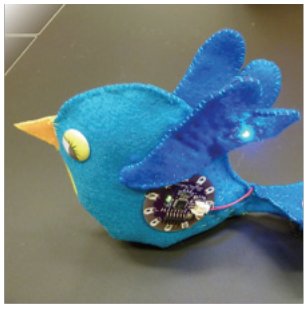
# LilyPad Drag and Drop

## // Drag and Drop Programming for the LilyPad Development Board

LilyPad technology is a way to sew robot brains, sensors, lights and even wireless communication into any fabric-based project. People used LilyPad to light up their jackets, wirelessly communicate between purses or stuffed animals, and create fashion that also helps blind people navigate the world. Sew it into a kite, a sweater, a backpack… or even your leather jacket!


But before you can start sewing the sensors, LEDs and buzzers into your beanbag chair, you need to program how you want the robot brain (also called a microcontroller) to interact with the input (that's the sensors) and the output (LEDs and buzzers). For example, maybe I need to be able to tell my e-textile beanbag chair to light up if the room is dark and someone sits in it. In order to do this I need to be able to take some readings from my sensors and tell the LEDs to turn on or off depending on what the sensors read.

Now you can learn how to program using Drag and Drag programming for LilyPad technology. You can't do everything with Drag and Drop programming that you can do with regular programming, but you can get started at a really young age!

**Note: Do not snap apart the LilyPad dev board yet, if you do you will not be able to easily program it!**



**SparkFun Electronics LilyPad Drag and Drop Educational Material**
**Material by Lindsay Craig | Design by Amanda Clark | Edited by Chelsea Moll**

LilyPad Drag and Drop
Worksheets v. 1.0

**Name:**
**Date:**

## // LilyPad Drag and Drop Goals:

1. To get kids excited about Science, Technology, Engineering and Mathematics through e-textile project creation.

2. Gain an understanding of the concepts of analog and digital as well as how these concepts relate to the input and output circuits as well as the microcontrollers of the systems the students are creating.

3. Gain an understanding of the concepts of input and output as well as how these concepts relate to the circuits and microcontrollers of the systems the students are creating.

4. Establish baseline computer programming skills including: Constructive file saving methods, using variables, nested “if” statements, culminating in an individualized project that students can embed in a textile project.

5. Engage students in representing and solving problems involving multiplication and division.

6. Solve problems involving measurement and conversion of measurements as well as converting like measurements within a given measurement system.

7. Support independent student pursuit of representing and interpreting data.

8. Use the four operations with whole numbers to solve problems.

9. Encourage students to generate and analyze patterns as well as data dependent relationships.

10. Use an integer variable counter to track time and control an aspect of output depending on this counter variable.

11. Extend students understanding of fraction equivalence and ordering.

12. Engage students in writing and interpreting numerical expressions through project driven learning.

13. Apply and extend previous understanding of multiplication and division.

14. Gain an understanding of ratio concepts and use ratio reasoning to solve real world problems.

15. Have students demonstrate the ability to reason about and solve one-variable equations and inequalities.

16. Involve students in representing and analyzing quantitative relationships between dependent and independent variables.

**Name:**
**Date:**

# // Standards and Materials:

**Common Core Math Standards:**

6.RP.1, 6.RP.3.b, 6.RP.3.d, 6.NS.5, 6.NS.6.b, 6.NS.7.a, 6.NS.7.b, 6.NS.7.c, 6.NS.8, 6.EE.1, 6.EE.2.a, 6.EE.2.b, 6.EE.2.c, 6.EE.5, 6.EE.6, 6.EE.7, 6.EE.8, 6.EE.9
7.RP.1, 7.RP.2.b, 7.RP.2.c, 7.RP.2.d, 7.RP.3, 7.NS.1.b, 7.NS.1.c, 7.NS.2.a, 7.NS.3, 7.EE.2, 7.EE.3, 7.G.4, 7.G.5
8.EE.1, 8.EE.2, 8.EE.7.a, 8.EE.7.b, 8.F.1, 8.F.3, 8.F.4, 8.G.3, 8.G.6, 8.G.7, 8.G.8
N-Q.1, N-VM.3, N-VM.4.b, A-CED.1, A-CED.2, A-CED.3, A-CED.4, A-REI.1, A-REI.3, F-IF.1, F-BF.1.a, F-BF.2, F-LE.1.b, F-LE.5, F-TF.1, F-TF.2, F-TF.7, G-SRT.8, G-C.5, G-MG.1, G-MG.3

**Materials:**

1. A computer with Arduino and Modkit Link installed, and an internet connection (or if previous to Modkit Micro's stable release use: http://material.media.mit.edu/?p=874)

2. SparkFun's Lily Pad Dev Board

3. A USB to mini cable

4. Additional materials needed for embedding: Conductive thread, sewing needles, non conductive fabric (for insulating circuits)

**About Modkit**

www.modkit.it

Modkit Micro helps you make (almost) anything smarter - from stuffed animals that respond to hugs, to window blinds that squint for you.

Modkit has an intuitive Drag and Drop interface similar to Scratch from MIT Media Labs, making it easier to introduce a younger population to microcontroller programming.

Modkit is:
**Ed Baafi** – Founder – Business and Engineering Lead
**Amon Millner** – Co-Founder – Design Lead
**Collin Reisdorf** – Co-Founder – Learning Lead

**Name:**
**Date:**

# // Questions:

**Suggested Questions Prior to Activity:**

1. What are some other examples of technology that uses inputs and outputs to perform a function?

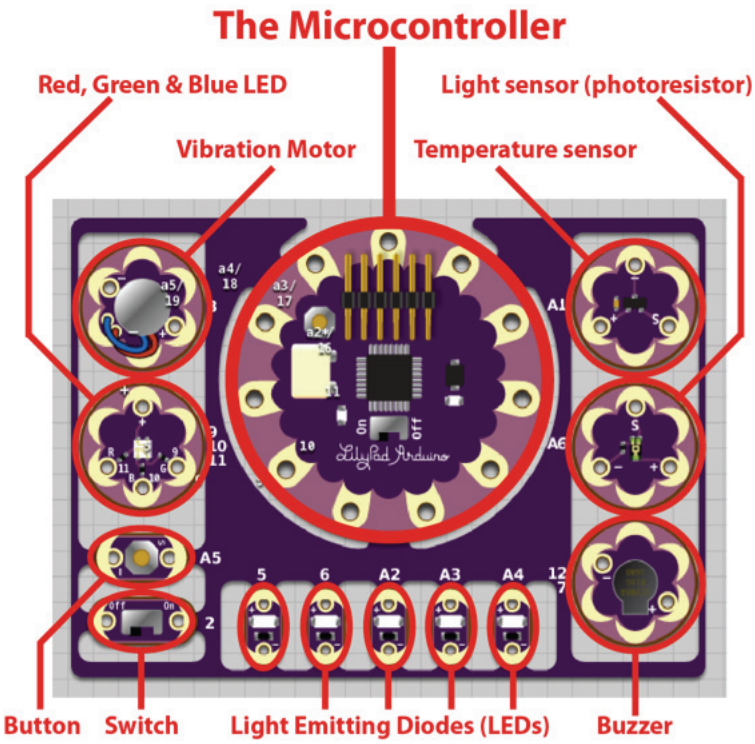
2. The Lily Pad Development Board has a light sensor, a temperature sensor, a switch and a button on it. What other kinds of inputs would you connect to the Lily Pad Dev Board?

