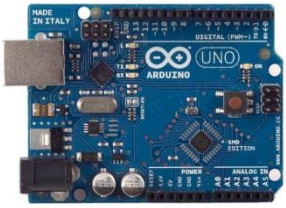
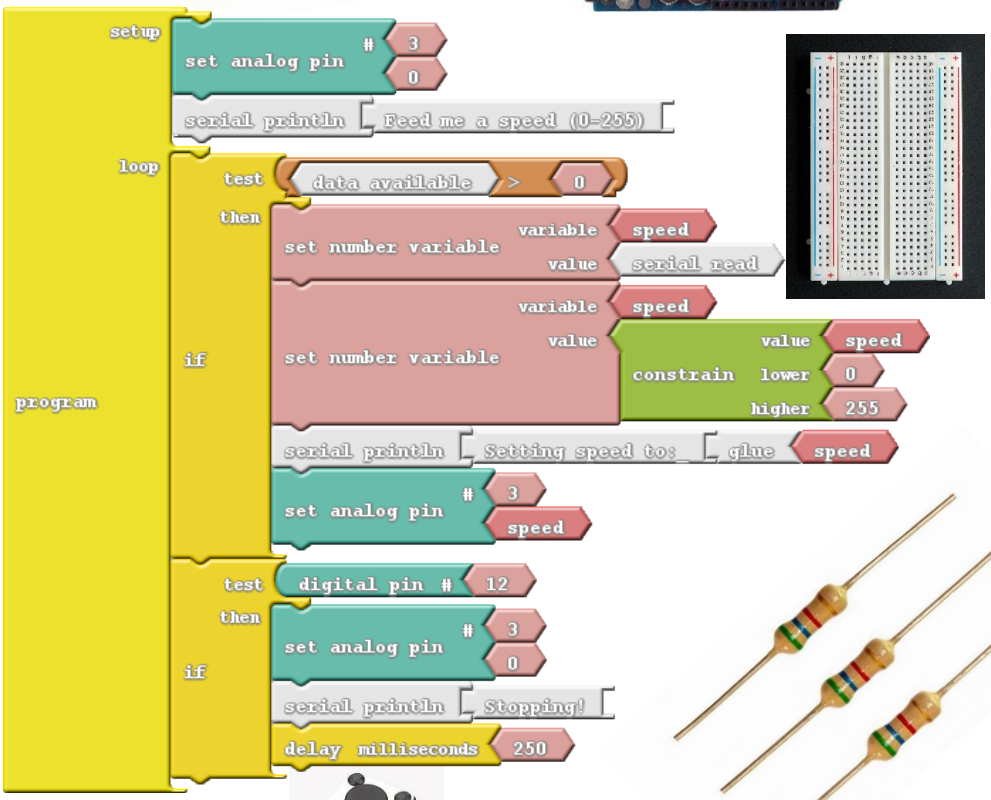
Arduino Basics – Intro to ArduBlocks

Materials:



