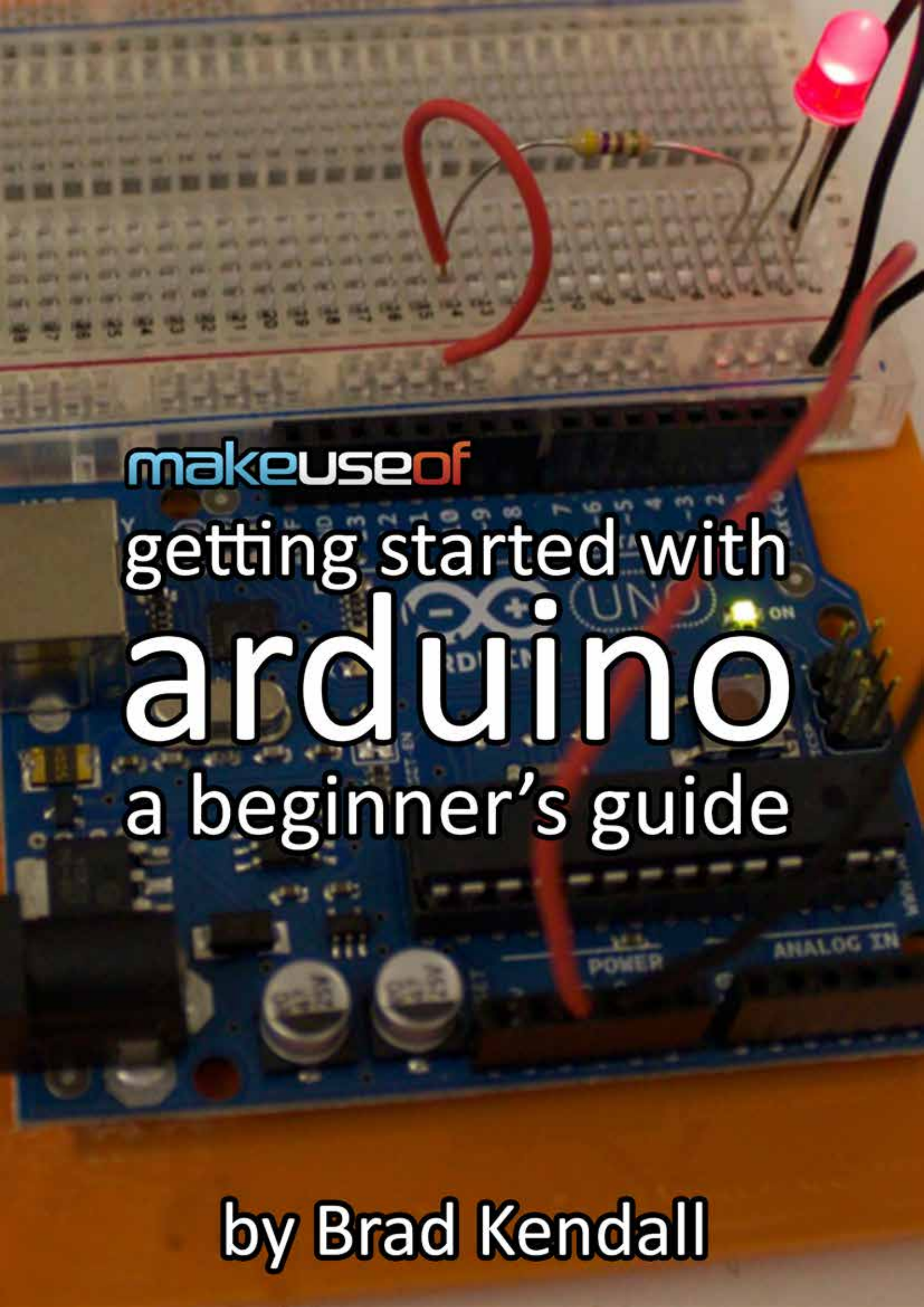

makeuseof
getting started with
arduino
a beginner's guide
by Brad Kendall

By Brad Kendall
http://www.bradkendall.ca/

Published August 2013

share: 

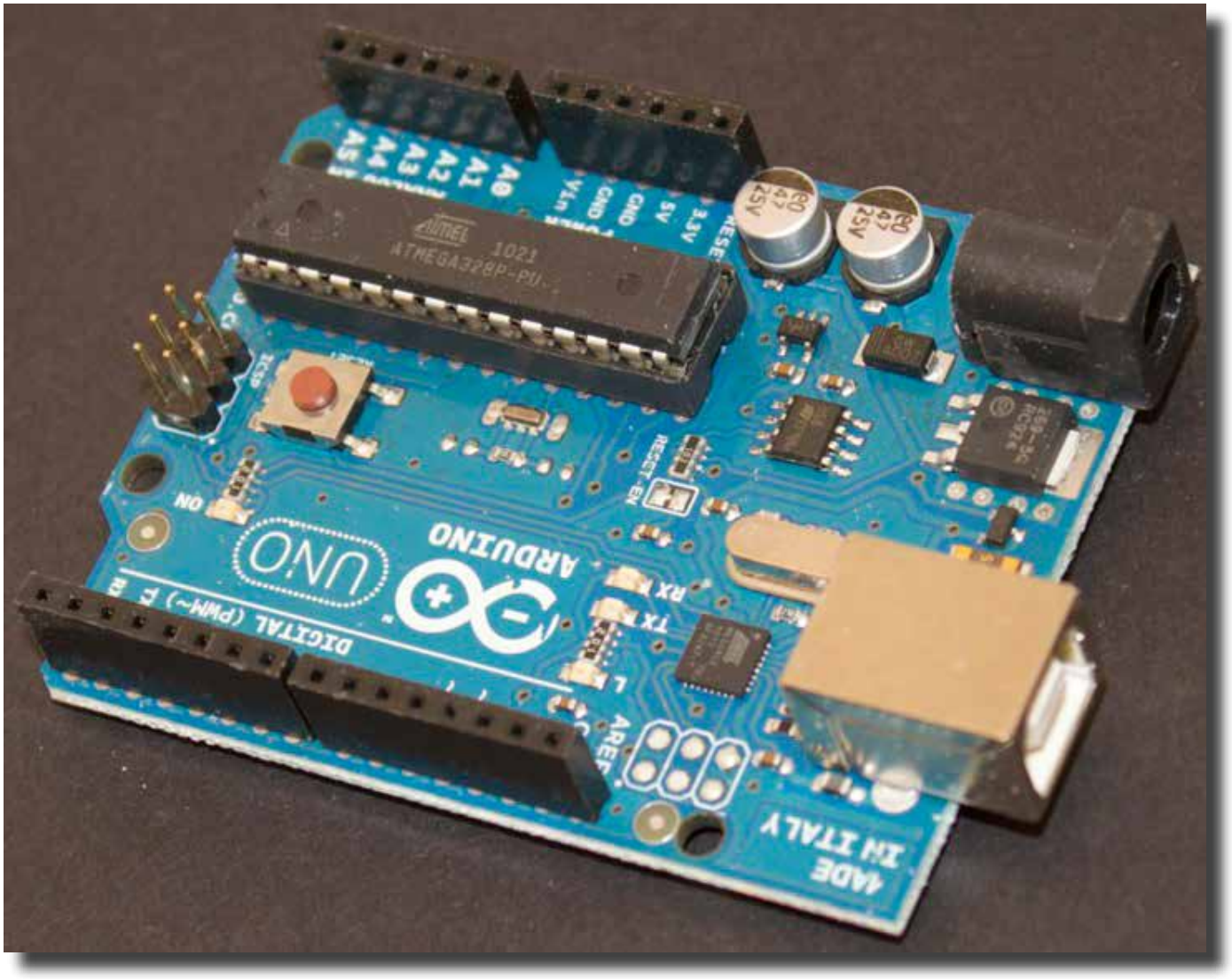
Think you've got what it takes to write a manual for MakeUseOf.com? We're always willing to hear a pitch!
Send your ideas to justinpot@makeuseof.com.

Table Of Contents

1. Intro to the Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language and the Arduino Development Environment. Arduino projects can be stand-alone, or they can communicate with software running on a computer.



There are plenty of other microcontrollers available. So you may be asking, why choose the Arduino? Arduino really simplifies the process of building projects on a microcontroller making it a great platform for amateurs. You can easily start working on one with no previous electronics experience.

That is what this guide is about.

In addition to Arduino's simplicity, it is also inexpensive, cross-platform and open source. The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced hobbyists and professionals can make their own version of the Arduino, extending it and improving it.

Believe it or not, even relatively inexperienced users can build a version of the Arduino module on a breadboard in order to understand how it works and save a little bit of money.

