THE MOST COMPLETE STARTER KIT TUTORIAL FOR MEGA 2560

V1.0.19.07.15

Preface

Our Company

Established in 2011, Elegoo Inc. is a thriving technology company dedicated to research & development, production, and marketing of open-source hardware. Located in Shenzhen, the Silicon Valley of China, we have grown to over 150+ employees with a 10,763+ square ft.factory.

Our product lines include DuPont wires, UNO R3 boards and complete starter kits designed for customers of any level to learn Arduino knowledge. In addition, we also sell Raspberry Pi accessories like TFT touch screens. Additionally, we plan to expand our offerings to include other technologies, including products related to 3D printing. All of our products comply with international quality standards and have been praised by our customers in a variety of different marketplaces throughout the world.

Official website:

http://www.elegoo.com

US Amazon storefront: http://www.amazon.com/shops/A2WWHQ25ENKVJ1
CA Amazon storefront: http://www.amazon.ca/shops/A2WWHQ25ENKVJ1
UK Amazon storefront: http://www.amazon.co.uk/shops/AZF7WYXU5ZANW
DE Amazon storefront: http://www.amazon.de/shops/AZF7WYXU5ZANW
FR Amazon storefront: http://www.amazon.fr/shops/AZF7WYXU5ZANW
ES Amazon storefront: http://www.amazon.es/shops/AZF7WYXU5ZANW
IT Amazon storefront: http://www.amazon.it/shops/AZF7WYXU5ZANW

offer you excellent products and quality service as to meet your expectation and you can reach out to us by simply drop a line at service@elegoo.com or EUservice@elegoo.com. We look forward to hearing from you and any of your critical comment or suggestion would be much valuable tous.

And any of problems and questions you have with our products will be promptly replied by our experienced engineers within 12 hours (24hrs during holiday)

Packing List

Red LED
5PCS

Yellow LED
5PCS

Blue LED
5PCS

Green LED
5PCS

White LED
5PCS

RGB LED
1PC

Diode Rectifier
1PC

Thermistor
1PC

NPN Transistor
(PN2222)
1PC

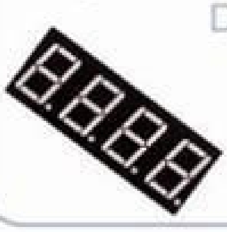
Tilt Ball Switch
1PC

74HC595 IC
1PC

Button(Small)
5PCS

Active Buzzer
1PC

Passive Buzzer
1PC

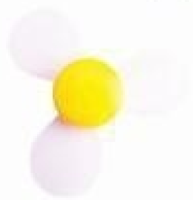

4 digit 7 - segment
Display
1PC

Photoresistor (photocell) 1PC

Resistor 120PCS

9V Battery with DC 1PC

5V Relay 1PC



Fan blade 1PC

3-6VMotor (with wire) 1PC

LCD 1602 Module (with pin header) 1PC

Mega 2560 Controller Board 1PC

9V Battery 1PC

Power Supply Module 1PC

USB Cable 1PC

Breadboard 1PC

Breadboard Jumper Wire 65PCS

F-M Dupont Wire 10PCS

Potentionmeter (10K) 2PCS

Content

Lesson 0 Installing IDE

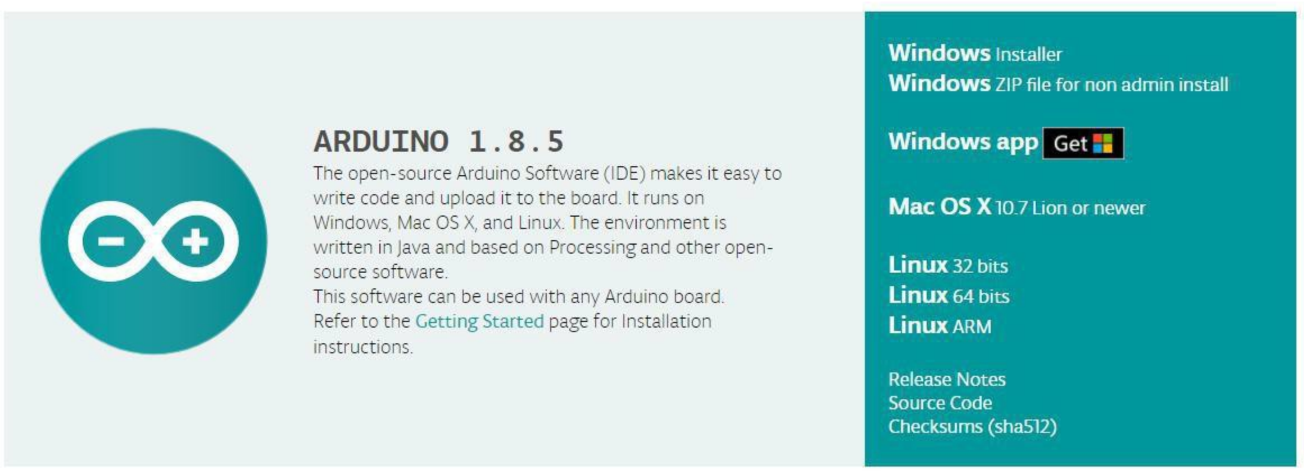
Introduction

The Arduino Integrated Development Environment (IDE) is the software side ofthe Arduino platform.

In this lesson, you will learn how to setup your computer to use Arduino and how to set about the lessons thatfollow.

The Arduino software that you will use to program your Arduino is available for Windows, Mac and Linux. The installation process is different for all three platforms and unfortunately there is a certain amount of manual work to install the software.

STEP 1: Go to https://www.arduino.cc/en/Main/Software and find below page.



ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Get

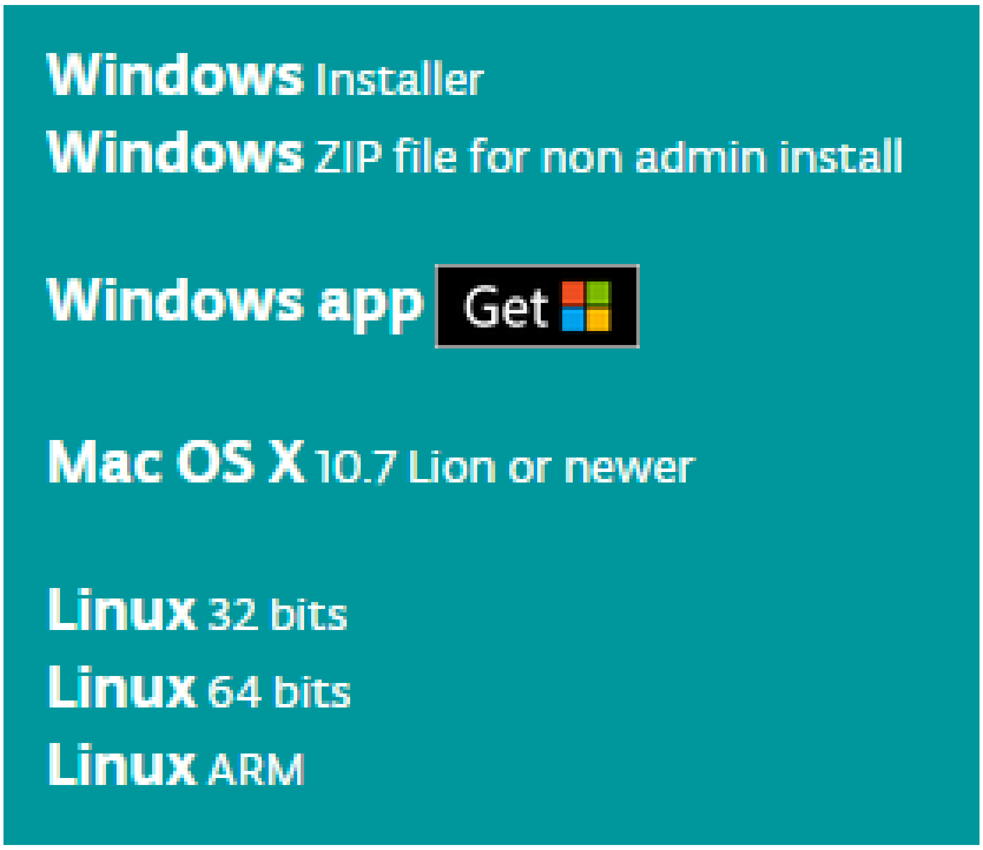
Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

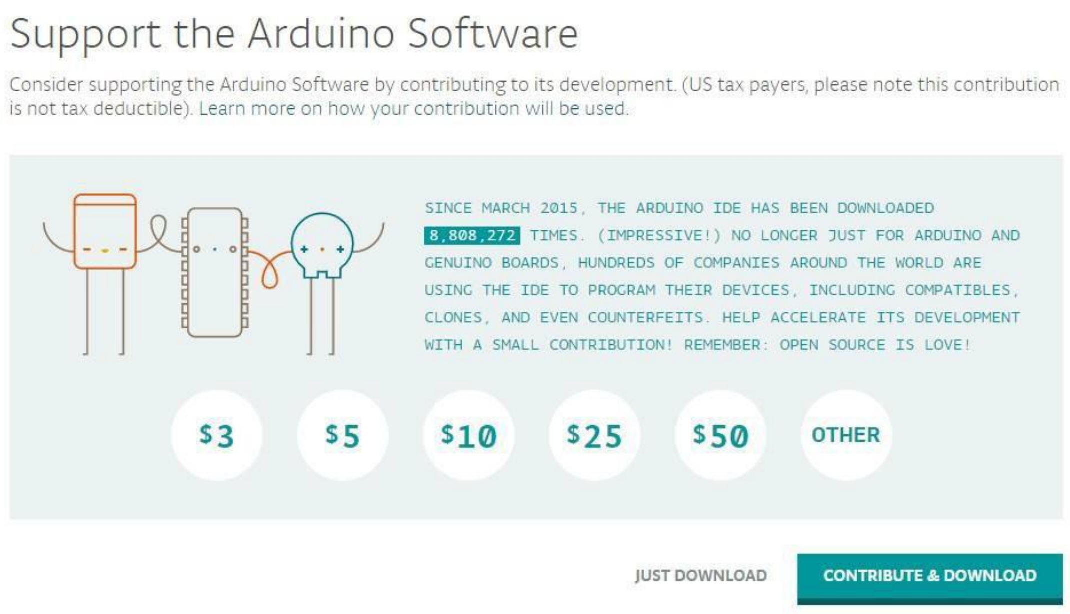
Release Notes
Source Code
Checksums (sha512)

The version available at this website is usually the latest version, and the actual version may be newer than the version in the picture.

STEP2: Download the development software that is compatible with the operating system of your computer. <u>Take Windows as an example here.</u>


Windows Installer
Windows ZIP file for non admin install
Windows app Get
Mac OS X 10.7 Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM

Click *Windows Installer*.


Support the Arduino Software

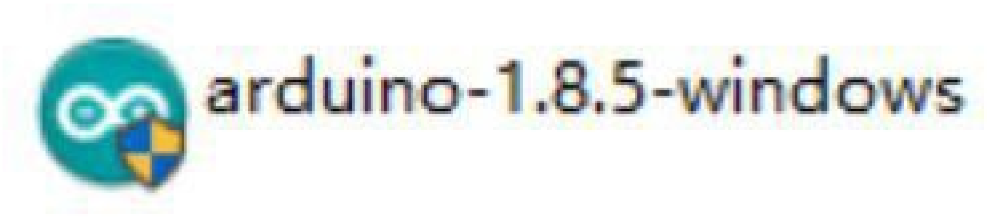
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

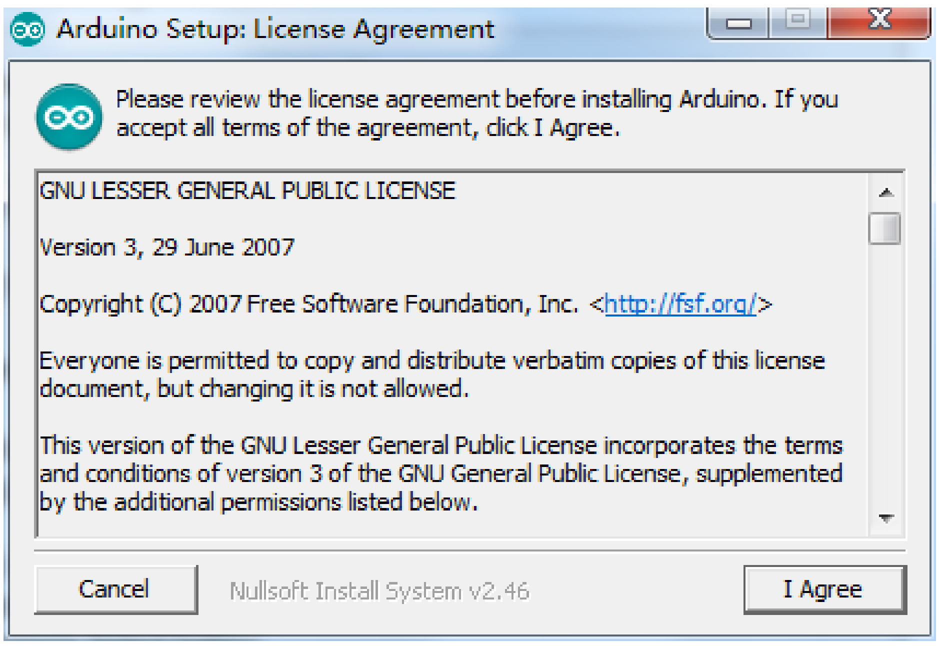
SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 8,808,272 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

Click *JUST DOWNLOAD*.

Installing Arduino (Windows)

Install Arduino with the exe. Installation package.





Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

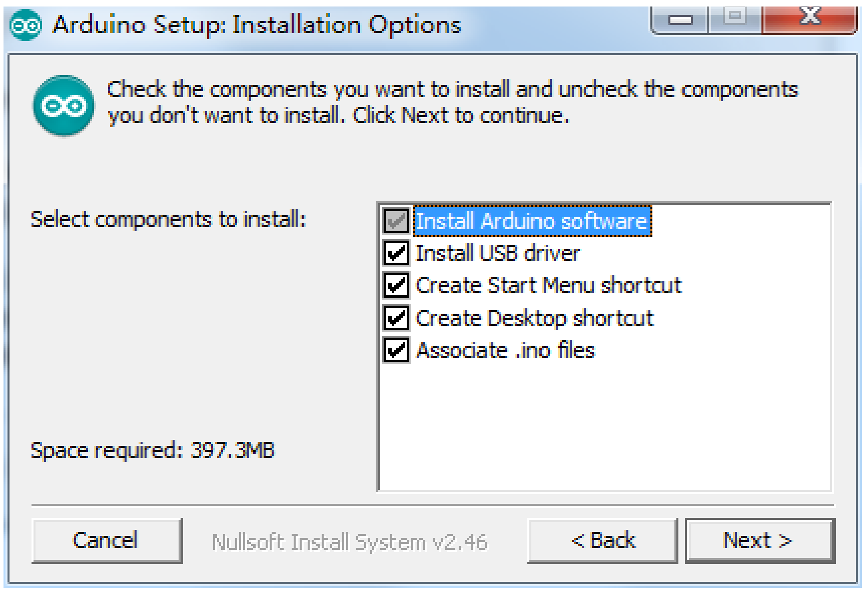
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Click *I Agree* to see the followinginterface


Arduino Setup: Installation Options

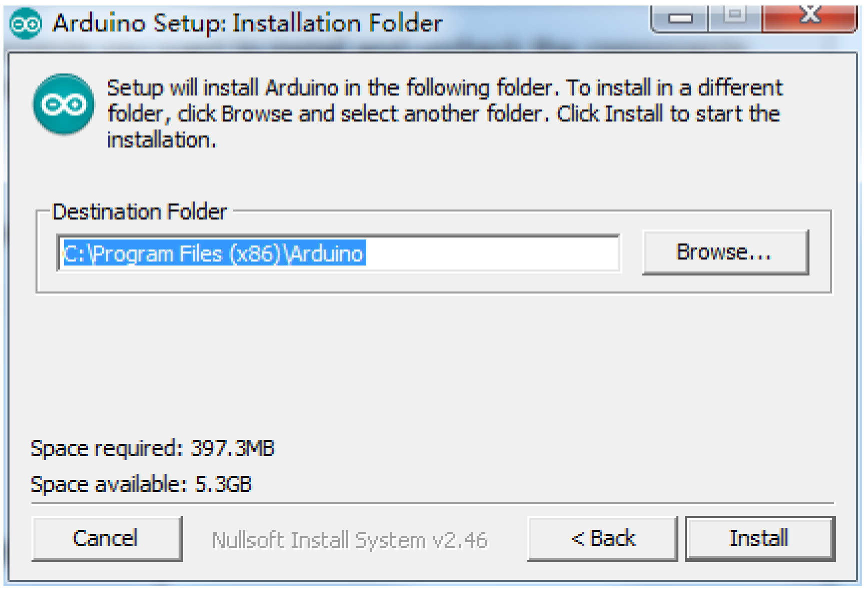
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [x] Install Arduino software
- [x] Install USB driver
- [x] Create Start Menu shortcut
- [x] Create Desktop shortcut
- [x] Associate .ino files

Space required: 397.3MB

Click *Next*


Arduino Setup: Installation Folder

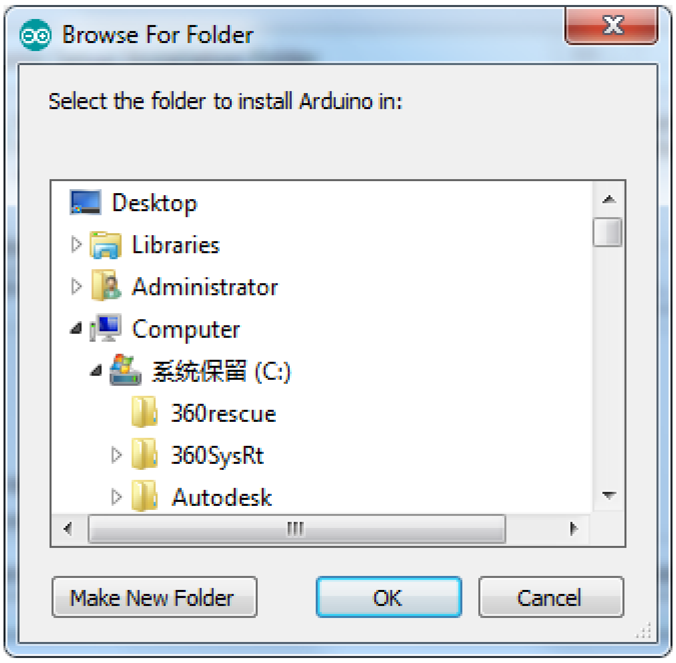
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino

Space required: 397.3MB

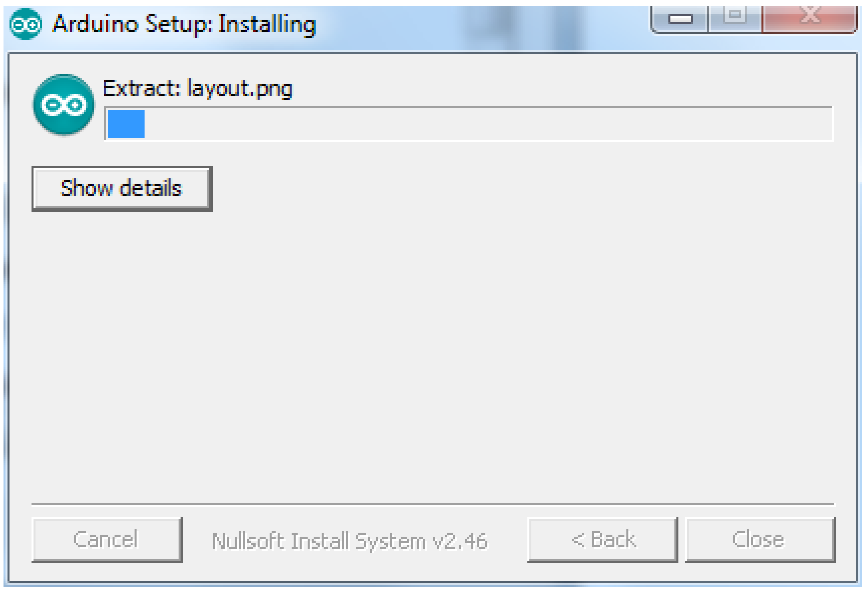
Space available: 5.3GB

You can press Browse... to choose an installation path or directly type in the directory you want.


Browse For Folder

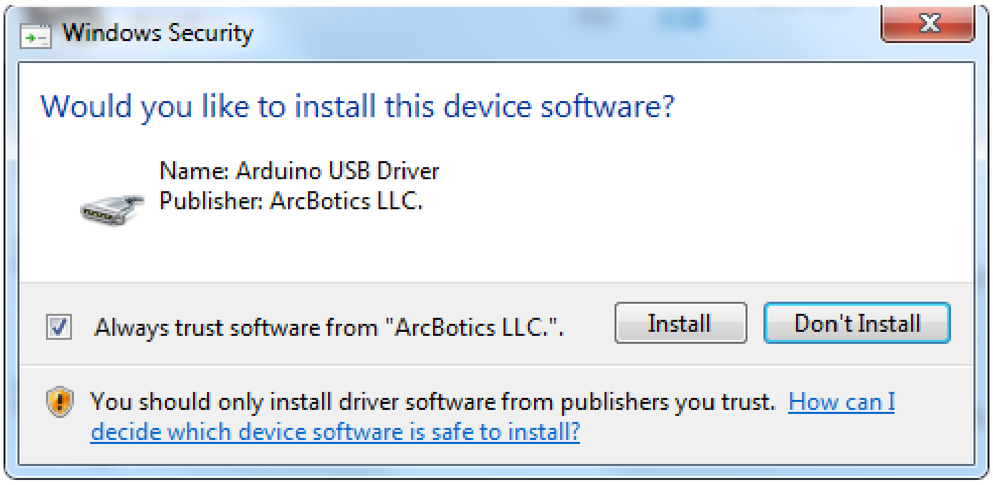
Select the folder to install Arduino in:

Click *Install* to initiate installation


Arduino Setup: Installing

Extract: layout.png

Finally, the following interface appears, click *Install* to finish the installation.


Windows Security

Would you like to install this device software?

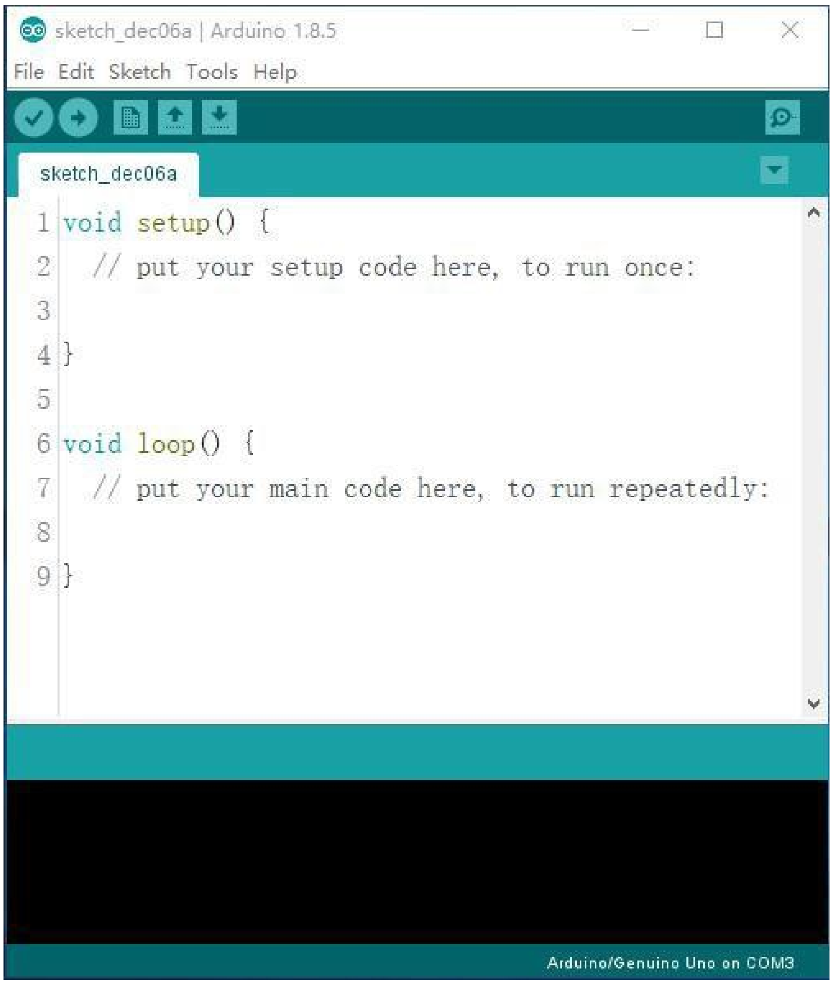
Name: Arduino USB Driver
Publisher: ArcBotics LLC.

☑ Always trust software from "ArcBotics LLC.".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Next, the following icon appears on thedesktop

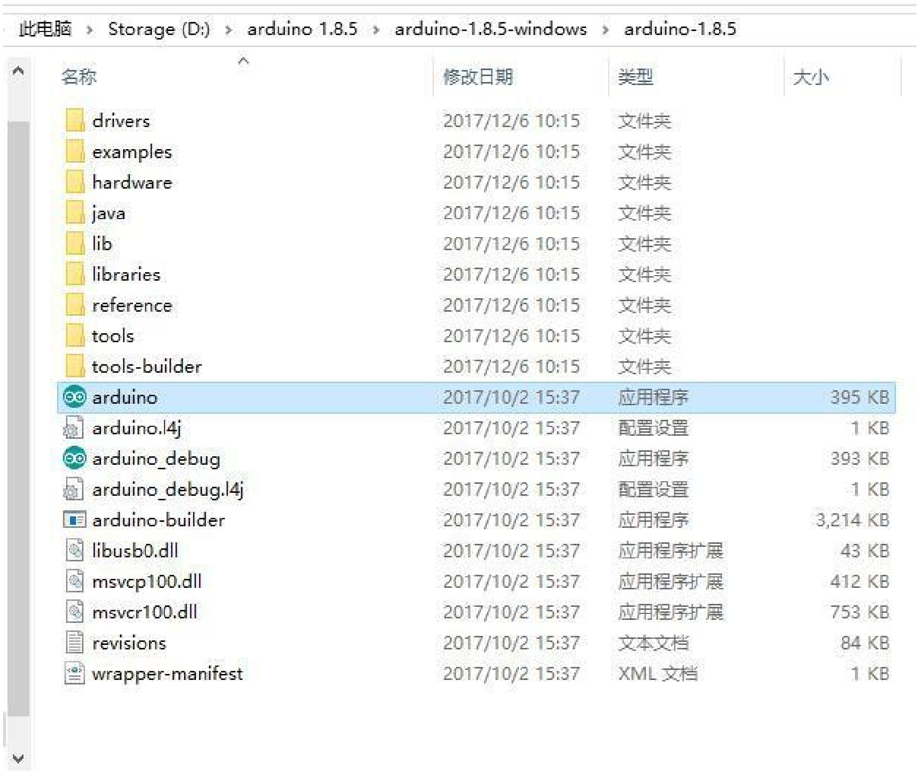
Double-click to enter the desired development environment


sketch_dec06a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_dec06a
Arduino/Genuino Uno on COM3

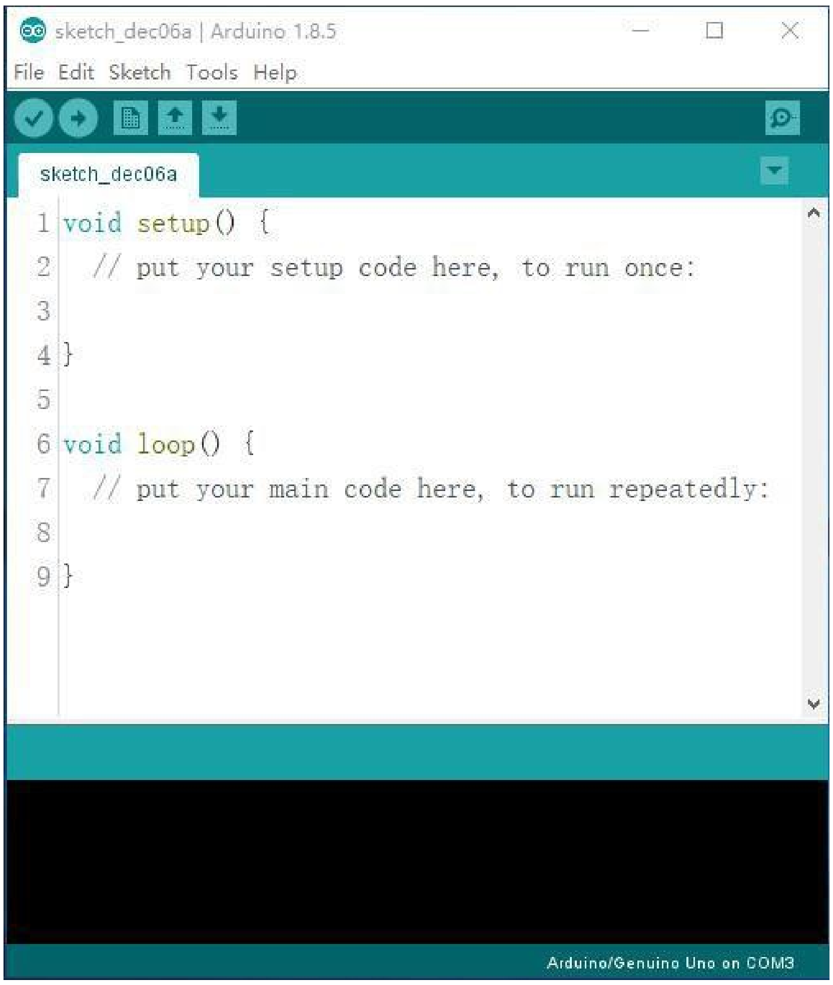
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

You may directly choose the installation package for installation and skip the contents below and jump to the next section. But if you want to learn some methods other than the installation package, please continue to read the section.

