Mioduino Guide

Breadboard Your Own Arduino Compatible

by Daniel M. Porrey

Version 1.0.4

Summary

This guide is intended as a basic introduction to the Arduino platform through the building of an Arduino compatible on a breadboard. The structure consists of two classroom sessions and various labs that will result in the student having a solid foundation for further exploration of the Arduino platform.



In this guide, the student will learn to build their own Arduino compatible on a breadboard and use her board to complete a series of labs. The required components are outlined and clear instructions and photos are provided to guide the student through the lab process.

This guide also provides background information on a few circuit components and basic circuit concepts to help understand the labs; however, this guide is not intended to be a complete electronics tutorial. See the reference at the end of this document for recommended reading.



Edited by Hannah Porrey

Table of Contents

Contents

Table of Contents 3
Introduction 8
Tool Required 8
Install the Arduino IDE 8
Install the Mioduino Board 9
Install the Libraries 12
Mioduino Library 12
MCP9808 Library 13
Check Your Components 15
Components List 15
Ordering Components 21
Full Kit for 15 Students 22
Full Kit for 10 Students 22
Single Kit 22
Classroom 1 23
Using a Breadboard 23
Using Jumper Wires 24
Understanding the Components 25
Overview 25
The Microcontroller 25
Power Supply 25
Clock 26
Reset 26
The LEDs 27

Programming Interface 27
Powering the Board 27
Lab 1: Build Your Mioduino 29
Objective 29
Steps to Follow 29
Assemble the Voltage Supply 29
Test the Voltage Supply 30
Assemble the Mioduino 31
Completed Board 32
Lab 2: Mioduino Blinky 36
Objective 36
Run the Sketch 36
Classroom 2 37
The Microcontroller 37
Electronic Components 37
The Resistor 37
The Capacitor 38
The Diode 38
The LED 40
Voltage Regulator 41
Photoresistor 41
Switch 42
Electronic Circuit Concepts 42
Passive and Active Components 42
Ohms Law 43
Series and parallel Circuits 43
Voltage Divider 44
Kirchhoff's Voltage Law 44

Pulse-Width Modulation 45
Analog and Digital Circuits 45
Arduino Basics 47
Arduino IDE 47
Arduino Sketch 48
Loading the Lab Sketches 48
Selecting the Board and Programmer 49
Lab 3: My Blinky 51
Objective 51
Prepare for the Lab 52
Schematic Diagram 52
Breadboard Diagram 52
Build the Circuit 53
Run the Sketch 53
Lab 4: LED Brightness 55
Objective 55
Prepare for the Lab 55
Schematic Diagram 55
Breadboard Diagram 56
Build the Circuit 56
Run the Sketch 57
Lab 5: Push Button Monitor 58
Objective 58
Prepare for the Lab 59
Schematic Diagram 59
Breadboard Diagram 59
Build the Circuit 60
Run the Sketch 61

Lab 6: Night Light 62
Objective 62
Prepare for the Lab 62
Schematic Diagram 62
Breadboard Diagram 62
Build the Circuit 63
Run the Sketch – Part A 64
Run the Sketch – Part B 65
Lab 7: Ohm's Law 66
Objective 66
Prepare for the Lab 66
Schematic Diagram 66
Breadboard Diagram 66
Build the Circuit 67
Run the Sketch 68
Lab 8: Analog Sensor 69
Objective 69
Prepare for the Lab 69
Schematic Diagram 70
Breadboard Diagram 71
Build the Circuit 71
Run the Sketch 72
Lab 9: Digital Sensor 1 73
Objective 73
Prepare for the Lab 73
Schematic Diagram 73
Breadboard Diagram 74
Build the Circuit 74

Run the Sketch 75
Lab 10: Digital Sensor 2 77
Objective 77
Prepare for the Lab 77
Schematic Diagram 77
Breadboard Diagram 78
Build the Circuit 78
Run the Sketch 79
Additional Resources 81
My Notes 84
My Notes 85
My Notes 86
My Notes 87
My Notes 88
My Notes 89
My Notes 90
My Notes 91
My Notes 92
My Notes 93
My Notes 94
My Notes 95

Introduction

In this guide, you will learn to make an Arduino compatible on a breadboard called the Mioduino. After building the board, you will be guided through the connection and use of various sensors.

Tool Required

There are a few tools required to complete the labs in this guide that are not supplied in the kit.

Needle-Nose Pliers	These pliers are needed to break-away the header pins used to connect the FTDI cable to the breadboard.	*REQUIRED*
Multimeter	A multimeter is helpful to measure the output of the power supply. It is also useful for making measurements during the lab.	*OPTIONAL (HIGHLY RECOMMENDED)*
Wire Cutters	These are required if you plan to make your own jumper wires from a roll of 22 AWG wire.	*OPTIONAL*

Install the Arduino IDE

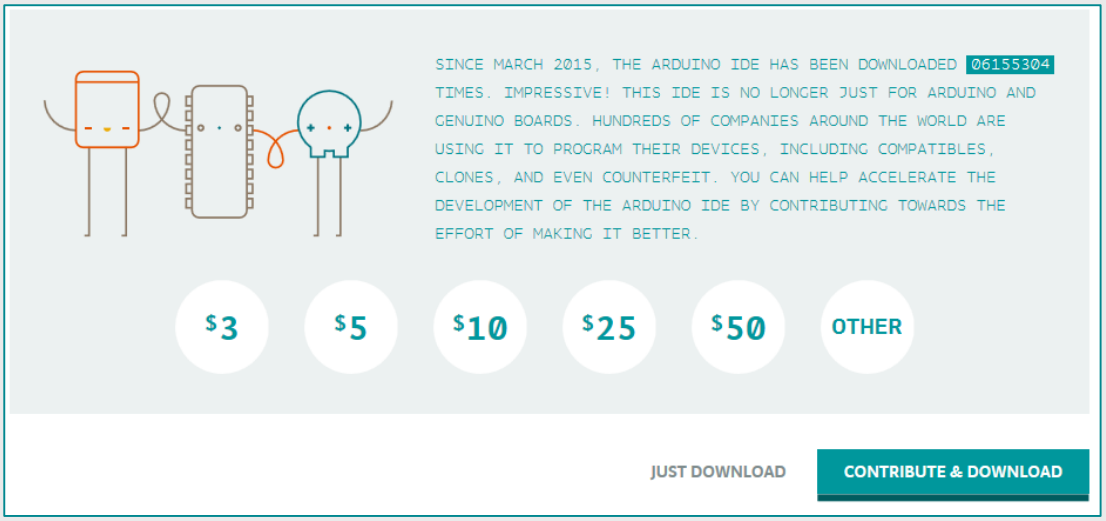
Before getting started you will need the Arduino IDE installed on your computer, along with some libraries. It's best to get this step completed before starting the labs.

1. Go to *HTTPS://WWW.ARDUINO.CC/* and choose **Download** from the menu.
2. At the time of this publication the latest version is **1.6.8**. Download the latest version appropriate for your operating system.



3. Click the **WINDOWS INSTALLER** link near the top of the section.

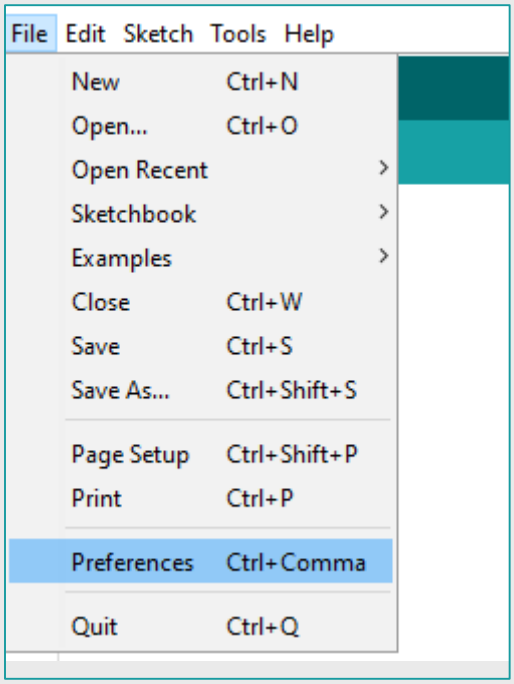
4. On the next page you will be asked to contribute to the Arduino Software development. For this exercise click the **JUST DOWNLOAD**. If you are not purchasing products directly from ARDUINO.CC, you should consider making a donation to support the development efforts.



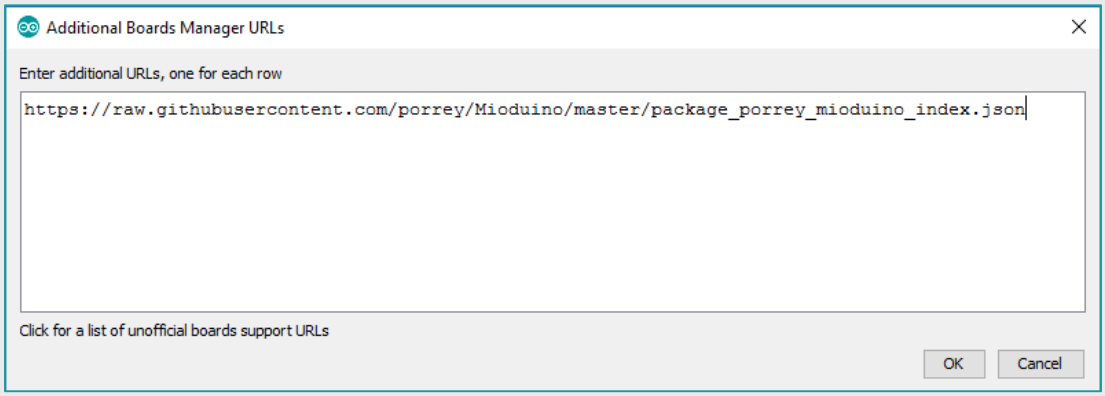
5. Now run the installer. If you already have an older version installed, you will be prompted to uninstal that version first. The uninstall will be done automatically for you.

INSTALL THE MIODUINO BOARD

Now that the IDE is installed, you need to add the *MIODUINO* to the list of boards. Select **PREFERENCES** from the **FILE** menu.



In this window click the button all the way to the right of the **ADDITIONAL BOARDS MANAGER URLS** list and enter the URL below into the window.

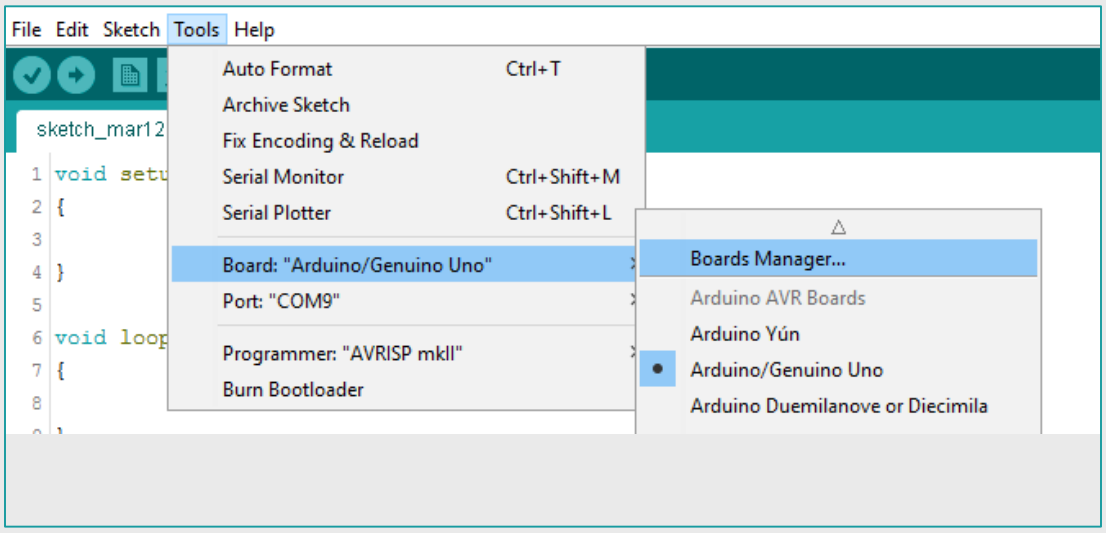


https://raw.githubusercontent.com/porrey/Mioduino/master/package_porrey_mioduino_index.json

This is a long URL! Enter the URL http://bit.ly/mioduino-ide into your browser. This link will forward you to the link above which you can copy and paste into the Arduino IDE.

Click **OK** to save the URL, then click **OK** again to save the preferences.

Next, click the **TOOLS** menu and select the **BOARD:** to open the board manager menu. Select **BOARDS MANAGER** at the top of this menu.



Scroll down the list of boards until you find the board named **MIODUINO**. Select the board and click **INSTALL**.

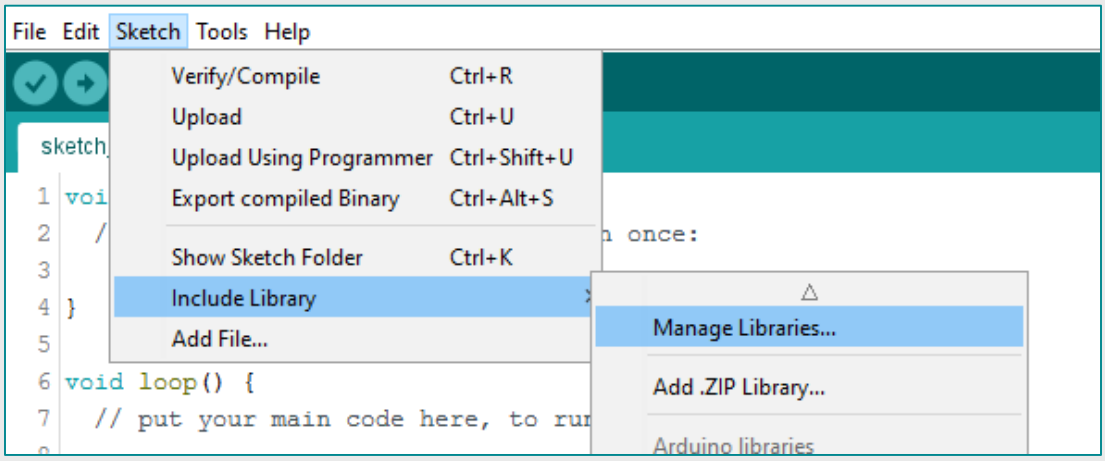


Click **CLOSE** after the installation has completed.

Install the Libraries

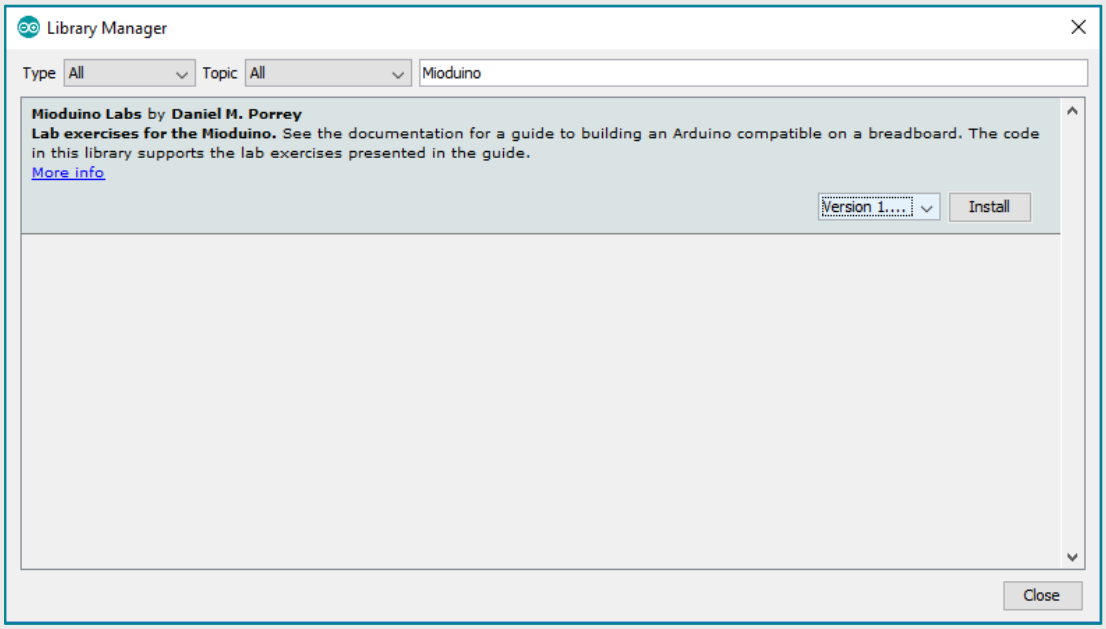
Mioduino Library

Select the **Sketch**, **Include Library** and **Manage Libraries...** to open the library manager.

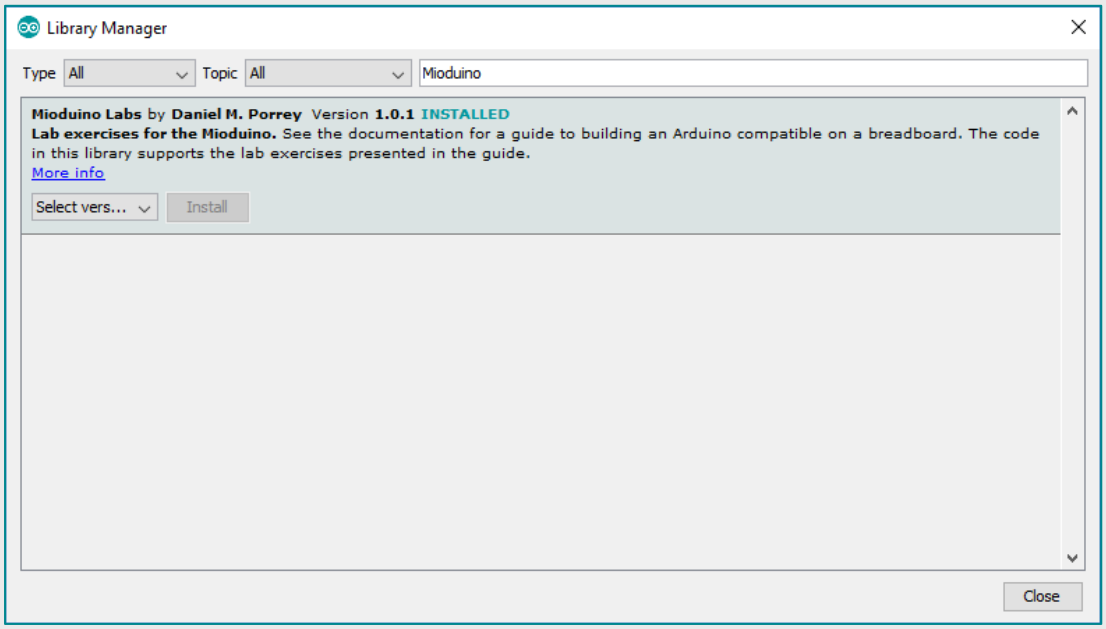


In the Library Manager window, in the search text start typing *MIODUINO* to find the support library for this guide.