Arduino

ArduBlocks Software



Arduino IDE

Laptop

Breadboard

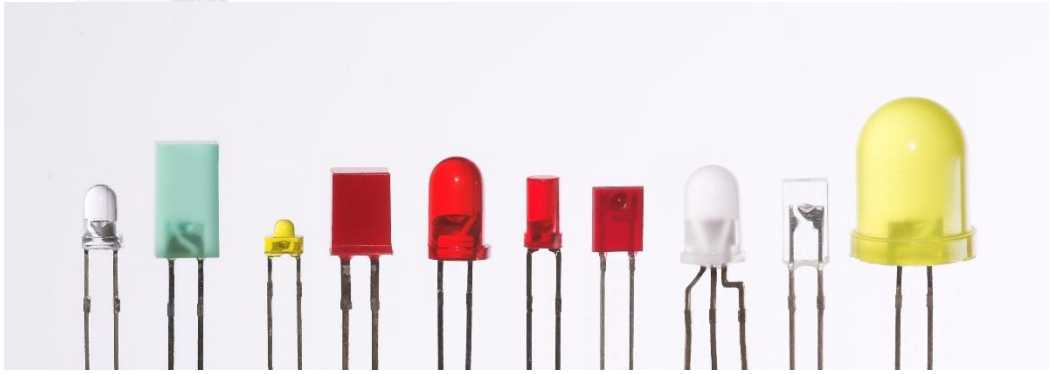
Wires

Resistors

LEDs

Potentiometer

Temporary Push Button



By: Mackenzie Ridley

7/15/2015

Get the Software

Download Arduino IDE

https://www.arduino.cc/en/Main/Software

Download ArduBlocks to Arduino IDE

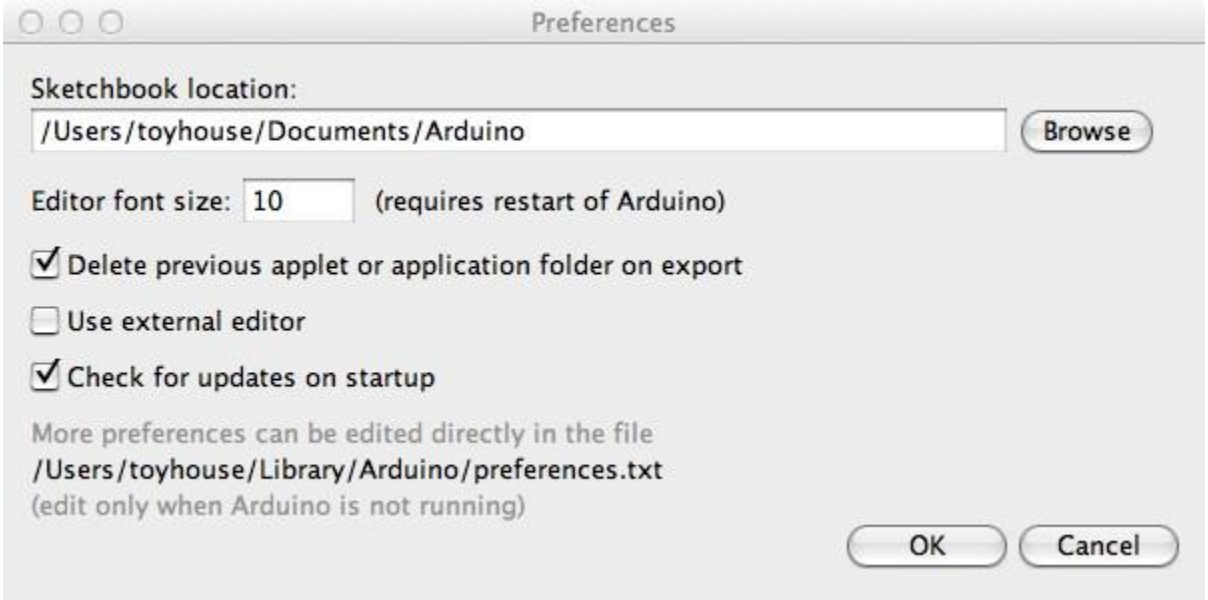
http://sourceforge.net/projects/ardublock/?source=typ_redirect

How to Connect ArduBlocks to Arduino IDE

1.Download ardublock-all.jar ArduBlock

2. In Arduino IDE, open menu "Arduino" -> "Preferences"

3. Find "Sketchbook location:"



- In Mac, it's by default "Documents/Arduino" under user's home directory
- In Linux, it's by default "sketchbook" under user's home directory
- In Windows, it's by default "Documents\Arduino" under user's home directory

4. Copy ardublock-all.jar to tools/ArduBlockTool/tool/ardublock-all.jar under "Sketchbook location". Assume the user is "abu,"