2. What Can You Do With an Arduino?

There is a lot you can do with an Arduino. An Arduino can basically do anything by interfacing sensors with a computer. This would allow you to take any sensor and have any action applied with the readings. For example (in one of our projects) we will read the level of light in a room and adjust an LED's brightness to react based on that input. This of course is a simple example of what you can do with an Arduino. A more complicated example would be to read from multiple sensors and use that data to affect other outputs. Think of the possibility of wiring your house with all sorts of different sensors (photocells, oxygen sensors, thermometers) and having it adjust your blinds, air conditioner and furnace and make your house a more comfortable place.

Hackers have used Arduinos to create some amazing electronics projects. Things like:

- *Robots*
- *Breathalyzers*
- *Remote controlled cars*
- *3d printers*
- *Video games*
- *Home automation systems*

And much more. Read about more great examples of Arduino projects.

3. What Is Inside an Arduino?

Although there are many different types of Arduino boards available, this manual focuses on the Arduino Uno. This is the most popular Arduino board around. So what makes this thing tick? Here are the specifications:

- *Processor: 16 Mhz ATmega328*
- *Flash memory: 32 KB*
- *Ram: 2kb*
- *Operating Voltage: 5V*
- *Input Voltage: 7-12 V*
- *Number of analog inputs: 6*
- *Number of digital I/O: 14 (6 of them pwm)*

The specs may seem meager compared to your desktop computer, but remember that the Arduino is an embedded device. We have a lot less to process than your desktop.

Another wonderful feature of the Arduino is the ability to use what are called “Shields”. Although we will not be covering shields in this manual, an Arduino shield will give you crazy functionality like you wouldn’t believe. Check out this list of some really cool Arduino shields to take your projects to the next level.

4. What You Will Need For This Guide

Below you will find a shopping list of the components we will use for this manual. All these components should come in under $50.00 USD. This should be enough to give you a good understanding of basic electronics and have enough components to build some pretty cool projects.

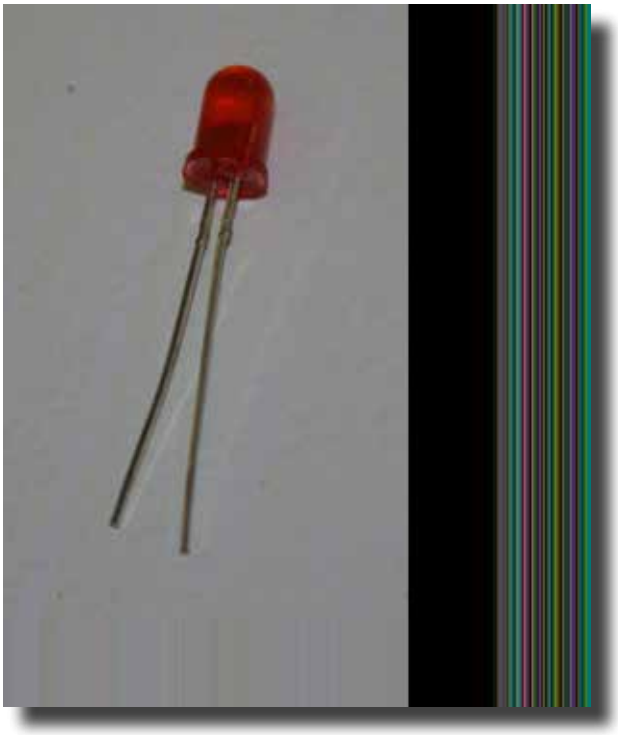
- *1x Arduino Uno Microcontroller*
- *1 x USB A-B Cable (same as your printer takes)*
- *1x Breadboard*
- *2 x LEDs*
- *1 x Photo Resistor*
- *1 x Tactile Switch*
- *1 x Piezo Speaker*
- *1 x 10 k-Ohm Resistors*
- *1 x 2 k-Ohm Resistors*
- *2 x 1 K-Ohm Resistors*
- *1 x Jumper Wire Kit*

5. Electrical Component Overview

5.1 What is a Breadboard?

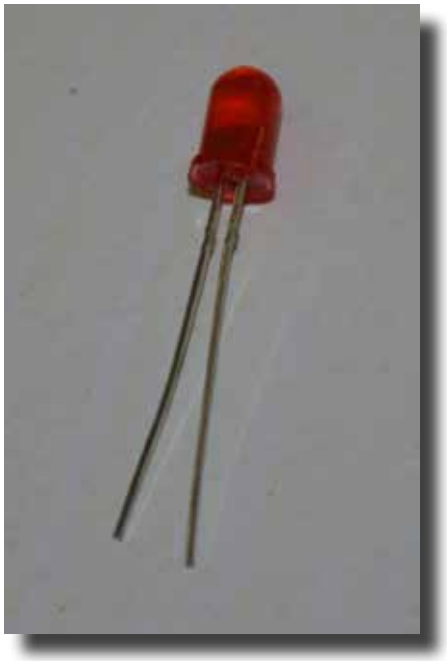
Breadboards are blocks of plastic with holes into which wires can be inserted. The holes are connected electrically, so that wires stuck in the connected holes are also connected electrically.

The connected holes are arranged in rows, in groups of five, so that up to five parts can be quickly connected just by plugging their leads into connected holes in the breadboard. When you want to rearrange a circuit, just pull the wire or part out of the hole, and move it or replace it. The breadboard I recommended also includes power and ground lanes on each side for easy power management.



5.2 What is an LED?

An LED, short for Light Emitting Diode, is a semiconductor light source. LEDs are typically used as visual indicators. For instance, your new Arduino microcontroller has an LED on pin 13 that we frequently use to indicate an action or event.

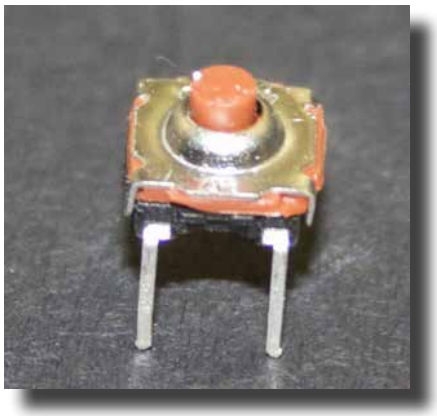


5.3 What is a Photo Resistor?

A photo resistor allows us to measure light by decreasing its resistance when it detects an increase of light intensity.

5.4 What is a Tactile Switch?

A tactile switch is an electric switch that controls the flow of electricity. When pressed, the switch completes the circuit. Basically, it is a button.



5.5 What is a Piezo Speaker?

A piezo speaker is a single frequency beeper that converts an electrical signal into a tone. This will allow your Arduino to sing to you.

5.6 What is a Resistor?

A resistor is an electrical component that limits or regulates the flow of electricity.

5.7 What are Jumper Wires?

Jumper wires are short wires that are used for prototyping circuits. These are what you will use to connect the various components electrically to your Arduino.

6. Programming Overview

If you're not too familiar with programming, this guide should get you used to some of the fundamentals. If you'd like to learn more about Arduino-specific functions, http://www.arduino.cc/en/Reference/HomePage is an excellent resource.

6.1 Variables

A variable is defined as a quantity that can assume any of a set of values. In the Arduino programming language, variables have types associated with them, which provide the set of valid values the variable can hold. Some languages are not strict and allow a variable to hold nearly anything, but that is out of the scope of this manual.

For example, a variable with type 'int' can only hold integer values like 1 or 12, and not 12.5 or "cats". Unfortunately, no variable is capable of holding a cat, something the programming world is quite upset about.

Variables are an excellent resource, as they improve code readability and reuse, and are extremely convenient for use as temporary storage.

Before using a variable, you must declare it. This merely lets the Arduino compiler know what data type your variable will hold.

An example of a variable declaration is as follows:

```
int itemCount;
```
In this case, the variable will be of type int, and therefore will only accept integers.

Here are a few example assignments and operations.

```
itemCount = 4; itemCount = itemCount + 8; // itemCount now holds the value 12. itemCount = "10"; // This will not compile.
```

6.2 Functions

A function is essentially a group of instructions that perform a specific task. There are many built-in functions, such as digitalWrite() or tone(). In those cases, you don't necessarily have to see the code, but can still reap the benefits. You can also specify your own functions.

The general form of a function is:

```
[return type] [function name] ({arguments}) { [ Code to execute ] }
```
Note that functions can return data, as illustrated by the function having a return type.

In many cases, there is no data to return, and in that case, the keyword 'void' would be used.

The function name is a user-friendly 'handle' to reference later (digitalWrite would be the function name for the digitalWrite function).

A function can accept zero or more arguments. Each argument must be of the form [datatype] [identifier]. For example, if we called a function foo as such:

```
foo(10);
```
The function header for foo would have to look like:

```
void foo(int number) { }
```
In the function, code can reference 'number' to retrieve the passed value. Outside of the function, 'number' would be undefined.