Click the **Install** button to add the library. After the installation completes, click the **Close** button. Always ensure that you select the latest version.



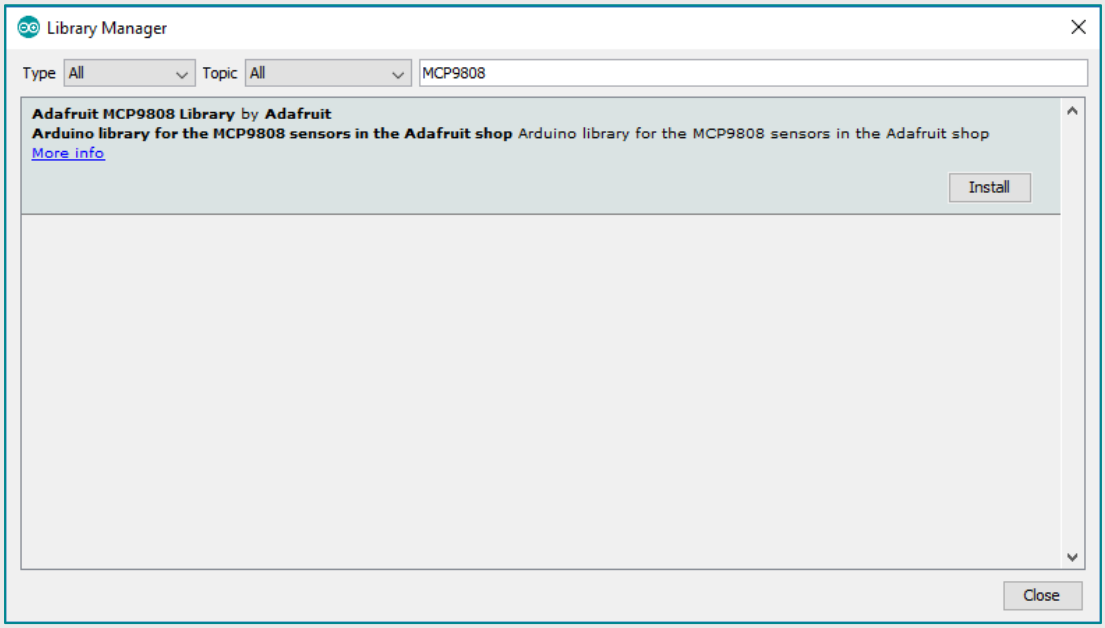
The library will now show as installed.



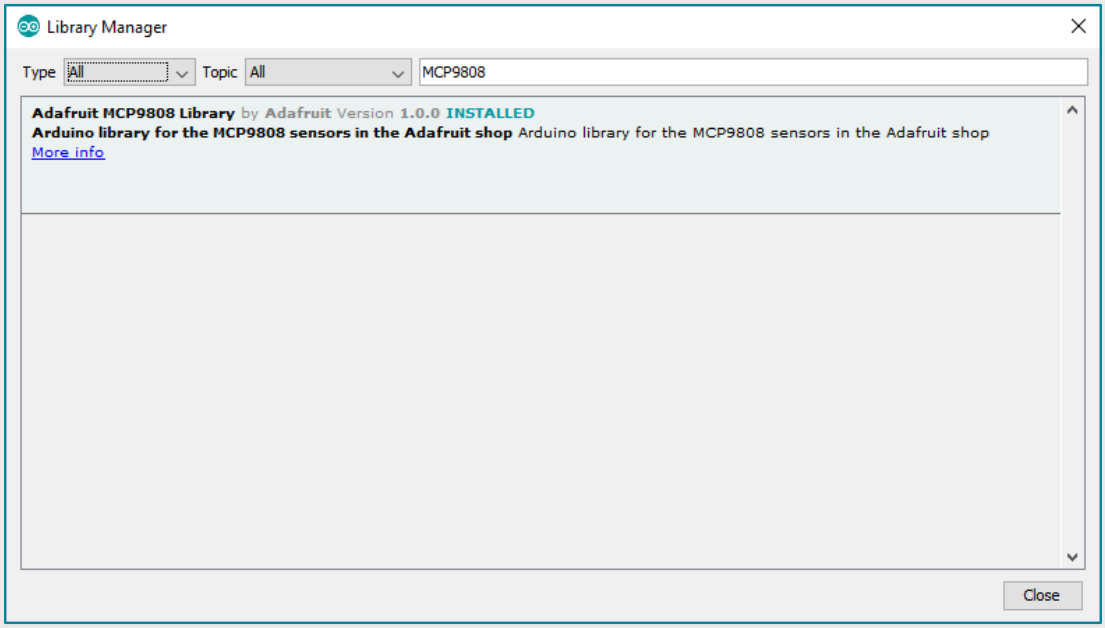
MCP9808 LIBRARY

Again, select the **SKETCH**, **INCLUDE LIBRARY** and **MANAGE LIBRARIES...** to open the library manager.

In the Library Manager window, in the search text start typing *MCP9808* to find the support library for this guide. Always ensure that you select the latest version. Click the **INSTALL** button to add the library. After the installation completes, click the **CLOSE** button.



The library will now show as installed.

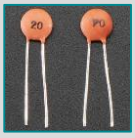


Check Your Components

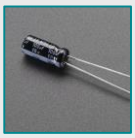
This guide can be used by anyone that wants to build the Mioduino on a breadboard. The list below will indicate if a component is required or recommended. If you are participating in a Hacker Weekend event, then these components have been supplied to you.

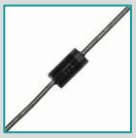
The next step is to ensure that you have everything you need to complete these labs. Use this list to ensure you have all of the necessary components before attempting to build the Mioduino or perform the labs.

Components List

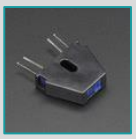
	Full sized breadboard	The full size contains 830 tie-points and two sets of power rails. This board is large enough to contain the Mioduino and still have room for additional components. http://adafru.it/239	*REQUIRED*	*x1*
	Arduino bootloader-programmed chip (Atmega328P)	This is the preprogrammed Atmega328P chip needed to make your own Arduino-compatible board. It contains a boot-loader already programmed on the board. http://adafru.it/123	*REQUIRED*	*x1*
	Adafruit AVR Sticker for Breadboard Arduino-compatibles	These stickers fit right on top of the Atmega32P chip and help identify the pins as they are referenced in the Arduino IDE. If you are receiving this as part of the Hacker Weekend this sticker is already applied to your microcontroller. http://adafru.it/554	*OPTIONAL*	*x1*
	16 MHz Crystal	This is the 16.000 MHz crystal required to provide timing for the Atmega328P microcontroller. http://adafru.it/2215	*REQUIRED*	*x1*
	20pF Capacitor	20pF capacitors are used in conjunction with the 16MHz crystal. http://adafru.it/2215	*REQUIRED*	*x2*

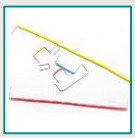
	5V Voltage Regulator	This linear voltage regulator takes an input voltage between 7 and 35V and regulates it to a constant 5V with a 1.5 A limit. This regulator has a ~2V linear drop-out. That means you must give it at least 7V to get a clean 5V out. There is a constant 'quiescent' current draw of 6mA. This is the TO-220 version. This device allows the breadboard circuit to be used with the 9V power supply while providing a clean 5-volts to the Atmega328P. http://adafru.it/2164	*REQUIRED*	*x1*
	2.1 mm DC Barrel Jack	This power jack is designed to fit 2.1mm power plugs snugly and securely into a breadboard provides the source point on the breadboard for the 9V power supply. http://adafru.it/373	*REQUIRED*	*x1*
	9V 1 A Power Supply	This is a 9V switching power supply that can provide up to 1 A of current. The input is 100-240V AC and provides the input voltage to our regulator portion of the circuit. http://adafru.it/63	*REQUIRED*	*x1*
	TO-220 Clip-on Heat Sink	This heatsink will clip onto the 7805 Voltage Regulator to provide proper heat dissipation. Note the 7805 can dissipate 2W of power without a heatsink. http://adafru.it/977	*OPTIONAL*	*x1*
	In-line power switch for 2.1 mm DC Barrel Jack	This power switch will sit in between the 2.1mm power supply and the barrel jack connector to provide an on and off switch. This component allows the power to be connected and disconnected without the need to physically remove the power adapter each time. http://adafru.it/1125	*OPTIONAL*	*x1*
	10µF 50V Electrolytic Capacitor	Electrolytic capacitors work great for low frequency filtering. http://adafru.it/2195	*REQUIRED*	*x2*

	100µF 50V Electrolytic Capacitor	Electrolytic capacitors have many other purposes. In our circuit, the we will use one of these to filter voltage spikes that can occur when the circuit is first turned on (low frequency filtering). This capacitor will protect our ATmega328P from voltage spikes that can damage the chip. http://adafru.it/2193		
	47µF 50V Electrolytic Capacitor	Electrolytic capacitors work great for low frequency filtering. In our circuit, the we will use one of these to filter voltage spikes that can occur when the circuit is first turned on (low frequency filtering). This capacitor will protect our voltage regulator from voltage spikes that can damage the chip. http://adafru.it/2194		
	0.1µF Ceramic Capacitor	Ceramic capacitors are great for high frequency filters. In our circuit, we will use these to filter out noise in the output of the voltage regulator to provide a very clean 5V signal to the microcontroller. The addition of these into our circuit will improve stability by filtering out power supply spikes and noise that can cause flaky behavior. http://adafru.it/753	*REQUIRED*	*x4*
	FTDI Serial TTL Cable	Just about all electronics use TTL serial for debugging, bootloading, programming, serial output, etc. But it's rare for a computer to have a serial port anymore. This is a USB to TTL serial cable, with a FTDI FT232RL USB/serial chip embedded in the head. It has a 6-pin socket at the end with 5V power and ground, as well as RX, TX, RTS and CTS at 3V logic levels. These are perfect for use with the Mioduino and all other Arduino clones and derivatives and whenever you want to communicate with a TTL serial device http://adafru.it/70	*REQUIRED*	*x1*
	Extra-long break-away pins	Since the breadboard contains all female connectors and the FTDI Serial TTL cable has a female end, this header will allow the cable to be easily connected to the breadboard. Just break off the necessary number of pins and you're ready to go! http://adafru.it/400	*REQUIRED*	*x1*

	BREADBOARD WIRING BUNDLE	In this kit, you are supplied with low profile jumpers to assemble your Mioduino on the breadboard. This bundle of jumper wires is provided in various sizes which is great for when you begin experimenting with other components. http://adafru.it/153	*REQUIRED*	*x1*
	RED 3MM LED	Diffused Red 3mm LED. This LED is used as a power indicator for your Mioduino. These are rated for 1.9 – 2.3V Forward Voltage, at 20mA current. http://adafru.it/777	*REQUIRED*	*x1*
	BLUE 3MM LED	Diffused Blue 3mm LED. This LED will be used for various labs. These are rated for 3.0 - 3.4V Forward Voltage, at 20mA current. http://adafru.it/780	*OPTIONAL*	*x1*
	GREEN 3MM LED	Diffused Green 3mm LED. This LED is used as a communication status indicator for your Arduino. It can also be programmed directly in a sketch. These are rated for 2.0 - 2.5V Forward Voltage, at 20mA current. http://adafru.it/779	*REQUIRED*	*x1*
	WHITE 3MM LED	Diffused White 3mm LED. This LED will be used for various labs. These are rated for 3.0 - 3.4V Forward Voltage, at 20mA current. http://adafru.it/778	*OPTIONAL*	*x1*
	ROUND TACTILE BUTTON	Normally open momentary push button switch. In our circuit, the first one will be used as the reset button. The second switch will be used in the labs. http://adafru.it/1009	*REQUIRED*	*x2*
	1N40001 DIODE	The 1N40001 is a classic power blocking diode. In our circuit, this diode will be used to protect the circuit from reverse polarity connections with the power source. http://adafru.it/755	*1 REQUIRED* *1 OPTIONAL*	*x2*

	1K Ω Resistor	Standard size resistor used for the two LEDs on the Mioduino. Available from various sources.	*REQUIRED*	*x2*
	220 Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2780	*OPTIONAL*	*x3*
	2.2K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2782	*OPTIONAL*	*x3*
	22K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2785	*OPTIONAL*	*x3*
	470 Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2781	*OPTIONAL*	*x4*
	4.7K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2783	*OPTIONAL*	*x3*
	47K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2786	*OPTIONAL*	*x3*
	10K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2784	*OPTIONAL*	*x2*
	100K Ω Resistor	Standard size resistor used for various labs. http://adafru.it/2787	*OPTIONAL*	*x3*
	100 Ω Resistor	Standard size resistor used for various labs. Available from various sources.	*OPTIONAL*	*x3*

	10K Ω Potentiometer	This breadboard trim potentiometer has an adjustable resistance from 0 to 10K Ω. When turning the knob, the resistance between the left pin and the center pin will increase while the resistance between the center pin and the right in will decrease. http://adafru.it/356	*REQUIRED*	*x1*
	Reflective IR Sensor	This Reflective IR Sensor is a simple plastic casing with two elements - an IR LED and an IR phototransistor. You can control the IR LED and turn it on to bounce IR off objects and determine their reflectivity. White & light colored stuff will bounce the light, so you can detect it. Black & dark colored stuff will absorb the IR light so that it isn't detected. Likewise, if something isn't in the way of the sensor, it won't trigger either. http://adafru.it/2349	*REQUIRED*	*x1*
	Photoresistor	Photoresistors behave like standard resistors, except the resistance changes with the amount of light they are exposed to. These are also referred to as Photo Cells or a CDS. When it's light, the resistance is about 5-10K Ω, when dark it goes up to 200K Ω. http://adafru.it/161	*REQUIRED*	*x1*
	TMP36 Analog Temperature Sensor	Wide-range, low-power temperature sensor outputs an analog voltage that is proportional to the ambient temperature. To use, connect pin 1 (left) to power (between 2.7 and 5.5V), pin 3 (right) to ground, and pin 2 to analog in on your microcontroller. The voltage out is 0V at -50°C and 1.75V at 125°C. You can easily calculate the temperature from the voltage in millivolts: Temp °C = 100*(reading in V) – 50. http://adafru.it/165	*REQUIRED*	*x1*
	MCP9808 High Accuracy I2C Temperature Sensor	This I2C digital temperature sensor is one of the more accurate/precise sensors, with a typical accuracy of ±0.25°C over the sensor's -40°C to +125°C range and precision of +0.0625°C. It works great with any microcontroller using standard I2C. There are 3 address pins so you can connect up to 8 to a single I2C bus without address collisions. Best of all, a wide voltage range makes is usable with 2.7V to 5.5V logic! http://adafru.it/1782	*REQUIRED*	*x1*

	BI-FOLD COMPARTMENTS PART BOX	This is a storage box for your components. http://adafru.it/796	*OPTIONAL*	*x1*
	"I LEARNED MICRO-CONTROLLERS" STICKER	You are learning micro-controllers! This sticker is for your achievement. Don't stick this to anything until you have successfully built your Mioduino. http://adafru.it/689	*OPTIONAL*	*x1*
	HACKSPACE PASSPORT	Keep track of the hackspaces you visit with this passport. Don't forget to get it stamped! http://adafru.it/769	*OPTIONAL*	*x1*
	PRE-FORMED JUMPER WIRES	These preformed jumper wires will assist you in getting your Mioduino built faster. The number and sizes have been selected for you. Check your kit for the following jumpers: 1. 1x Tiny (no color) 2. 1x Small Red 3. 3x Small Orange 4. 4x Small Yellow 5. 2x Medium Green 6. 1x Medium Blue 7. 1x Medium Purple 8. 5x Large Red http://bit.ly/mioduino-jumper	*REQUIRED*	*x18*
	CARDBOARD STORAGE BOX	This cardboard box is provided to store and protect your Mioduino after you have completed it. http://bit.ly/mioduino-box	*OPTIONAL*	*x1*
	WHITE 3 X 5 INDEX CARD	This card is used to test the reflective sensor. http://bit.ly/mioduino-card	*OPTIONAL*	*x1*

ORDERING COMPONENTS

All of these parts and components can be purchased from Adafruit (www.adafruit.com) with the exception of the jumper wires which can be purchased through Elenco.

The links below can be used to view and order a complete list of parts.

Full Kit for 15 Students

https://www.adafruit.com/wishlists/393585

Full Kit for 10 Students

https://www.adafruit.com/wishlists/ 397786

Single Kit

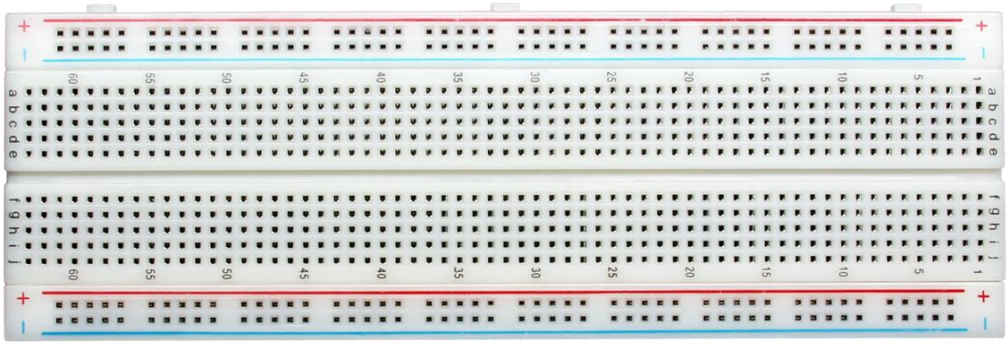
https://www.adafruit.com/wishlists/396935

CLASSROOM 1

In this session we want to cover the basics of the Arduino and ensure you are ready to build your first Mioduino.