3. The Lily Pad Development Board has some LEDs, a buzzer, a RGB LED and a vibration motor on it. What other kinds of outputs would you want to connect to the Lily Pad Dev Board?

4. What is the most interesting thing you can think of that you could sew circuits into?

**The LilyPad Development board:**

First, let's look at the board and talk about all the different parts on it. The board can be divided into three different categories: The microcontroller, input and output.

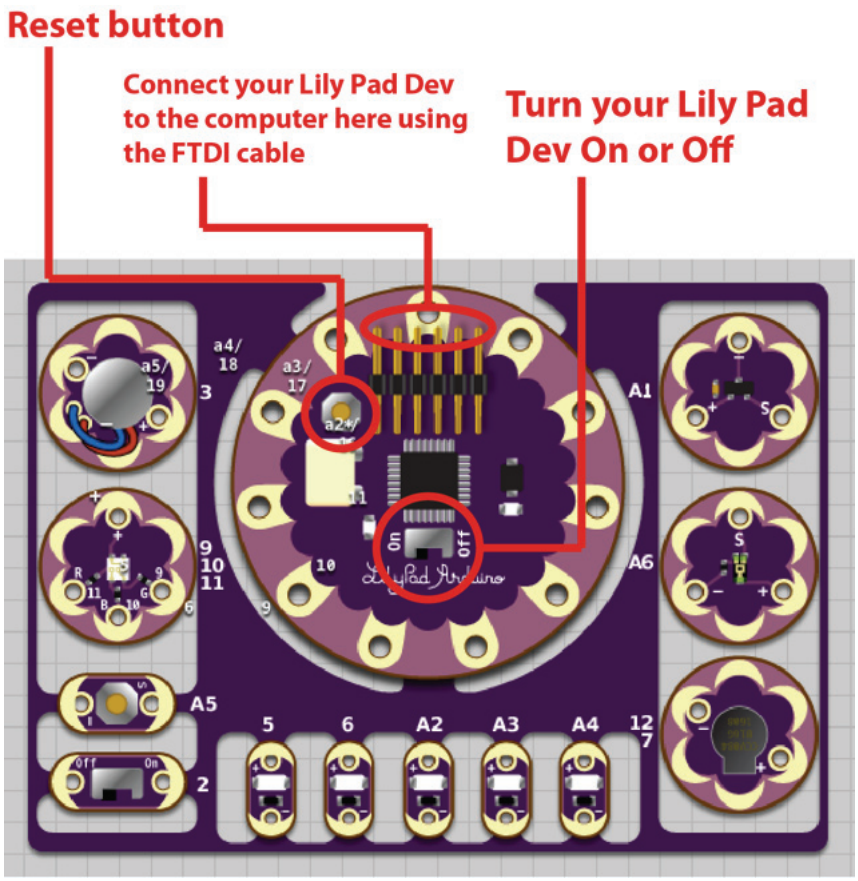


**Name:**
**Date:**

## // Microcontroller:

The microcontroller is the most important part of the board you are holding. The microcontroller is like the body and brain of the system you are creating. You can connect sensors (inputs) and LEDs (outputs) and many other things to the microcontroller. The microcontroller takes information from the inputs (sensors, button and switch). Then the microcontroller decides what to do with the information it gets. What the microcontroller decides to do with this input depends on the code you put in the microcontroller's brain. Then the microcontroller tells the outputs to turn on or off depending on the code in its brain decided to do.

Let's talk about the microcontroller and some of the parts on the microcontroller.



**Name:**
**Date:**

# // Plugging It All In:

**Plugging it all in: The Microcontroller, the FTDI, the USB Cable and Your Computer:**

First, plug your LilyPad Dev Board into your computer using the USB cable and FTDI device.
The FTDI device looks like this:

Make sure you line up the ‘B’ on the LilyPad Dev Board with the letters “BLK” on the FTDI.
On the other side of the LilyPad Dev Board and the FTDI device are the letters “GRN” and ‘G’.
Make sure these are lined up as well.


The FTDI and USB cable supplies power to your board, as well as a way to put the code you write on the microcontroller. Now you can turn the board on using the power switch. When you turn it on you will see a yellow LED light up near the On/Off switch.

You can also reset the board by pressing the reset button. This will turn your board off for a second and then turn it back on.

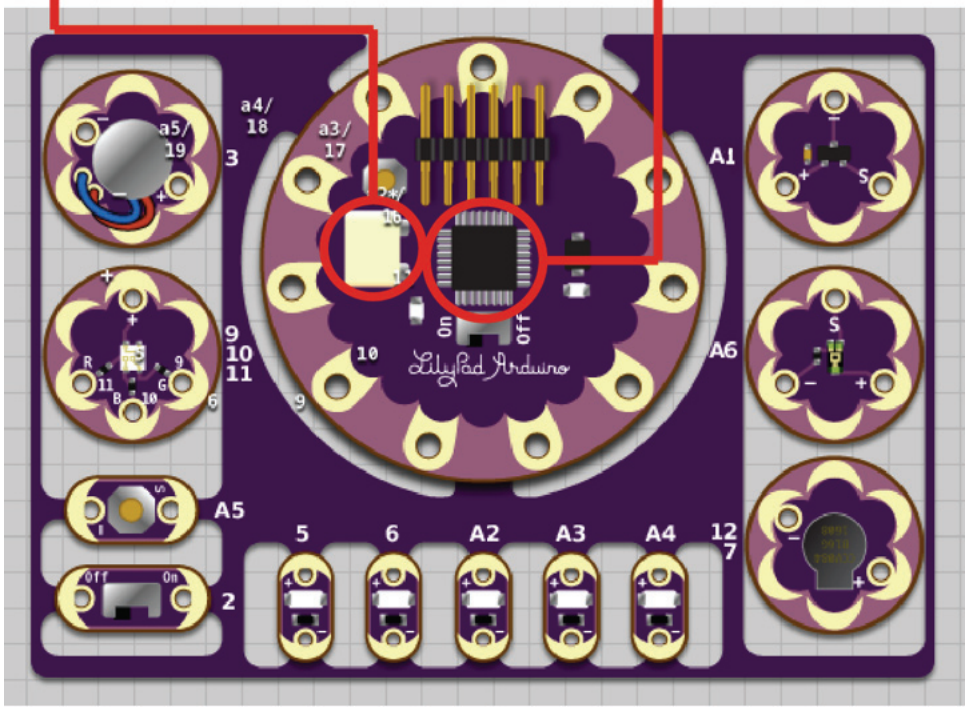
**Name:**
**Date:**

## // The Microcontroller Brain and Power:

There are three important parts of the microcontroller we haven't talked about yet. There is the actual computer chip (called an ATMega 328) that stores the code you upload to the microcontroller, and the battery connection in case you want to unplug the LilyPad Dev from the computer and power it with a battery or plug it into the wall.