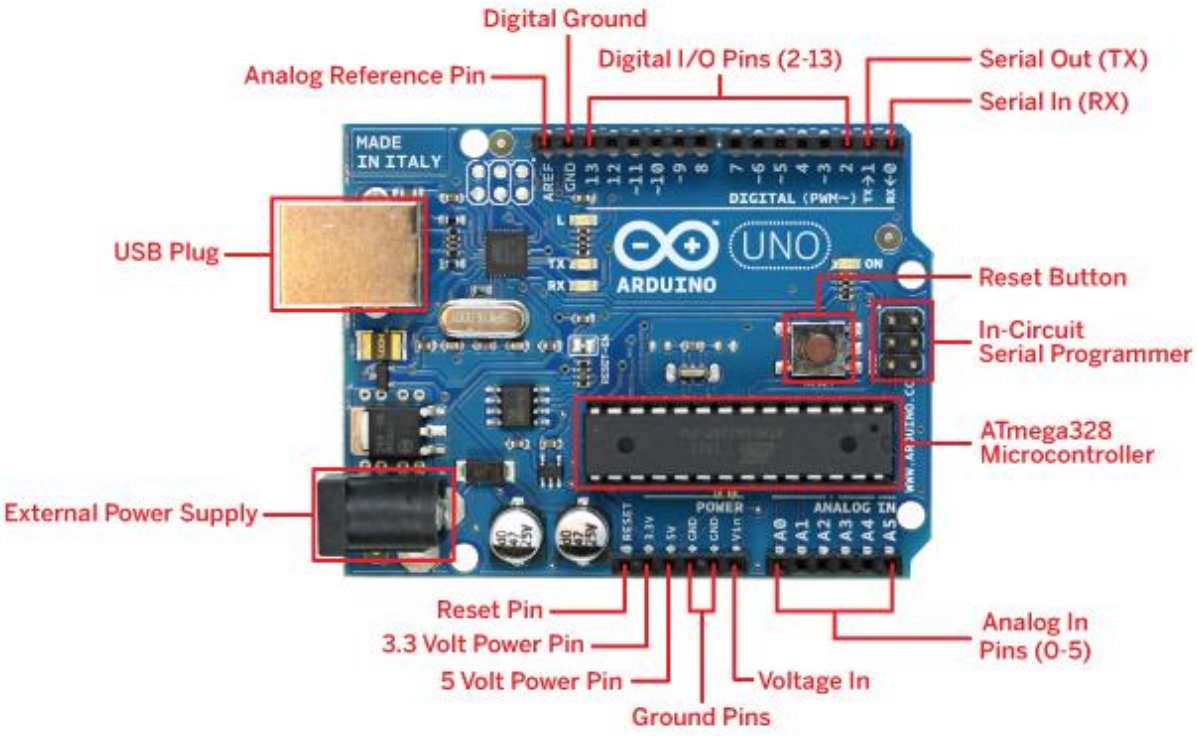
- In Mac, /Users/abu/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar
- In Linux, /home/abu/sketchbook/tools/ArduBlockTool/tool/ardublock-all.jar
- In Windows, C:\Users\abu\Documents\Arduino\tools\ArduBlockTool\tool\ardublock-all.jar

* Be careful, the name of folder "ArduBlockTool" under tools folder **is case sensitive**.

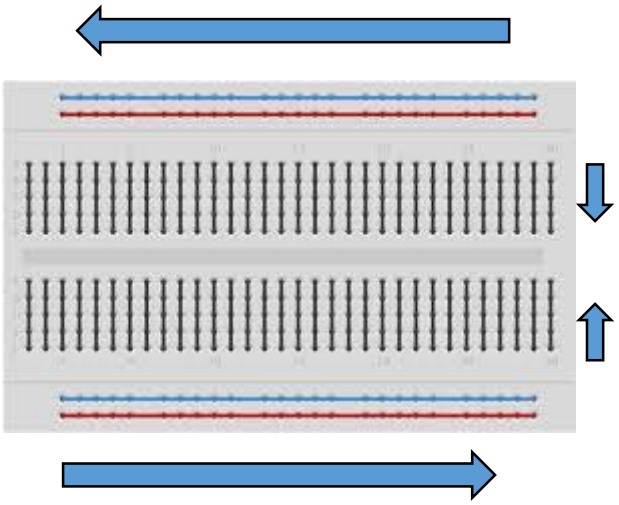
5. Start the Arduino IDE and find ArduBlock under the Tool menu