USING A BREADBOARD

A solderless breadboard (normally shortened to *breadboard*) is a device that allows temporary circuits to be built quickly either for experimenting or for testing prior to fabrication on a PCB (printed circuit board). This procedure is called **PROTOTYPING**.

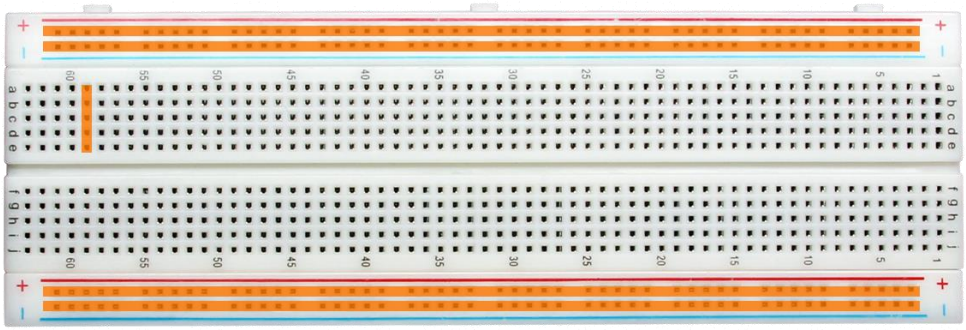


To understand the structure, consider the board to be laid out so that the long edge is horizontal and the short edge is vertical. The horizontal pins are the rows and the vertical pins are the columns.

Breadboards come in various shapes and sizes but all have the same relative structure. There are two primary sections called **STRIPS**. The inner strip is where most of the components will be located. This strip is also subdivided into two sections with two sets of columns. All of the pins in a single column are connected together and are labeled **A** through **E** on the top side and **F** through **J** on the bottom side. These two sets of columns are separated down the middle to provide a convenient location for integrated circuits using what is called a Dual Inline Package (**DIP**).

The outer strips have four rows of pins running the horizontal length of the board to provide power and are referred to as the **POWER RAILS**. There are two sets of independent power rails on most breadboards and are often marked with a **+** and **–**

to help make prototyping easier. It is up to the prototype to make the proper power connections to these power rails.



This image highlights how the underlying connections between the pins are made.

*The labs will refer to connections to 5V and GROUND. On the breadboard, **5V** is the rail labeled **+** while **Ground** is the rails labeled **-**.*

USING JUMPER WIRES

It is convenient to have jumpers wires already made when prototyping on a breadboard to expedite the process. Jumper wires can be made from any standard 22 AWG solid copper wire or they can be purchased premade.

In this guide there are two types of jumper wires used. The first set is premade stripped wires that are meant to be place snuggly on the breadboard. This type will be used to connect the various components of the Mioduino and are primarily used for two reasons. The first reason is to ensure they do not get in the way when we begin to experiment with various components and sensors on the breadboard. The second reason is that we can tell them apart from the other jumper wires that will be added and removed frequently to and from the board. We do not want to inadvertently pull out a wire necessary to keep the Mioduino working. The second type of wire is the flexible jumper wire that has pins at either end. The pins make it

easy to plug into the breadboard and the flexibility is suitable for prototyping. These wires will be used during the various labs.

Understanding the Components

Overview

Much like properly written software, hardware can be thought of as building blocks where smaller, simpler blocks are joined together to create larger complex blocks. This Mioduino (the name of our Arduino on a breadboard) is a combination of some simple circuit blocks.

The Microcontroller

Of course, the most important part of our kit is the Atmega328P, the microcontroller. This chip contains everything that makes our board an Arduino, but, it requires help from a few external blocks to be complete.

Power Supply

The ATMega328P requires 5V to operate efficiently. It is possible to overclock this chip but that is beyond the scope of this guide. Our power supply block has at its heart a 7805 which is a common 5V linear voltage regulator. It can handle 7 to 35V on its input side and provides a constant 5V on its output side. This chip can deliver up to 1.5A for the circuit which is plenty of current for our Atmega328P and our labs!



In this guide, a 9V, 1A power supply has been recommended. This means that a theoretical maximum of 1A can be drawn from this device. No need to fear: if you need more current, use a larger power supply such as a 12V, 1.5A or higher.

This chip can also take a TO-220 heat-sink. The voltage regulator will generate heat when the power reaches a certain point. The higher the input voltage and output current, the more heat it will generate. The chip is capable of dissipating up to 2W of power on its own without a heat-sink. The formula for determining the power of your setup is

$$P = (V_{in} - 5V) \times I$$

where I is the average current drawn from the device and V_{in} is the input voltage supplied to the regulator. Using the equation above, and assuming a 9V power supply, the circuit can draw up to *.5A* (or 500mA) before we need to add the heat-sink.

Clock

The microcontroller performs its function one instruction at a time and is controlled by a clock. In this block, the clock is built using the 16 MHz crystal and two 20 pF capacitors.



The ATMega328 has an internal clock that runs at 8 MHz eliminating the need for an external clock. This is achieved by changing the values of what are called "fuses" on the chip. These fuses tell the chip, among other things, where the clock is (internal or external) and the speed of the clock. Our chips are set up already to expect an external 16 MHz clock.

Reset

The ATmega328P supports reset function on pin one that instructs the chip to restart the firmware. This reset is activated whenever the reset pin is connected to ground. The reset block consists of a little more than a switch that connects the pin to ground when pushed. There is a pull-up resistor that keeps the pin at a higher voltage to ensure the microcontroller does not inadvertently reset while running. In

addition, we have wired our Serial TTL to the reset switch through the .1 µF capacitor to allow the chip to be reset after programming (resetting after programming starts the newly loaded firmware).

The LEDs

There are two LEDs in our circuit. The first one, red, is the power indicator. When this LED is on, we know that we have power on our board. The second LED, green, is used to indicate that our board is performing some internal work (while booting for example or when we are communicating with the board through the serial interface. This LED can also be controlled through code and is the target of the first lab.

It is important to be able to identify the two leads of an LED. Typically, one lead is longer than the other and is referred to as the **Anode** or **Positive** lead. The other shorter lead is the **Cathode** or **Negative** lead.

If both leads of an LED are the same, it is still possible to identify the Anode and the Cathode. Examining the inside of the LED will reveal that there are to pieces of metal separated by a small gap. The smaller of these two is the Anode.

One more way to identify the leads is to look at the bottom ring, where the leads go into the plastic housing. The bottom of this housing is a circle with a flat edge. The lead on the flat edge side is the Cathode.

Programming Interface

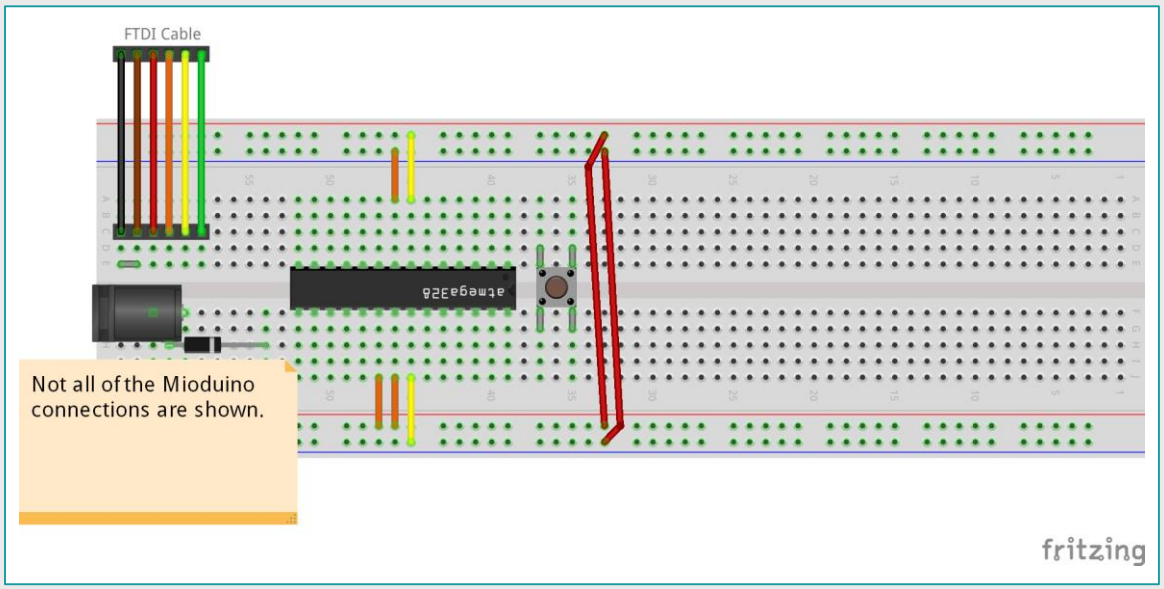
The Mioduino is not very useful until we can load our software, called firmware, onto the board. This is done is through a Serial TTL interface. Since most computers no longer have serial ports, we will use a special cable with a USB to Serial chip embedded within the cable.

Powering the Board

There are two ways to power this board. The first is through the power supply using the DC Barrel connector. Any power supply with this type of connector and an output voltage between 7 and 36 V can be used. The voltage regulator will ensure that only 5V will be supplied to our Atmega328P. The second method is via the

Serial TTL USB cable. This power option will be used whenever you are programming the board. *NOTE THAT YOU SHOULD NEVER CONNECT BOTH THE SERIAL TTL CABLE AND THE DC POWER SUPPLY AT THE SAME TIME*. This circuit does not have any protection built in to prevent voltage or current spikes emanating from the power supply from damaging the USB port on your computer.

The image below shows how the FTDI cable should be connected to the Mioduino.



Lab 1: Build Your Mioduino

Objective

In this lab, you will build your Arduino compatible on the breadboard using the supplied components. Please follow the instructions carefully while checking your progress each step of the way. Before you begin this lab, make sure that you have verified that you have all of the correct components.

Steps to Follow

Assemble the Voltage Supply

Assemble the board slowly and carefully paying particular attention to the placement of each component and wire. Some components are symmetrical which means the component can be connected in any direction and it will work. Other components are directional or polarized and need to be placed in a specific manner or they will not function the intended way. Components with three or more leads are very specific to their placement and can be damaged if placed incorrectly.



Your board will look great if you trim the leads on some of the components and keep them low to the board; however, it is best to wait until you have fully assembled the board and have tested it. Once this has been done, you can remove components one at a time, trim the leads, and place them back into their positions.

1. Place the DC Barrel jack into **H62**, **F61** and **F63** with the Plug facing the outside of the board.
2. Place a .1µF capacitor between **G54** and **G55**. This is a symmetric component, so it does not matter which way it is placed into the board.

3. Place the voltage regulator into **F54**, **F55** and **F56** with the flat edge adjacent to the power rail on the **F-J** side of the board. When looking at the flat edge of the component, the right-most pin is **V_{IN}**, the middle pin is **GROUND**, and the left-most pin is **V_{OUT}**.
4. Place the 1N4001 Diode between **H61** and **H54**. Note the diode is directional so ensure that the end with the white stripe is in **H54**.
5. Place a medium blue jumper between **I63** and **GROUND**.
6. Place a .1 µF capacitor between **GROUND AND 5V** (into the power rail) in the third position from the DC Barrel end of the board on the F-J side.
7. Connect a small yellow jumper between **J56** and **5V**.
8. Connect a medium purple jumper between **H55** and **GROUND**.
9. Place a 1K Ω resistor between **I57** and **5V**.
10. Place the red LED between **J57** and **GROUND** with the *ANODE* at **J57** and the *CATHODE* at GROUND. The Anode of the LED is the longer of the two leads.
11. Place the 47 µF electrolytic capacitor between **I55** and **I54**. The electrolytic capacitor is *POLARIZED* so ensure that the negative lead is in position I55. You will need to trim both leads to the same length to get them to fit into the breadboard.
12. Place the 100 µF electrolytic capacitor between **GROUND** and **V5**. Ensure that the negative lead is in GROUND. You will need to trim both leads to the same length to get them to fit into the breadboard.

Test the Voltage Supply

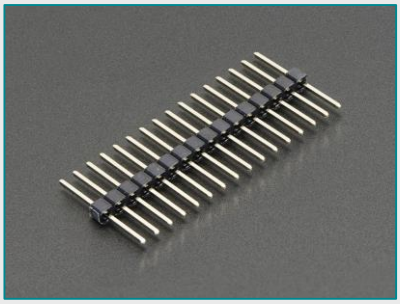
1. Double-check all of your connections.
2. Connect the 9V power supply to one end of the inline switch.
3. Connect the other end of the inline switch to the DC Barrel jack on the breadboard.
4. Turn the switch to the off position.
5. Connect the power supply to a power source.

6. Turn the power switch on. The red LED should turn on.
7. With a power meter, measure the voltage at the power rail on the F-J side of the breadboard. The voltage should be ~5V.
8. If the voltage reading is correct then proceed to the next section. If it is not correct, turn the power off, check your connections and try again.

Assemble the Mioduino

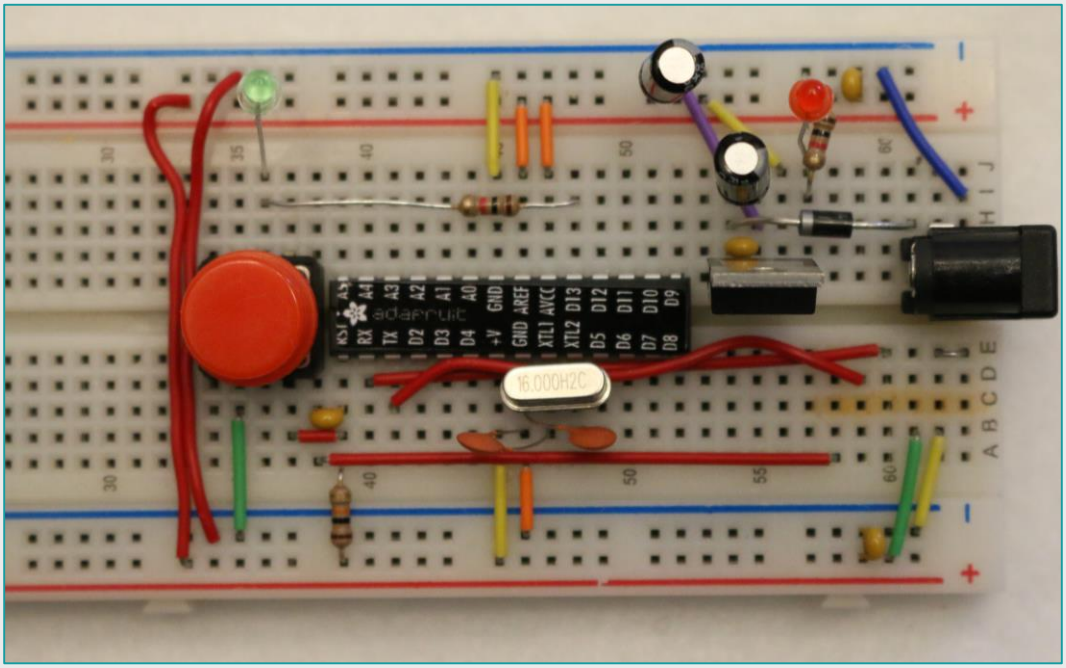
1. Place the Atmega328P chip onto the breadboard placing the reset pin at position **E39** and the **D9** pin at **F52** (you will need to carefully push the pins inward towards the chip to make them fit into the breadboard).
2. Align the four pins to **D35**, **D37**, **G35** and **G37** and place the pushbutton switch onto the breadboard
3. Connect a medium green jumper between **C35** and **Ground**.
4. Place a .1µF capacitor between **C38** and **C39**.
5. Connect a small red jumper between **B37** and **B39**.
6. Place a 10K Ω resistor between **A39** and **5V**.
7. Connect a small yellow jumper between **A45** and **5V**.
8. Connect a small orange jumper between **A46** and **GROUND**.
9. Connect a long red jumper between **A38** and **A58**.
10. Place a .1 µF capacitor between **GROUND and 5V** (into the power rail) in the third position from the DC Barrel end of the board on the A-E side.
11. Connect a small yellow jumper between **B62** and **GROUND**.
12. Connect a medium green jumper between **B61** and **5V**. Optionally, you can replace this jumper with a 1N4001 Diode by placing the cathode the end with the white stripe) at **5V** and the cathode at **B61**. This diode, although not required, can provide extra protection for your FTDI cable.
13. Connect a long red jumper from **GROUND** on one power rail to **GROUND** on the other power rail.
14. Connect a long red jumper from **5V** on one power rail to **5V** on the other power rail.

15. Place a 1K Ω resistor between **I36** and **I48**.
16. Place the green LED between **J36** and **5V** with the *ANODE* at J36 and the *CATHODE* at Ground. The Anode of the LED is the longer of the two leads.
17. Connect a small yellow jumper between **J45** and **GROUND**.
18. Connect a small orange jumper between **J46** and **5V**.
19. Connect a small orange jumper between **J47** and **5V**.
20. Connect a long red jumper between **D40** and **E60**.
21. Connect a long red jumper between **C41** and **D59**.
22. Place the tiny jumper between **E62** and **E63**.
23. Place the first 20 pF capacitor between **C46** and **B48**.
24. Place the second 20 pF capacitor between **B46** and **B47**.
25. Place the 16 MHz crystal between **C47** and **C48**.
26. Using pliers, break away 6 pins from the long headers and place them into the board between **C58** and **C63**. After plugging the FTDI cable in for the first time, the header will remain in the cable. Use a highlighter to mark these pin positions on the breadboard to easily identify the insertion point each time the cable is connected to the breadboard. Remember the black wire end of the cable is always near the edge of the board.

