only real constants if you want to change the style of the switch are conductive thread, conductive fabric, a non-conductive material to insulate between them and an adhesive.

- Conductive Fabric
- Conductive Thread
- Stiff, Non-conductive fabric (I used felt)
- Non-conductive ribbon
- Hot Glue
- Chenille Needle

First, cut about 15 small strips of ribbon, roughly as long as you'd like the longest point of your flower petals to be. Mine were about 2 inches long, but go ahead and customize it to your project. Trim each strip into an oblong with one pointy end and one flat end. I highly recommend having one flat side, as it makes them easier to line up when you're creating your flower, but beyond that, the shape and quantity of the petals is an aesthetic decision- feel free to pack more petals in or change the shape as you see fit!

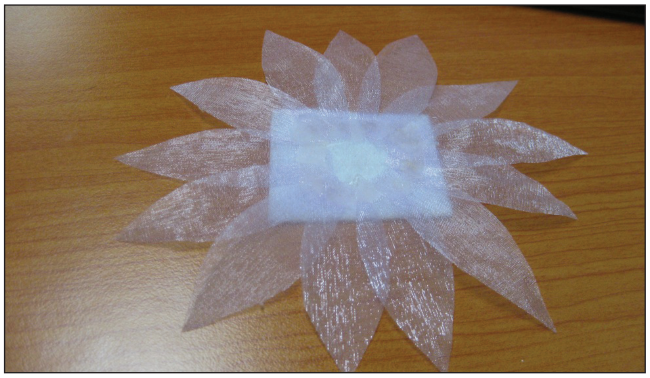


Name:
Date:

// More Concepts with LilyPad and Conductive Materials:

Cut out conductive fabric petals in the same shape, slightly smaller. If you're having problems, try using one of the ribbon petals as a template.

Cut a small piece of your non-conductive fabric. The shape is largely irrelevant, since you'll be covering it with petals, but make sure it's big enough to support two layers of petals and still have room to be sewn or glued to the garment. Glue the non-conductive petals down, overlapping slightly, making sure that you leave a bare circle in the middle.



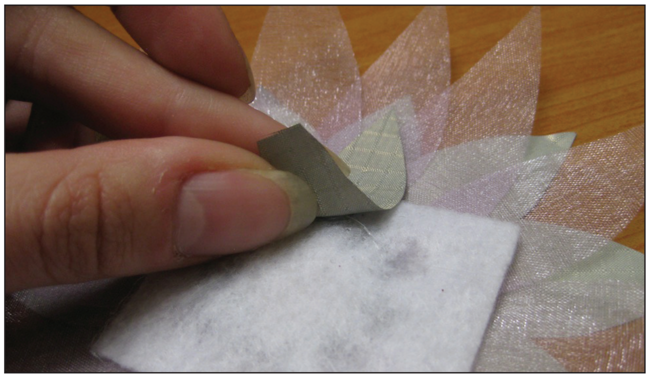
Glue the conductive petals on top of the ribbon petals. Just like the ribbon petals, you want them to overlap, and you want to leave an open circle in the middle. Make sure that you don't use too much glue here- you don't want to insulate the petals. try to make sure that each petal makes uninsulated contact with both of the petals next to it.

Repeat this process until you have as many flowers as you need for your project.

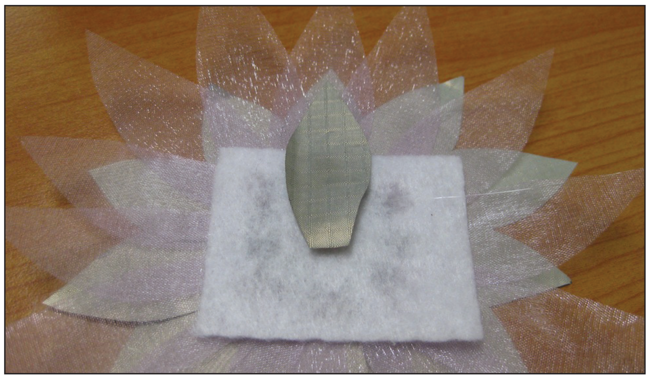
Name:
Date:

// More Concepts with LilyPad and Conductive Materials:

Turn your flower over and glue another conductive petal to the back of the non-conductive fabric, being careful to ensure that the flat edge of the petal overlaps the empty circle in the middle of the flower.
You want both tips of the petal free for sewing, so place your dot of glue towards the middle of the petal.