Unzip the zip file downloaded, Double-click to open the program and enter the desired development environment


此电脑 › Storage (D:) › arduino 1.8.5 › arduino-1.8.5-windows › arduino-1.8.5

名称	修改日期	类型	大小
drivers	2017/12/6 10:15	文件夹	
examples	2017/12/6 10:15	文件夹	
hardware	2017/12/6 10:15	文件夹	
java	2017/12/6 10:15	文件夹	
lib	2017/12/6 10:15	文件夹	
libraries	2017/12/6 10:15	文件夹	
reference	2017/12/6 10:15	文件夹	
tools	2017/12/6 10:15	文件夹	
tools-builder	2017/12/6 10:15	文件夹	
arduino	2017/10/2 15:37	应用程序	395 KB
arduino.l4j	2017/10/2 15:37	配置设置	1 KB
arduino_debug	2017/10/2 15:37	应用程序	393 KB
arduino_debug.l4j	2017/10/2 15:37	配置设置	1 KB
arduino-builder	2017/10/2 15:37	应用程序	3,214 KB
libusb0.dll	2017/10/2 15:37	应用程序扩展	43 KB
msvcp100.dll	2017/10/2 15:37	应用程序扩展	412 KB
msvcr100.dll	2017/10/2 15:37	应用程序扩展	753 KB
revisions	2017/10/2 15:37	文本文档	84 KB
wrapper-manifest	2017/10/2 15:37	XML 文档	1 KB



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino/Genuino Uno on COM3

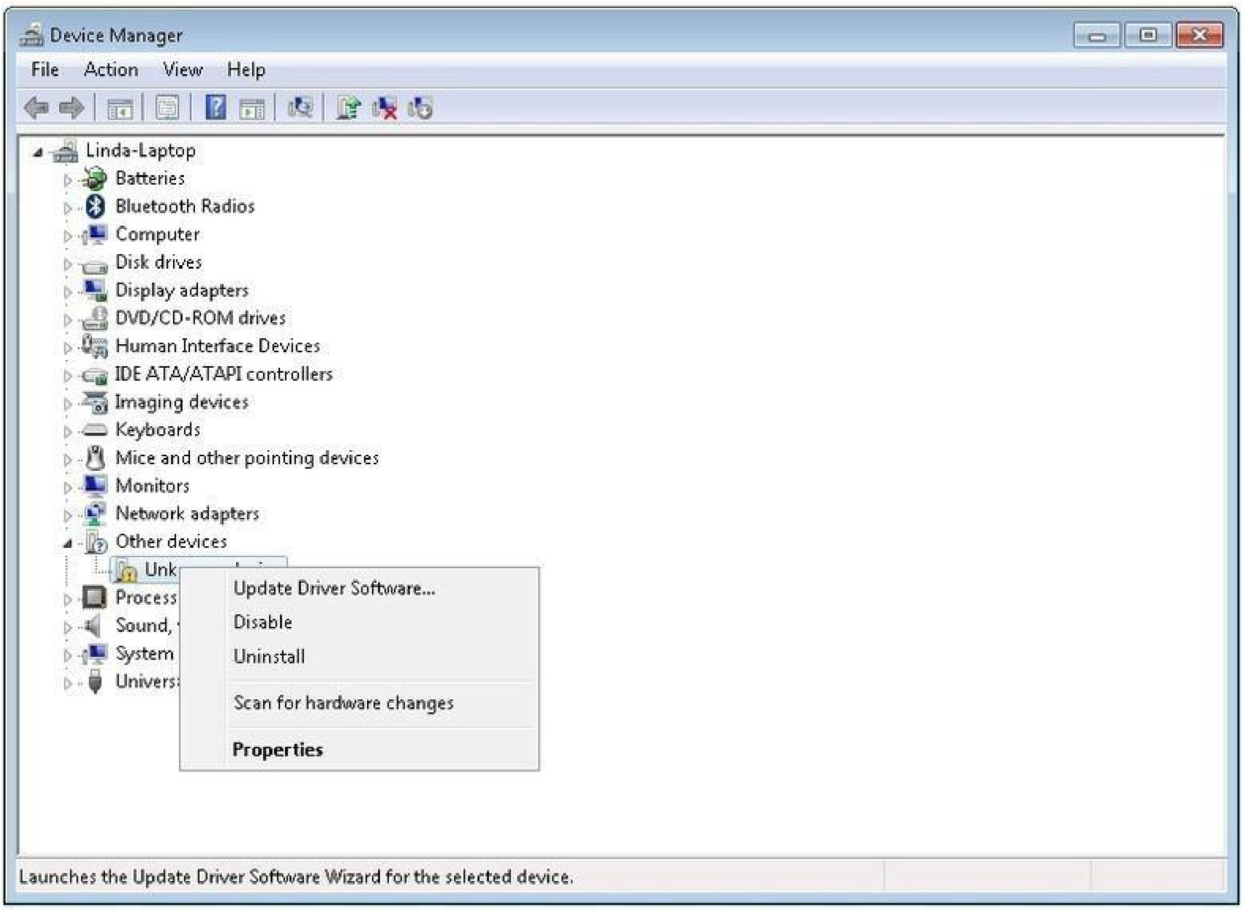
However, this installation method needs separate installation of driver.

The Arduino folder contains both the Arduino program itself and the drivers that allow the Arduino to be connected to your computer by a USB cable. Before we launch the Arduino software, you are going to install the USB drivers.

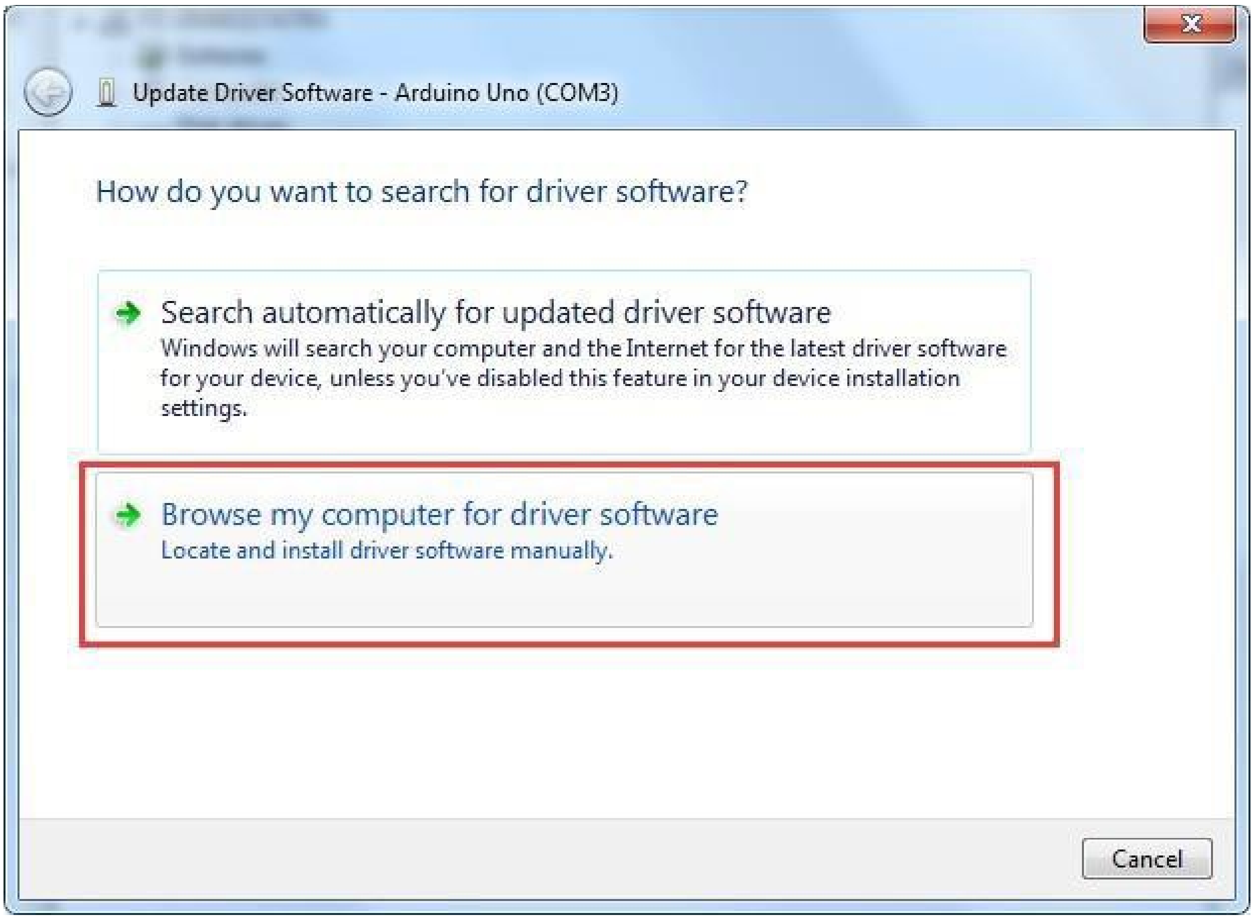
Plug one end of your USB cable into the Arduino and the other into a USB socket on your computer. The power light on the LED will light up and you may get a 'Found New Hardware' message from Windows. Ignore this message and cancel any attempts that Windows makes to try and install drivers automatically for you.

The most reliable method of installing the USB drivers is to use the Device Manager. This is accessed in different ways depending on your version of Windows. In Windows 7, you first have to open the Control Panel, then select the option to view Icons, and you should find the Device Manager in the list.

Under 'Other Devices', you should see an icon for 'unknown device' with a little yellow warning triangle next to it. This is your Arduino.



Launches the Update Driver Software Wizard for the selected device.

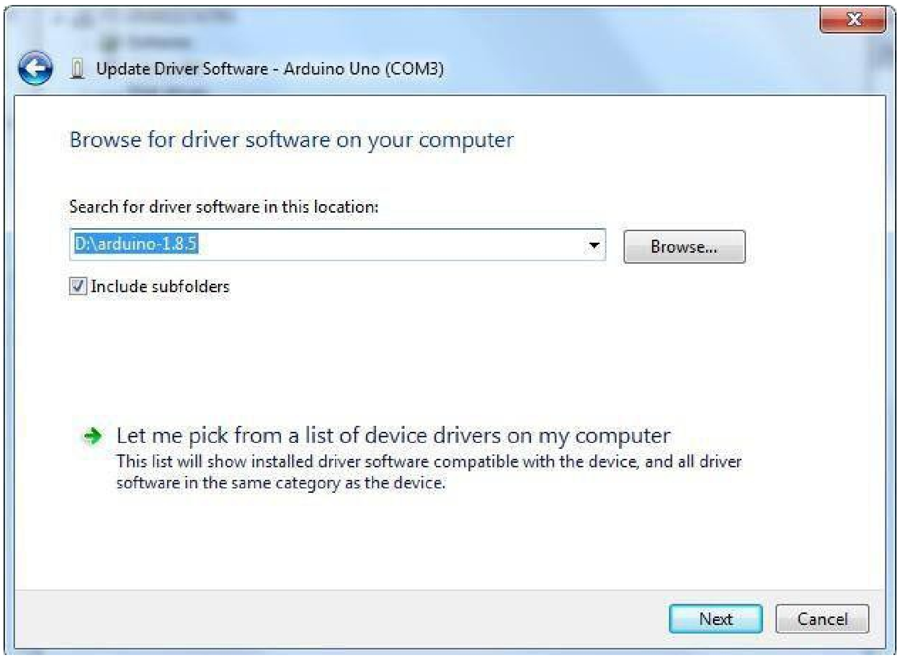

Update Driver Software - Arduino Uno (COM3)

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

Right-click on the device and select the top menu option (Update Driver Software...). You will then be prompted to either 'Search Automatically for updated driver software' or 'Browse my computer for driver software'. Select the option to browse and navigate to theX\arduino1.8.0\drivers.