Say we want to write a function to multiply two numbers, for whatever reason. This function would look like:

```
int multiply(int num1, int num2) { int result; result = num1 * num2; return result; }
```
Note that this could simply look like:

```
int multiply(int num1, int num2) { return num1 * num2; }
```
It's usually a good idea to be liberal with the use of spaces, as it makes for much easier debugging. To each their own, however.

6.3 Logic Overview

You'll often find yourself wanting to execute certain code under certain conditions. This will give you a quick overview of the logical operators you have to work with.

First up, with the exception of the NOT operator, each logical operation takes two operands.

== - The Equals operator

This operator ensures that both operands are equal to one another. To test whether or not the operands are not equal to one another, use the != (not-equals) operator.

Example:

```
4 == 4 (true) 4 == 5 (false) 4 != 5 (true)
```

&& - The AND operator

The AND operator is quite similar to the equals operator, except it does not evaluate to true when both operands are false.

For example: (true && true) evaluates to true, while (true && false) and (false && false) both evaluate to false.

|| - The OR operator

The OR operator will evaluate to true so long as at least one of the two operands is true.

The only time OR will evaluate to false is if both the operands are false.

! - The NOT operator

This simply flips the truthiness of the operand specified. !false == true.

Using Multiple Expressions

Sometimes you'd like to have more than one test. Fortunately, since (as above), something like (false == true) will evaluate to false, nesting statements in brackets works, and the statements in brackets will be evaluated first.

For example:

```
if (( a != b) && (b > 12))
```

a != b and b > 12 will have to be evaluated first, as their outcome determines whether the entire logical expression is true.

The past two sections should have given you enough basic knowledge to get started with our projects below. If it all seems a little complicated, don't worry. It will make a lot more sense when we apply it in a practical sense.

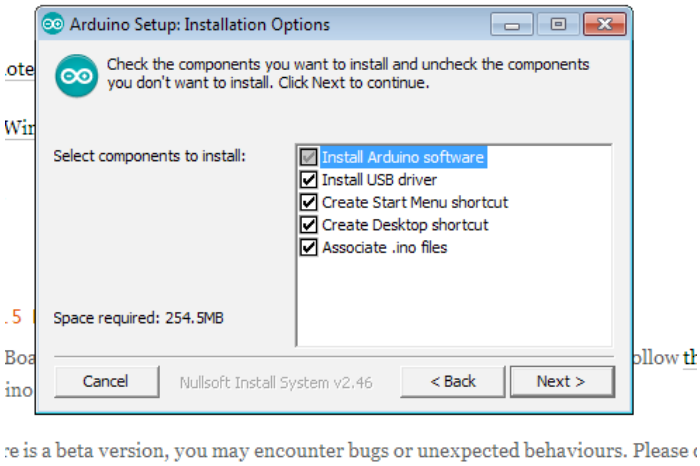
7. Setting Up Your Arduino

Before we can start on our projects, we first need to get your Arduino talking to your computer. We need to do this so you can compile and send code for your Arduino to execute.

7.1 Installing the Arduino IDE on Windows

Step 1: Download the Arduino software

Go to http://arduino.cc/en/Main/Software and download the Arduino Software for your Windows.



Step 2: Install the software

Install the Drivers:

- *Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts.*
- *Click on the Start Menu, and open up the Control Panel.*
- *While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.*
- *Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)".*
- *Right click on the "Arduino UNO (COMxx)" port and choose the "Update Driver Software" option.*
- *Next, choose the "Browse my computer for Driver software" option.*
- *Finally, navigate to and select the Uno's driver file, named "ArduinoUNO.inf", located in the "Drivers" folder of the Arduino Software download.*
- *Windows will finish up the driver installation from there.*

7.2 Installing the Arduino IDE on Mac OS X

Step 1: Download the Arduino software

Go to http://arduino.cc/en/Main/Software and download the Arduino Software for your Mac OS X.

Step 2: Install the software

The disk image (.dmg) should mount automatically. If it doesn't, double-click it. It should look like the following image.

Copy the Arduino application into the Applications folder (or elsewhere on your computer). Since you're using an Arduino Uno, you don't have any drivers to install.

7.3 Installing the Arduino IDE on Ubuntu/ Linux

Install gcc-avr and avr-libc from the Terminal.

```
sudo apt-get install gcc-avr avr-libc
```

If you don't have openjdk-6-jre already, install and configure that too:

```
sudo apt-get install openjdk-6-jre sudo update-alternatives --config java
```

Select the correct JRE if you have more than one installed.

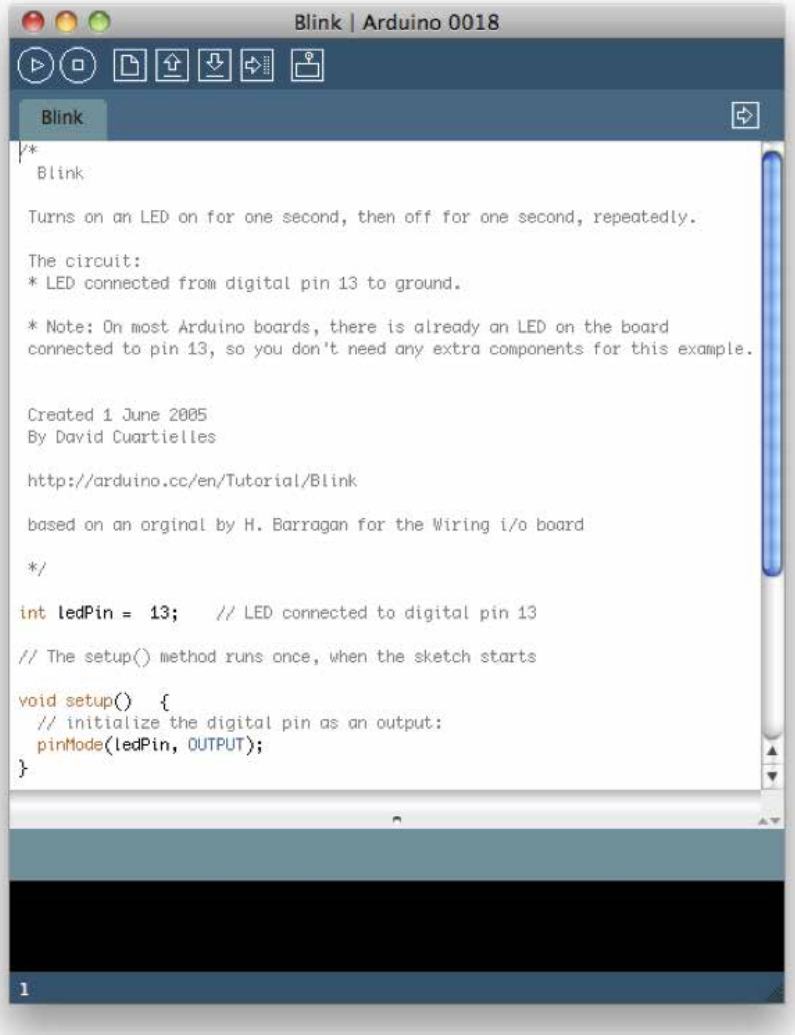
Go to http://arduino.cc/en/Main/Software/ and download the Arduino Software for Linux. You can untar and run it with the following command:

```
tar xzvf arduino-x.x.x-linux64.tgz cd arduino-1.0.1 ./arduino
```

7.4 Running the Arduino Software

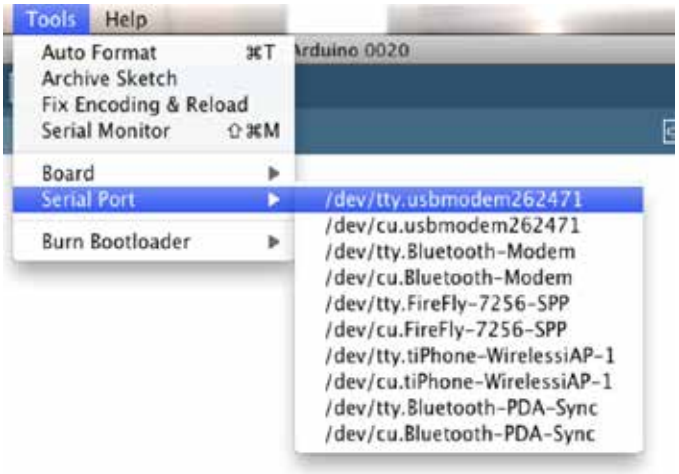
Now that our software is installed and our Arduino is setup, let's verify everything is working. The easiest way to do this is by using the "Blink" sample application.