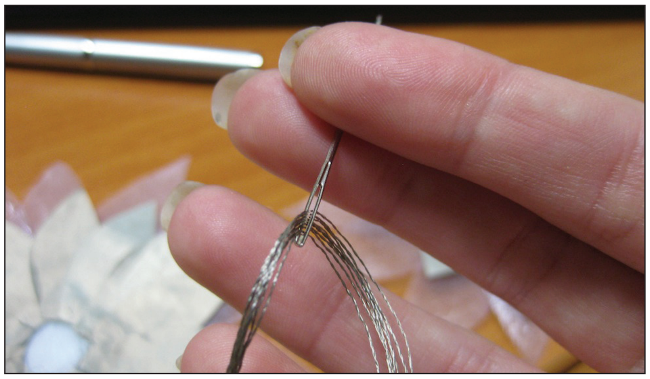
Double check that your bottom conductive petal is in NO CONTACT with your top conductive petals. Ideally, the ribbon petals should insulate between them, but there can be gaps, so check now. If the top and bottom petals touch, your circuit will always be closed, which means that whatever you're controlling with the switch will be always turned on.



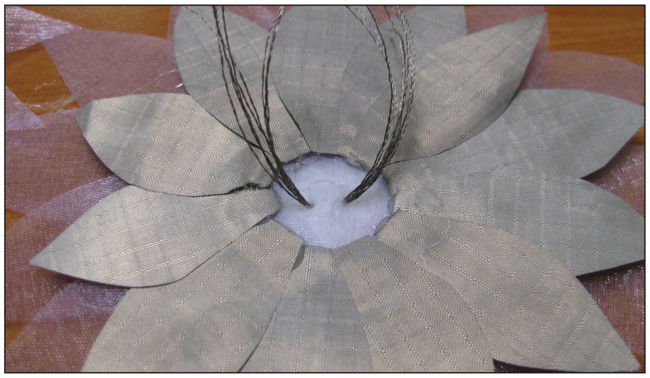
Cut 8 strands of conductive thread, about 3 inches long per flower, and twist them all together. Thread the entire twist through a single needle- this will be easiest with a fairly fine thread, such as the 2-ply conductive, and a large chenille needle, available at craft and fabric stores or in our needle set.

Name:
Date:

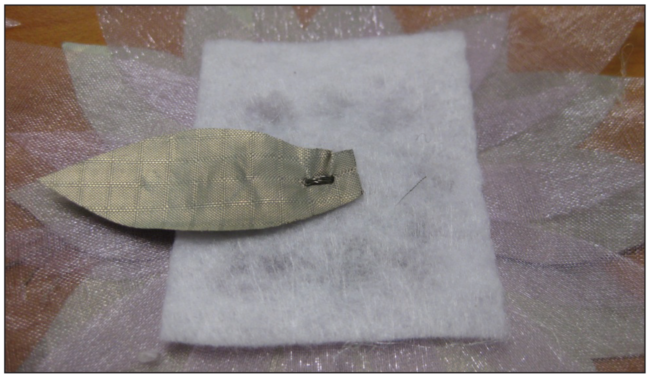
// More Concepts with LilyPad and Conductive Materials:



Make one stitch all the way through the bare center of your flower, making sure that it goes through the bottom petal, and back up, without pulling the thread tight- you want about an inch and a half of thread on each side of the stitch, on top of the flower, like a tailor's tack. (Many thanks to Miss P for putting up such a great tutorial on the subject!)