Update Driver Software - Arduino Uno (COM3)

Browse for driver software on your computer

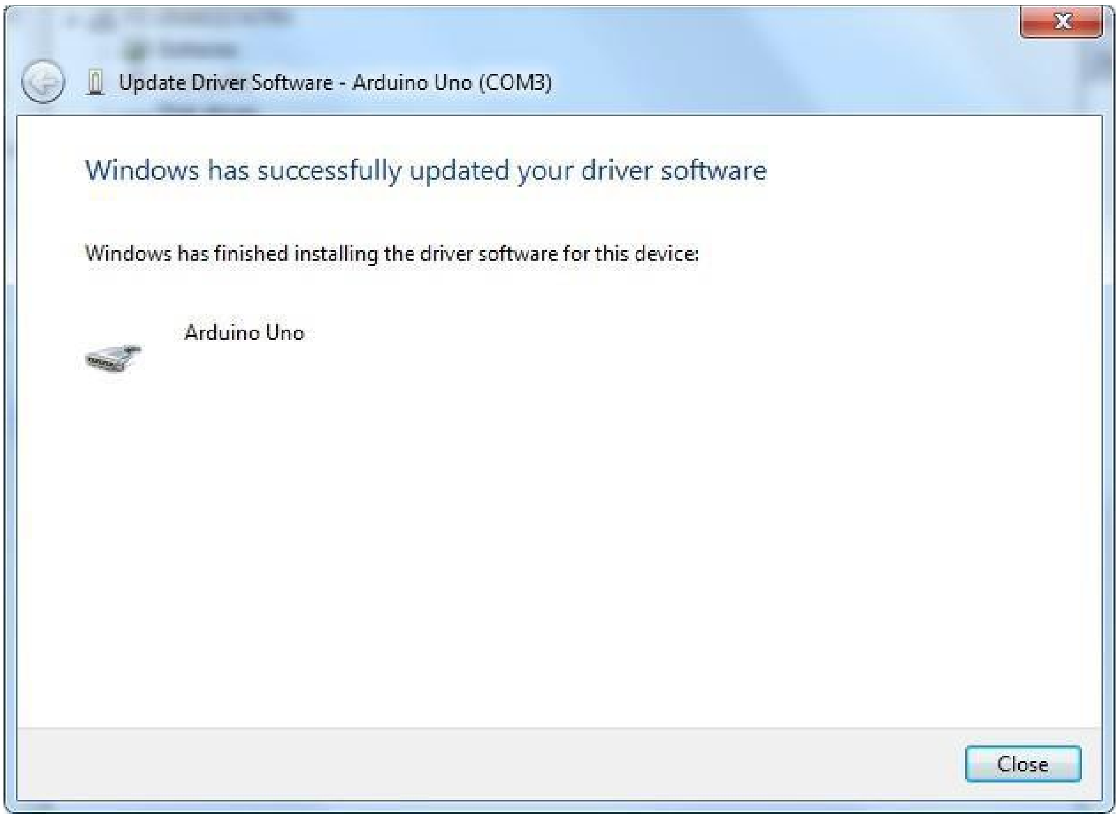
Search for driver software in this location:

D:\arduino-1.8.5

Include subfolders

Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Click 'Next' and you may get a security warning, if so, allow the software to be installed. Once the software has been installed, you will get a confirmation message.


Update Driver Software - Arduino Uno (COM3)

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

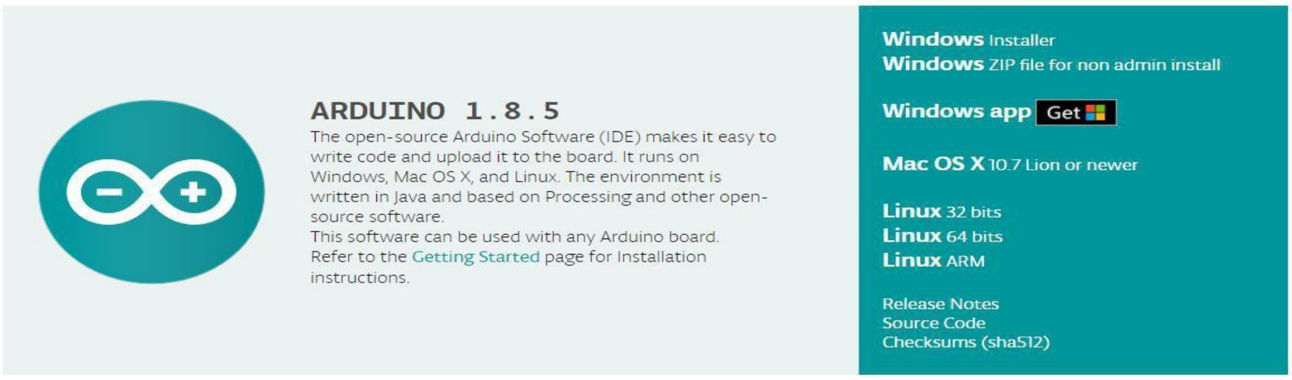
Arduino Uno

Windows users may skip the installation directions for Mac and Linux systems and jump to Lesson 1. Mac and Linux users may continue to read this section.

Installing Arduino (Mac OS X)

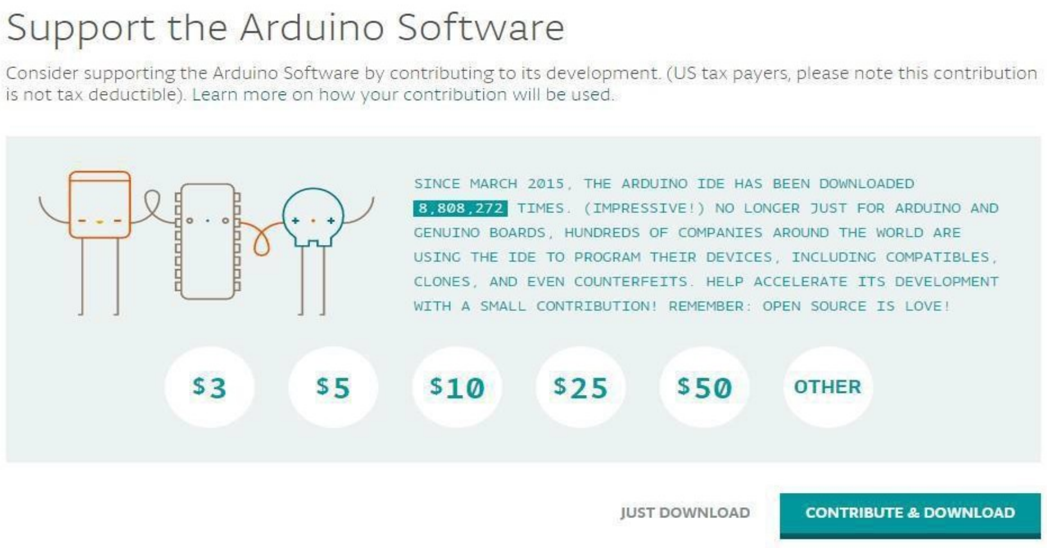
Step One: Download Arduino Software (IDE)

- Open the URL: https://www.arduino.cc/en/Main/Software with browser


ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

- Click Mac OSX 10.7 Lion ornewer



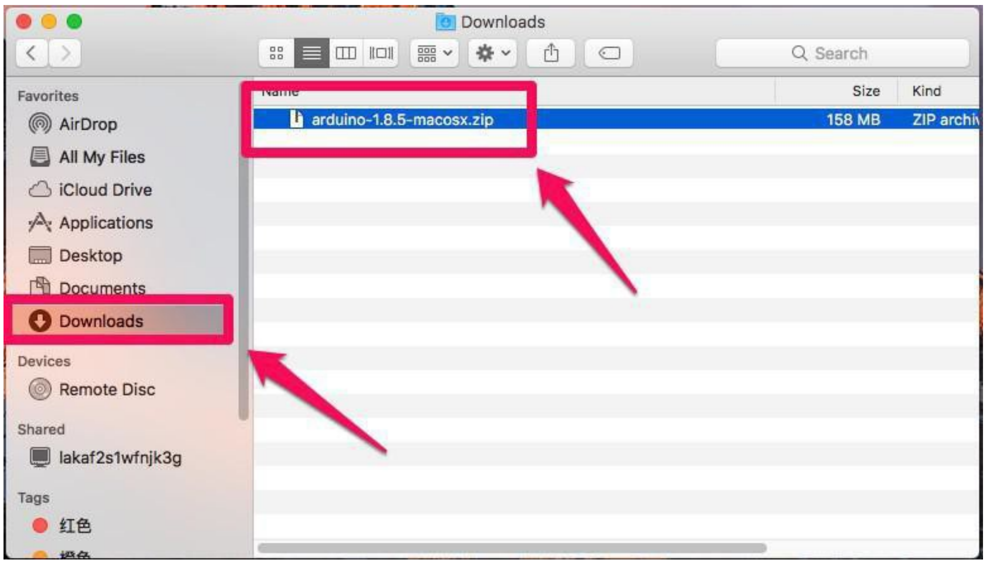
Support the Arduino Software

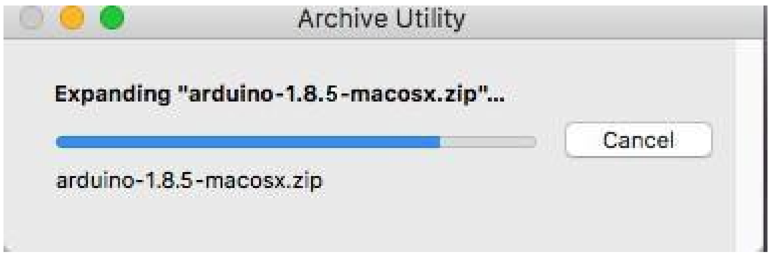
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 8,808,272 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

• Click JUST DOWNLOAD

Download and Unzip the zip file, double click the Arduino. app to enter Arduino IDE; the system will ask you to install Java runtime library if you don’t have it in your computer. Oncethe installation is complete you can run the Arduino IDE.

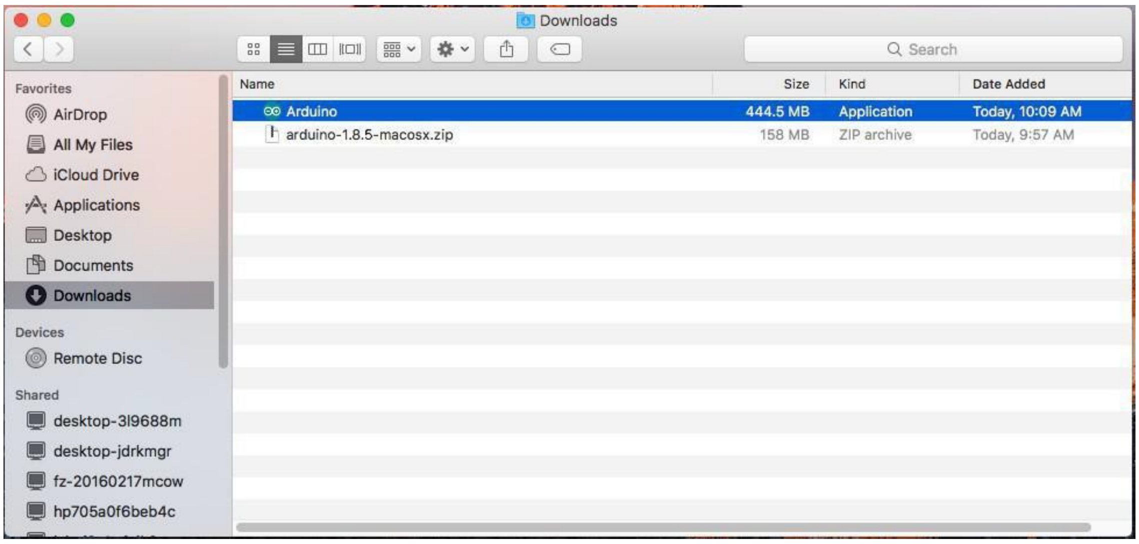




Expanding "arduino-1.8.5-macosx.zip"...

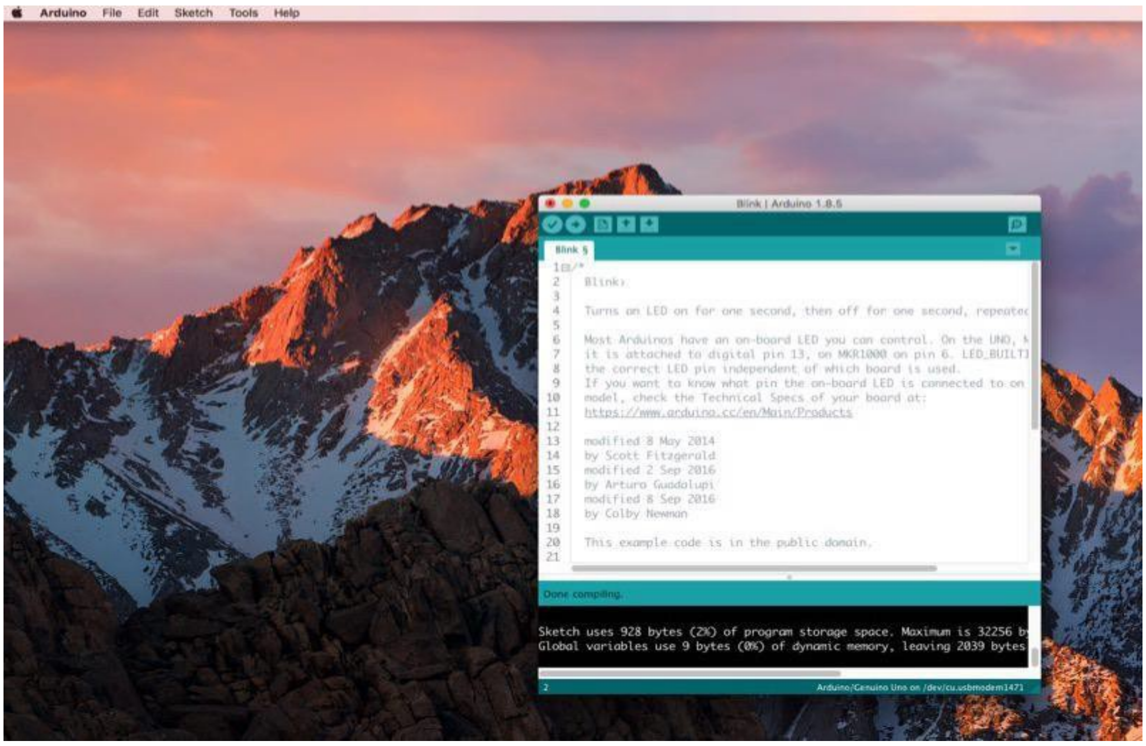
arduino-1.8.5-macosx.zip

• arduino file is the Arduino application, just double click to open it



Name	Size	Kind	Date Added
Arduino	444.5 MB	Application	Today, 10:09 AM
arduino-1.8.5-macosx.zip	158 MB	ZIP archive	Today, 9:57 AM

• Arduino IDE



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatec

  Most Arduinos have an on-board LED you can control. On the UNO, M
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTI
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.
```

Done compiling.

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 b
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes
```

2   Arduino/Genuino Uno on /dev/cu.usbmodem1471

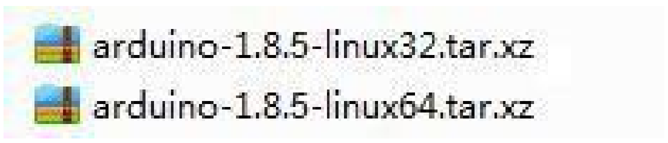
Installing Arduino (Linux)

You will have to use the “make install” command. If you are using the Ubuntu system, it is recommended to install Arduino IDE from the Ubuntu Software Center.