Materials

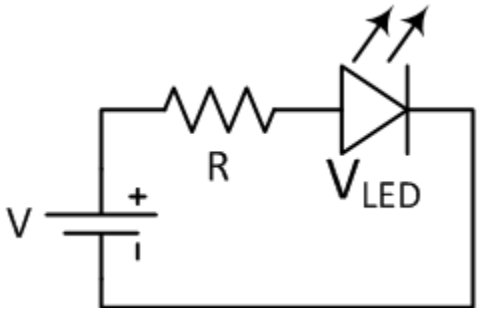
Arduino UNO

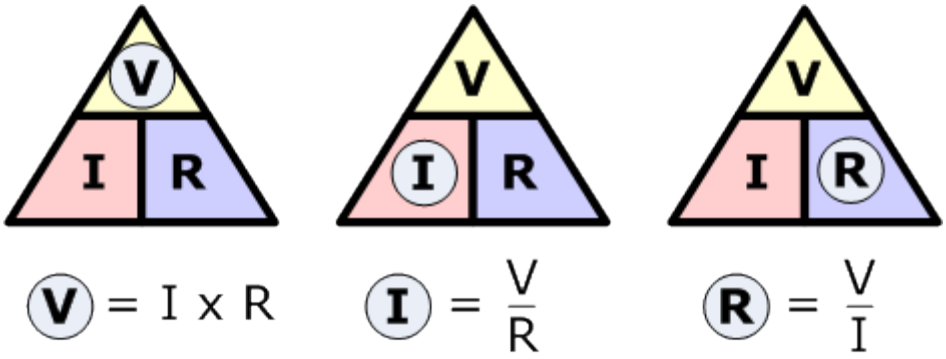


Breadboard



Circuits





Resistors

Here's an example of a 4.7kΩ resistor with four color bands:

Potentiometers

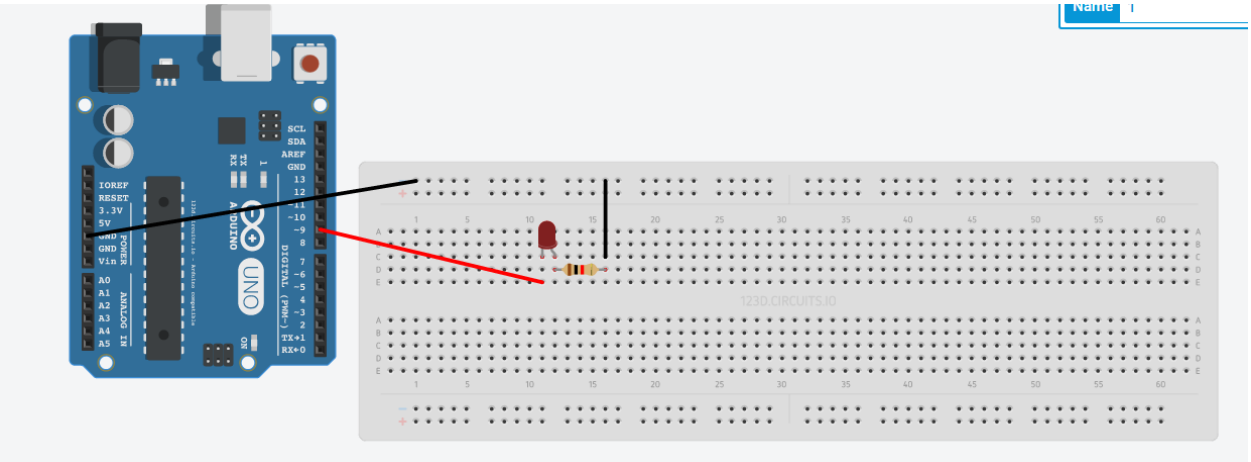
ArduBlocks

- All blocks within a 'loop do' function, and only one loop per code.
- Digital functions have rounded edges, and Analog functions have points (backwards chevrons).
- Double click on a block to drag it.
- Drag blocks over Categories and release to remove from screen.
- When blocks are close, click on one to make Sure Blocks snap together.

Categories Explained

Project 1 – Turn on an LED

What do we want: What we want to do is to tell the Arduino that a specific pin on our right-side row of the Arduino board is an output. We want that output to be fully open. It does not matter if this pin is Digital (ON or OFF, TRUE or FALSE, 1 or 0) or Analog (smooth resolution of 5 volts, spread between integers 0 and 255). Both will allow us to turn the LED on as an output.

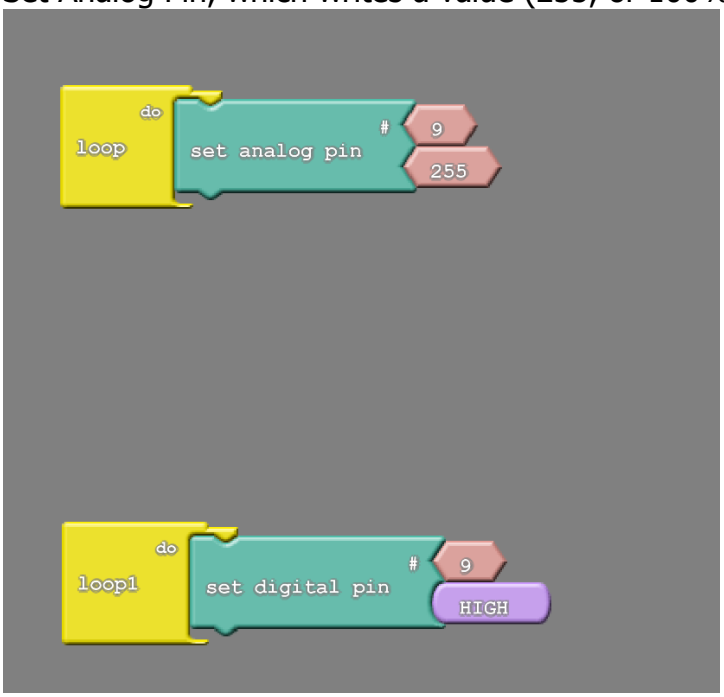


Steps:

1. Start out with 'loop' function, found in the top control category.
2. Set Digital Pin, which writes a binary value (HIGH or LOW, 1 or 0) to a pin #(9).