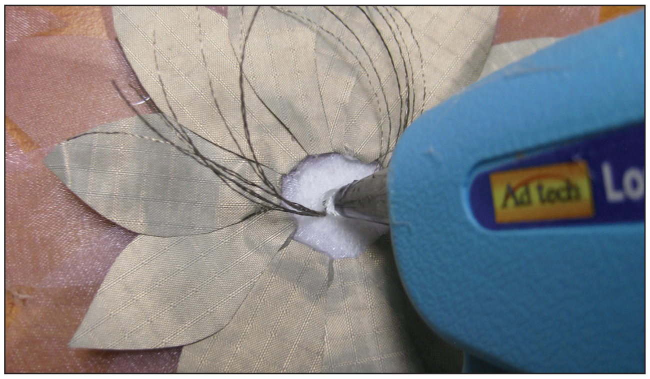
The stitch should be solidly in contact with the back petal, as shown. This is going to create the electrical connection between the two petals and stamen, closing the circuit when the switch is activated.



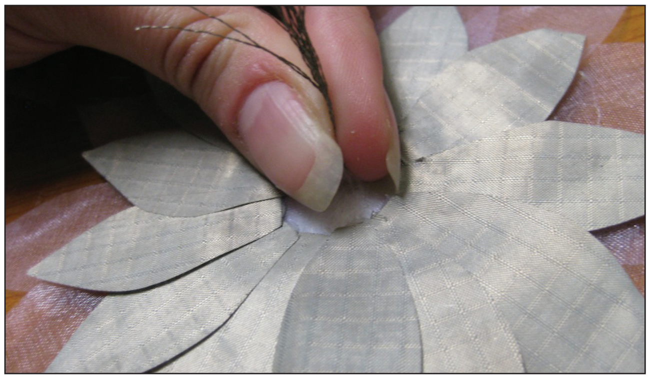
It's important for the stamen to stand up so that it's only in contact with the petals when you brush it. To help ensure this, and also to keep the stitch from coming out, dot a little bit of hot glue in between the two bundles of strands.

Name:
Date:

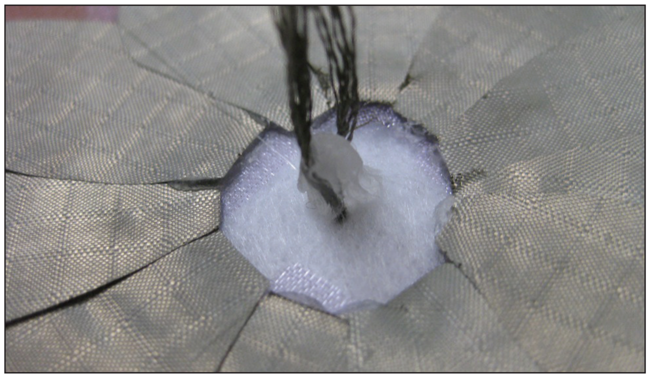
// More Concepts with LilyPad and Conductive Materials:



Press both bundles into the blob, being careful not to burn yourself. I'm using a low temperature glue gun, which makes this a little easier.



This results in one nice tall bundle of thread poking out of the top of a (slightly unsightly!) blob of glue. Feel free to try another technique here, or cover this junction with glitter, beads, whatever works for you. I left mine alone, because it was pretty small in relation to the whole dress, and unlikely to draw much notice.

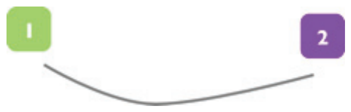
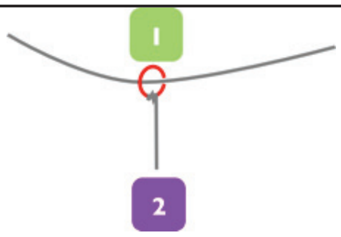


And here are some finished flowers! These can be sewn into a circuit in the same way that you would sew a LilyPad Button Board.