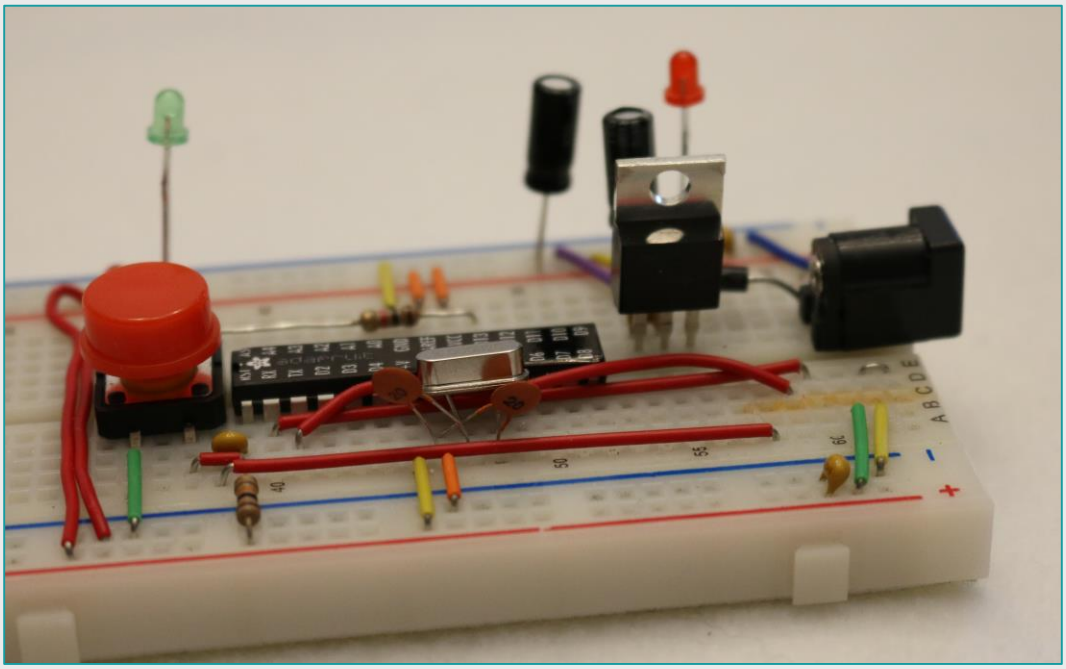


Completed Board

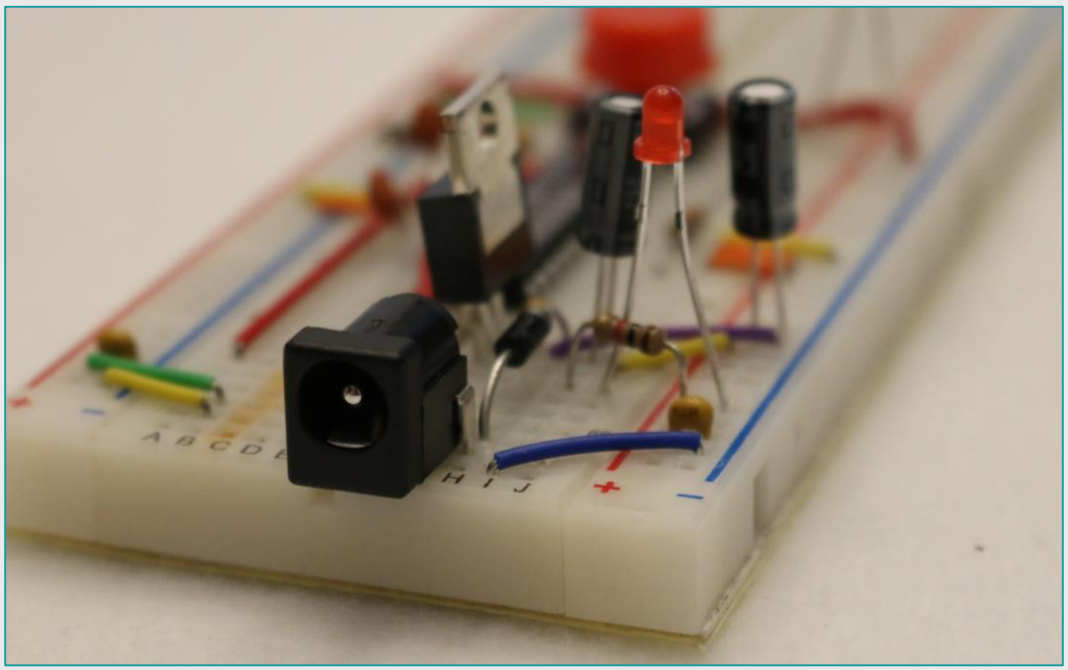
The images in this section show a completed board that will help guide you during the build process.



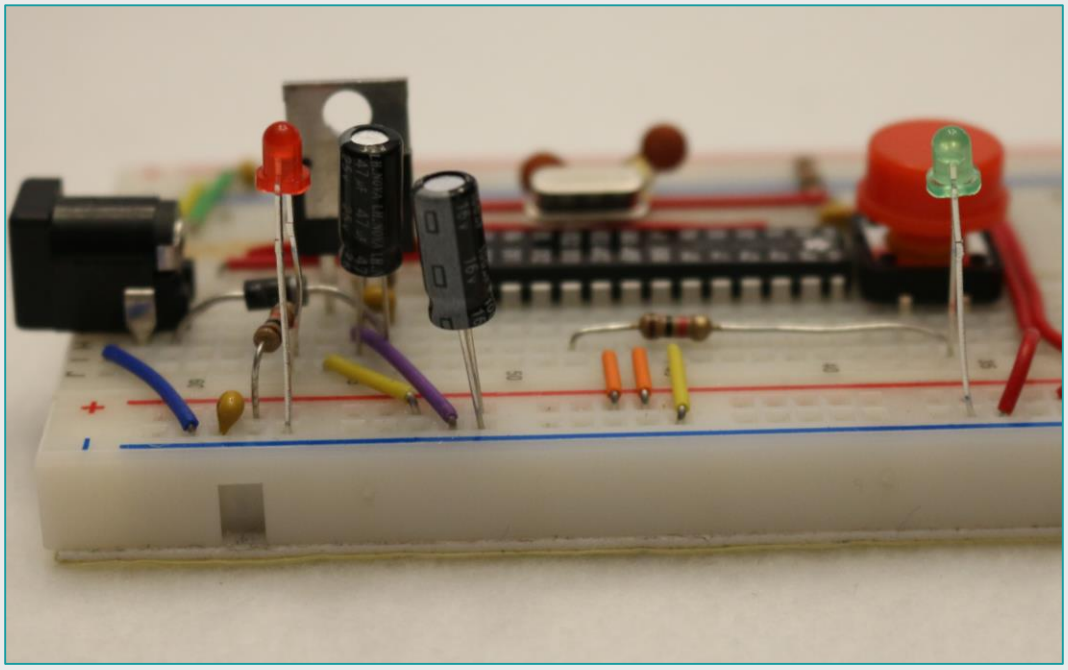
TOP VIEW OF THE MIODUINO.



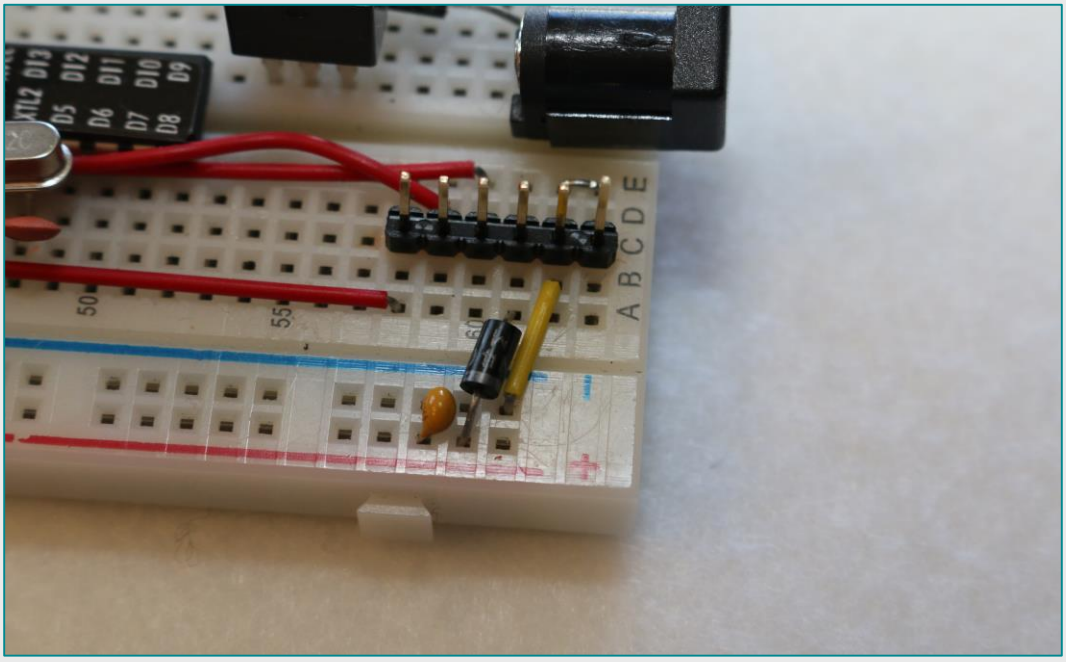
SIDE VIEW OF THE MIODUINO SHOWING THE CLOCK, RESET AND FTDI CONNECTION.



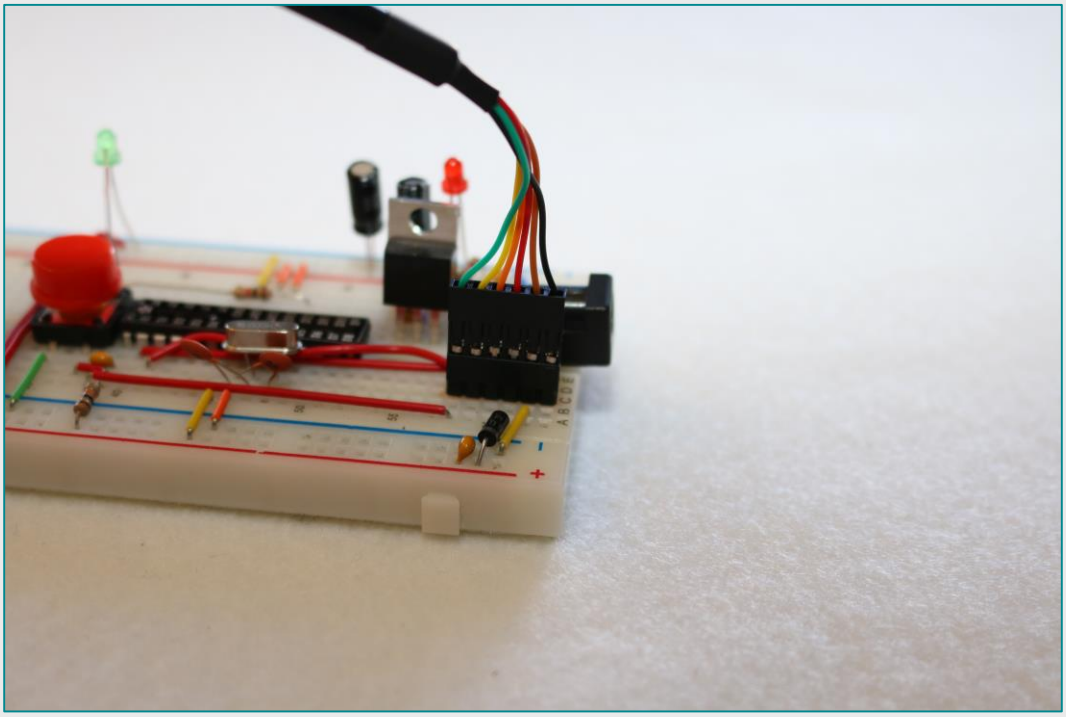
DC POWER CONNECTOR ON THE MIODUINO.



5V DC POWER SUPPLY ON THE MIODUINO.



THE GREEN JUMPER REPLACED BY THE OPTIONAL 1N40001 DIODE. THE 6-PIN JUMPER HAS BEEN INSERTED INTO THE BOARD TO SHOW IT'S PLACEMENT.



THE FTDI CABLE CONNECTED TO THE 6-PIN HEADER ON THE BREADBOARD.

Lab 2: Mioduino Blinky

Objective

The purpose of this lab is to test your Mioduino by making the built in LED blink.

Run the Sketch

1. Open the Arduino IDE.
2. Load the sketch for **Lab 2**.
3. Select the board to use from the **Tools/Board** menu and choose **Mioduino**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **Serial TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **Tools/Port** menu choose the correct COM port for your cable (in most cases, you will only have one COM port listed).
7. Open the serial port monitor by choosing **Serial Monitor** from the **Tools** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **Upload** from the **Sketch** menu.
9. The *GREEN* LED will flash while the code is loading.
10. When loading is complete, the *GREEN* LED should flash on and off every second.

Classroom 2

Before getting started, it is a good idea to understand the components that make up the Arduino and the purpose they serve.

The Microcontroller

A microcontroller is a small computer with the following (typical) characteristics:

1. Support devices such as RAM, ROM, IO and CPU are internal and contained on a single IC.
2. Found embedded in devices (to control their behavior or interaction).
3. Dedicated to one task or one set of tasks.
4. Low power.
5. Dedicated/fixed inputs.
6. Small and inexpensive.
7. No operating system. The software running on the microcontroller is referred to as firmware.

Electronic Components

The Resistor

This device is a two terminal passive component used to reduce the flow of current or to reduce voltage in a circuit. The typical resistor has a fixed amount of resistance which can be determined by the color coding on each component. The resistors used in this lab are four-band resistors and can be decoded using the chart below.

Band Color	1st Band	2nd Band	3rd Band	4th Band
	Digit 1	*Digit 2*	*Multiplier*	*Tolerance*
Black	0	0	1	
Brown	1	1	10	+/- 1%
Red	2	2	100	+/- 2%
Orange	3	3	1K	
Yellow	4	4	10K	
Green	5	5	100K	+/- 0.50%
Blue	6	6	1M	+/- 0.25%

Violet	7	7	10M	+/- 0.10%
Gray	8	8	100M	
White	9	9	1000M	
Gold	-	-	.1	+/- 5%
Silver	-	-	.01	+/- 10%

Some resistors have **FIVE** bands while others even have **SIX**. With a five band resistor, the third band is digit three, the fourth band is the multiplier and the fifth band is the tolerance. A six band resistor is the same as a five band resistor with the addition of the sixth band which represents the temperature coefficient.

The resistance is measured in units called **OHMS** and is represented by the Greek Symbol Omega (Ω).

There are also variable resistors, called potentiometers, that allow the resistance value to be adjusted. These types are usually noted by their maximum resistance value. If you are participating in the hacker Weekend event, then you were provided as part of your kit a 10K Ω (10,000 ohms) potentiometer than can be adjusted by turning the knob on top of the device.

THE CAPACITOR

A capacitor is a passive device used to store an electric charge, consisting of one or more pairs of conductors separated by an insulator. Common uses of this component include:

1. Filtering
2. Storage
3. Power conditioning
4. Signal coupling
5. Decoupling
6. Oscillators

THE DIODE

The diode is a two-terminal semiconductor with the primary purpose of controlling the direction of current flow. Current passing through a diode can do so only in one direction. This direction is referred to as *FORWARD DIRECTION*. Current attempting to move in the opposite direction, *REVERSE DIRECTION*, is blocked.

When current flows in the forward direction, an ideal diode behaves like a wire with zero resistance. When current tries to move in the reverse direction, the ideal diode behaves like an open circuit.

The two terminals on a diode are *POLARIZED*, which means that you need to pay attention to how you connect a diode in a circuit (opposed to a resistor that is symmetric and can be reversed without changing the behavior). The positive end of the diode is called the *ANODE*, and the negative end is called the *CATHODE*. The current flows from the anode to the cathode. Most diodes will have a stripe on cathode end.

Actual diodes, non-ideal, have what is called a *FORWARD VOLTAGE*. This is the minimum voltage required across the terminals to start the flow of current. This positive voltage varies based on the type diode. When a positive voltage is applied in this manner, the diodes is considered to be *FORWARD BIASED*.

When a negative voltage is applied to a diode, and blocks the flow of current, then the diode is considered to be *REVERSED BIASED*.

Actual diodes also have a reverse *BREAKDOWN VOLTAGE* and will conduct current in the reverse direction when this negative voltage is reached. A diode will maintain this voltage across its terminals regardless of the amount of current flowing in the reverse direction. This makes for some very interesting uses besides just controlling current that are beyond the scope of this guide.

There are several types of diodes that are designed for specific purposes. The descriptions below provide very basic information on the varying types diodes. See the references at the end of this guide if you want to get a deeper understanding of diodes and dhow the various types of diodes are used.

Signal Diodes

The most common type of diode is the **SIGNAL DIODE** designed for standard use with lower voltages and currents.

Rectifier Diodes

The **RECTIFIER DIODE** has larger current ratings and is typically used to convert AC currents into DC currents.

Zener Diodes

The **ZENER DIODE** is designed to be used in reverse-voltage mode because it has a very precise breakdown voltage and is typically used to regulate voltage in a circuit.

Schottky Diodes

The Schottky diode has a lower-than-normal forward voltage and a much higher **REVERSE RECOVERY TIME** (the amount of time the diode switches from conducting to non-conducting mode). This type of diode is typically used as a **VOLTAGE CLAMP** to protect circuits from over-voltage. They are also commonly used to protect circuits from reverse current that can otherwise damage sensitive components.

Photodiodes

The **PHOTODIODE** is designed to capture light and create electrical current.

Light Emitting Diodes

The light emitting diode (LED) is very common because they emit light at low voltage and low current levels. These are covered in more detail in the next section.

THE LED

Whether you have known it or not, you have seen Light Emitting Diodes (LEDs). These special diodes give off light when they are forward biased (positive voltage and a forward current). The light emitted by these can be in both the visible and non-visible spectrum.