**Battery connection:**
**Sometimes you will want to power your project when it is not plugged into the computer**

**ATMega 328:**
**This is the robot brain, also called a chip, an IC or a microprocessor**



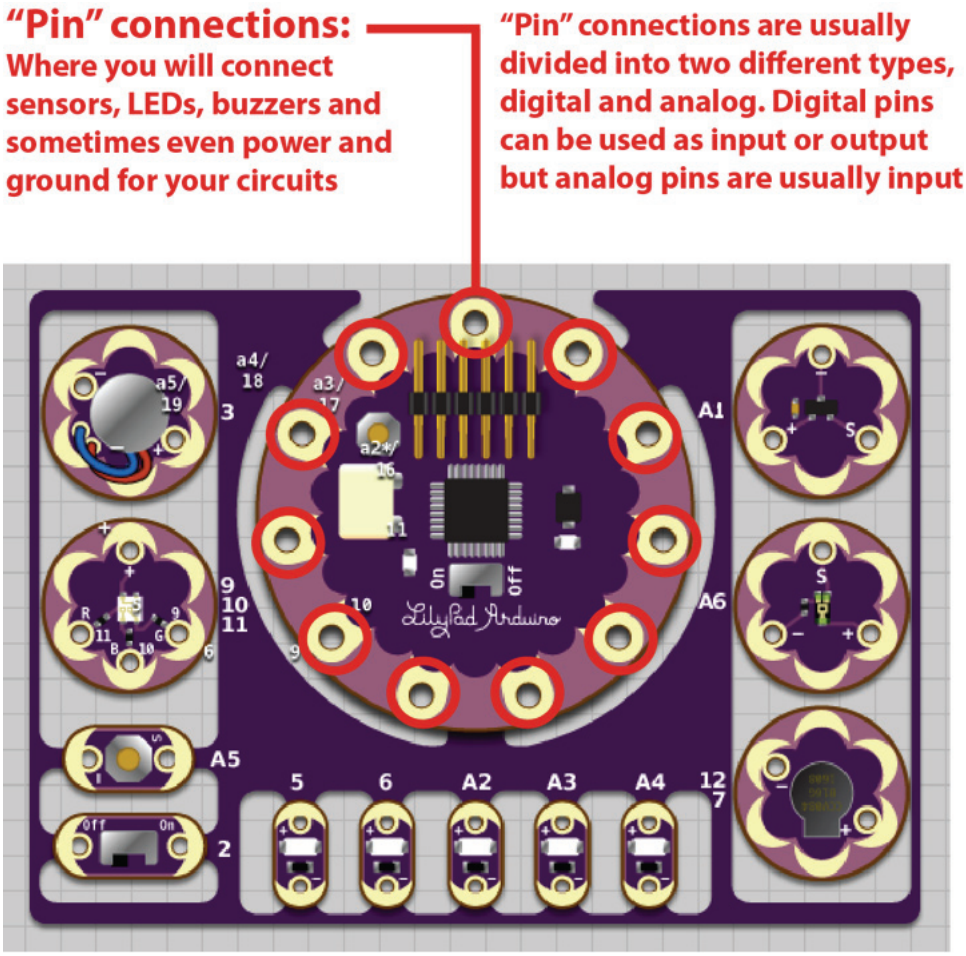

The LilyPad Dev Board needs at least a 5V power supply to work. You can use 4 AA batteries or a 9V "wall wart."

**Name:**
**Date:**

# // The “Pin” Connections:

**The “Pin” Connections Between the Microcontroller and the Other Components on the LilyPad Dev Board:**

The “Pin” Connections are where you will connect the microcontroller to the sensors, LEDs and other components that make up your input and output circuits later on.



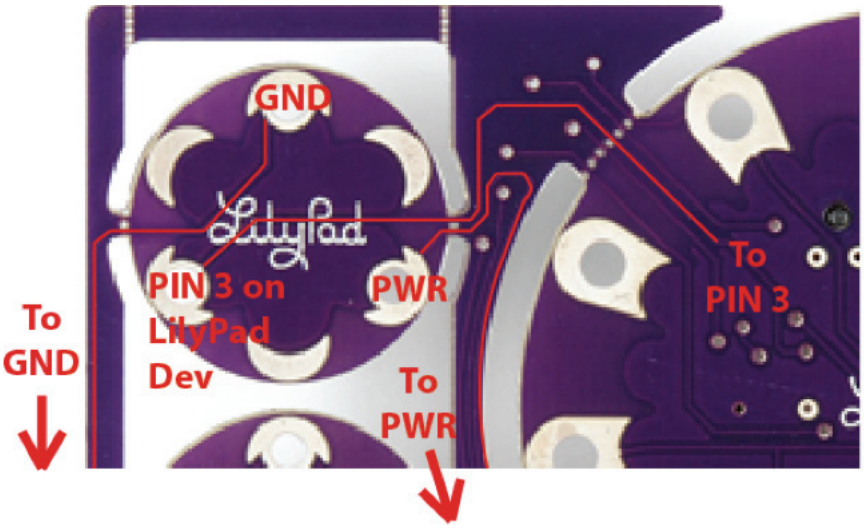

The Digital Pin connections are numbered one through eleven or twelve. There are also six Analog Pins named A1 through A6. The regular LEDs are connected to Digital Pins five and six and Analog Pins A2, A3 and A4.

**Name:**
**Date:**

## // The “Pin” Connections:

**The “Pin” Connections Between the Microcontroller and the Other Components on the LilyPad Dev Board:**

We’ll go more into these connections later. For now take a look at the back of the board. You will be able to see little tiny wires, or “traces,” that run from these Pin Connections out to the various sensors, buttons, LEDs and other circuits that are already connected to the LilyPad Dev Board. This means that the microcontroller can already talk to the inputs and outputs on the board without making any connections!



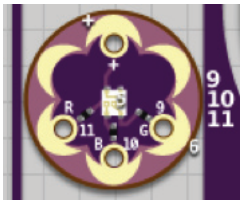
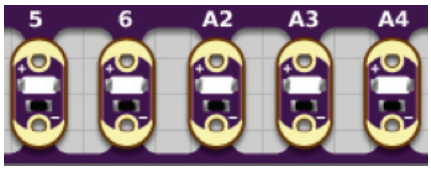

This is an example of how the temperature sensor circuit is connected to the rest of the LilyPad Dev Board. I have highlighted the traces that actually connect this sensor to the other pin connections on the microcontroller. The pin 3 connection is how the microcontroller gets a reading off of the sensor. The PWR connection is where the sensor is connected to a power supply. The GND connection is where the circuit is completed by running to the Ground connection on a power supply. As you can see, there are pin connections on the input and output circuits as well.