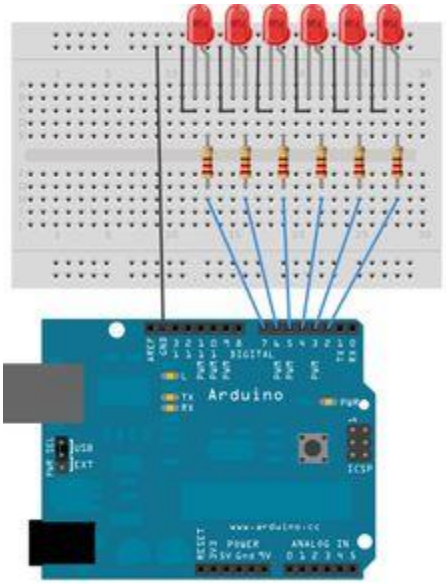
OR

3. Set Analog Pin, which writes a value (255, or 100% ON) to a pin # (9).



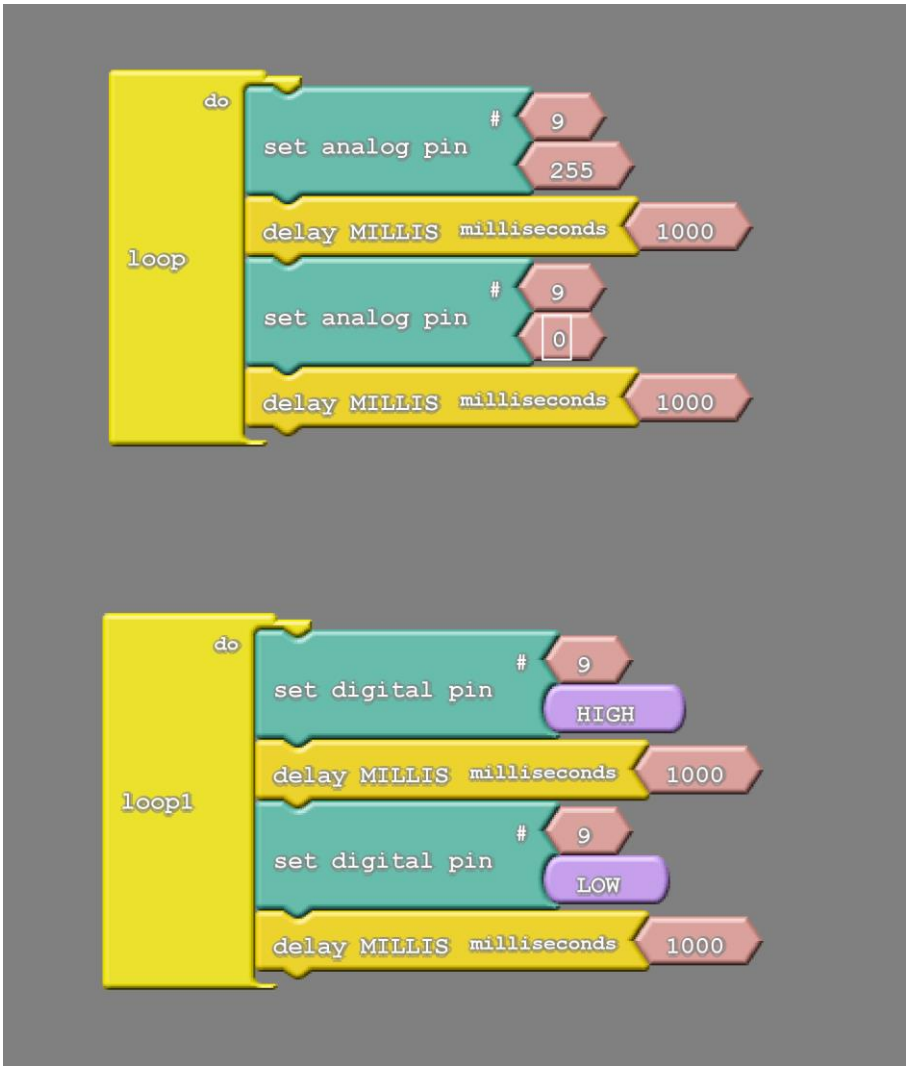
Project 2 – Blink LED

How do we make an LED turn on and off for a given time? We need a way to Turn the LED on, pause for a moment of time, turn the LED off, pause for another moment in time, and then repeat the cycle. Since we are putting this code in a 'loop' function, the repeating part is done for us by the Arduino.