LEDs have many useful purposes other than being cool to look at. For example, when coupled with a photosensor device (such as a photodiode for example), they can be used to isolate current in one circuit from another. The first circuit can apply a signal to the LED and the photosensor picks up the light and converts it to a current on the second circuit. This technique is referred to as **OPTICAL ISOLATION** and is quite common when isolating high voltage circuits from low voltage circuits. See my *HACKSTER.IO* project ***Opto-Isolated AC Voltage Sensor*** at *HTTPS://WWW.HACKSTER.IO/PORREY/VACSENSOR*.

An LED has two leads, one called the **ANODE** and the other is called the **CATHODE**. The Anode is the positive terminal and the cathode the negative terminal.

VOLTAGE REGULATOR

A voltage regulator is a device that maintains a constant voltage with varying current demand. These devices can be quite complex and are usually contained in a single package. The common characteristics of a package will include the output voltage, the input voltage range and the maximum current.

The output voltage is the constant voltage maintained by the device. Voltage regulators are chosen for a particular circuit based on the target voltage required.

The input voltage is usually expressed in a range allowing varying types of voltage supplies to be used. Regardless of the input voltage, as long as it is in the specified range, the output voltage is maintained.

The current rating is important. A circuit that depends on the voltage regulator should never draw more current than the what is rated for the device or one of two things may happen. First, the device may lose its ability to maintain the constant voltage. Second, it may overheat and become damaged. It is also important to note that the power supply providing the input voltage be rated at or higher than the rated current of the device. If the circuit draws more current that the input device is supplying the regulator will fail to work properly.

Voltage regulators are often used in conjunction with step down transformers and rectifier circuits to convert AC signals to DC. This first stage converts the very large voltage, from a wall outlet in a house for example, down to a manageable DC voltage which is then fed into a voltage regulator to create a very “clean” DC supply. Voltage regulators can also be used to stabilize the voltage from batteries and solar cells.

PHOTORESISTOR

The photoresistor, also referred to as a Light Dependent Resistor (LDS) or a photocell, is a device that works much like a resistor, but whose resistance is dependent on the amount of light on the device. The resistance of the device decreases with the intensity of the light.

SWITCH

Switches are simple devices that open and close a circuit loop usually through a mechanical process like the pushing of a button. They are not limited, however, to mechanical operation. Some switches can be "electronic", meaning they are activated and deactivated by another portion of the circuit or by another circuit all together. A relay is one type of switch that works in this manner. A relay can be both mechanical in design or built from solid state components.

Although switches are very common and easy to understand, they are important to discuss because they can introduce some complexity into a circuit. One type of complexity is **BOUNCING**. When a button is pressed, the mechanism that makes the contact may actually open and close a few times before finally closing the circuit. In some cases, this effect can be ignored (like a light switch on a wall), because it does not have any tangible side effect, but not always. In digital circuits, this bouncing effect will almost always have a side effect that cannot be ignored. In these cases, we need a solution called **DEBOUNCING**. Debouncing is a technique to remove the bounce effect which can be accomplished using electrical components or in software. In most cases software debouncing works fine and it is the easiest of the two to implement.

ELECTRONIC CIRCUIT CONCEPTS

There a few basic circuit concepts that are common and are important to understand when you are beginning with the Arduino platform. The concepts presented here in no way include everything there is to learn but are the most common concepts you will encounter when working on basic Arduino circuits.

PASSIVE AND ACTIVE COMPONENTS

All electrical components can be classified into one of two categories: **ACTIVE** or **PASSIVE**. Official definitions vary on exactly how to describe these terms but the easiest way to think about this is to consider how the component affects the circuit. In the most basic form, a passive component alters current flow or the voltage of a circuit and consumes power in the circuit (dissipates electrical energy). An active

component can alter current flow or voltage but does so by increasing (or maintaining) the overall power in the circuit.

A few typical passive components are resistors, capacitors and inductors. Active components include things like power sources, integrated circuits and transistors.

Interestingly, a diode can be considered both an active and passive component, depending on how it is made and used in the circuit. A diode is passive unless it has a negative differential resistance, in which case it would be considered active. A full description of this concept is out of scope for this guide.

In order for a circuit to be considered an "electrical" circuit it must contain at least one active component.

Ohms Law

A very important concept to understand is that of Ohm's Law. Simply put, it states that the current through a conductor between two points is directly proportional to the voltage across those two points. This then implies the component has a "resistance" to electrical current flow that can be measured and stated as follows:

$$I = \frac{V}{R}$$

Where I is the current measured in amperes, V is the voltage measured in volts and R is the resistance measured in ohms. This concept, and equation forms the basis for many of the more complex calculations required to understand an intricate circuit.

Series and parallel Circuits

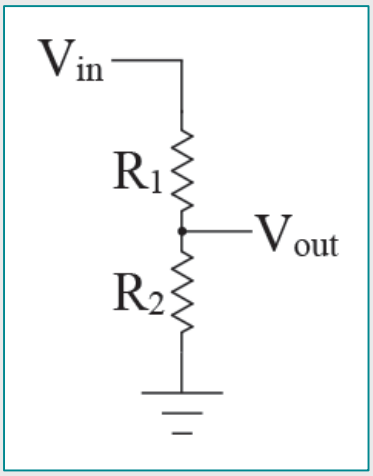
Electronic components can be arranged in one of two ways, series or parallel. Most circuits are made up of multiple parallel and multiple series circuits combined to form complex circuits.

A series circuit is one in which the components are aligned to create a single path for the flow of current. In this type of circuit, the current through each component is the same.

A parallel circuit is one in which the components are connected to form multiple paths for the current. In this type of circuit, the voltage is always the same across all of the components.

Voltage Divider

One of the most common basic circuits types you will come across in your endeavors is the voltage divider. The voltage divider is a circuit does what it states, it divides a voltage. This circuit takes an input voltage and divides it so that the output voltage is a fraction of the input voltage. It is a simple construction based on two resistors in series. A diagram of it is shown here.



Determining the output voltage based on the input voltage and the two resistors is simple using Ohm's Law and can be represented with the formula shown here.

$$V_{out} = V_{in} \times \frac{R_2}{(R_1 \times R_2)}$$

Kirchhoff's Voltage Law

This law states that "*The directed sum of the electrical potential differences (voltage) around any closed network is zero*". Although it is beyond the scope of this guide to get into the specifics, we will use this law to help bias our LEDs properly.

Given a circuit with a 5-volt source, a resistor and an LED with a forward voltage of 3.3V connected in series, this rule states that the voltage drop across the resistor will be 1.7V. Since this is a series circuit, we also know that the current through the LED and the resistor are the same. This is helpful in determining the size of the resistor so we do not pass too much current through the LED, which could result in damage to the LED. If the maximum current of the LED is 20 mA, then from Ohms Law and

Kirchhoff's Voltage Law, we know we would need at least an 85 Ω resistor. In this case, we would most likely choose a standard 100 Ω resistor since this is more common.

Pulse-Width Modulation

Pulse-Width Modulation (PWM) is the encoding of information on a digital signal (square wave) where the width of the HIGH portion of the signal varies in order to convey the encoded information. This type of signal can also be used to control the amount of power to a device, most commonly, an LED.

Since LEDs have specific forward voltage specifications and a maximum current specification, we would not typically control the intensity of the LED by adjusting either the voltage or the current. Doing so could, at the least, diminish the life of the LED, and worse case, permanently damage it.

Pulse-Width Modulation uses a square wave signal to turn the LED on and off at such a high rate that our eye cannot see that this is actually happening. In a given time frame the pulse, the amount of time the LED is on and the amount of time the LED is off, can be varied (called the **DUTY CYCLE**). If the LED is on longer, it will appear to our eye as being more intense (brighter). If the LED is off longer, it will appear to our eye as less intense. Thus, the apparent intensity of the LED is adjusted by varying the duty cycle.

This technique is very common with RGB LEDs (red, green and blue) to vary the perceived color of the LED. See my *HACKSTER.IO* project called ***Many Colors with Variable RGB LED*** at https://www.hackster.io/porrey/rgbled.

Analog and Digital Circuits

The circuits you will encounter will either be analog or digital or a combination of both. When dealing with computers, they typically work with digital signals. A type of analog circuit would be an audio amplifier.

Analog Circuits

Analog circuits have voltages that can vary in a specified range from negative voltages to positive voltages. This is considered a continuous voltage model in which the voltage signals can appear at any level at any point in the circuit.

Digital Circuits

Digital circuits, or binary circuits, deal with two voltages commonly referred to as the HIGH voltage and the LOW voltage. The value of the HIGH and LOW voltages is determined by the reference voltage of the microprocessor and is called the **LOGIC LEVEL**. The two most common logic levels are 5V and 3.3V (often written 3V3).

The Arduino platform is a 5V platform (although some version operates at 3V3) while the Raspberry Pi works at 3V3. It is important to know the logic level of your platform and of the devices you use. Mixing these levels can cause permanent damage to your microcontroller, sensors and even USB ports on your computer when you are interfacing to these circuits. Many devices perform what is called logic level shifting to allow 3V3 devices to communicate with 5V systems and vice-versa.

Floating State

There is a third level in digital circuits, which we typically want to avoid, called **FLOATING**. Floating means that we are neither HIGH nor LOW and unless the microprocessor or sensor is designed to deal with it, it should be avoided. Floating state is easy to avoid if we design our circuits properly. Take a switch, for example, that has one terminal connected to a pin on the Arduino and another pin connected to Ground. When the switch is pushed, our pin is LOW, but when the button is released, the pin is floating because it is not connected to anything. If we were to read this pin on the Arduino it would randomly read HIGH and LOW because the Arduino is not designed to deal with this state. We can easily solve this problem by connecting a 50K or 100K Ω resistor between the pin and 5V. This resistor is called a **PULL-UP** resistor because it pulls the pin to HIGH. When the switch is pushed the pin is not connected directly to Ground and the pin is now LOW. If we reversed the connections so that the switch connects us to 5V and the resistor is connected to ground, then the resistor would be called a **PULL-DOWN** resistor because it pulls the state of the pin down to a LOW.

Conversion

It is very common to mix analog and digital circuits. In this guide we will take a look at a couple of analog devices in our labs. To work with analog signals in a digital circuit, we need a process to convert the signals. Converting a digital signal to analog is called Digital-to-Analog Conversion (**DAC**). There are many devices that can do this conversion.



The function in the Arduino platform called analogWrite does not actually perform DAC, but is actually a PWM output.

Converting the opposite way, from analog to digital (ADC) is more common. In fact, the Arduino has built-in 10-bit ADC's that can read an analog voltage and convert it to a number between 0 and 1023. Knowing the reference voltage, which in most cases is 5V (unless we connect an external source), we can easily convert this to a voltage value in our code.



The conversion of audio to digital is considered ADC and from digital to analog is DAC. Typically, with audio, the process is more complex due to the addition of compress techniques to reduce the size of the converted signals.

Arduino Basics

Arduino IDE

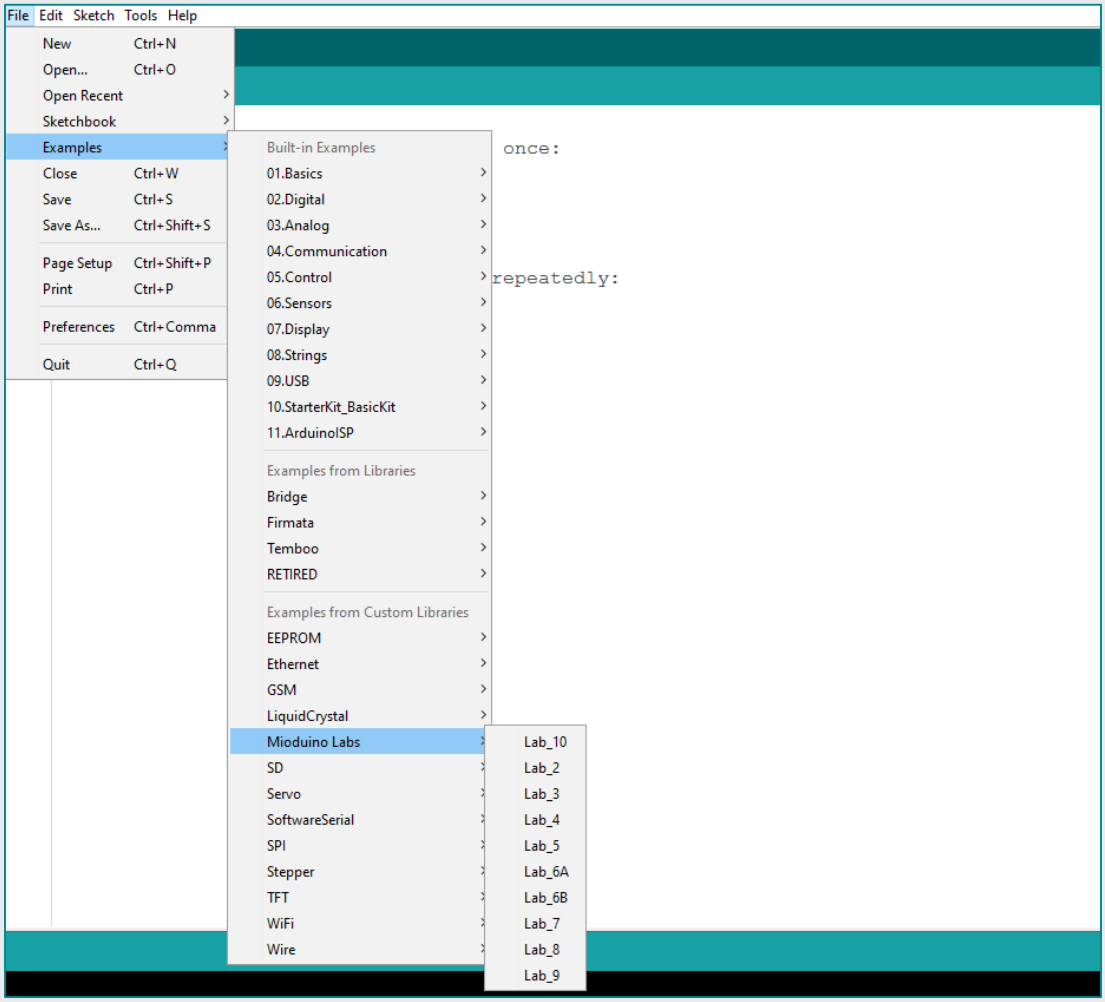
The development environment for the Arduino is the Arduino IDE which can be downloaded from **Arduino.cc**. This environment includes everything that is needed to get started with the basic boards. The IDE is also extensible, allowing additional hardware platforms and software libraries to be installed and used within your code.

ARDUINO SKETCH

The primary language for the Arduino platform is C/C++. The Arduino does not run an operating system but instead runs the code you write and is referred to as firmware. The firmware is contained in a main code file called the **SKETCH** which can then reference other C++ libraries to expand the capabilities of your firmware.

LOADING THE LAB SKETCHES

The sketches for the labs can be found in the **FILES** menu under **EXAMPLES**.

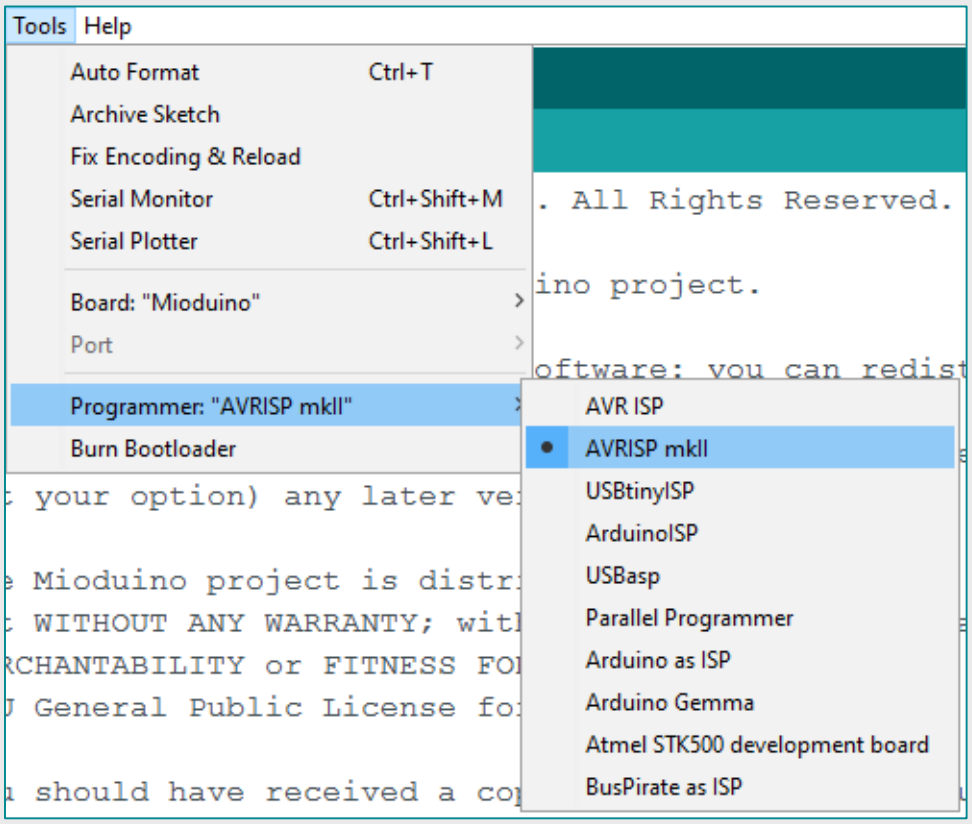


Select the appropriate lab from the menu and a new window will load with that sketch.