1. *Open the Arduino Software by Double-clicking the Arduino Application (./arduino on Linux).*
2. *Make sure the board is still connected to your computer.*
3. *Open the LED blink example sketch: File > Examples > 1.Basics > Blink. You should see the code for the application open and it should look like this:*



4. *You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino. Select the Arduino Uno Option.*

5. *Select the serial device of the Arduino board from the Tools > Serial Port menu. On Windows, This is likely to be COM3 or higher. On the Mac or on Linux, this should be something with '/dev/tty.usbmodem ' in it.*



6. *Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX LEDs on the Arduino flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.*

A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink. If it does, congratulations! You've got your Arduino up and running.

8. Starter Projects

Okay, now is when the real fun begins. Let's get started.

8.1 Communicating Between Your Arduino and Your PC

Most of the communication you'll be doing with the Arduino (for now) will be done via the Serial port (The USB cord). This is quite trivial to set up on the Arduino. Merely add the following line to your setup() method:

```
Serial.begin(9600);
```

9600 is the baud rate, something we will not get into here (it essentially means the number of signal changes made per second, and merely ensures that the PC and the Arduino are on the same page in regards to this). Whenever you would like to write something to the serial port, simply use the Serial.print or Serial.println function, as so:

```
Serial.print("Hello world!");
```

Reading from the Serial Port

Note that you will have to read in a single character at a time via the serial port, which is rather unfortunate. If you take a peek at the sample code for our calculator application, specifically the waitForNum() method, you will see an example of how to read in all characters entered, albeit in this case for a number.

8.2 Building a Calculator

To tie all of your new found programming knowledge together, we submit to you the following program that performs basic mathematical operations. We have clearly commented the code, so you should be able to understand each step. There is a download available for people who don't like typing at: http://www.bradkendall.ca/arduino

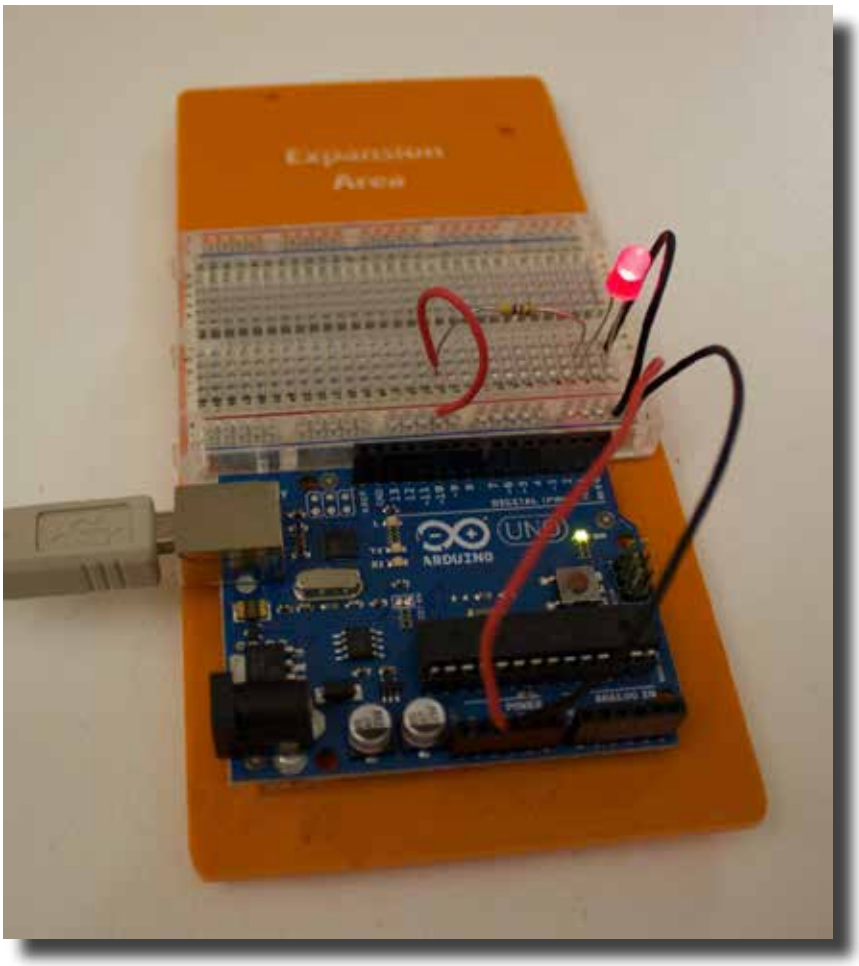
Here we go!

```
/*
Example Arduino Calculator
Communication protocol: Send an 'A', 'S', 'M', or 'D' via serial, than two numbers. The arduino
will reply with the result of the operation on the two numbers, (first number first). Note that
the division will no doubt look strange - it is an integer division and therefore there will
not be anything after the decimal point.
*/
void setup() {
Serial.begin(9600);
Serial.println("Calculator initiated.");
}
/* loop()
This code gets executed over, and over, and over, and over, and over, and over, and over, and
over, and over, and over, and over, and over, and over, and over, and over, and over again.
Our loop pretty much starts the 'waiting for input' stage, where we wait for the user to input
a character (the mathematical operation), then two operands.
After we output the result, we let the loop get hit again, and joy is had by all!
*/
void loop() {
char operation;
int number1;
int number2; // hehe, Number 2.
int result; // Hold the result of the operation.
boolean success;
// Indicates whether the operation
// was successful (we knew what to
// do - nothing bad was inputted)
success = true;
// Go ahead and set success to true ;
// The only time we will be updating
// this variable now is to set it to
// false if we've encountered a
// problem.
```

```
Serial.println("Pick an operation: 'A'dd, 'S'ubtract, 'M'ultiply, or 'D'ivide (Simply input the
first letter in quotes.)");
// We have to wait for the user to send something
// here; the easiest way to do so is to simply loop
// and waitfor Serial.available() to be true.
while(Serial.available() == 0) {
; // ; indicates an empty statement. Or a sea
// monster in Nethack. God those suck.
}
// This loop will continue executing while Serial.
// available() == 0. Thus, it will be stuck here until
// the serial has a character waiting.
operation = Serial.read();
// We have to do the same thing to get the two
// operands (numbers).
// I have factored this code into a function so that
// I do not have to rewrite it twice. See if you can
// determine why I would not be able to use it (at
// least intuitively) to get the operation.
Serial.println("Okay, now please enter the two numbers, one at a time!");
number1 = waitForNum();
Serial.print("Read: ");
Serial.println(number1);
number2 = waitForNum();
Serial.print("Read: ");
Serial.println(number2);
// Now we have read in all the data we need. It is
// time to calculate the result. We will have to
// determine what operation the user specified, and
// perform the calculation from there.
Serial.print("Operation: ");
if(operation == 'A')
{
// This checks to see if the user sent along the
// character 'A', specifying an add.
Serial.println("ADD (Look, a kitty!)");
result = number1 + number2;
} else if(operation == 'S')
{
// Note that the above condition will only be
// tested for if operation is not equal to 'A' -
// hence the else.
// This code executes if the operation is 'S' for
// subtract.
Serial.println("Subtract");
result = number1 - number2;
} else if(operation == 'M')
{
// In this case, we will be multiplying.
Serial.println("Multiply");
result = number1 * number2;
} else if(operation == 'D')
{
// Here we will be dividing.
Serial.println("Divide");
result = number1 / number2;
} else{
// This code will be used if the character
// specified doesn't match anything - in other
// words, the user did not send A, S, D, or M,
// and we don't know what to do.
// Hence, set 'success' to false
success = false;
```

```
}
// Now we should have our result. Time to send the
// user back something! (Then start over again! Joy!)
if(success)
{
// Note that print will not start a new
// line, and the next print statement will
// continue writing right
// where the previous one left off.
// Output the result.
Serial.print("Result: ");
Serial.println(result);
} else
{
Serial.print("Sorry, I don't understand what you want me to do! (You inputted '");
Serial.print(operation);
Serial.println("')");
}
}
int waitForNum()
{
int ret;
while(Serial.available() == 0) { ;
}
// Why minus '0'? The value we'll get from Serial.
// read() will be a character. What this means is
// that its numeric value will not necessarily
// reflect the number it represents. (Look at an
// ASCII table, the character '0' actually has a
// decimal value of 48!)
// The take-away from this is that, since fortunately
// all the numbers are in sequence, you can simply
// subtract the decimal value of '0' from
// whatever you read in, and you'll be left with the
// number itself. '5' - '0' = 5 .
ret = Serial.read() - '0';
// To handle numbers that span more than one
// character (like 124, which spans three), we must
// loop until there is no more input, and multiply
// each number we read by one (as 124 would come in
// like: 4 2 1
// And the number we build would be:
// (((1 * 10) + 2) * 10) + 4,
// or 124! The joys of the decimal numbering system!
// Note that the delays are merely to slow things
// down a bit - removing them would have the
// code execute too quickly to 'notice' more
// characters waiting to come in from Serial.
// A little strange, neh? Welcome to the joys of this
// type of thing. =]
delay(10);
while(Serial.available() != 0)
{
ret = ret * 10;
ret += Serial.read() - '0';
delay(10);
}
return ret;
}
```

8.3 Turning on an LED



What You Need:

- *1 – LED*
- *1 – Resistor – 1 KOhm (brown, black, red)*
- *4 – Jumper Wires*

You will build a circuit by plugging the LED and resistor leads into small holes called sockets on the breadboard.