You can refer to the following links which offer you the specific installation steps.

https://www.arduino.cc/en/guide/linux

INSTALL ARDUINO ON LINUX.pdf


arduino-1.8.5-linux32.tar.xz
arduino-1.8.5-linux64.tar.xz

Lesson 1 Add Libraries and Open Serial Monitor

Installing Additional Arduino Libraries

Once you are comfortable with the Arduino software and using the built-in functions, you may want to extend the ability of your Arduino with additional libraries.

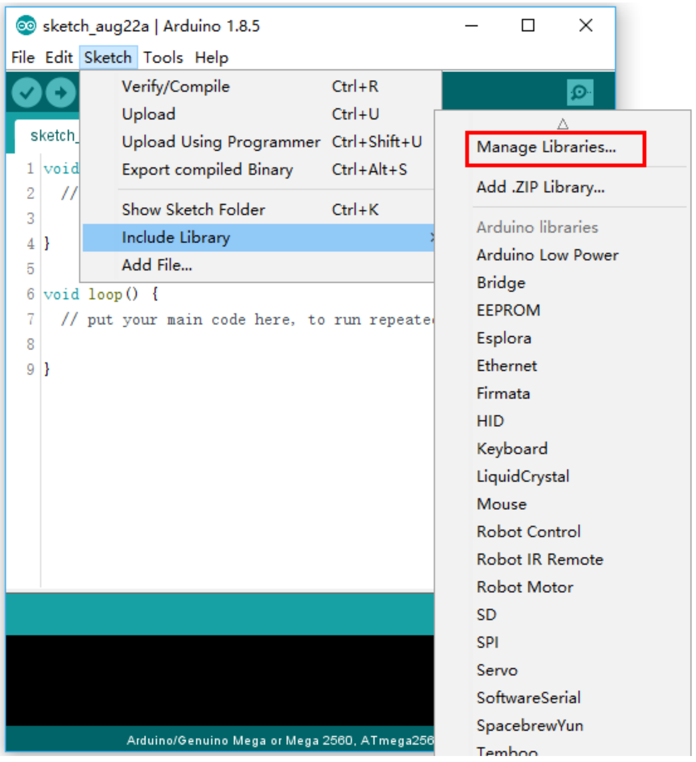
What are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in Liquid Crystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

How to Install a Library

Using the Library Manager

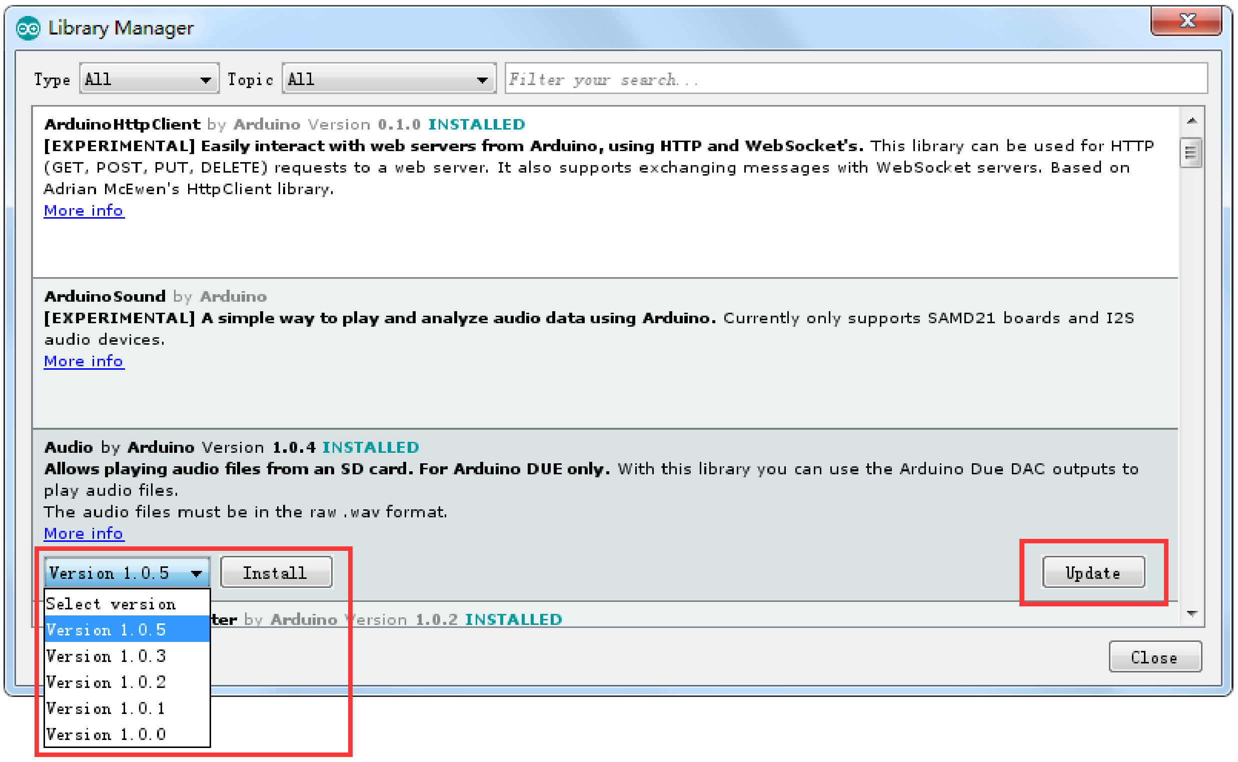
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



```
void
  //

}

void loop() {
  // put your main code here, to run repeate

}
```

Then the library manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry: it isnormal.

There are times you have to be patient with it, just as shown in the figure. Please refresh it and wait.



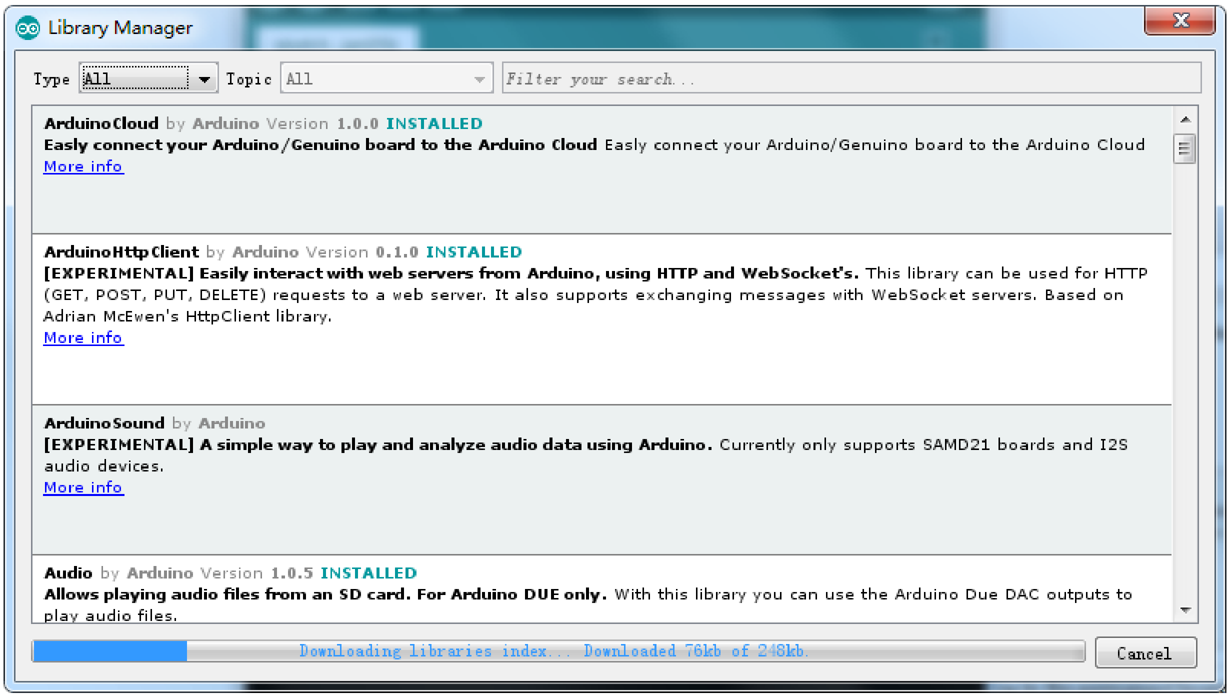
Library Manager

Type All Topic All Filter your search...

ArduinoHttpClient by Arduino Version 0.1.0 INSTALLED
[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's. This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian McEwen's HttpClient library.
More info

ArduinoSound by Arduino
[EXPERIMENTAL] A simple way to play and analyze audio data using Arduino. Currently only supports SAMD21 boards and I2S audio devices.
More info

Audio by **Arduino** Version **1.0.4** INSTALLED
Allows playing audio files from an SD card. For Arduino DUE only. With this library you can use the Arduino Due DAC outputs to play audio files.
The audio files must be in the raw .wav format.
More info



Library Manager

Type All Topic All Filter your search...

ArduinoCloud by Arduino Version 1.0.0 INSTALLED
Easily connect your Arduino/Genuino board to the Arduino Cloud Easily connect your Arduino/Genuino board to the Arduino Cloud
More info

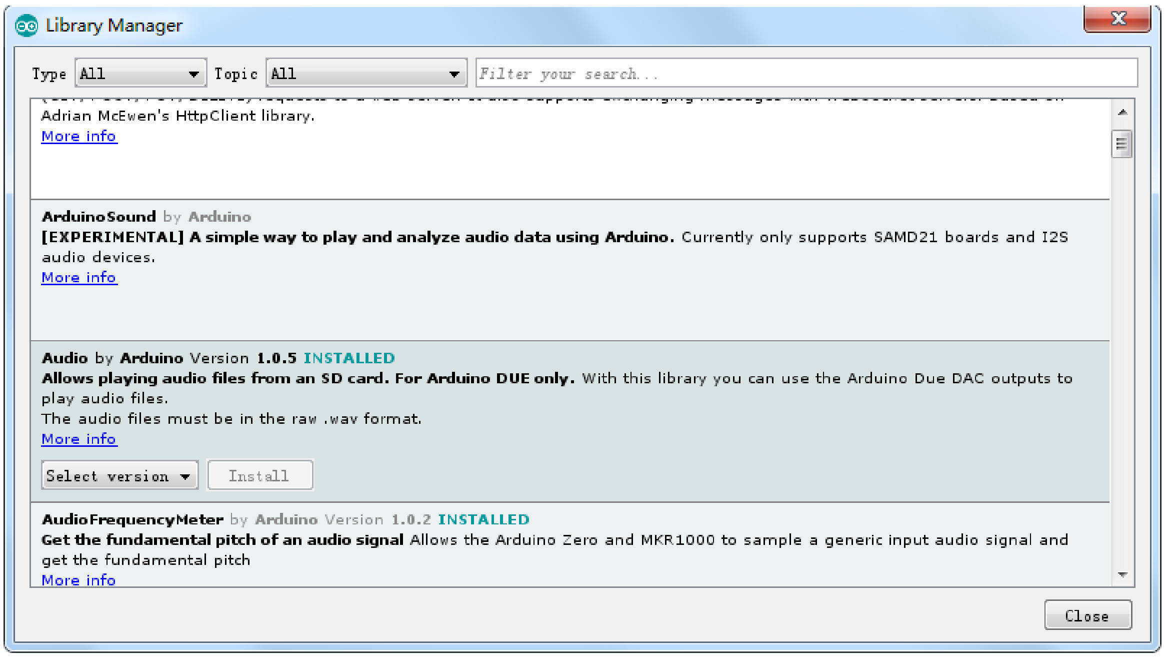
ArduinoHttpClient by Arduino Version 0.1.0 INSTALLED
[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's. This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian McEwen's HttpClient library.
More info

ArduinoSound by Arduino
[EXPERIMENTAL] A simple way to play and analyze audio data using Arduino. Currently only supports SAMD21 boards and I2S audio devices.
More info

Audio by Arduino Version 1.0.5 INSTALLED
Allows playing audio files from an SD card. For Arduino DUE only. With this library you can use the Arduino Due DAC outputs to play audio files.

Downloading libraries index... Downloaded 76kb of 248kb.

Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.



Adrian McEwen's HttpClient library.
More info

ArduinoSound by Arduino
[EXPERIMENTAL] A simple way to play and analyze audio data using Arduino. Currently only supports SAMD21 boards and I2S audio devices.
More info

Audio by **Arduino** Version **1.0.5** INSTALLED
Allows playing audio files from an SD card. For Arduino DUE only. With this library you can use the Arduino Due DAC outputs to play audio files.
The audio files must be in the raw .wav format.
More info

AudioFrequencyMeter by Arduino Version 1.0.2 INSTALLED
Get the fundamental pitch of an audio signal Allows the Arduino Zero and MKR1000 to sample a generic input audio signal and get the fundamental pitch
More info

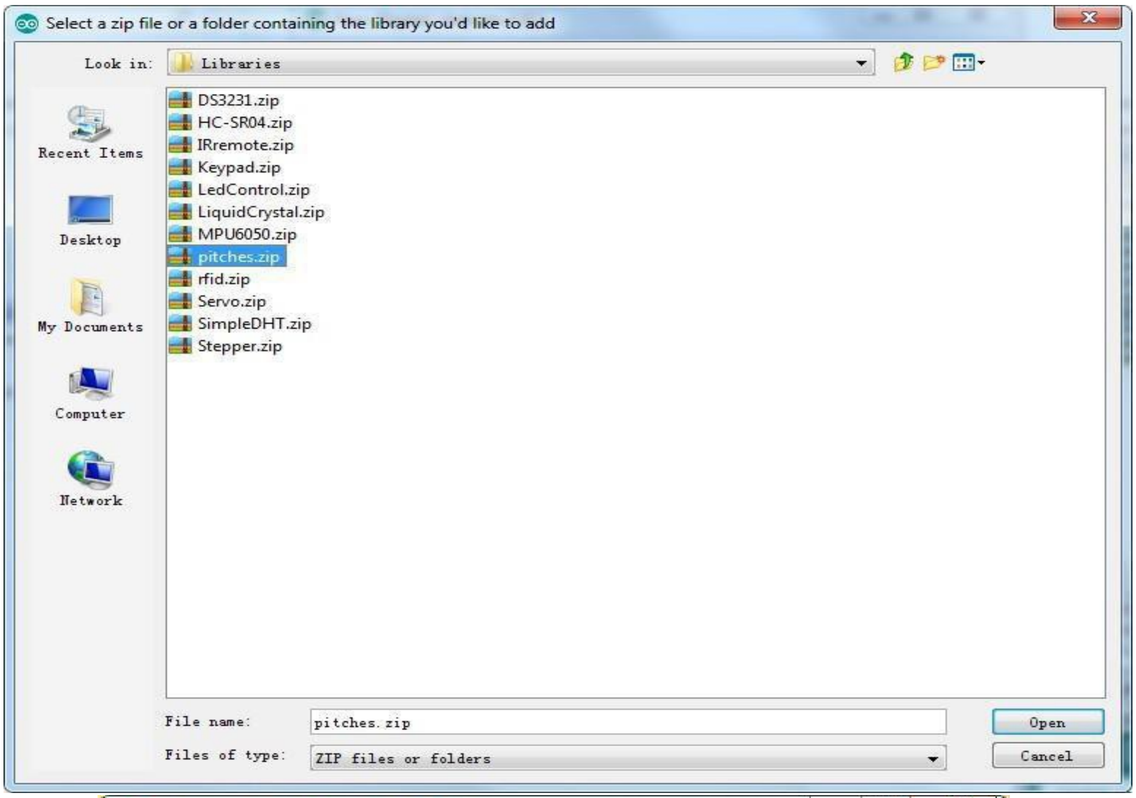
You can now find the new library available in the Include Library menu. If you want to add your own library open a new issue on Github.