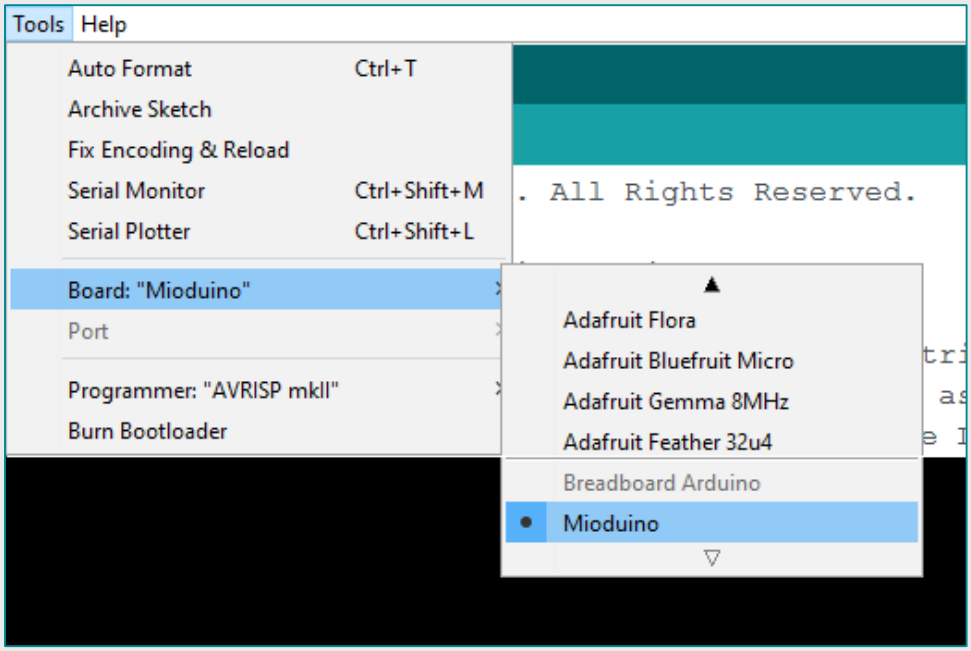
SELECTING THE BOARD AND PROGRAMMER

The board selection for the labs in this guide is the Mioduino. If you have not installed this board, please go to **INSTALL THE MIODUINO BOARD** section.

To select the board, click the Tools menu and select Boards.



Scroll down the list and click **MIODUINO**.



Also ensure that the correct programmer is selected. For the Mioduino, you will select **AVRISP MKII** from the **TOOLS** menu.

Lab 3: My Blinky

Objective

An LED can be activated by applying a voltage across the two terminals. If this voltage is too high, the LED will burn out and no longer emit light. To prevent this from happening, a resistor should be placed in series with the LED to limit the amount of current and apply the correct voltage to the LED. This resistor is often referred to as a current limiting resistor in this type of circuit.

When LEDs are manufactured, they have a specification for the accepted voltage range the amount of current they are designed to handle. It is possible to make an LED brighter by increasing the voltage or the current, but this is not recommended because doing so may decrease the life expectancy of the LED or damage it. The process of applying the correct DC voltage across the two terminals of the LED is called **BIASING**. It is important that we bias our components correctly to get the behavior we are expecting.



Since the biasing of an LED is fixed, it would appear that we are stuck with a fixed brightness, however, this is not the case. In the next lab we will examine the proper manner to adjust the apparent brightness of an LED

In this lab, you will build an LED circuit with a current limiting resistor. The resistor will be selected by applying Ohm's Law and Kirchhoff's Law using the known characteristics of the LED.



***Warning**: the LEDs chosen for this kit are super bight at the specified current. These can hurt your eyes when looking at them directly. They are designed to be seen during the day at all angles.*

Prepare for the Lab

First, you need to determine the correct size resistor to use with the blue LED. Take a look at the specifications in the component list for the blue LED. Using the formula here, calculate the correct size current limiting resistor:

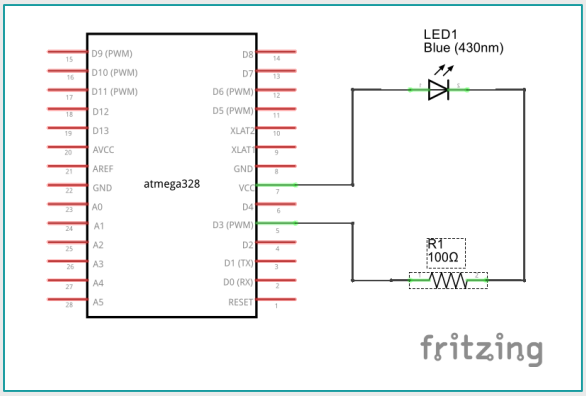
$$R = (5V - V_f) \div I_{max}$$

where V_f is the forward voltage of the LED (found in the specification) and I is the maximum current for the LED (also in the specification). Note the value for R will not match an exact resistor value so pick a resistor whose resistance is closest to the value you calculated.

Find the necessary components and set them aside.

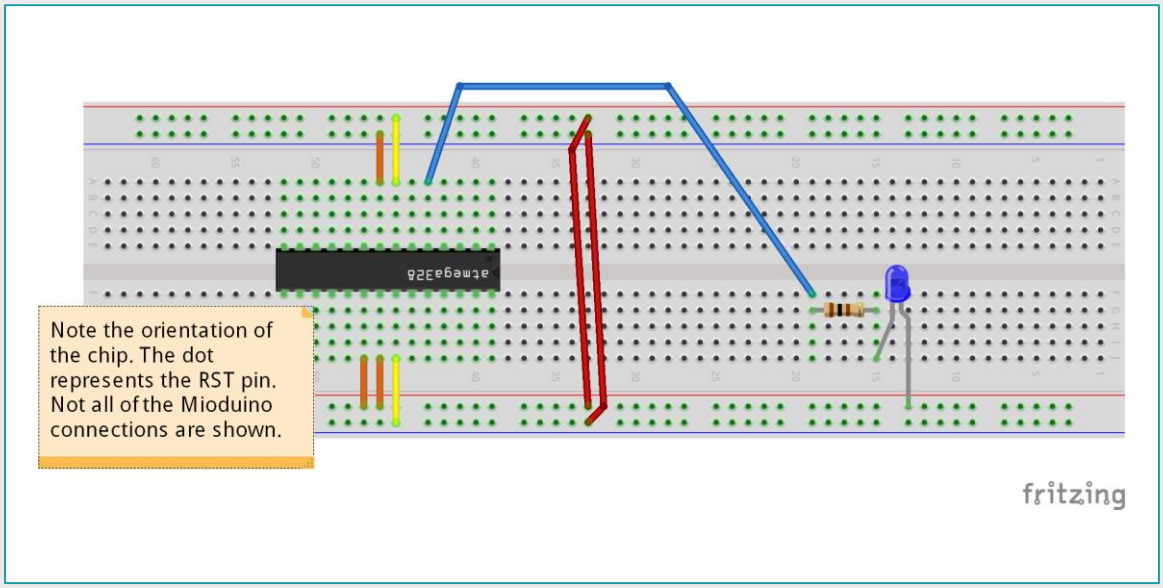
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Connect a flexible jumper wire between B43 to F19.	One end of this wire is connected to pin D3 on the Mioduino. The other end will be connected to the resistor in the next step.
Step 2		Place the Resistor between G15 and G19.	One end of the Resistor is connected to the wire from previous step. The other end will connect to the LED in the next step.
Step 3		Place the anode (long lead) of the blue LED in 5V and the cathode (short lead) into J15.	The short lead (cathode) of the LED is connected to the resistor from the previous step. The long lead (anode) is connected to 5V (+ rail).

Run the Sketch

1. Open the Arduino IDE.
2. Load the sketch for **Lab 3**.

3. Select the board to use from the **TOOLS/BOARD** menu and choose **MIODUINO**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The *GREEN* LED will flash while the code is loading.
10. When loading is complete, the *BLUE* LED should flash on and off every second.



*The actual measurements on the LED in this circuit using a **100 Ω** resistor yielded a forward voltage of **3.087 V** and a forward current of **18.49 mA** measured with a* FLUKE 87 TRUE MULTIMETER*. Using an **82 Ω** resistor yields a forward voltage of **3.129 V** and a current of **21.56 mA**.*

Lab 4: LED Brightness

Objective

We learned it is important to bias our LED correctly, but what if we want to change the brightness? Well, there is a technique called Pulse-Width Modulation that allows us to do exactly that.

In its simplest form, pulse-width modulation is a pulse (a high voltage followed by a low voltage) where the amount of time the voltage is high and the amount of time the voltage is low is varied. The pulse occurs at a high enough frequency that your eye does not see that the LED is really turning on and off.

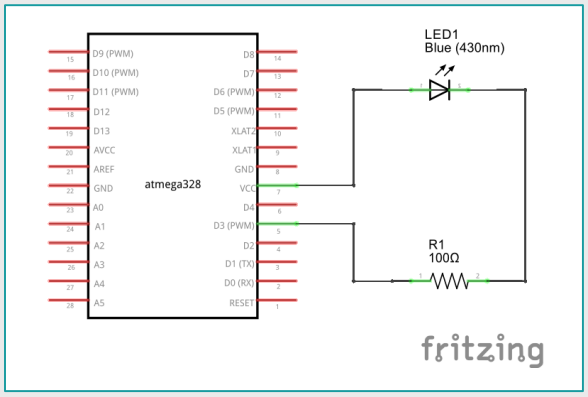
To use pulse-width modulation on the Arduino, the `analogWrite` function is used on a pin. The pulse frequency on most Arduino's is 490 Hz which means the length of the pulse is approximately 200 µS. When the `analogWrite` function is called with the value 0, the voltage of the entire pulse is **LOW** (0 volts) and the LED is always off (assuming the non-pin end of the LED is connected to ground). When the `analogWrite` function is called with a value of 255, the voltage of the entire pulse is **HIGH** (5V) and the LED is always on. Any value between 1 and 254 cause a portion of the pulse to be HIGH and a portion of the pulse to be LOW. When a large portion of the pulse is HIGH, your eye sees the LED as being brighter. When the large portion of the pulse is LOW, your eye sees the LED as dimmer. Therefore, adjusting the amount of time the pulse is HIGH and LOW varies the apparent brightness or intensity of the LED. The Arduino function `analogWrite` makes all of this easy for you!

Prepare for the Lab

Find the necessary components and set them aside.

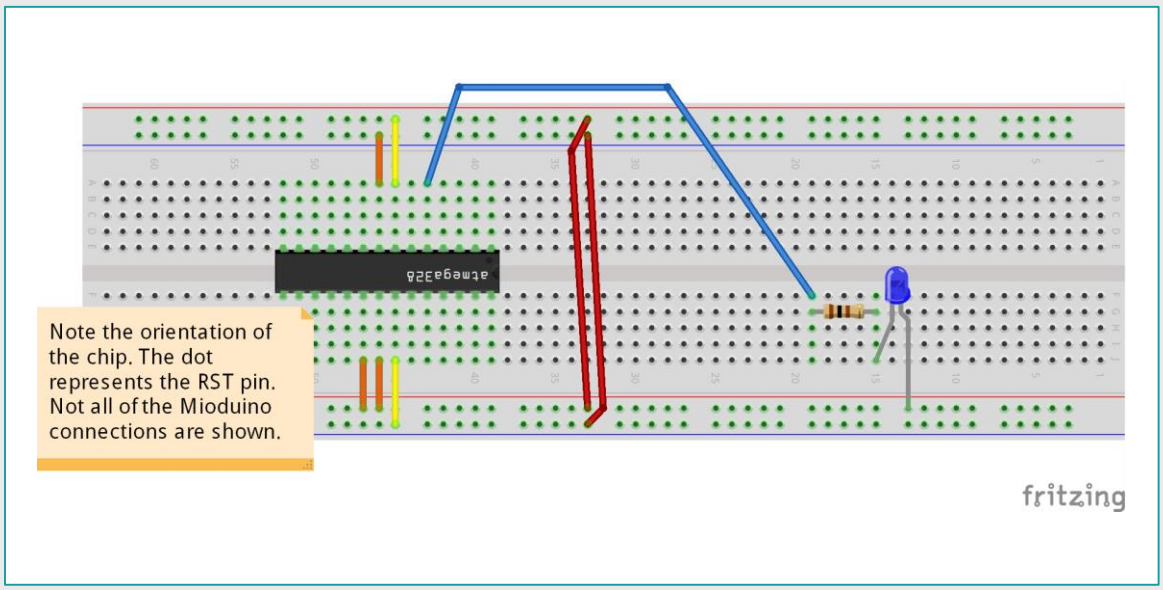
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference

STEP 1		Connect a flexible jumper wire between B43 to F19.	One end of this wire is connected to pin D3 on the Mioduino. The other end will be connected to the resistor in the next step.
STEP 2		Place the 100 Ω RESISTOR between G15 and G19.	One end of the 100 Ω RESISTOR is connected to the wire from previous step. The other end will connect to the LED in the next step.
STEP 3		Place the anode (long lead) of the blue LED in 5V and the cathode (short lead) into J15.	The short lead (cathode) of the LED is connected to the resistor from the previous step. The long lead (anode) is connected to 5V (+ rail).

RUN THE SKETCH

1. Open the Arduino IDE.
2. Load the sketch for **LAB 4**.
3. Select the board to use from the **TOOLS/BOARD** menu and choose **MIODUINO**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The *GREEN* LED will flash while the code is loading.
10. When loading is complete, the *BLUE* LED will fade on and off (this effect is called breathing and is often used to indicate a sleep mode or inactivity).

Lab 5: Push Button Monitor

Objective

So far we have used ports for digital and analog output but now we'll take a look at using a port for digital input.

When reading a digital port, the value read will either be **HIGH** or **LOW**. When connecting a device to a port we want to ensure that the state of the input pin is not **Floating** (neither HIGH nor LOW) because the Arduino `digitalRead` function cannot distinguish this state.

In this lab, we are going to use what is called a *PULL-UP RESISTOR* with a push button switch. This pull-up resistor connects the Arduino input pin to 5V using a resistor with a high resistance value to keep the pin HIGH and the current low. Selecting a high value resistor prevents the circuit from drawing a large amount of current.



When designing circuits, it is important to know the current draw of the circuit. If we are designing something that runs on a battery, we want to ensure long battery life.

The LEDs on our Mioduino use 1K Ω resistors. If you apply the formula from Lab 3, you probably would not have chosen these values. So why use 1K Ω resistors? Since neither of these LEDs are critical to the function of the Mioduino, we do not want them consuming a lot of current, especially the red LED which is on all of the time!

When pushed, the button in our circuit will connect the ground pin to the same Arduino input pin, thus changing the value read on the pin from HIGH to LOW.

When the button is released, the pin returns to its HIGH state due to the pull-up resistor.



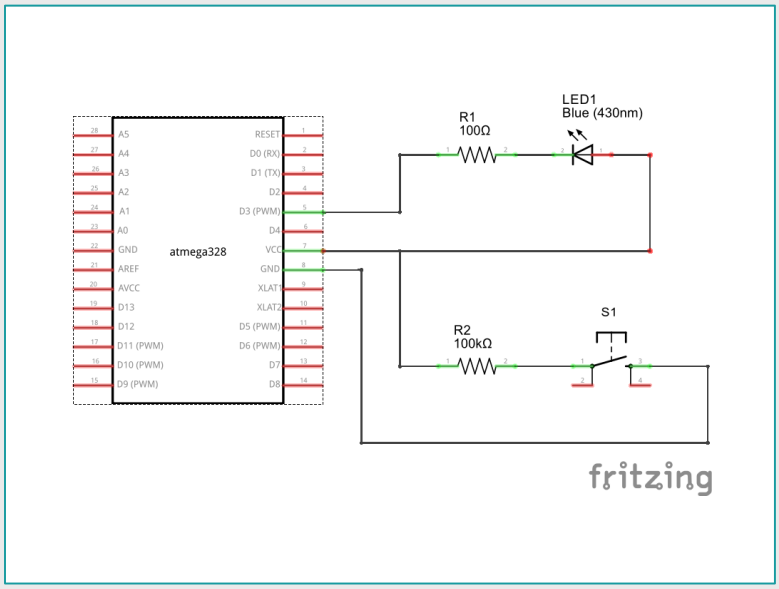
The Arduino has built-in pull-up resistors that can be enabled by specifying the INPUT_PULLUP mode in the pinMode function, but we will not use these resistors to gain first-hand experience.

Prepare for the Lab

Find the necessary components and set them aside.

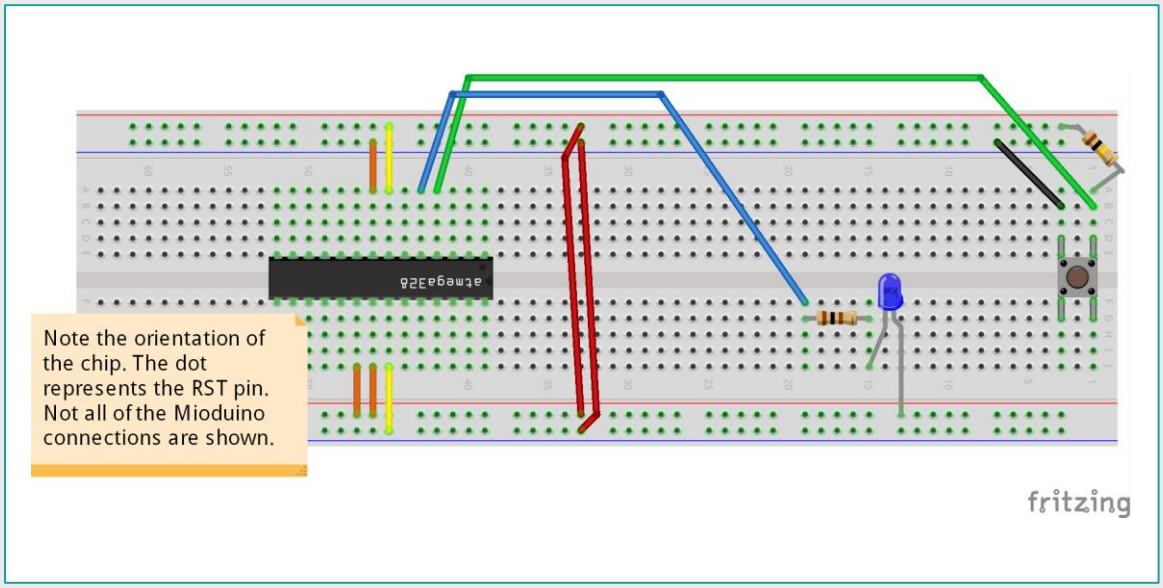
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Connect a flexible jumper wire between B43 to F19.	One end of this wire is connected to pin D3 on the Mioduino. The other end will be connected to the resistor in the next step.
Step 2		Place the 100 Ω Resistor between G15 and G19.	One end of the 100 Ω resistor is connected to the wire from previous step. The other end will connect to the LED in the next step.
Step 3		Place the anode (long lead) of the blue LED in 5V and the cathode (short lead) into J15.	The short lead (cathode) of the LED is connected to the resistor from the previous step. The long lead (anode) is connected to 5V (+ rail).
Step 4		Place the Push Button Switch into the board at G1, G3, D1 and D3.	Place the Push Button Switch into the breadboard so that it crosses over the center of the breadboard.
Step 5		Connect a flexible jumper between B42 and B1.	One end of the wire connects to the Push Button Switch while the other end is connected to D2.