Let's get started!

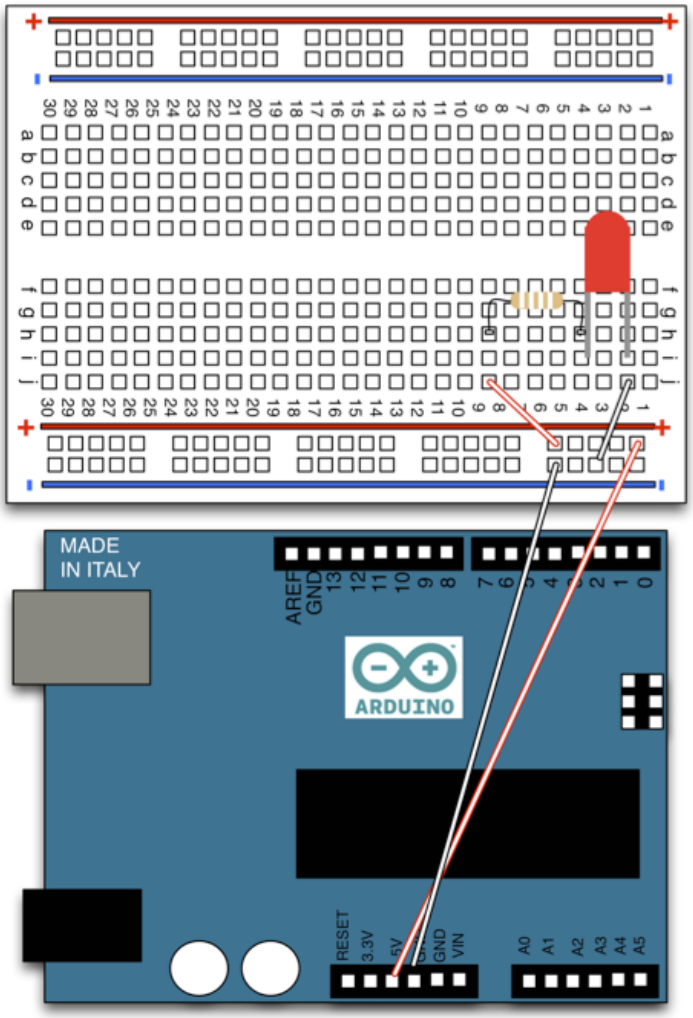
Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug a jumper wire from the 5V port on your Arduino into the positive section of your breadboard's power lane.*
- *Step 3 - Plug a jumper wire from the GND port on your Arduino into the negative section of your breadboard's ground lane.*
- *Step 4 - Plug the LED's cathode (the short lead) into the I-2 socket on your breadboard.*
- *Step 5 - Plug the LED's anode (the long lead) into the I-4 socket on your breadboard.*
- *Step 6 - Plug one of the resistor's leads into the H-4 socket on your breadboard.*
- *Step 7 - Plug the resistors other lead into the H-9 socket on your breadboard.*
- *Step 8 - Connect a jumper wire from your breadboard's power lane to the J-9 socket on your breadboard.*
- *Step 9 - Connect a jumper wire from your breadboard's ground lane to the J-2 socket on your*

breadboard.

- *Step 10 - Reconnect the USB cable to your Arduino.*

Summary: Once power is applied to the circuit, the LED will turn on. This is about as simple as a circuit gets.



8.4 Making Your LED Blink

What You Need:

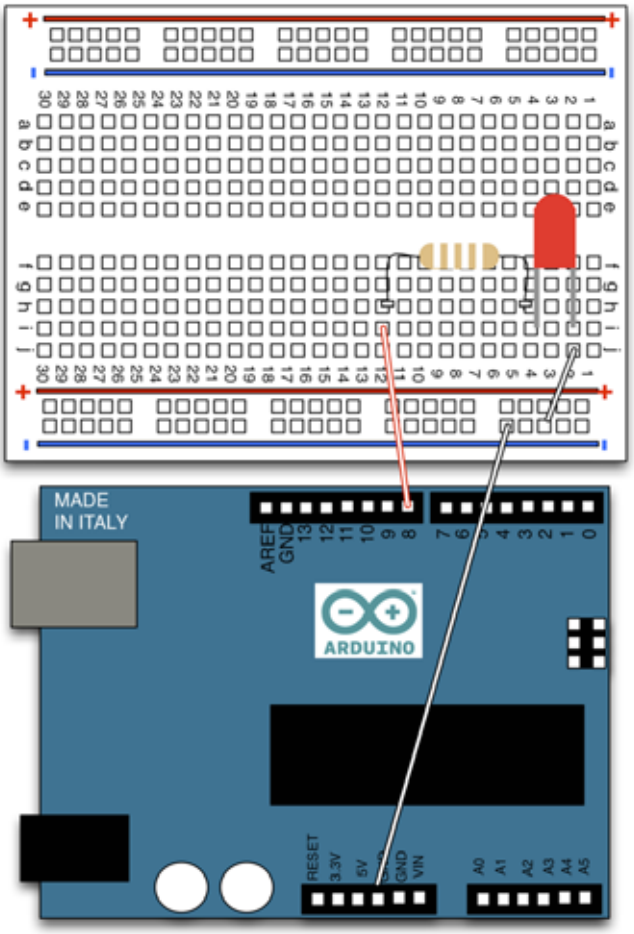
- *1 – LED*
- *1 – Resistor – 1 K Ohm (brown, black, red)*
- *4 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug a jumper wire from the Digital IO pin 8 into the I-12 socket on your breadboard.*
- *Step 3 - Plug one of the resistor's leads into the H-12 socket on your breadboard.*
- *Step 4 - Plug the resistor's other lead into the H-4 socket on your breadboard.*
- *Step 5 - Plug the LED's cathode (the short lead) into the I-2 socket on your breadboard.*
- *Step 6 - Plug the LED's anode (the long lead) into the I-4 socket on your breadboard.*
- *Step 7 - Connect a jumper wire from your breadboard's ground lane to the J-2 socket on your*

breadboard. Ensure that the ground lane is still grounded.

- *Step 8 - Reconnect the USB cable to your Arduino.*



Software Setup:

Open up your Arduino Development Environment and create a new sketch (*File > New*).

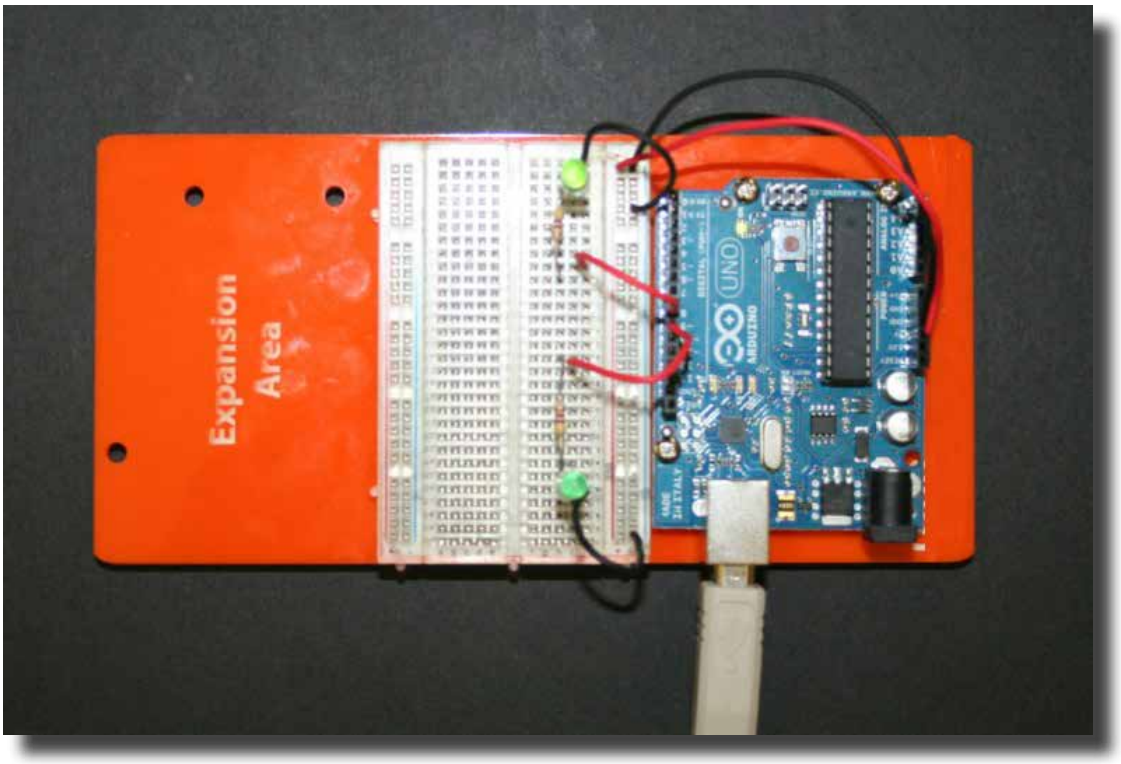
Enter the following code into your sketch:

```
void setup() {
// initialize the digital pin as an output.
// Pin 8 is our output pin
pinMode(8, OUTPUT);
}
void loop() {
digitalWrite(8, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(8, LOW); // set the LED off
delay(1000); // wait for a second
}
```

After you enter the code, press the upload button and your LED should start blinking.