The PWR and GND connections travel through the rest of the board, connecting to the other sensors as well. This is why only the pin 3 connection actually goes back to the microcontroller.

**Name:**
**Date:**

## // The Outputs:

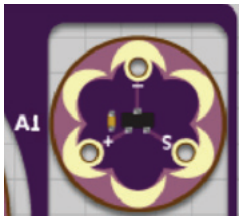
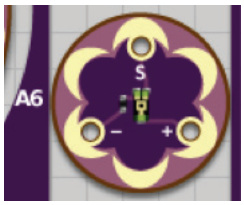
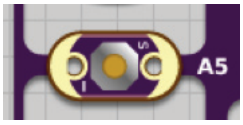
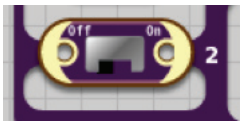
Let's talk about the outputs on the board. An "output" is any LED, motor or other component that can physically change the real world in some way. An output is a system that takes information from the microcontroller and sends that information out into the real world. In some cases an output is just an electrical signal that leaves the microcontroller and goes straight to another microcontroller! The LilyPad Dev Board's outputs are similar to your hands and feet. Outputs are how microcontrollers affect their environment. Just like your hands and feet, the microcontroller's outputs can make a system walk, pick stuff up or simply turn an LED on or off. Outputs cannot send information into the microcontroller system.

| Output Type | Output Image | Pin Connections |
|---|---|---|
| **Vibration motor:** *Vibrates.* |   | Pin 3 and ground |
| **Red, green and blue LED:** *Almost 17 million colors!* |   | Pin 9, pin 10, pin 11 and power<br>** Note: The RGB LED works the reverse as the regular LEDs below.* |
| **LED (one color):** *On and off or fade it up and down.* |   | Pin 5, pin 6, pin A2, pin A3, pin A4 and ground<br>** Note: There are 5 LEDs, each has 1 pin connection.* |
| **Buzzer:** *Plays music and beeps.* |   | Pin 7 and ground |

**Name:**
**Date:**

## // The Inputs:

Let's talk about the inputs on the board. An "input" is any sensor or system that takes information from the real world and sends that information into the microcontroller. The LilyPad Dev Board's inputs are similar to your eyes and ears. Inputs are how microcontrollers sense their environment. Just like your eyes and ears, information can only enter into the microcontroller through the inputs. Inputs cannot send information out from the microcontroller system.

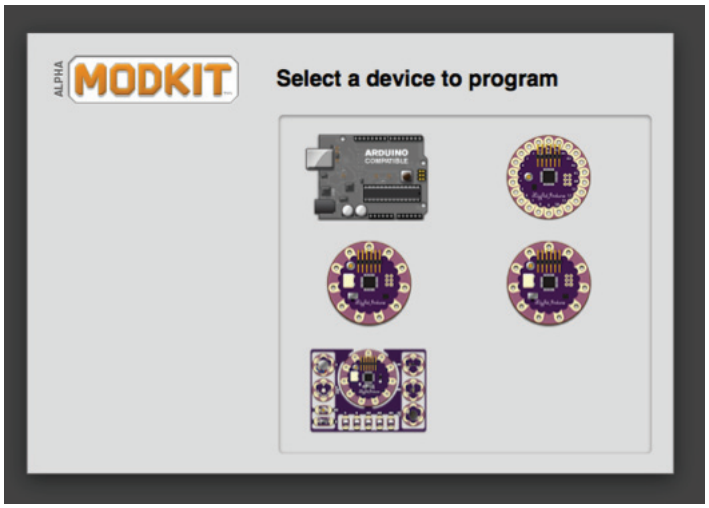
| Input Type | Input Image | Pin Connections |
|---|---|---|
| **Temperature sensor:** *Senses heat and cold.* |  | Pin A1, power and ground |
| **Light Sensor: (photoresistor)** *Senses light and shadow.* |  | Pin A6, power and ground |
| **Button Interface:** *Senses push of button.* |  | Pin 5A and ground |
| **Buzzer:** *Plays music and beeps.* |  | Pin 2 and ground |

**Name:**
**Date:**

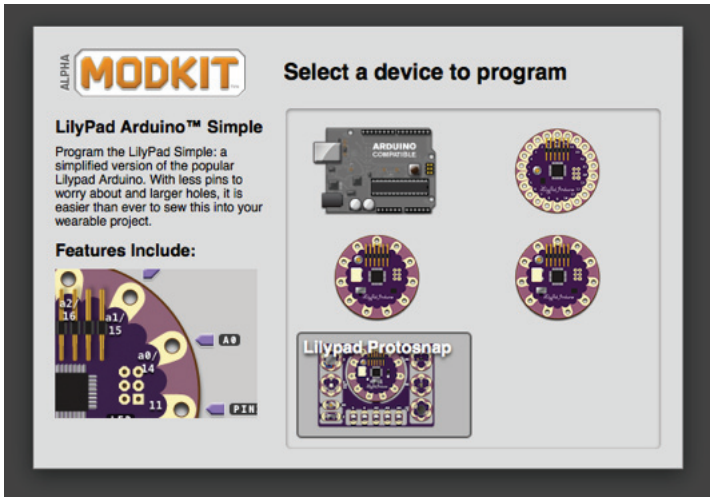
## // Opening the Modkit Application:

Now that you know a little about the LilyPad Dev Board you're ready to start writing code to load on the Board's microprocessor. Double click on the Modkit icon to open the application for writing code:


You should get a screen that looks something like this:



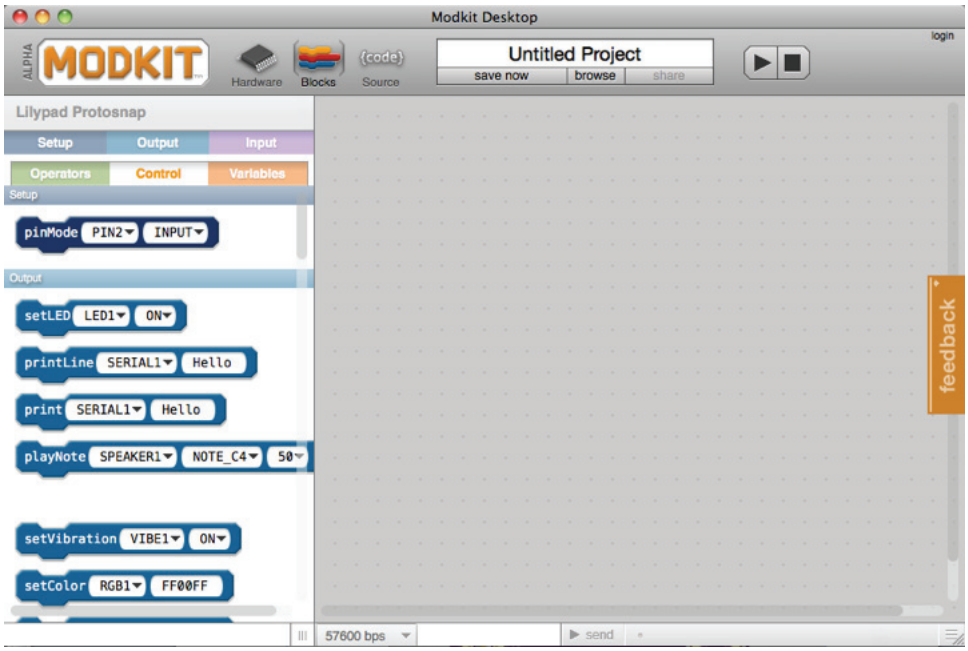

Select the LilyPad Development Board in the lower left.



**Name:**
**Date:**

# // Opening the Modkit Application:

We're almost ready to start writing code! Now your screen should look like this:




This is where you will be writing your code using the blocks on the left hand side of the screen. But wait! Before we can start programming we need to make sure the computer and the LilyPad Dev Board know how to talk to each other. That means we need to look at one of the other two views in Modkit.

In order to switch between views in Modkit you will use these three buttons in the upper left hand corner:

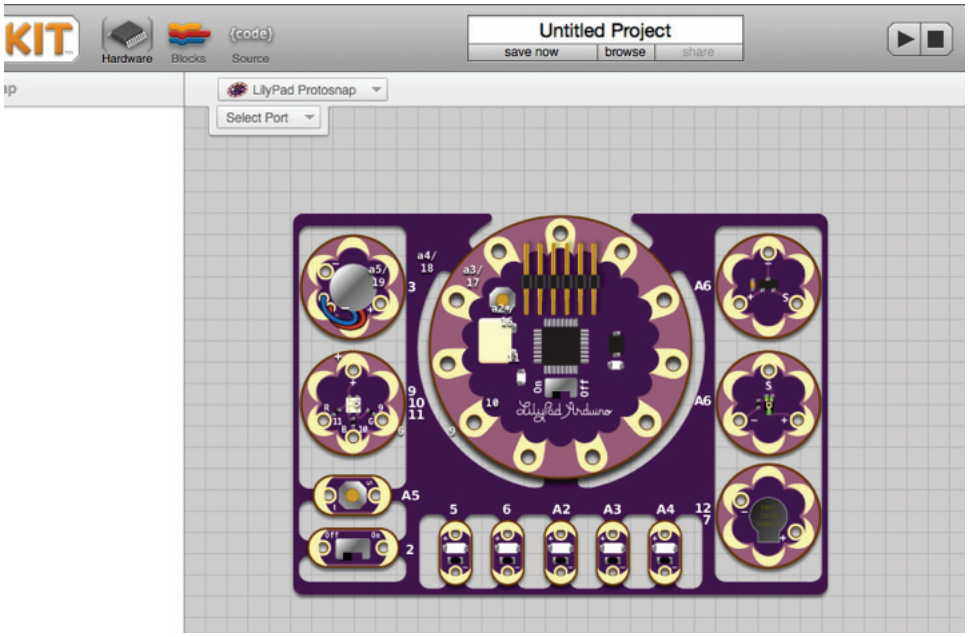

To tell the computer and the LilyPad Dev Board how to talk to each other, first click on the hardware icon to switch to the hardware view:

**Name:**
**Date:**

# // Opening the Modkit Application:

**Selecting board type:**

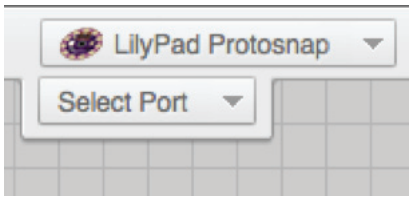
Now your screen should look like this:




Before you start programming any board with a microcontroller, you will usually need to check two things: the type of Board you are programming, and the COM Port the Board is plugged into.

Modkit already knows we have a LilyPad Dev Board because we chose the LilyPad at the beginning. We can see that we have the LilyPad ProtoSnap Development Board (that's the full name of the board. Phew, that's a lot to say!) selected in the upper right hand part of the screen:

You can plug in all types of different boards but for this activity we want to make sure we have LilyPad ProtoSnap selected.

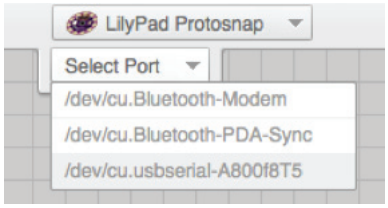


**Name:**
**Date:**

## // Selecting a Com Port:

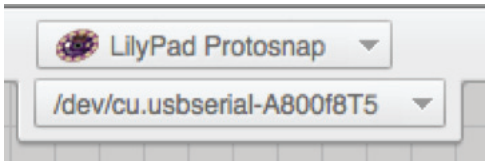
The other thing we have to select before we can put code on the LilyPad Board is the COM Port. COM Port stands for “communication port.” We plug all kinds of stuff into computers; from mice and keyboards, to cell phones and microcontrollers, with scanners and printers between! We usually plug all these machines into similar places on our computer: the trusty old USB Port. But if we have two things plugged into the computer how will Modkit know which one to send code to?

To figure out which COM Port you will be using click on the menu tab that says “Select Port” just below the Board selection tab:




Usually there are at least three options in the COM Port menu. My LilyPad is the bottom COM Port. Yours will look similar, but the number after “usbserial-“ will be a little different. Select the COM Port that says “usbserial-” if there are more than one COM Ports that says “usbserial-“ see the instructions below.

Each microcontroller has a completely different COM Port ID number. Take a second and look at your classmate’s COM Port ID number. It’s different from yours, isn’t it?




**Confused?** If you’re ever confused about which COM Port is your LilyPad Dev Board look at the COM Port menu tab, unplug the LilyPad Dev Board USB cable, and look at the COM Port menu tab again. The COM Port that is missing is your LilyPad Dev Board!

It’s important for Modkit to be able to talk to the LilyPad Dev Board through the COM Port, because otherwise you might wind up trying to upload code meant for the LilyPad Dev Board to your printer. That will really confuse your printer and your LilyPad Dev Board won’t do anything at all!