STEP 6		Connect a 100K Ω RESISTOR between A1 and 5V.	Connect one end of the 10K Ω RESISTOR to the same switch pin as the previous step. Connect the other end to 5V.
STEP 7		Connect a flexible jumper between B3 and GROUND.	Connect a flexible jumper between a second terminal of the PUSH BUTTON SWITCH on the same side of the breadboard) and GROUND (- rail).

RUN THE SKETCH

1. Open the Arduino IDE.
2. Load the sketch for **LAB 5**.
3. Select the board to use from the **TOOLS/BOARD** menu and choose **MIODUINO**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu, choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The *GREEN* LED will flash while the code is loading.
10. Push the button and the *BLUE* LED will light. The sketch monitors the button and turns the LED on when the button is pushed and off when it is released.

Lab 6: Night Light

Objective

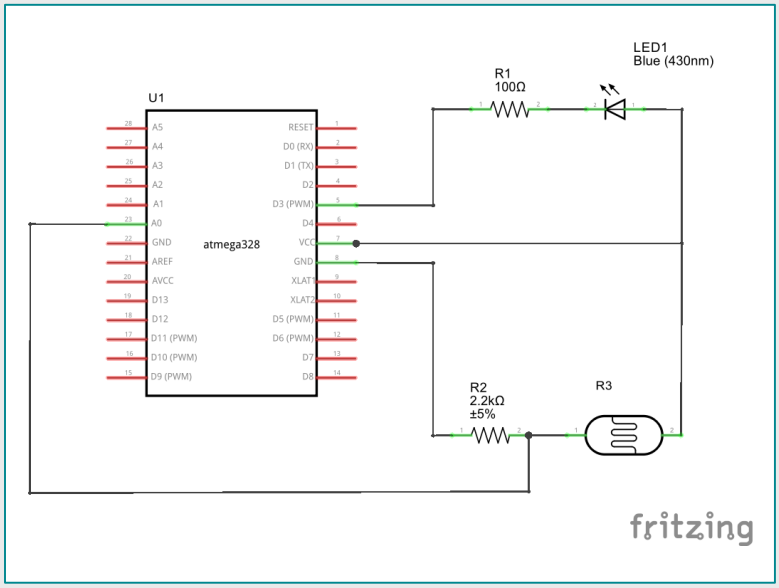
Create a night light from an LED and light sensor (photoresistor). When light detected by the sensor drops below a certain level, the LED will turn on.

Prepare for the Lab

Find the necessary components and set them aside.

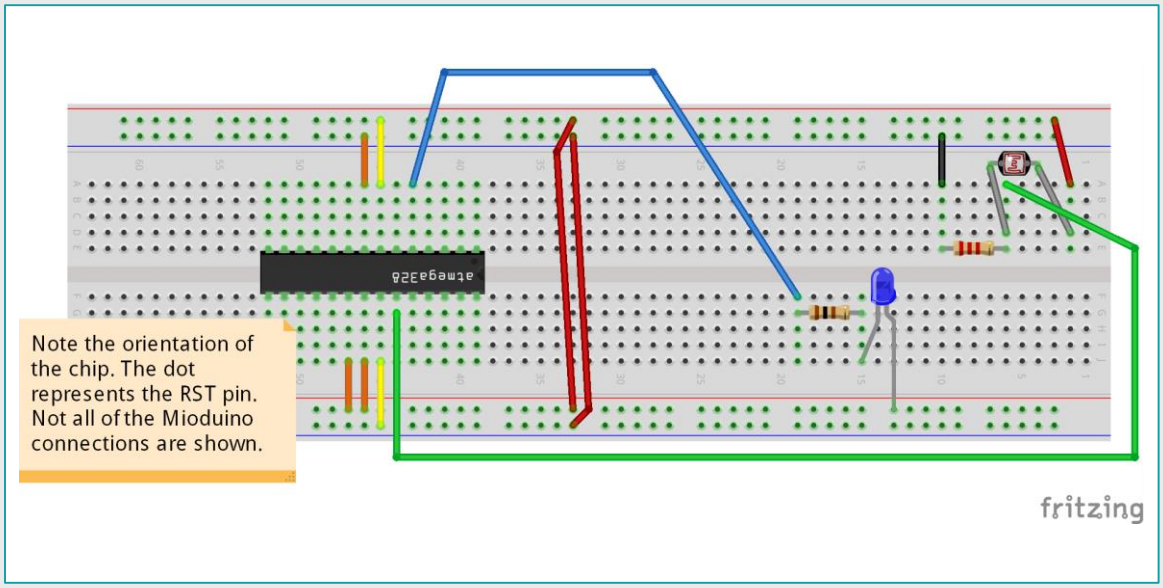
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Connect a flexible jumper wire between B43 to F19.	One end of this wire is connected to pin D3 on the Mioduino. The other end will be connected to the resistor in the next step.
Step 2		Place the 100 Ω Resistor between G15 and G19.	One end of the 100 Ω resistor is connected to the wire from previous step. The other end will connect to the LED in the next step.
Step 3		Place the anode (long lead) of the blue LED in 5V and the cathode (short lead) into J15.	The short lead (cathode) of the LED is connected to the resistor from the previous step. The long lead (anode) is connected to 5V (+ rail).
Step 4		Connect the Photoresistor between D2 and D6.	Place the Photoresistor on the breadboard.
Step 5		Connect a 2.2K Ω Resistor between E6 and E10.	Connect one end of the 2.2K Ω Resistor to one end of the Photoresistor.

Step 6		Connect a (black) flexible jumper wire from A10 to Ground.	Connect the other end of the 2.2K Ω Resistor to Ground (- rail).
Step 7		Connect a (red) flexible jumper wire from A2 to 5V.	Connect the other end of the Photoresistor to 5V (+ rail).
Step 8		Connect a flexible jumper wire from A6 to G44.	Connect the point where the 2.2K Ω Resistor and the Photoresistor are connected to pin A0.

Run the Sketch – Part A

1. Open the Arduino IDE.
2. Load the sketch called **Lab 6A**.
3. Select the board to use from the **Tools/Board** menu and choose **Mioduino**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **Serial TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **Tools/Port** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **Serial Monitor** from the **Tools** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **Upload** from the **Sketch** menu.
9. The *GREEN* LED will flash while the code is loading.
10. If there is NOT enough light on the sensor, the LED will turn on. Cover the sensor or shine some light on it to turn the LED on and off.

Run the Sketch – Part B

11. Load the sketch called **Lab 6B**.

12. Vary the light over the sensor. This version of the code will increase and decrease the intensity of the LED based on the amount of light detected on the sensor. As light on the sensor increases, the intensity of the LED decreases. As it gets darker, the LED will get brighter.

Lab 7: Ohm's Law

Objective

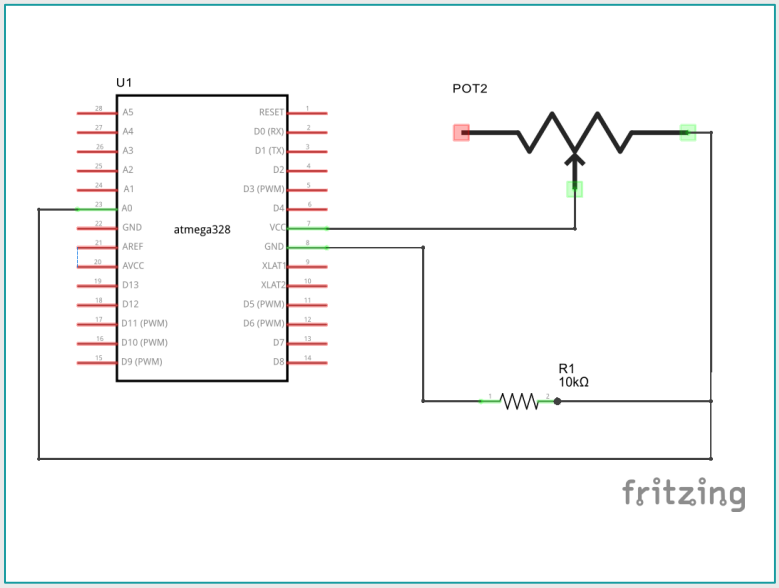
Using a Voltage Divider with a potentiometer as R2, measure the voltage on an analog port, and determine the resistance value of the potentiometer

Prepare for the Lab

Find the necessary components and set them aside.

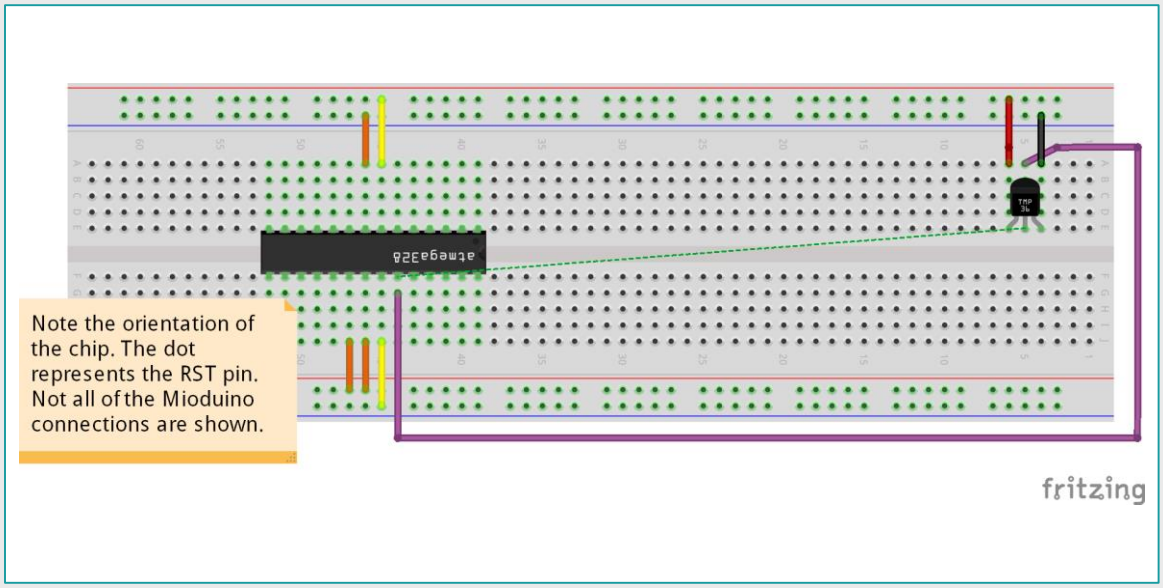
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Place the 10K Ω Breadboard Trim Potentiometer into the breadboard at E5, E6 and E7. Place the component so that the bulk of the potentiometer is covering the center gap in the breadboard.	Place the 10K Ω Breadboard Trim Potentiometer into the breadboard.
Step 2		Place a 10K Ω Resistor between B7 and B11.	Connect one end of a 10K Ω Resistor to one of the outer pins of the potentiometer.
Step 3		Place a (red) flexible jumper wire between A6 and 5V.	Connect the center pin of the potentiometer to 5V (+ rail).
Step 4		Place a (black) flexible jumper wire between A11 and Ground.	Connect the other end of the 10K Ω Resistor to Ground (- rail).

STEP 5		Place a flexible jumper between A7 and G44.	Connect the point where the resistor and the potentiometer are connected to pin A0.

RUN THE SKETCH

1. Open the Arduino IDE.
2. Load the sketch for **LAB 7**.
3. Select the board to use from the **TOOLS/BOARD** menu and choose **MIODUINO**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. *OPTIONAL:* The sketch has definitions (`#define` statements) for `R1`, `R2_MAX` and `V_REF` that are used in the equation to calculate the resistance value of the potentiometer. These values are close to the actual ones; however, you can use a meter to measure them and enter exact values in your sketch to get a more accurate calculation.
10. The *GREEN* LED will flash while the code is loading.
11. Depending on the current position of the potentiometer, the *BLUE* LED may turn on. Turning the potentiometer will update the calculation and send the results to the **SERIAL MONITOR**. It will also adjust the brightness of the LED using PWM via the `analogWrite` function (turning the potentiometer fully counter-clockwise will turn the blue LED off).

Lab 8: Analog Sensor

Objective

This lab demonstrates how to read an analog sensor. In this case, we will be reading temperature from a TMP36 which is a solid state sensor that sends the temperature information through an analog signal. The code will read the signal from an analog port and translate it to the correct temperature value.

The formula for this sensor based on the manufacturers specifications) is:

$$Temp\,°C = 100 * (reading\ in\ V) - 50$$

when the analog port is reading using `analogRead` function, a value from 0 to 1023 is returned. The value of *0* represents 0V and the value *1023* represents the maximum voltage, or the reference voltage (V_{ref}) which is applied to pin *21* on the Atmega38P. The Moduino has this pin connected to *5V*.

Calculating the voltage read by the port is done using this equation:

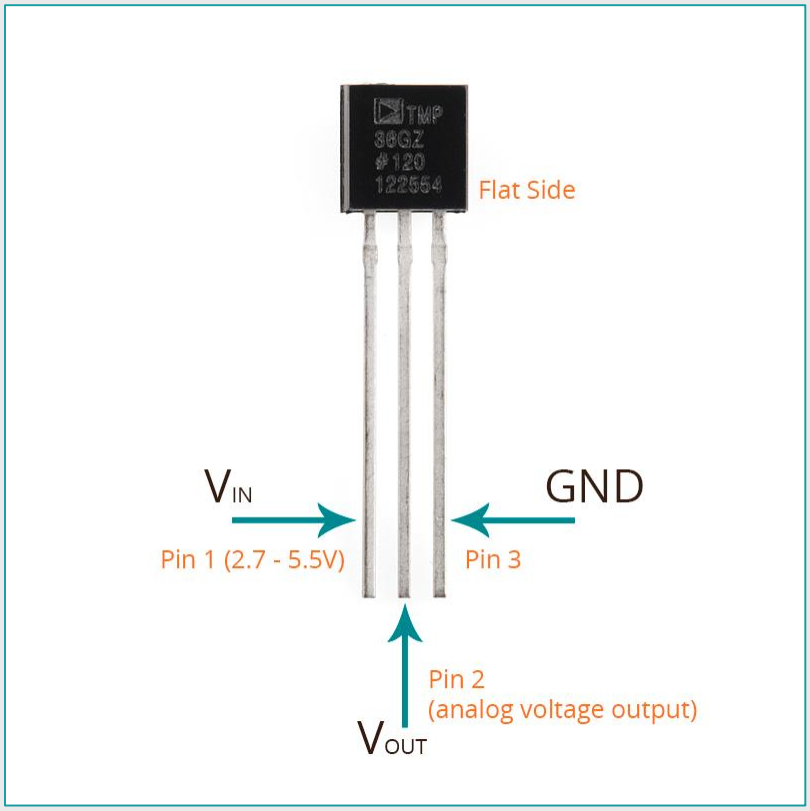
$$V_n = (A_n \div 1023) \times V_{ref}$$

Where V_n is the voltage on a given analog port, A_n is the value read from that port and V_{ref} is the reference voltage.

Prepare for the Lab

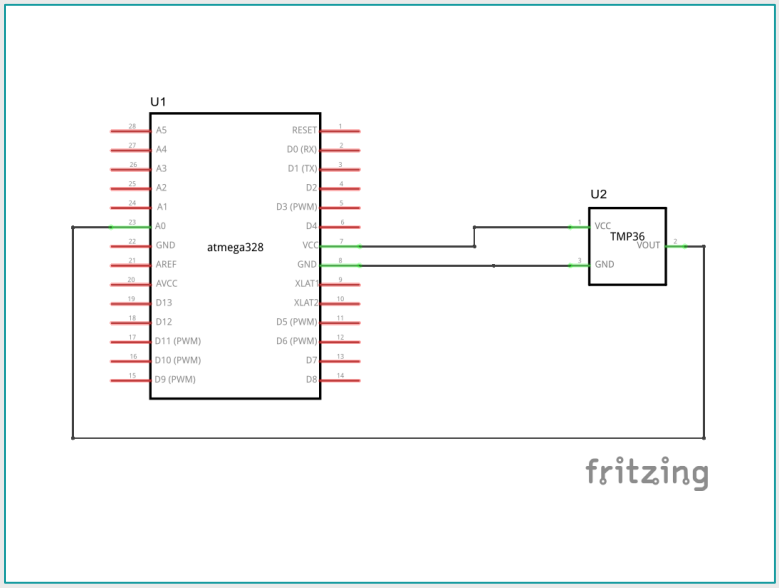
Find the necessary components and set them aside.

Note the pinout for the TMP36 sensor as shown below. Use this image as a reference when building the circuit.