Summary: The digitalWrite(8, HIGH); command sets the output pin 8 on the Arduino to 5V. The digitalWrite(8, LOW); command sets the output pin 8 on the Arduino to 0V. The delay(1000); command pauses execution on the Arduino for 1000 ms or 1 second. Since this in the loop() function, the code is called over and over again. Pretty cool, huh?

8.5 Making Multiple LEDs Blink

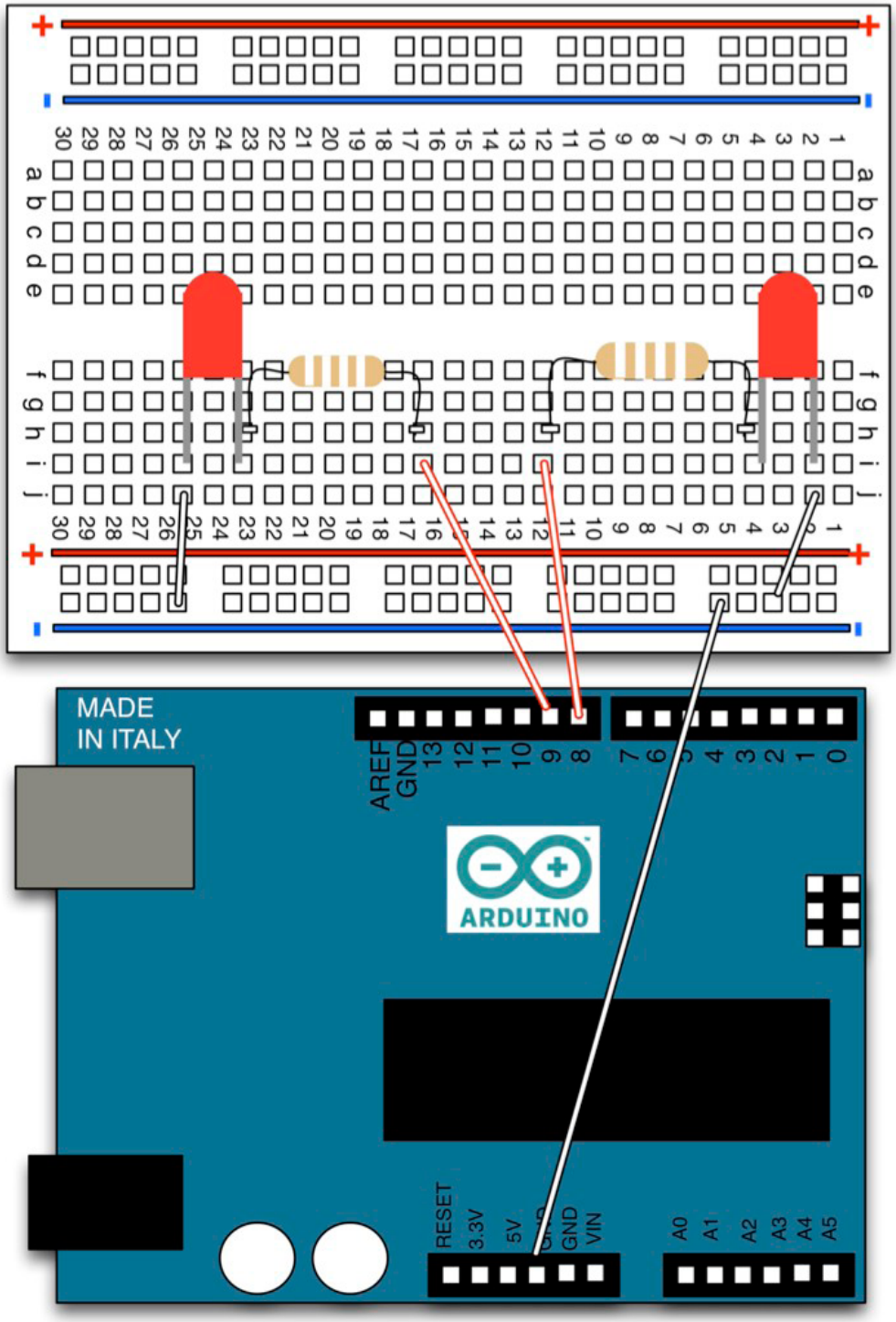


What You Need:

- *2 – LEDs*
- *2 – Resistor – 1 K Ohm (brown, black, red)*
- *4 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Setup the project board the same as in Project 3.*
- *Step 3 - Plug a jumper wire from the Digital IO pin 9 into the I-16 socket on your breadboard.*
- *Step 4 - Plug one of the resistor's leads into the H-16 socket on your breadboard.*
- *Step 5 - Plug the resistor's other lead into the H-24 socket on your breadboard.*
- *Step 6 - Plug the LED's anode (the long lead) into the I-24 socket on your breadboard.*
- *Step 7 - Plug the LED's cathode (the short lead) into the I-26 socket on your breadboard.*
- *Step 8 - Connect a jumper wire from your breadboard's ground lane to the J-26 socket on your breadboard. Ensure that the ground lane is still grounded.*
- *Step 9 - Reconnect the USB cable to your Arduino.*



Software Setup:

Open up your Arduino Development Environment and create a new sketch (*File > New*).

Enter the following code into your sketch:

```
void setup() {
// initialize the digital pins as an output.
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
}
void loop() {
digitalWrite(8, HIGH); // set the LED on
digitalWrite(9, LOW); // set the LED on
```

```
delay(1000); // wait for a second
digitalWrite(8, LOW); // set the LED off
digitalWrite(9, HIGH); // set the LED on
delay(1000); // wait for a second
}
```

After you enter the code, press the upload button and both your LEDs should start blinking.

Summary: This project is exactly the same as the last project, except we have added an additional LED on output pin 9 that turns off when the other LED is on. Can you think of any other ways to expand on this?

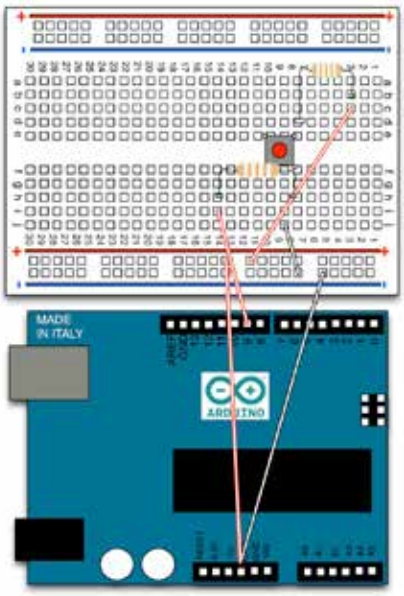
8.6 Pushbuttons with a Pull-up Resistor

What You Need:

- *1 – Resistor 2 K Ohm (red-black-red)*
- *1 – Resistor – 1 K Ohm (brown, black, red)*
- *1 – Tactile Switch*
- *5 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Connect a jumper wire from your breadboard's power lane to the C-3 socket on your breadboard. Ensure that the power lane is still connected.*
- *Step 3 - Plug one of the 2 K Ω resistor's leads into the B-3 socket on your breadboard.*
- *Step 4 - Plug the 2 K Ω resistor's other lead into the B-7 socket on your breadboard.*
- *Step 5 - Plug a tactile switch so the pins are in the F-9, F-7, E-9 and E-7 on your breadboard.*
- *Step 6 - Plug one of the 1K Ω resistor's leads into the H-7 socket on your breadboard.*
- *Step 7 - Plug the 1K Ω resistors' other lead into the H-14 socket on your breadboard.*
- *Step 8 - Plug a jumper wire from the Digital IO pin 9 into the I-14 socket on your breadboard.*
- *Step 9 - Connect a jumper wire from your breadboard's ground lane to the H-9 socket on your breadboard. Ensure that the ground lane is still connected.*
- *Step 10 - Reconnect the USB cable to your Arduino.*



Software Setup:

Open up your Arduino Development Environment and create a new sketch (*File > New*).