**Name:**
**Date:**

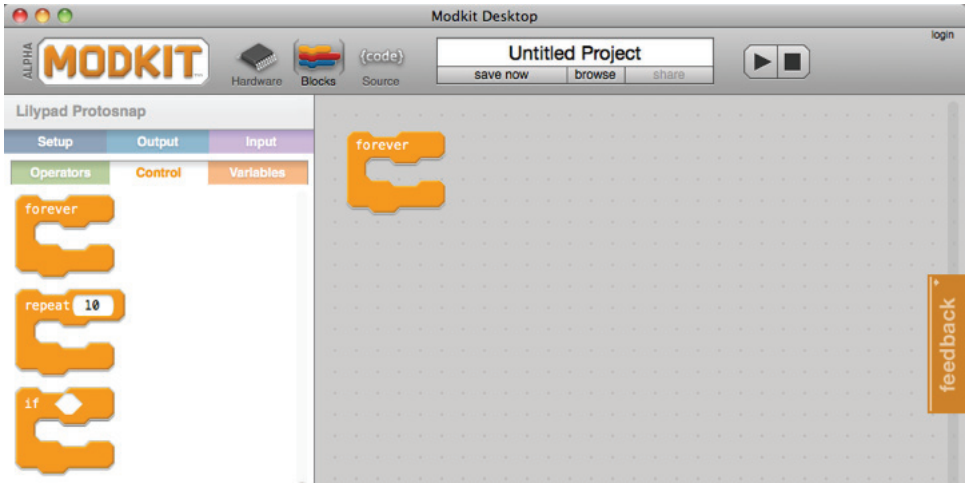
# // Writing Code:

Now go back to the code block view by clicking here:


The code you write to put onto your microcontroller is called a “Sketch.” Every single sketch needs a “forever loop.” A forever loop is what makes your LilyPad do the instructions in the code you put in its little robot brain over and over and over and, well… forever.

To create a forever loop first click on the Control section of your code library. 


Then click and hold on the forever loop from the code library on the left. Drag the forever loop into the coding window and let go of the mouse button. Your coding window should look like this now:

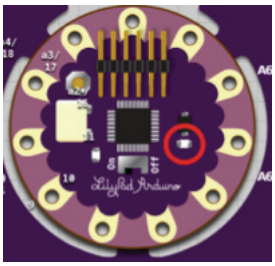



Once you load this code onto your microcontroller and start it, the microcontroller will run any code before the forever loop. Then, once the microcontroller enters the forever loop, it never exits it again. Once the code at the bottom of the forever loop is done running the microcontroller goes to the top of the forever loop and starts over. If the microcontroller loses power or gets reset everything starts over before the forever loop.

**Name:**
**Date:**

# // Writing Code:

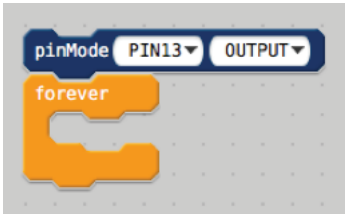
**We're writing code to upload onto the board!**



The first thing everyone learns to upload to a microcontroller is something called "Blink." This means turning an LED first on, then off. We're going to blink the little green LED on the microcontroller.

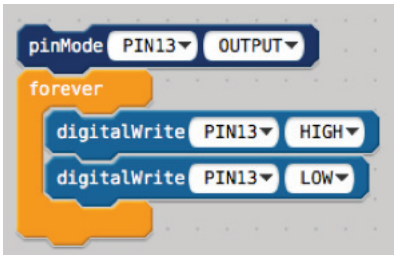
First click on Setup to see the Setup library code:


Drag a "pinMode" block to the top of the forever loop. Set the pin to pin13 and set the mode to OUTPUT.

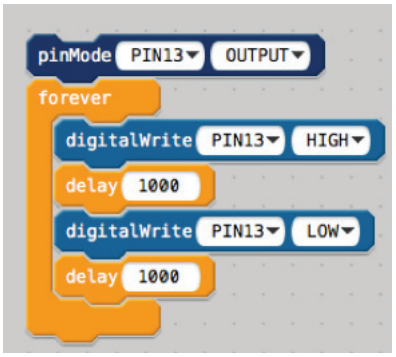



Now click on Output to see the Output library code:


Then drag two "digitalWrite" blocks into the forever loop. Set the pin to pin13 and set the first digitalWrite to HIGH and the second to LOW.




Next click on Control again, and drag two "delay" blocks into your forever loop. Put one below each of the digitalWrite blocks.



**Name:**
**Date:**

# // Uploading Code:

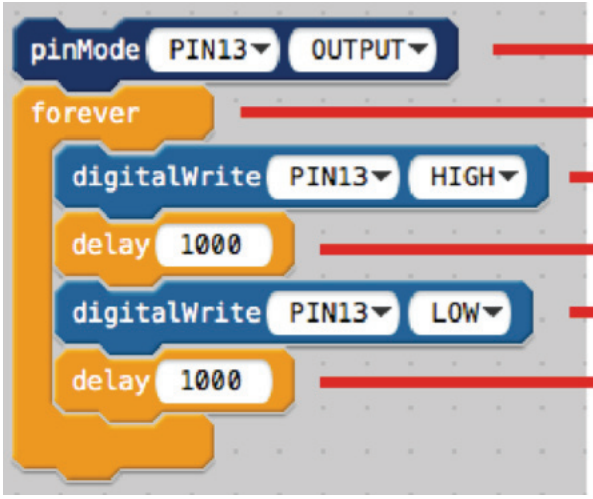
Guess What? You're ready to upload code onto your LilyPad Dev Board!!! Click on the upload button.

It looks kind of like a play button.

You should see red and green LEDs on your FTDI flashing on and off. This means the computer and the LilyPad Dev Board are talking to each other.

After your code is done uploading you should see the little green LED on the LilyPad Dev Board blinking.

Let's look at the Blink code you uploaded onto your microcontroller:




pinMode tells Pin 13 to act like an output.

Forever makes the code repeat.

digitalWrite turns the LED on (or HIGH).
Pause for a second.

digitalWrite turns the LED off (or LOW).
Pause for another second.

Try changing the numbers in delay and uploading your code again.

**Name:**
**Date:**

# // Digital Signals:

We've talked about both input and output. But the electricity (or electrical signals) that the microcontroller uses as inputs and outputs has two different types: Digital and Analog. It's important to understand the difference between the two so you can really master input and output.

Any input or output that can only be either on or off is called a "Digital signal." A digital output signal can only turn the LED all the way on or all the way off.

The Blink sketch you uploaded to your LilyPad Dev Board uses a digital signal to turn the LED on Pin 13 on and off. But in this case HIGH means "on" and LOW means "off." These are the only two options if you are using a digitalWrite command to control an output: On and off, or HIGH and LOW.

To send a digital signal from the Arduino to an output, use:


Set the menu that reads "PIN13" to the pin you want to control.

Set the menu that reads "HIGH" to either HIGH or LOW.