Name:
Date:

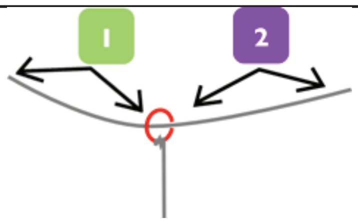
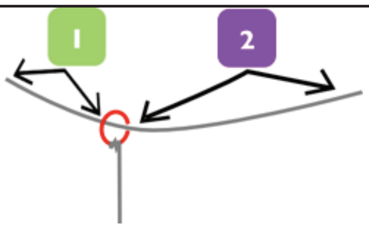
// Making Potentiometer (AKA Voltage Divider) Circuits:

Potentiometer is just a fancy way of say “dial”. But the reason why we say potentiometer is because this type of sensor can come in many different styles. Below is an explanation of how to create a potentiometer using conductive thread or yarn and some type of conductive clasp or circle.

	**Step 1:** First connect one side of a piece of conductive material to your power source (1) and connect the other side to a ground (2). You may need to deattach one of these connections, so make sure you can do so if need be.
	**Step 2:** Next place your conductive circular component (a metal ring or jewelry clasp) on the first piece of conductive material (1). The metal ring should be able to move freely on the conductive material while still making constant contact. Bear in mind that this ring will move a lot, so make sure it doesn’t wear or fray the conductive material. Tie another piece of conductive material to the metal ring and connect the end of this to an analog Input pin (2). Now as you move the metal ring you will get a different voltage to your Input pin (2) depending on where the metal ring is. The closer the ring is to the power source the higher the voltage you will see on your analog Input pin. The closer the ring is to the grounded connection the lower the voltage you will see on your analog Input pin. Bear in mind that the conductive clasp needs to make contact with all conductive material in order for this potentiometer to work. If you are getting an intermittent signal, make sure that your clasp is constantly in contact with the first piece of conductive material.

Name:
Date:

// Making Potentiometer (AKA Voltage Divider) Circuits:

	An explanation: The reason this works is that due to Ohm's Law the voltage will decrease the more resistance the electrical signal passes through before it reaches the Micro-Controller pin. So if it passes through a whole bunch of conductive material (remember, even though it's conductive it also has resistance) before it gets a chance to travel through the metal ring and into the analog Input pin then the signal will be kind of weak.
	But if the electrical signal doesn't have to travel through as much conductive material because the metal ring has been moved closer to the power source, then the signal will be stronger because there is less resistance. Here is the equation used by people who like equations to figure out exactly how much voltage will come out of this potentiometer into the analog Input pin:
Potentiometer (AKA Voltage Divider) Equation: **Vout = Vin * R2/(R2 + R1)**	In this equation Vout is the electrical signal we are solving for because it represents the amount of voltage we will find running into the analog Input pin. Vin is the amount of voltage the first piece of conductive material is connected to at the power source. R1 is the amount of resistance provided by the first section of the conductive material (1). R1 is the amount of resistance provided by the second section of the conductive material (2).

Example Code:

We have also included some example code to explain a couple different concepts.

LilyPadSnapDigitalButtonPatterns Sketch shows how you can use the button to switch between a couple different patterns of LEDs. You don't necessarily need to limit this to LEDs though, your code can switch between any number of functions.

TheWorstInstrumentEver Sketch uses the temperature sensor and the piezo buzzer to switch between two different patterns depending on the temperature. This code may require the user to change the temperature value that causes the pattern to switch depending on the ambient room temperature.

ProtoConditionals Sketch uses an analog pin to read the temperature sensor, we then take that analog value (between 0-255)and uses it in a conditional statement to control an LED. we will start reading the prevailing state of the sensor and we'll set the conditional values close to the prevailing state in order to compensate for fluctuation.