Importing a .zip Library

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is.

In the Arduino IDE, navigate to Sketch > Include Library. At the top of the drop down list, select the option to "Add .ZIP Library".

You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.

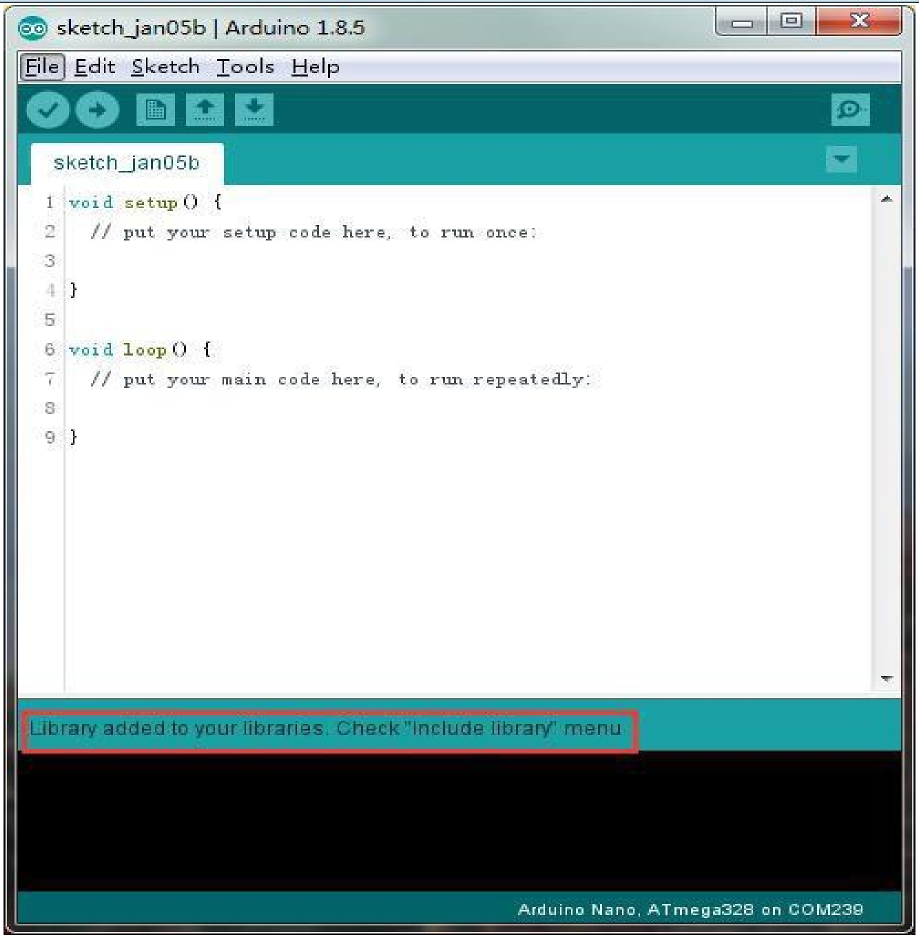

Select a zip file or a folder containing the library you'd like to add

Look in: Libraries

DS3231.zip
HC-SR04.zip
IRremote.zip
Keypad.zip
LedControl.zip
LiquidCrystal.zip
MPU6050.zip
pitches.zip
rfid.zip
Servo.zip
SimpleDHT.zip
Stepper.zip

File name: pitches.zip

Files of type: ZIP files or folders


sketch_jan05b | Arduino 1.8.5

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Library added to your libraries. Check "Include library" menu

Arduino Nano, ATmega328 on COM239

Return to the Sketch > Import Library menu. You should now see the library atthe bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory. NB: the Library will be available to use in sketches, but examples for the library will not be exposed in the File > Examples until after the IDE has restarted.

Those two are the most common approaches. MAC and Linux systems can be handled likewise. The manual installation to be introduced below as an alternative may be seldom used and users with no needs may skip it.

Manual installation

To install the library, first quit the Arduino application. Then uncompress the ZIP file containing the library. For example, if you're installing a library called "ArduinoParty", uncompressing ArduinoParty.zip. It should contain a folder calledArduinoParty, with files like ArduinoParty.cpp and ArduinoParty.h inside. (If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "ArduinoParty" and move into it all the files that were in the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.)

Drag the Arduino Party folder into this folder (your libraries folder). Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Your Arduino library folder should now look like this (onWindows):

```
MyDocuments\Arduino\libraries\ArduinoParty\ArduinoParty.cpp
My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.h
MyDocuments\Arduino\libraries\ArduinoParty\examples
```

or like this (on Mac and Linux):

```
Documents/Arduino/libraries/ArduinoParty/ArduinoParty.cpp
Documents/Arduino/libraries/ArduinoParty/ArduinoParty.h
Documents/Arduino/libraries/ArduinoParty/examples
```

....

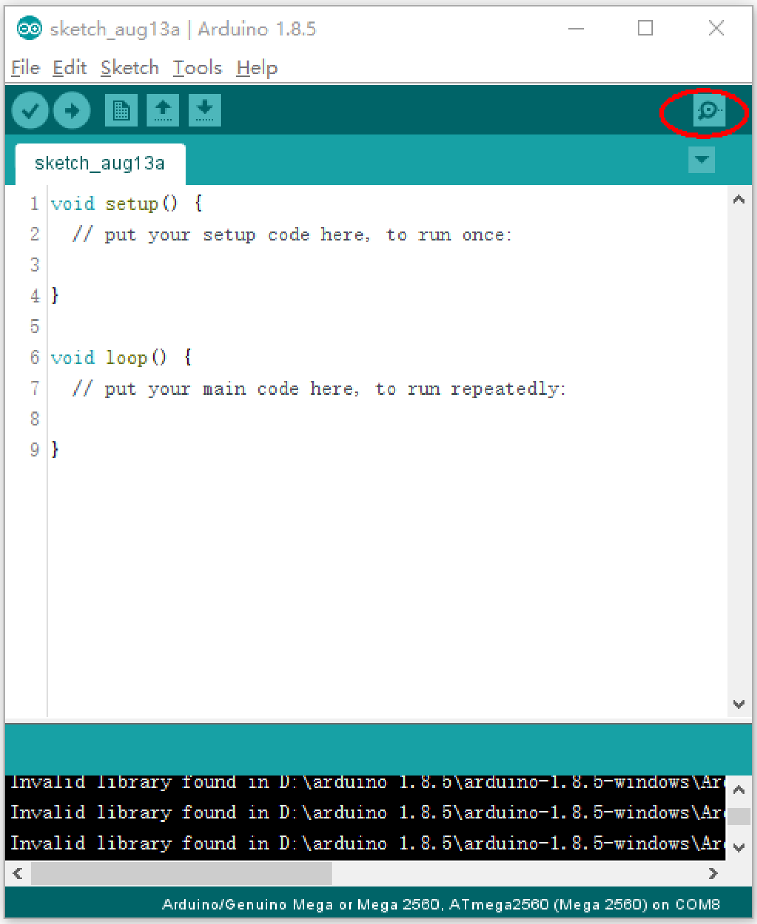
There may be more files than just the .cpp and .h files, just make sure they're all there. (The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder. For example: Documents\Arduino\libraries\ArduinoParty.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp won't work.)
Restart the Arduino application. Make sure the new library appears in the S k e t c h - >Import Library menu item of the software. That's it! You've installed a library!

Arduino Serial Monitor (Windows, Mac, Linux)

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. And, because using a terminal is such a big part of working with Arduinos and other microcontrollers, they decided to include a serial terminal with the software. Within the Arduino environment, this is called the Serial Monitor.

Making a Connection

Serial monitor comes with any and all version of the Arduino IDE. To open it, simply click the Serial Monitoricon.



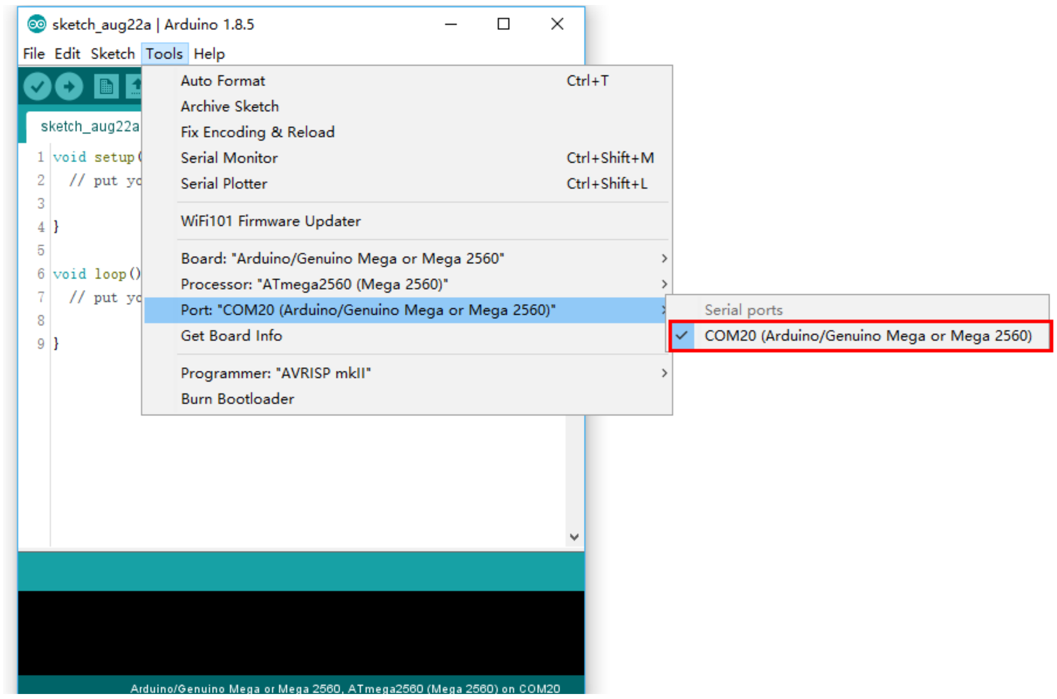
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ar
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ar
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ar
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM8

Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to Tools -> Serial Port, and select the correct port.

Tips: Choose the same COM port that you have in Device Manager.