And remember:


HIGH means on


LOW means off

**Name:**
**Date:**

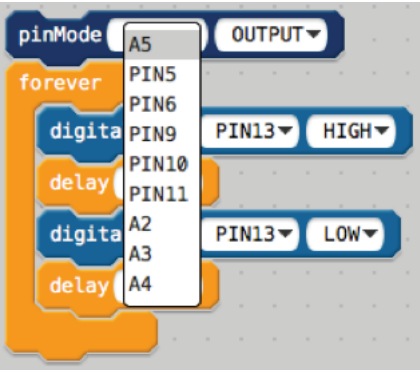
# // Using Analog:

**Making the LED light up halfway using analog:**

Now that you've got some experience turning an LED on and off using digitalWrite, let's change the code a little so that you can turn it on and off using analogWrite.

First thing we need to do is change the pin in the pinMode block. The reason we have to this is because most of the pins on your LilyPad Microcontroller are digital. But you can't turn a digital pin like pin 13 halfway off!

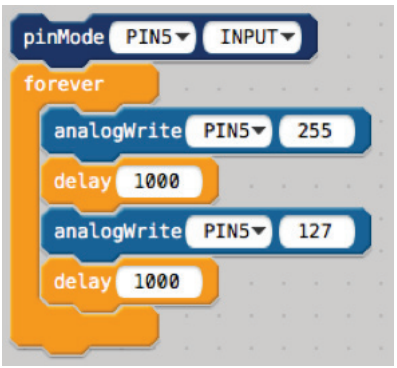
Luckily there are some pins on the microcontroller that can pretend to be analog output pins. We call these pins PWM. PWM stands for Pulse Width Modulation. That's really complicated though… the important thing to remember is that you can turn PWM pins a little on, halfway on, or all the way on. You can't do that with digital pins!




I changed my pinMode from PIN13 to PIN5.

The next thing we need to do is switch out the digitalWrite blocks of code for analogWrite blocks of code. Then we need to set the pin of the analogWrite blocks to PIN5 so we are talking to the right pins.

See the numbers in the analogWrite blocks? Those are how you control the amount of electricity the microcontroller sends to the pin. With the regular LEDs, "255" means all the way on and "0" means all the way off. All the rest of the numbers are somewhere in between. That means if you set the numbers to 127, your LED will turn on halfway. Play around with these numbers until you understand what they mean.



**Name:**
**Date:**

# // PWM Signals:

**PWM signals - kind of like analog:**

We've talked about digital signals, which can only be all the way on or all the way off. But a PWM signal is almost like an analog signal. It can be controlled like the sound on your radio. A PWM signal can be all the way on, all the way off, or any amount in between.

Any output that can be either on, off, or anything in between is called a "PWM signal." A PWM signal can turn your LED on or off or set it to anything in between the two.

The Blink sketch you uploaded to your LilyPad Dev Board uses a analog signal to turn the LED on Pin 5 all the way on and halfway on. 255 means "on" and 127 means "halfway on." There are 256 options if you are using a analogWrite command to control an output: 0 - 255.

**To send a PWM signal from the Arduino to an output use:**


Set the menu that reads "255" to any number between 0 and 255.

And remember:


255 is on


127 is 1/2 on and 1/2 off

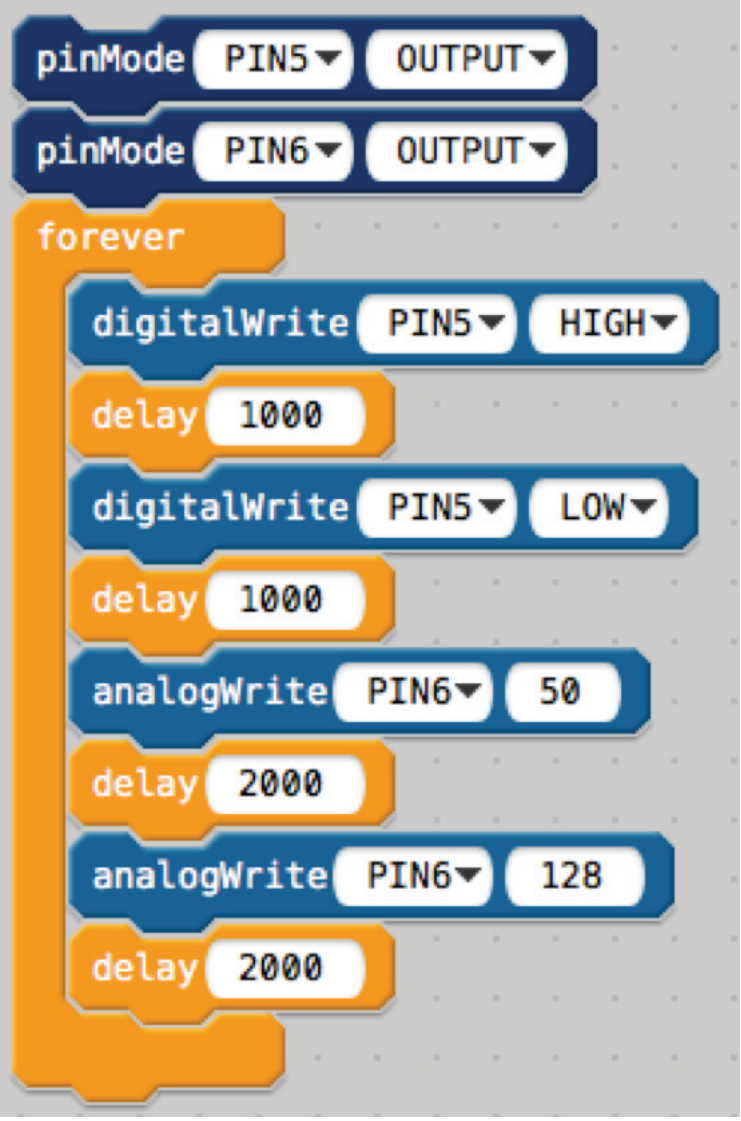

0 is off

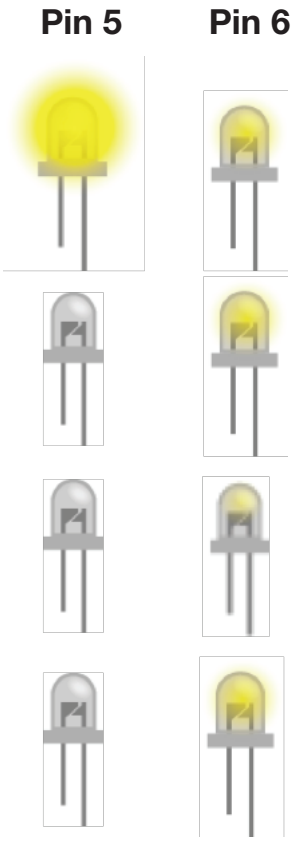
**Name:**
**Date:**

# // Blinking Two Lights:

**What if I Want to Blink Two Lights at Once?**

Blam. Here's how you do that.