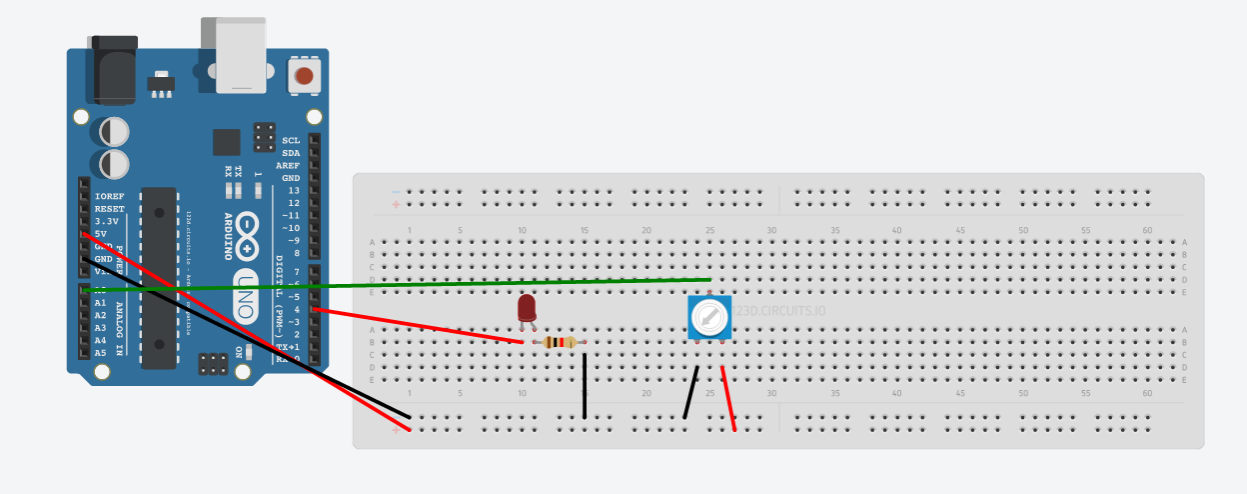
Steps:

1. Rebuild the code block from Project 1
2. Search for a time delay block. This is found within the Generic Hardware category. (*1000 milliseconds = 1 second).
3. We want to turn the light off. Right click on your first Set Analog/Set Digital pin.
4. Play around with delay times! The way the code is written, the light will blink on and off for as long as we keep it powered.


do
loop
set analog pin
#
9
255
delay MILLIS
milliseconds
1000
set analog pin
#
9
0
delay MILLIS
milliseconds
1000
do
loop1
set digital pin
#
9
HIGH
delay MILLIS
milliseconds
1000
set digital pin
#
9
LOW
delay MILLIS
milliseconds
1000

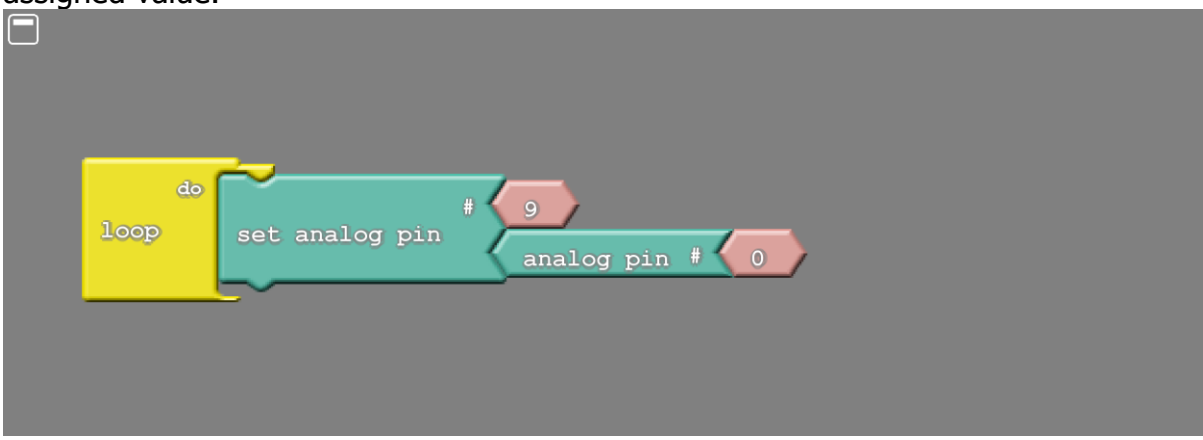
Project 3 – LED Dimmer with Potentiometer

We want to turn on an LED with a pin from the Arduino. Then, we want to read the resistance on the potentiometer with an analog input pin (bottom left side of Arduino). From here, we want to define the LED's brightness as the value of the potentiometer.