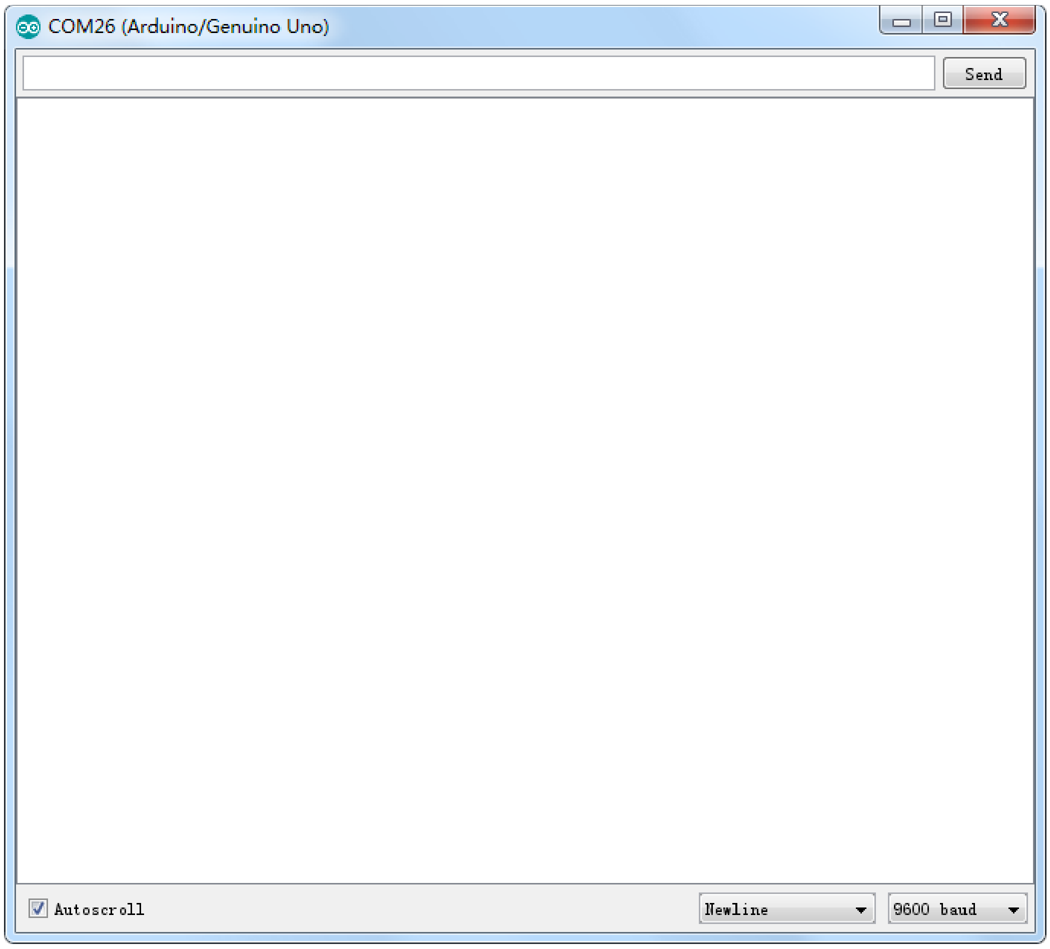
sketch_aug22a | Arduino 1.8.5

File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "COM20 (Arduino/Genuino Mega or Mega 2560)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM20 (Arduino/Genuino Mega or Mega 2560)

```
void setup(
  // put yo

}

void loop()
  // put yo

}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

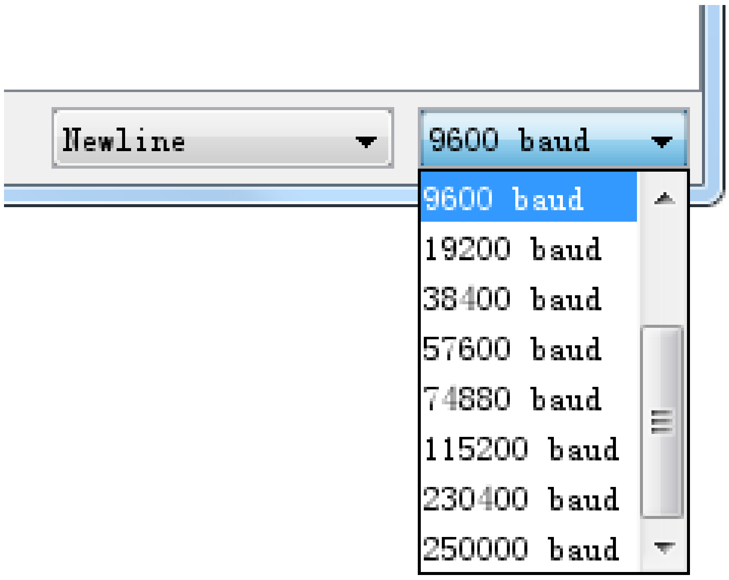
Once open, you should see something like this:



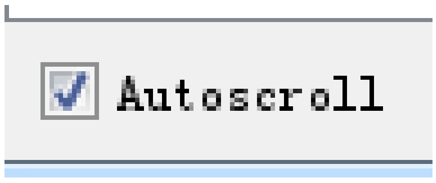
COM26 (Arduino/Genuino Uno)

Settings

The Serial Monitor has limited settings, but enough to handle most of your serial communication needs. The first setting you can alter is the baud rate. Click on the baud rate drop-down menu to select the correct baud rate. (9600 baud)



Last, you can set the terminal to Auto scroll or not by checking the box in the bottom left corner.



Pros

The Serial Monitor is a great quick and easy way to establish a serial connection with your Arduino. If you're already working in the Arduino IDE, there's really no need to open up a separate terminal to displaydata.

Cons

The lack of settings leaves much to be desired in the Serial Monitor, and, for advanced serial communications, it may not do the trick.

Lesson 2 Blink

Overview

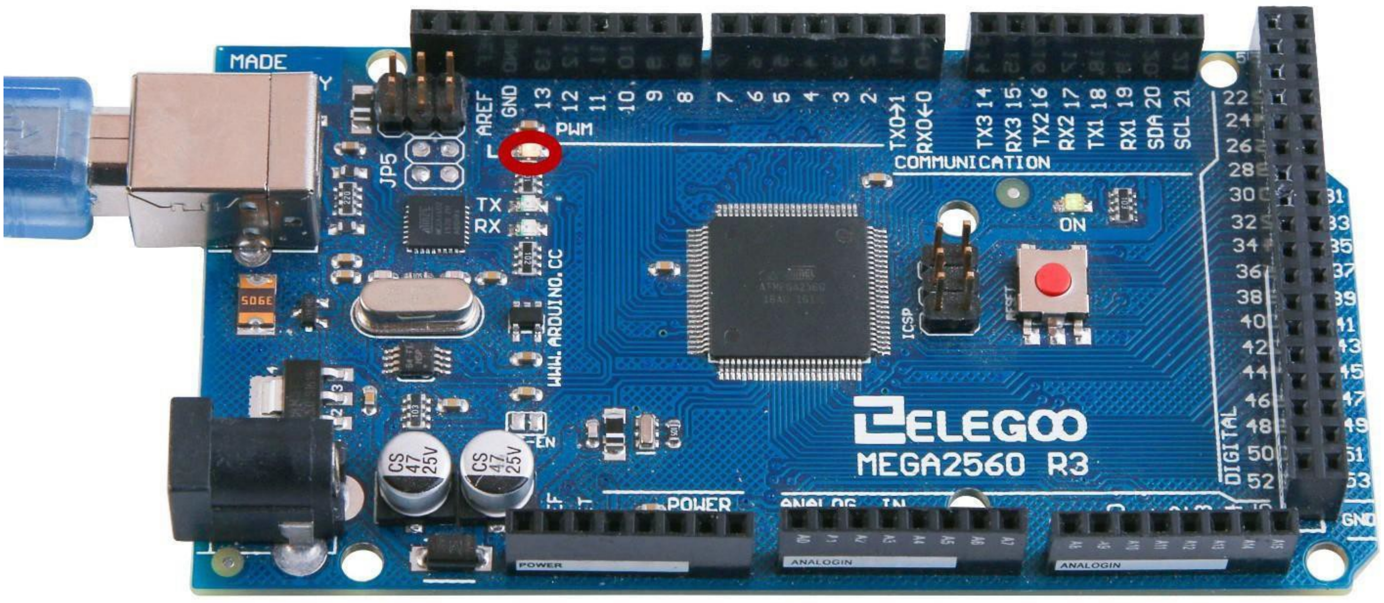
In this lesson, you will learn how to program your MEGA 2560 R3 controller board to blink the Arduino's built-in LED, and how to download programs by basic steps.

Component Required:

(1) x Elegoo Mega 2560R3

Principle

The MEGA2560 R3 board has rows of connectors along both sides that are used to connect to several electronic devices and plug-in 'shields' that extends its capability. It also has a single LED that you can control from your sketches. This LED is built onto the MEGA2560 R3 board and is often referred to as the 'L' LED as this is how it is labeled on the board.



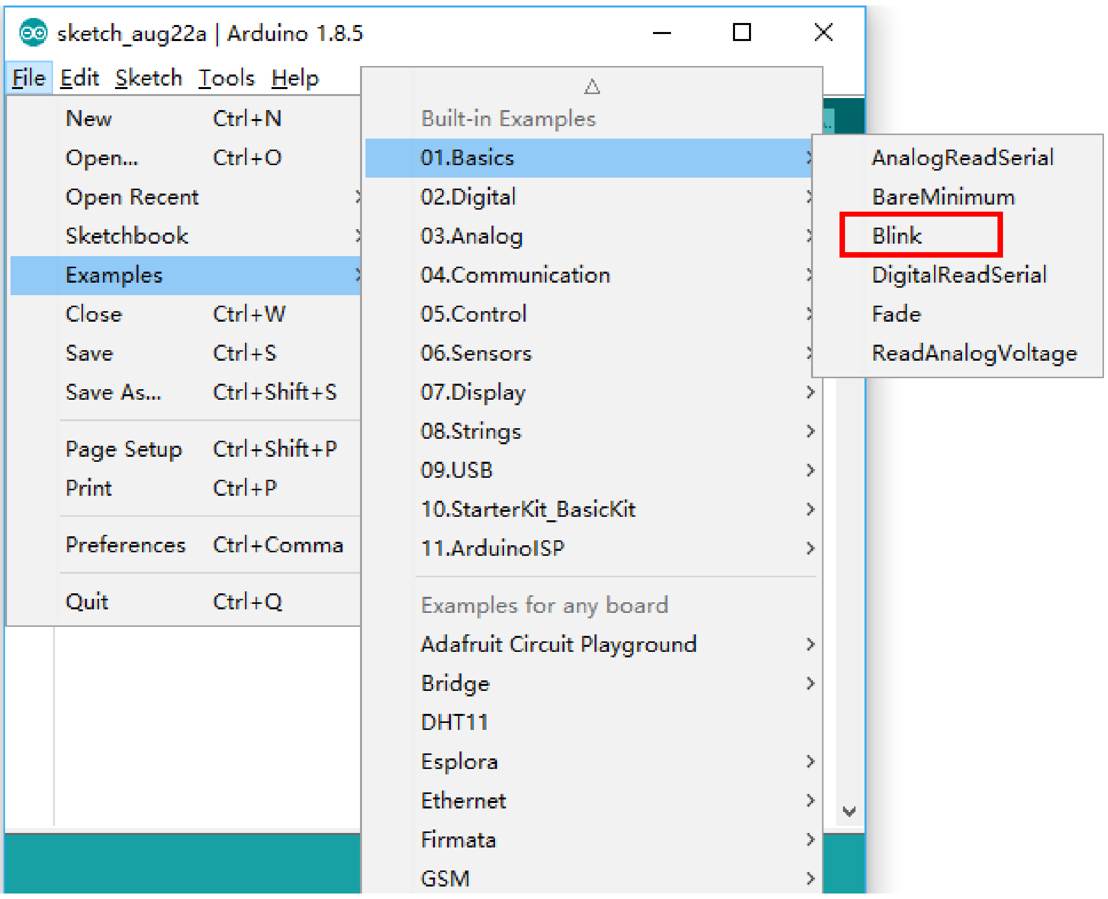
You may find that your MEGA 2560 R3 board's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

In this lesson, we will reprogram the MEGA 2560 R3 board with our own Blink sketch and then change the rate at which it blinks.

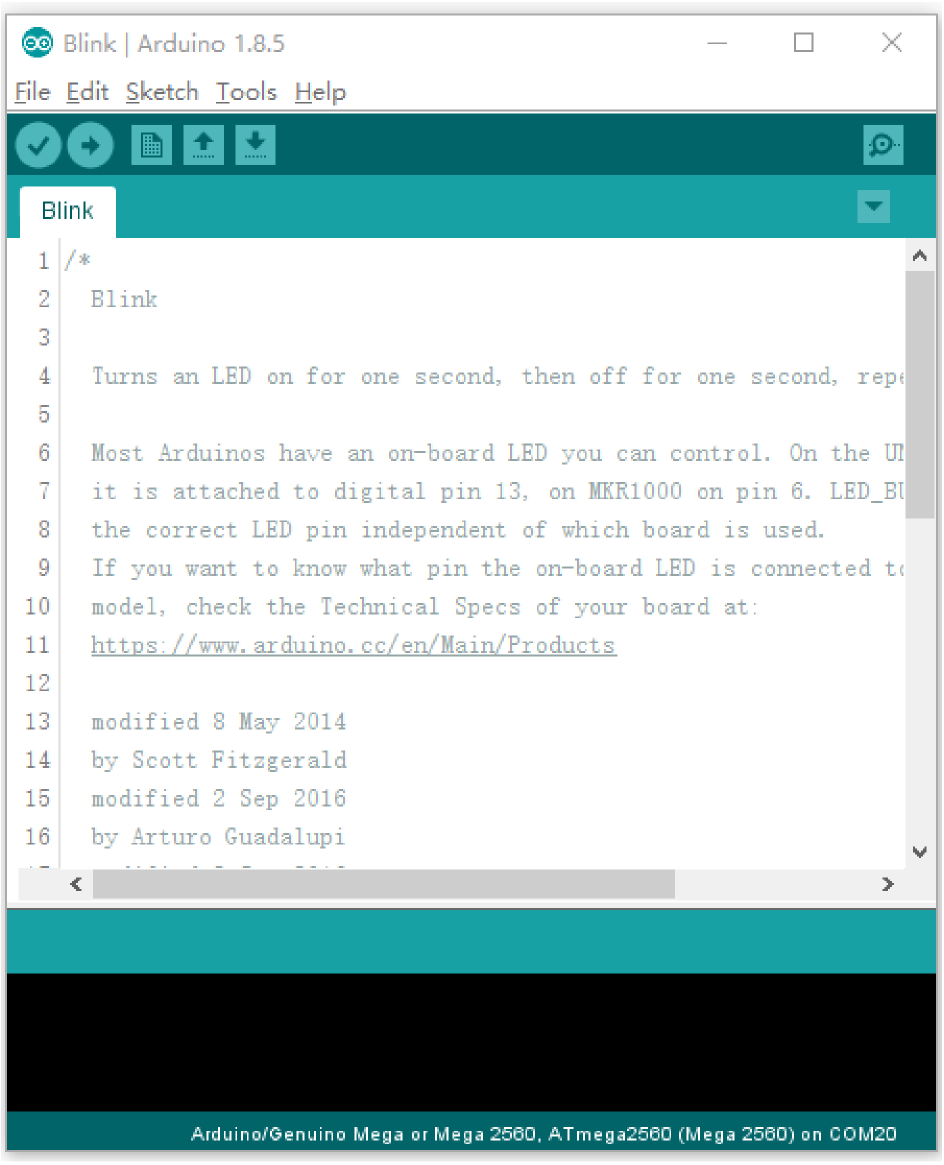
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your MEGA 2560 R3 board. The time has now come to put that connection to the test and program your MEGA 2560 R3 board.

The Arduino IDE includes a large collection of example sketches that you can load up and use. This includes an example sketch for making the 'L' LED blink.

Load the 'Blink' sketch that you will find in the IDE's menu system under File > Examples > 01. Basics



When the sketch window opens, enlarge it so that you can see the entire sketch in the window.



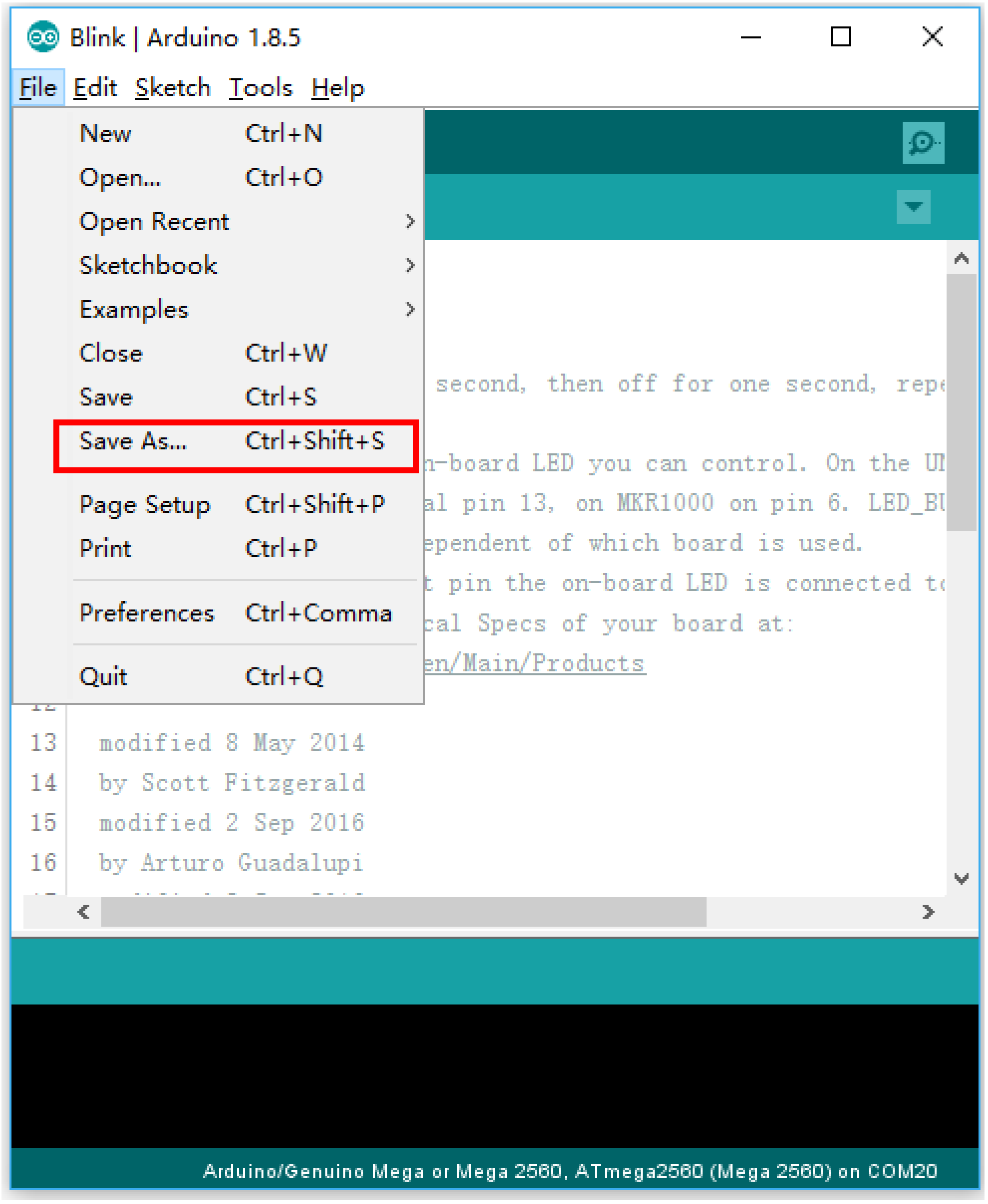
```
/*
  Blink