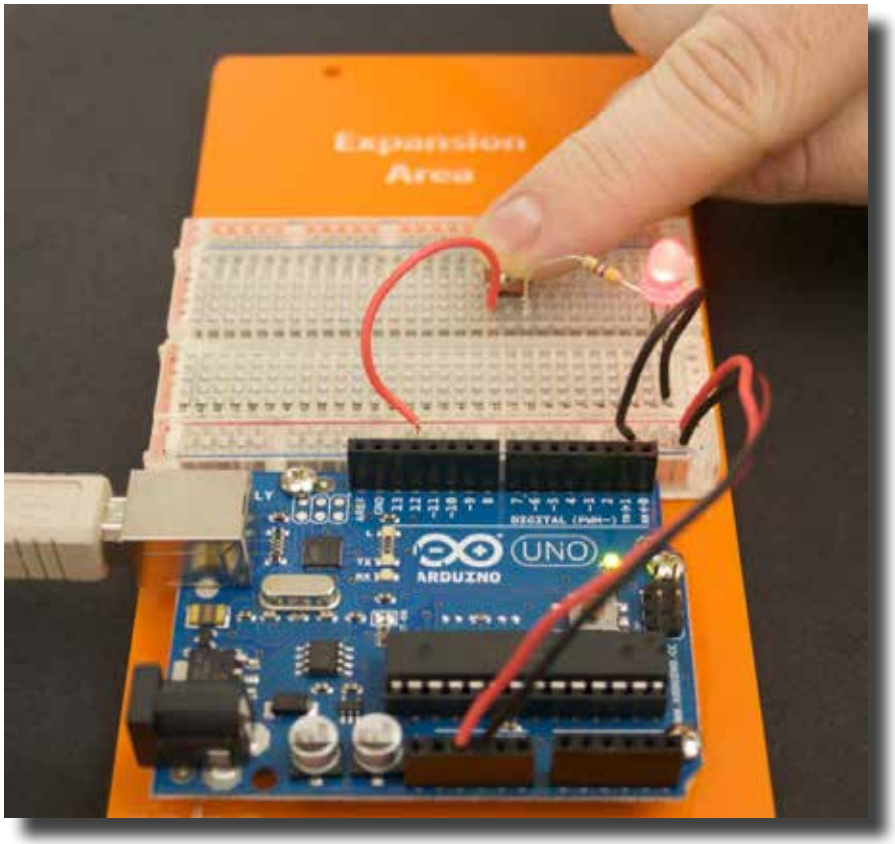
Enter the following code into your sketch:

```
void setup() {
// initialize the digital pin 9 as an input.
pinMode(9, INPUT);
// initialize the serial port.
Serial.begin(9600);
}
void loop() {
int buttonStatus = digitalRead(9);
if (buttonStatus == LOW) //The button is down
{
Serial.println("The button is down");
}
}
```

After you enter the code, press the upload button and open the Serial Monitor (*Tools > Serial Monitor*). When you press the tactile switch, the serial monitor should print "The button is down".

Summary: This project reads the digital input for 5v (HIGH). When the button is pressed, the voltage is set to 0v (LOW) and the Arduino executes the code in our if statement.

8.7 Turning on an LED with a Pushbutton

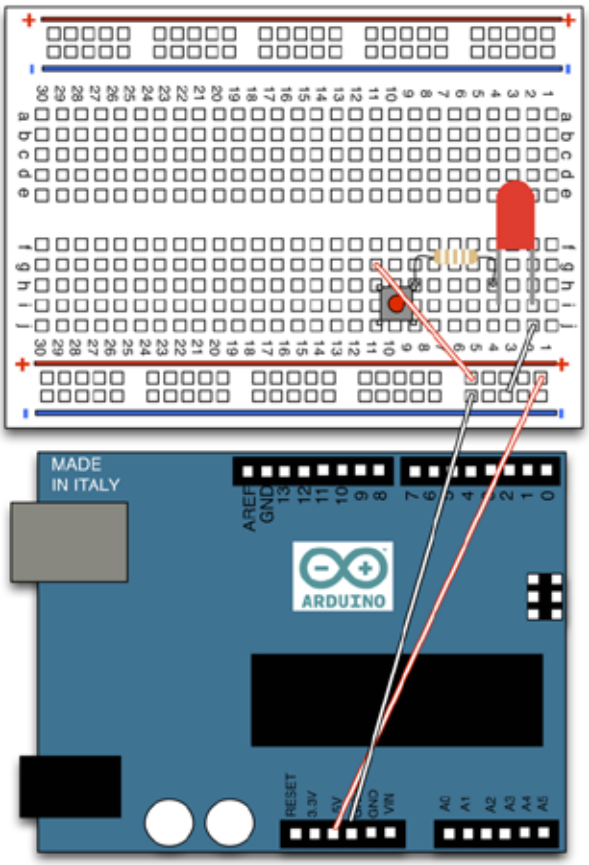


What You Need:

- *1 - LED*
- *1 – Resistor – 1 K Ohm (brown, black, red)*
- *1 – Tactile Switch*
- *4 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug a tactile switch so the pins are in H-9, H-11, J-9 and J-11 on your breadboard.*
- *Step 3 - Plug a jumper wire from the GND port on your Arduino into the negative section on your breadboard's ground lane.*
- *Step 4 - Plug the LED's cathode (the short lead) into the I-2 socket on your breadboard.*
- *Step 5 - Plug the LED's anode (the long lead) into the I-4 socket on your breadboard.*
- *Step 6 - Plug one of the resistor's leads into the H-4 socket on your breadboard.*
- *Step 7 - Plug the resistor's other lead into the H-9 socket on your breadboard.*
- *Step 8 - Connect a jumper wire from your breadboard's power lane to the G-11 socket on your breadboard.*
- *Step 9 - Connect a jumper wire from your breadboard's ground lane to the J-2 socket on your breadboard.*
- *Step 10 - Reconnect the USB cable to your Arduino.*



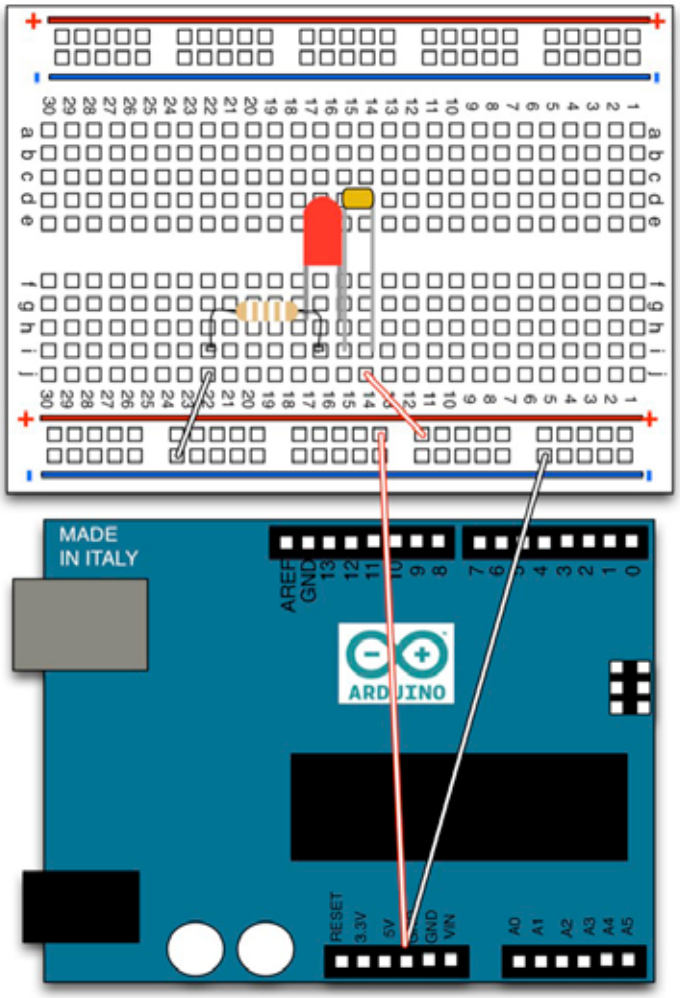
8.8 Control an LED's Brightness

What You Need:

- *1 – Photo Resistor*
- *1 – Resistor – 2 K Ohm (red-black-red)*
- *1 – LED*
- *5 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug one of the photo resistor's leads into the I-14 socket on your breadboard.*
- *Step 3 - Plug the photo resistor's other lead into the I-15 socket on your breadboard.*
- *Step 4 - Plug the LED's cathode (the short lead) into the H-17 socket on your breadboard.*
- *Step 5 - Plug the LED's anode (the long lead) into the H-15 socket on your breadboard.*
- *Step 6 - Plug one of the resistor's lead into the I-17 socket on your breadboard.*
- *Step 7 - Plug the resistor's other lead into the I-22 socket on your breadboard.*
- *Step 8 - Connect a jumper wire from your breadboard's power lane to the J-14 socket on your breadboard. Ensure that the power lane is still connected.*
- *Step 9 - Connect a jumper wire from your breadboard's ground lane to the J-22 socket on your breadboard. Ensure that the ground lane is still connected.*
- *Step 10 - Reconnect the USB cable to your Arduino.*



Summary: As you see, the resistance of the photo resistor decreases with more light. The lower the resistance, the brighter the LED. Combine this with the pull up resistor project (Chapter 8.6) and watch the opposite effect.