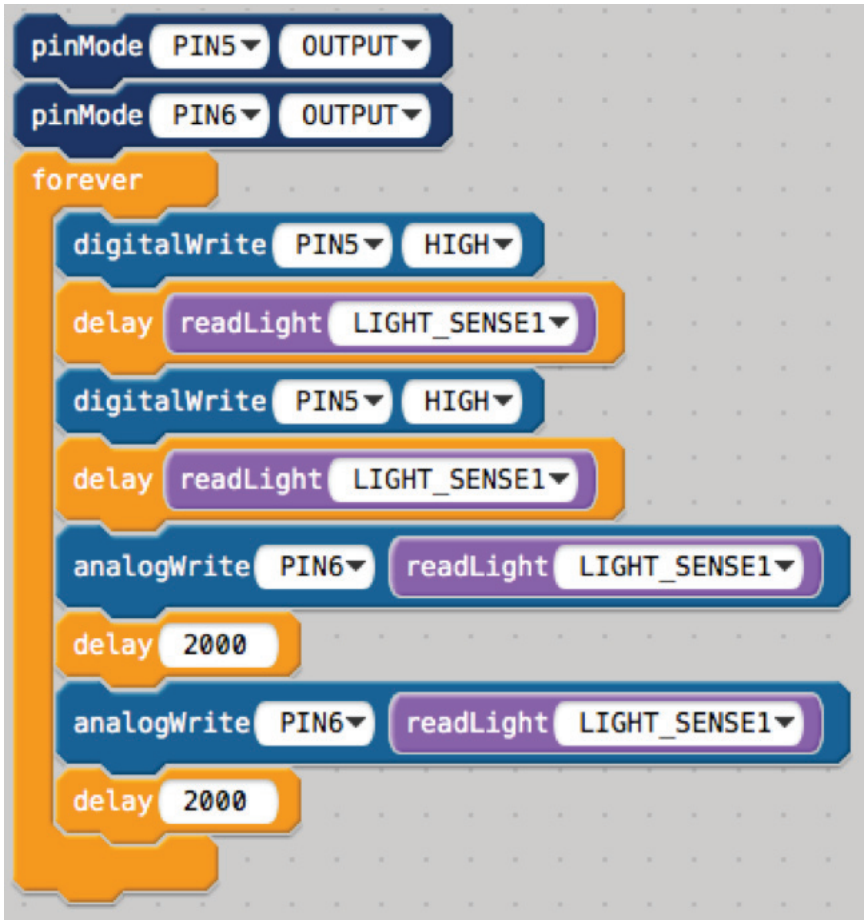
**Name:**
**Date:**

# // Analog Inputs:

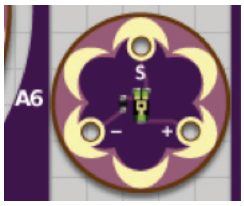
**How do I use analog inputs?**

Here's how you can use the light sensor to control some LEDs.

To find the purple input blocks, click on the Input section of your code library.


This input uses the light sensor.
The light sensor is an analog sensor.
The light sensor goes up to 1023.



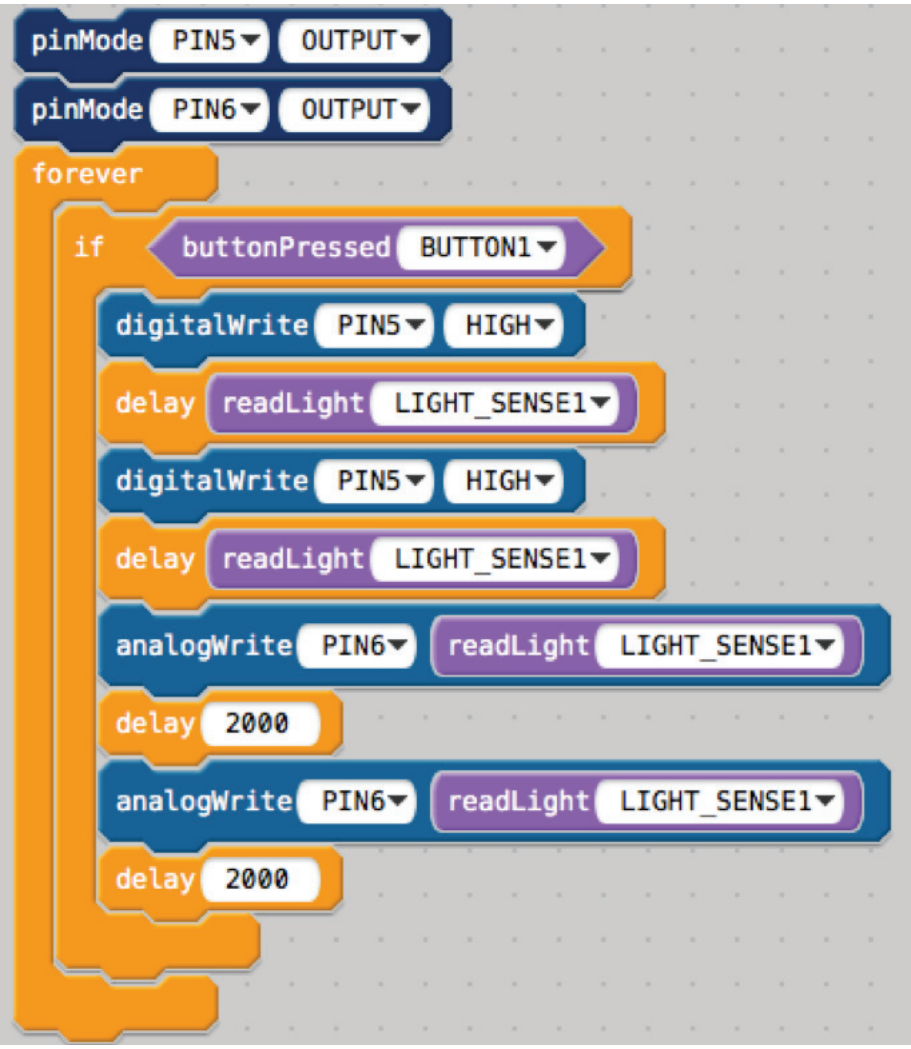
**Name:**
**Date:**

# // Digital Inputs:

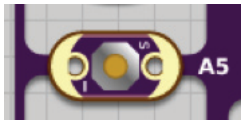
**How do I use digital inputs?**

Here's how you can use a button to change the brightness of an LED.

Add an "if" block from the Control section of your code library.
Then add a buttonPressed block from the input section of your library.




This code uses the button.To make your lights blink and change brightness, you have to press the button.

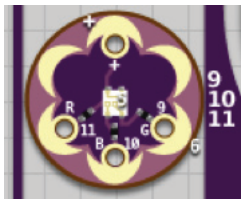


**Name:**
**Date:**

# // RBG LED:

**One last thing about the RGB LED….**

The Red, Green and Blue LED is backward! You have to send it the opposite of what you used with the other LEDs.




**This guy is weird.**


LOW means on


HIGH means off

It also means that:


0 is on


127 is 1/2 on and 1/2 off


255 is off