Steps:

1. Start off with our 'do loop'.
2. Add the 'AnalogWrite' function. Define a pin # as an output, and write a value to the output.
3. The output is automatically an interger. Let's delete the interger and add our own value. This will be the value of the potentiometer.
4. We want to read the potentiometer. Fill in 'AnalogRead' block as the assigned value.

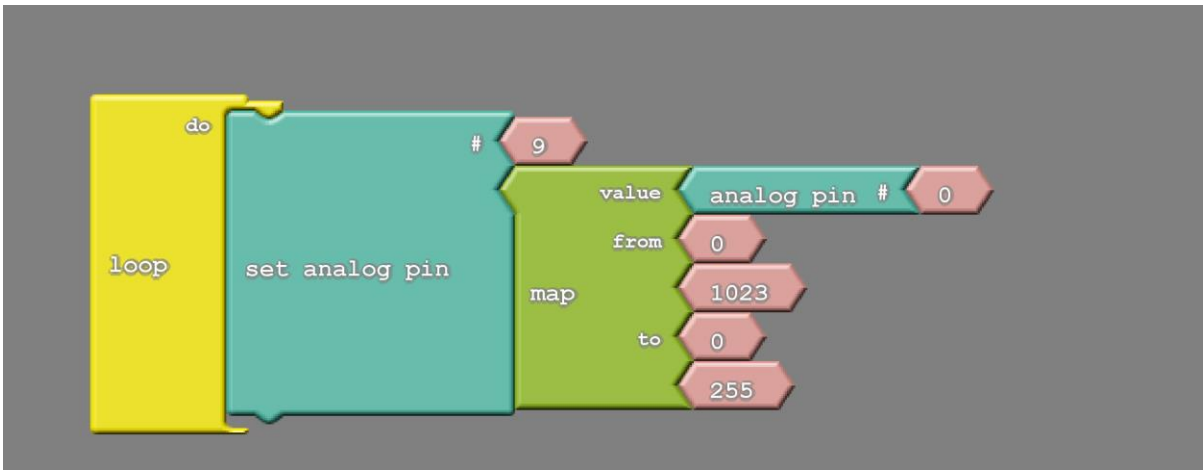


…. Did it work? ….

The Arduino has an input analogRead resolution of 0-1023, and an analogWrite resolution of only 0-255. The potentiometer has a resolution that's 4 times greater than what can be defined in the LED, so the LED runs through four cycles

of a dimmer. If we could only somehow change the resolution of the potentiometer from 0 – 1023 to 0 - 255…

5. Look down at the bottom loop, and notice the 'map' block. This does just what we need; change the pot resolution down to 0 – 255, so that it matches the LED output. You'll find that map block in the bottom of the category 'Math Operators'.



Project 4 – More with the potentiometer

Dimming might not be for everyone. What if we want to control our very own light show?

What we are going to do is use the potentiometer to control how fast the light turns on and off. This involves setting up a blinking light, just as we accomplished earlier. Instead of the light blinking for 1 second, let's set it to blink at the rate of our potentiometer!

This will use a similar circuit as the last project.

Steps:

1. Start with 'loop'.
2. Define and Write analog (or digital) output pin to an ON value.
3. Add a delay of time
4. Redefine the pin output to be OFF.

5. Remove time integers in delay blocks (1000).
6. Replace with our Map function of the potentiometer's Analog input value.