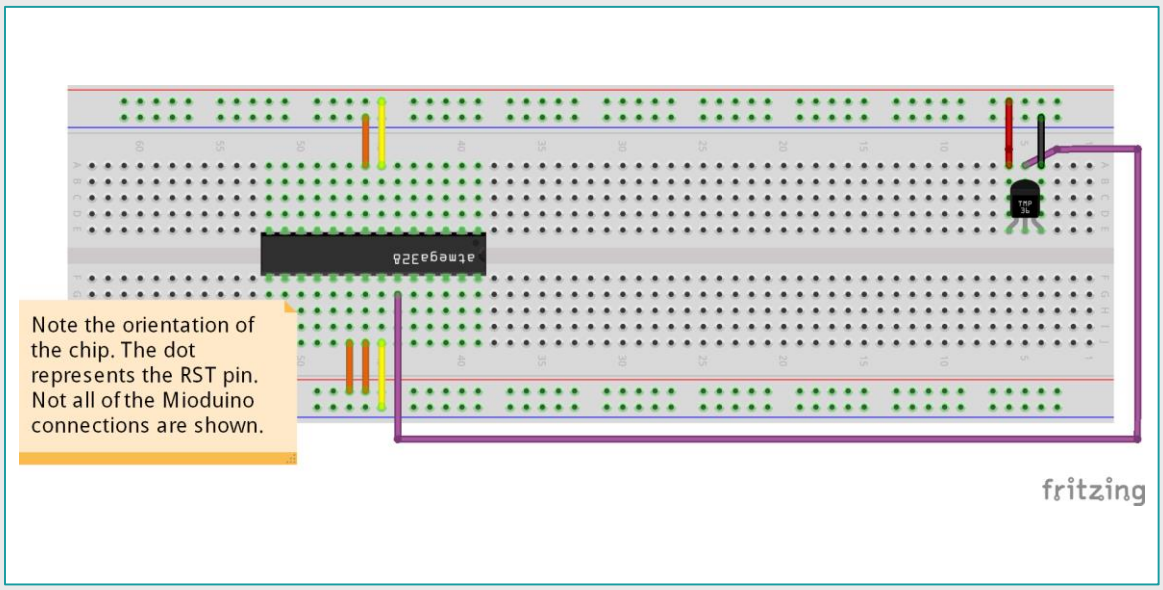
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Place the TMP36 sensor into the breadboard with the center lead at E5 and the flat side of the sensor facing the center of the breadboard. The *VOLTAGE SUPPLY* pin of the TMP36 will be in position E6 and the *GROUND* pin of the TMP36 will be in position E4.	Place the TMP36 in the breadboard.
Step 2		Connect a (black) flexible jumper from A4 to Ground on the breadboard.	Connect the GND pin of the TMP36 to Ground (- rail).

STEP 3		Connect a (red) flexible jumper from A6 to 5V on the breadboard.	Connect the V_{IN} pin of the TMP36 to 5V (+ rail).
STEP 4		Connect a flexible jumper from A5 to G44.	Connect the V_{OUT} pin of the TMP36 to pin A0.

RUN THE SKETCH

1. Open the Arduino IDE.
2. Load the sketch for **LAB 8**.
3. Select the board to use from the **TOOLS/BOARD** menu and choose **MIODUINO**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The *GREEN* LED will flash while the code is loading.
10. Watch the **SERIAL MONITOR** to view the reading from sensor. The code will take a reading every two seconds.

Lab 9: Digital Sensor 1

Objective

In this lab we will use a Reflective IR Sensor to detect reflective materials over the sensor. This sensor consists of two elements built into a single plastic housing. The first element is an IR LED and the second element an IR Phototransistor. When powered, the IR LED will emit infrared light that can be detected by the IR Phototransistor. This type of device is useful to detect if something has passed by the sensor.

This sensor is considered a digital sensor simply due to the way the circuit is designed and how the code will interact with it. It will be setup in a manner that will give us a HIGH or LOW pulse indicative of whether or not something is in the vicinity of the sensor (~ 5 mm; range is 2 – 10 mm).

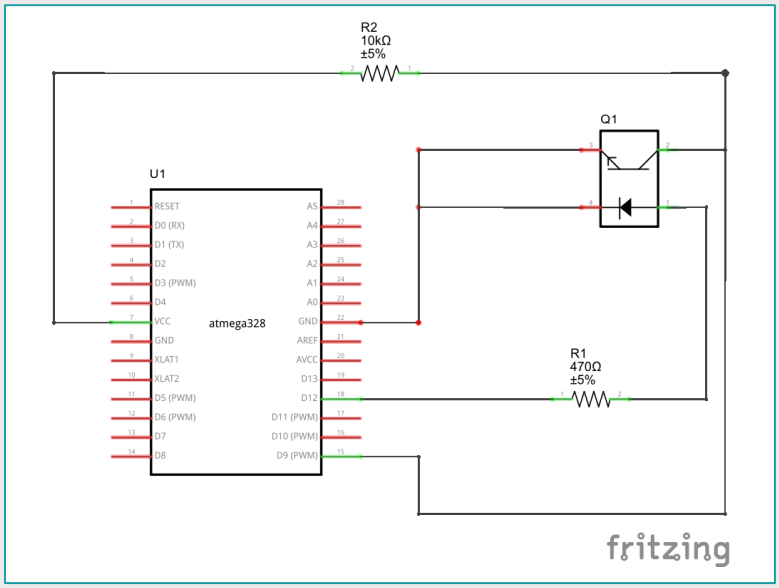
Note that this sensor can also be wired as an analog sensor in which the amount of reflected light can be detected.

Prepare for the Lab

Find the necessary components and set them aside.

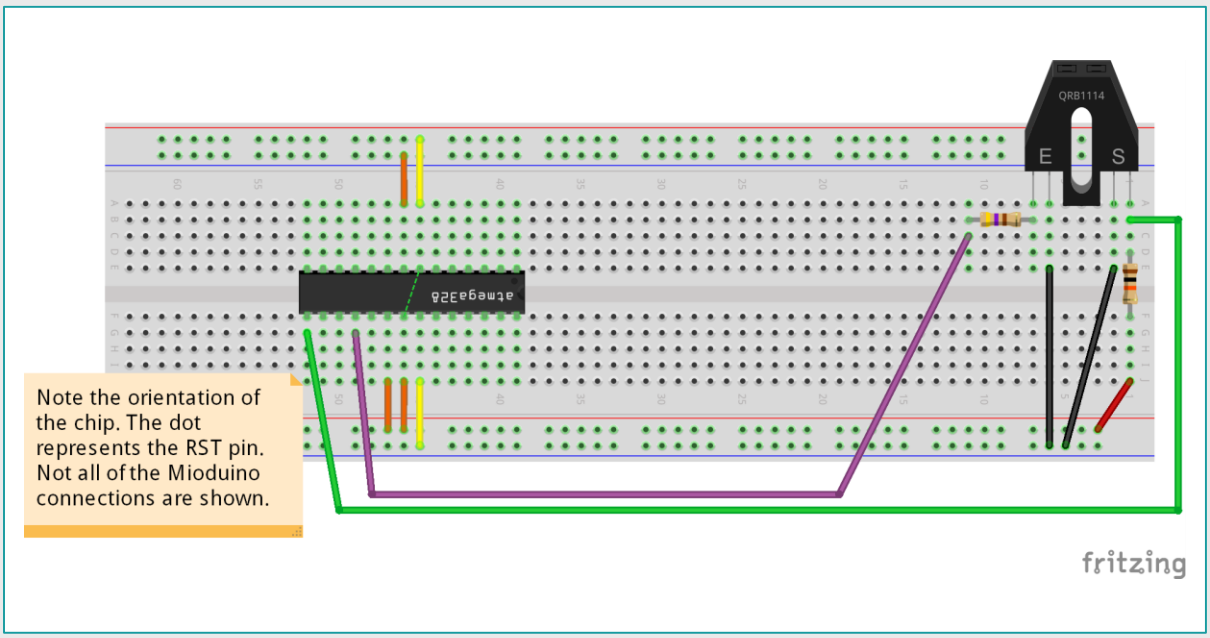
Schematic Diagram

The schematic for the circuit in this lab is shown below.



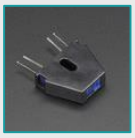
Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Place the Reflective IR Sensor onto the breadboard at A1, A2, A6 and A7 and the lettering facing the center of the breadboard.	Place the Reflective IR Sensor on the breadboard.
Step 2		Connect a (black) flexible jumper from E6 to Ground.	Connect the GND (cathode) pin of the Emitter side to Ground (- rail).
Step 3		Connect a (black) flexible jumper from E2 to Ground.	Connect the GND pin of the Sensor side to Ground (- rail).
Step 4		Place a 470 Ω Resistor between B7 and B11.	Connect a 470 Ω Resistor to the anode of the Emitter side.
Step 5		Connect a flexible jumper from C11 to G49.	Connect the other end of the 470 Ω Resistor to D12.
Step 6		Place a 10K Ω Resistor between D1 and F1.	Connect a 10K Ω Resistor to the second pin on the Emitter side.
Step 7		Connect a (red) flexible jumper from J1 to 5V.	Connect the other end of the 10K Ω Resistor to 5V (+ rail).
Step 8		Connect a flexible jumper from B1 to G52.	Connect the second pin on the Emitter side (the one connected to the 10K Ω Resistor) to pin D9.

Run the Sketch

1. Open the Arduino IDE.
2. Load the sketch for **Lab 9**.
3. Select the board to use from the **Tools/Board** menu and choose **Mioduino**.

4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **SERIAL TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **TOOLS/PORT** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **SERIAL MONITOR** from the **TOOLS** menu.
8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The *GREEN* LED will flash while the code is loading.
10. Hold a piece of white paper over the sensor about 5 mm from the top. Move the piece of paper back and forth and watch the **SERIAL MONITOR**. The green built-in LED will light up whenever the sensor detects a reflective material.

Lab 10: Digital Sensor 2

Objective

This lab will demonstrate how to connect a complex sensor that communicates over the I2C bus to retrieve the sensor data. The data are exchanged by sending various commands to the sensor and then reading the response.

The sensor used in this lab is the Adafruit MCP9808 High Accuracy I2C Temperature Sensor Breakout Board. This board is built and configured for us in a ready to use fashion.



The I2C bus was designed by Philips in the early '80s to allow easy communication between components that reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006.

The name I2C translates into "Inter IC". Sometimes the bus is called IIC or I^2C bus.

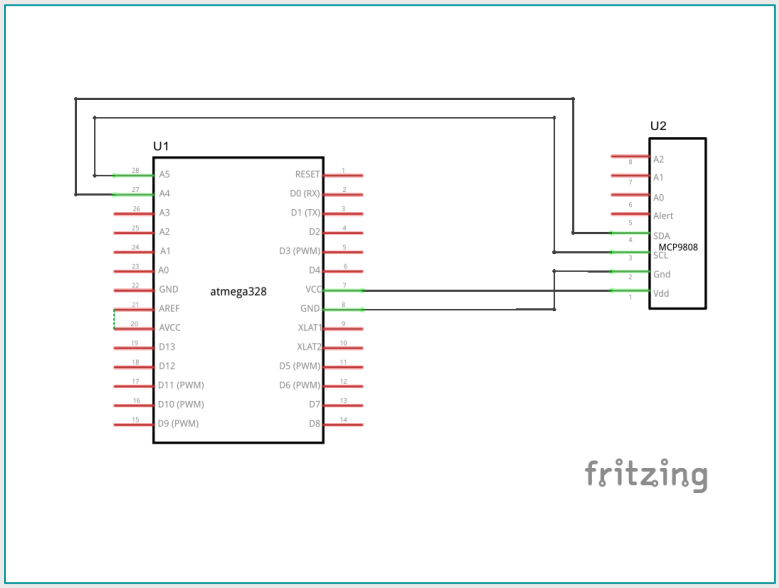
http://www.i2c-bus.org/

Prepare for the Lab

Find the necessary components and set them aside.

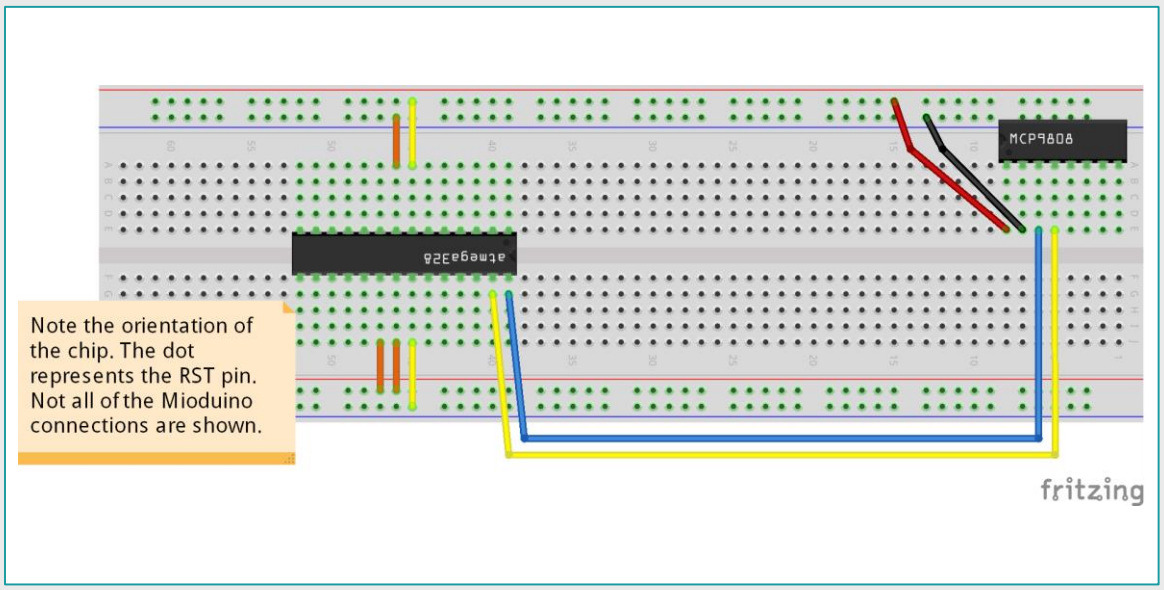
Schematic Diagram

The schematic for the circuit in this lab is shown below.



Breadboard Diagram

The diagram below is a rendered model of the breadboard for this lab. Use this as a guide while building your own circuit.



Build the Circuit

Follow the steps below to build the circuit for this lab.

Step	Image Reference	Breadboard Reference	Component Reference
Step 1		Place the MCP9808 Sensor onto the breadboard at A1 through A8 with the pin labeled V_{DD} in the A8 position.	Place the MCP9808 onto the breadboard.
Step 2		Connect a (red) flexible jumper from E8 to 5V.	Connect the V_{DD} pin of the MCP9808 to 5V (+ rail).
Step 3		Connect a (black) flexible jumper from E7 to Ground.	Connect the GND pin of the MCP9808 to Ground (- rail).
Step 4		Connect a flexible jumper from E6 to G39.	Connect the SCL pin of the MCP9808 to pin A5.
Step 5		Connect a flexible jumper from E5 to G40.	Connect the SDA pin of the MCP9808 to pin A4.

Run the Sketch

1. Open the Arduino IDE.
2. Load the sketch for **Lab 10**.
3. Select the board to use from the **Tools/Board** menu and choose **Mioduino**.
4. Disconnect the DC power supply if connected.
5. Connect your Mioduino to the computer using the **Serial TTL** cable. Place the black connector into the breadboard using the 6 pin header (the end of the connector with the black wire should be at the edge of the breadboard). Plug the **USB** end of the cable into your computer.
6. In the **Tools/Port** menu choose the correct COM port for your cable (in most cases you will only have one COM port listed).
7. Open the serial port monitor by choosing **Serial Monitor** from the **Tools** menu.

8. Upload the sketch by clicking the toolbar button with the arrow or by choosing **UPLOAD** from the **SKETCH** menu.
9. The green LED will flash while the code is loading.
10. Watch the **SERIAL MONITOR** for the output of this sensor. The sensor is read every two seconds and the results are displayed.

Additional Resources

The following are great resources for further reading. All of these are available on Amazon (available for Kindle) or on Adafruit.

Programming Arduino Next Steps: Getting Started with Sketches by Simon Monk

Using clear, easy-to-follow examples, *Programming Arduino: Getting Started with Sketches* reveals the software side of Arduino and explains how to write well-crafted sketches using the modified C language of Arduino. No prior programming experience is required! The downloadable sample programs featured in the book can be used as is or modified to suit your purposes.

- Understand Arduino hardware fundamentals.
- Install the software, power it up, and upload your first sketch.
- Learn C language basics.
- Write functions in Arduino sketches.
- Structure data using arrays and strings.
- Use Arduino's digital and analog inputs and outputs in your programs.
- Work with the Standard Arduino Library.
- Write sketches that can store data.
- Program LCD displays.
- Use an Ethernet shield to enable Arduino to function as a web server.
- Write your own Arduino libraries.

Programming Arduino Next Steps: Going Further with Sketches by Simon Monk

Take your Arduino skills to the next level!

In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Featuring coverage of the Arduino Uno, Leonardo, and Due boards, *Programming Arduino Next Steps: Going*

Further with Sketches shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download.

Learn advanced Arduino programming techniques, including how to:

- Use hardware and timer interrupts
- Boost performance and speed by writing time-efficient sketches
- Minimize power consumption and memory usage
- Interface with different types of serial busses, including I2C, 1-Wire, SPI, and TTL Serial
- Use Arduino with USB, including the keyboard and mouse emulation features of the Leonardo and Due boards
- Program Arduino for the Internet
- Perform digital signal processing
- Accomplish more than one task at a time—without multi-threading
- Create and release your own code library

Practical Electronics for Inventors, Third Edition by PAUL SCHERZ and SIMON MONK

Spark your creativity and gain the electronics skills required to transform your innovative ideas into functioning gadgets. This hands-on, updated guide outlines electrical principles and provides thorough, easy-to-follow instructions, schematics, and illustrations. Find out how to select components, safely assemble circuits, perform error tests, and build plug-and-play prototypes. Practical Electronics for Inventors, Third Edition, features all-new chapters on sensors, microcontrollers, modular electronics, and the latest software tools.

Exploring Arduino: Tools and Techniques for Engineering Wizardry by JEREMY BLUM

Written by Arduino expert Jeremy Blum, this unique book uses the popular Arduino microcontroller platform as an instrument to teach you about topics in

electrical engineering, programming, and human-computer interaction. Whether you're a budding hobbyist or an engineer, you'll benefit from the perfectly paced lessons that walk you through useful, artistic, and educational exercises that gradually get more advanced. In addition to specific projects, the book shares best practices in programming and design that you can apply to your own projects. Code snippets and schematics will serve as a useful reference for future projects even after you've mastered all the topics in the book.

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes

My Notes