8.9 Observing Light with your Arduino

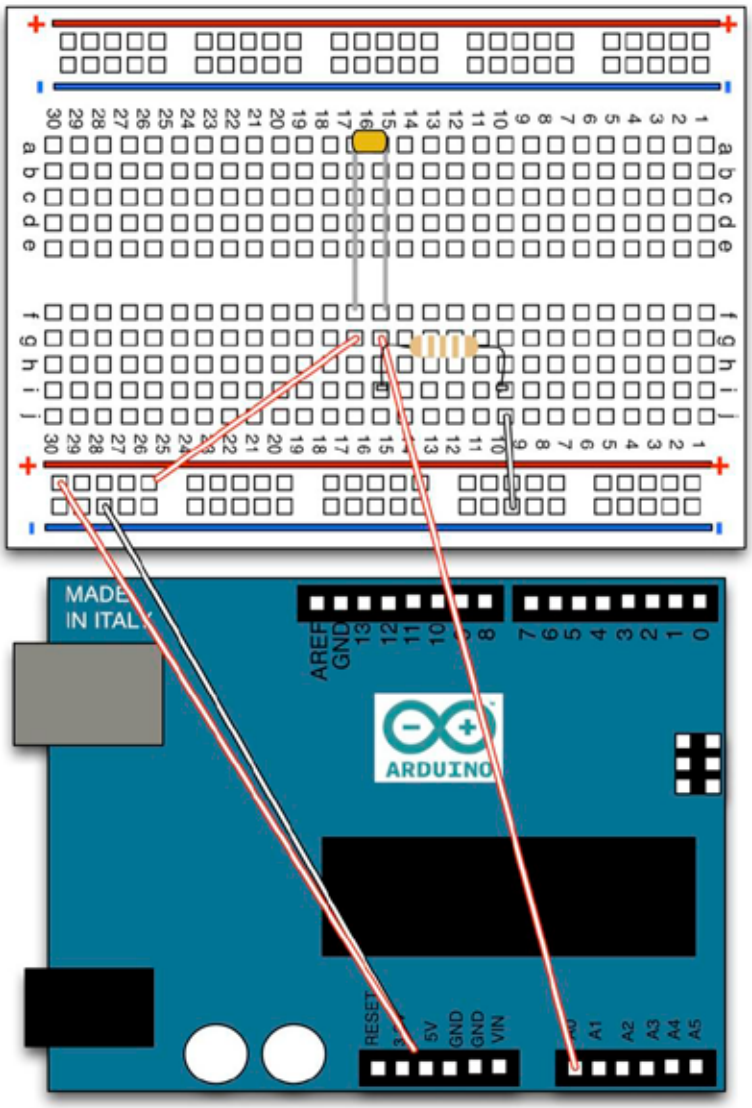
What You Need:

- *1 – Photo Resistor*

- *1 – Resistor – 10 K Ohm (brown-black-orange)*
- *5 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug one of the photo resistor's leads into the F-16 socket on your breadboard.*
- *Step 3 - Plug the photo resistor's other lead into the F-15 socket on your breadboard.*
- *Step 4 - Plug one of the resistor's leads into the I-15 socket on your breadboard.*
- *Step 5 - Plug the resistor's other lead into the I-10 socket on your breadboard.*
- *Step 6 - Connect a jumper wire from your breadboard's power lane to the G-16 socket on your breadboard. Ensure that the power lane is still connected.*
- *Step 7 - Connect a jumper wire from your breadboard's ground lane to the J-10 socket on your breadboard. Ensure that the ground lane is still connected.*
- *Step 8 - Plug a jumper wire from the Analog IO pin 0 into the G-15 socket on your breadboard*
- *Step 9 - Reconnect the USB cable to your Arduino.*



Software Setup:

Open up your Arduino Development Environment and create a new sketch (*File > New*).

Enter the following code into your sketch:

```
int lightPin = 0; //define a pin for Photo resistor
void setup()
{
Serial.begin(9600); //Begin serial communication
}
void loop()
{
//Write the value of the photo resistor to the serial //monitor.
int lightValue = analogRead(lightPin);
Serial.println(lightValue);
delay(1000); //pause for 1000 ms or 1 second.
}
```

After you enter the code, press the upload button and open the Serial Monitor (*Tools > Serial Monitor*). The console should give a light reading in the form of an integer. When you reduce the amount of light, the number will be lower.

Summary: This project is the same as the previous project, except we are reading the values from your Arduino instead of outputting to an LED. The resistance of the photo resistor decreases with more light. You could use logic to reverse this effect!

8.10 Making Music with your Arduino

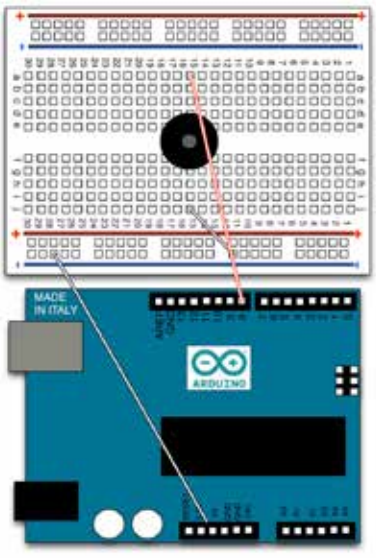
What You Need:

- *1 – Piezo Speaker*
- *3 – Jumper Wires*

Hardware Setup:

- *Step 1 - Unplug the USB cord from your Arduino.*
- *Step 2 - Plug the positive lead of your piezo speaker into the E-15 socket on your breadboard.*
- *Step 3 - Plug the negative lead of your piezo speaker into the F-15 socket on your breadboard.*
- *Step 4 - Plug a jumper wire from the Digital IO pin 8 into the A-15 socket on your breadboard.*
- *Step 5 - Connect a jumper wire from your breadboard's ground lane to the J-15 socket on your breadboard. Ensure that the ground lane is still connected.*
- *Step 6 - Reconnect the USB cable to your Arduino.*

Easy, right?



Software Setup:

This project is included in the Examples Section. No typing on this one! Open up your Arduino Development Environment .Open the toneMelody Example Sketch (*File > Examples > Digital > toneMelody*).

After you enter the code, press the upload button and your piezo speaker will start making noise. You can modify the sound by modifying the melody[] and noteDurations[] arrays.

Summary: This project produces sound out of the piezo speaker.

9. Where to go From Here

As you can see, the Arduino is an easy way to get into electronics and software. Hopefully you have seen that it is easy to build simple electronic projects with it. I hope you have realized that your projects don't have to stay simple. You can build way more complex projects on top of these simple ones. Here are some of my favorite projects that would be a great next step:

- *Create Christmas light ornaments*
- *Arduino Traffic light controller*
- *Arduino Shields to superpower your project*
- *Make your own Arduino*
- *Build your own pong game with an Arduino*
- *Connect your Arduino to the internet*
- *Create a home automation system with your Arduino*

Did you like this PDF Guide? Then why not visit MakeUseOf.com for daily posts on cool websites, free software and internet tips?

If you want more great guides like this, why not subscribe to MakeUseOf and receive instant access to 50+ PDF Guides like this one covering wide range of topics. Moreover, you will be able to download free Cheat Sheets, Free Giveaways and other cool things.

Home:	**http://www.makeuseof.com**
MakeUseOf Answers:	**http://www.makeuseof.com/answers**
PDF Guides:	**http://www.makeuseof.com/pages/**
Tech Deals:	**http://www.makeuseof.com/pages/hot-tech-deals**

Follow MakeUseOf:

RSS Feed:	**http://feedproxy.google.com/Makeuseof**
Newsletter:	**http://www.makeuseof.com/pages/subscribe-to-makeuseof-newsletter**
Facebook:	**http://www.facebook.com/makeuseof**
Twitter:	**http://www.twitter.com/Makeuseof**

Think you've got what it takes to write a manual for MakeUseOf.com? We're always willing to hear a pitch! Send your ideas to justinpot@makeuseof.com.

Download Other MakeUseOf PDF Guides!

http://makeuseof.com/pages