  Turns an LED on for one second, then off for one second, repe

  Most Arduinos have an on-board LED you can control. On the UN
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BU
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

The example sketches included with the Arduino IDE are 'read-only'. That is, you can upload them to an MEGA2560 R3 board, but if you change them, you cannot save them as the same file.

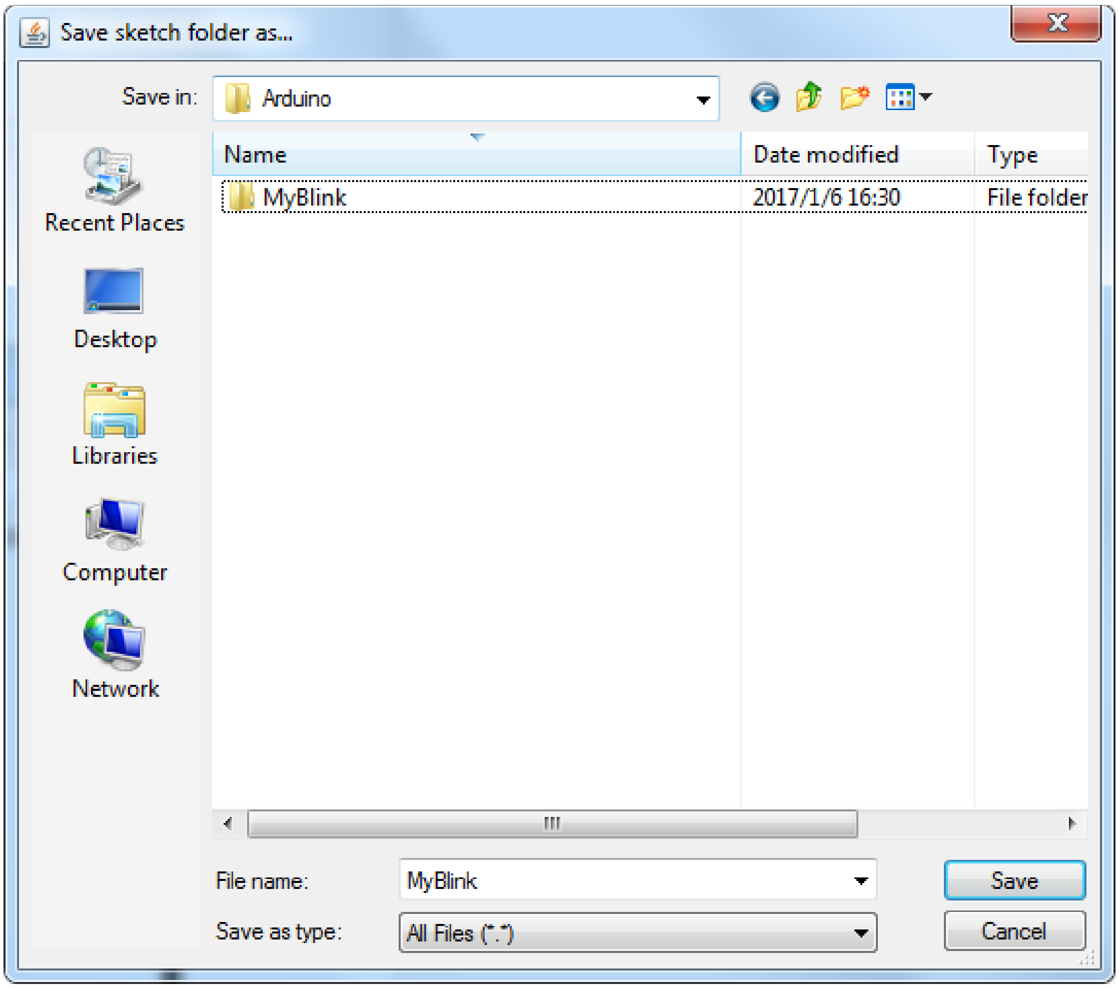
Since we are going to change this sketch, the first thing you need to do is save your own copy.

From the File menu on the Arduino IDE, select 'Save As..' and then save the sketch with the name 'MyBlink'.



```
second, then off for one second, repe
n-board LED you can control. On the UN
al pin 13, on MKR1000 on pin 6. LED_BU
ependent of which board is used.
t pin the on-board LED is connected to
cal Specs of your board at:
en/Main/Products

modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

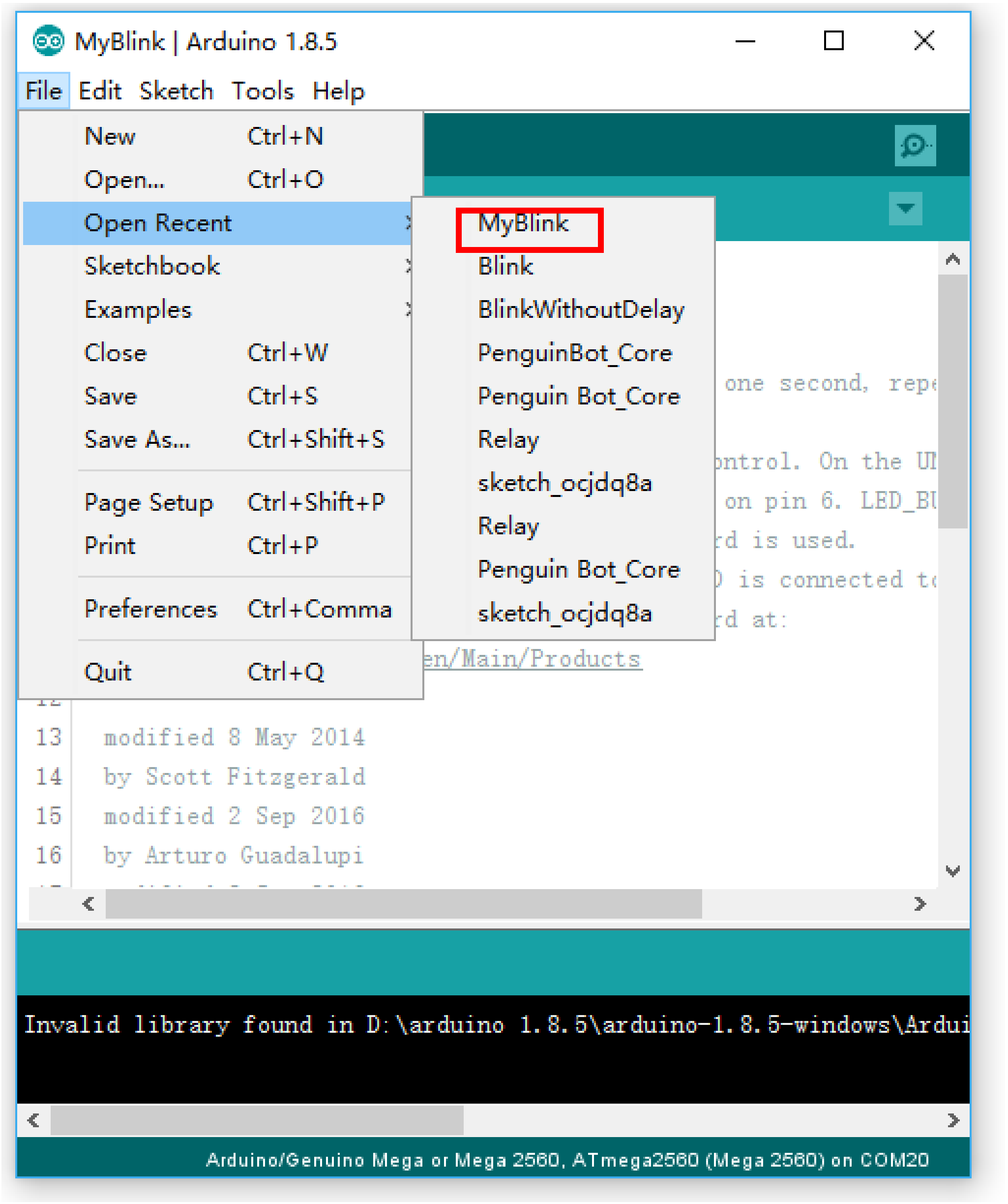


Name	Date modified	Type
MyBlink	2017/1/6 16:30	File folder

File name: MyBlink

Save as type: All Files (*.*)

You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File > Sketchbook menu option.

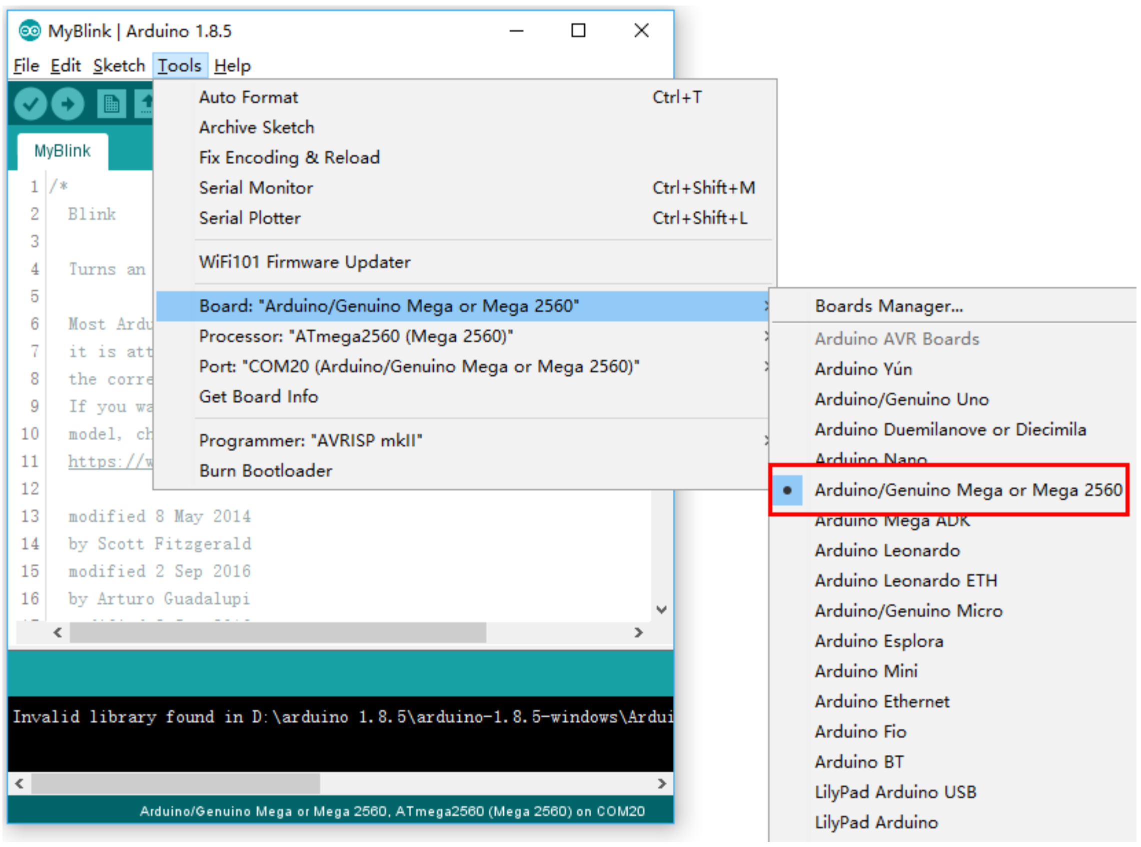


```
one second, repe
ontrol. On the UI
 on pin 6. LED_BU
rd is used.
) is connected to
rd at:
en/Main/Products

modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
```

```
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ardui
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.



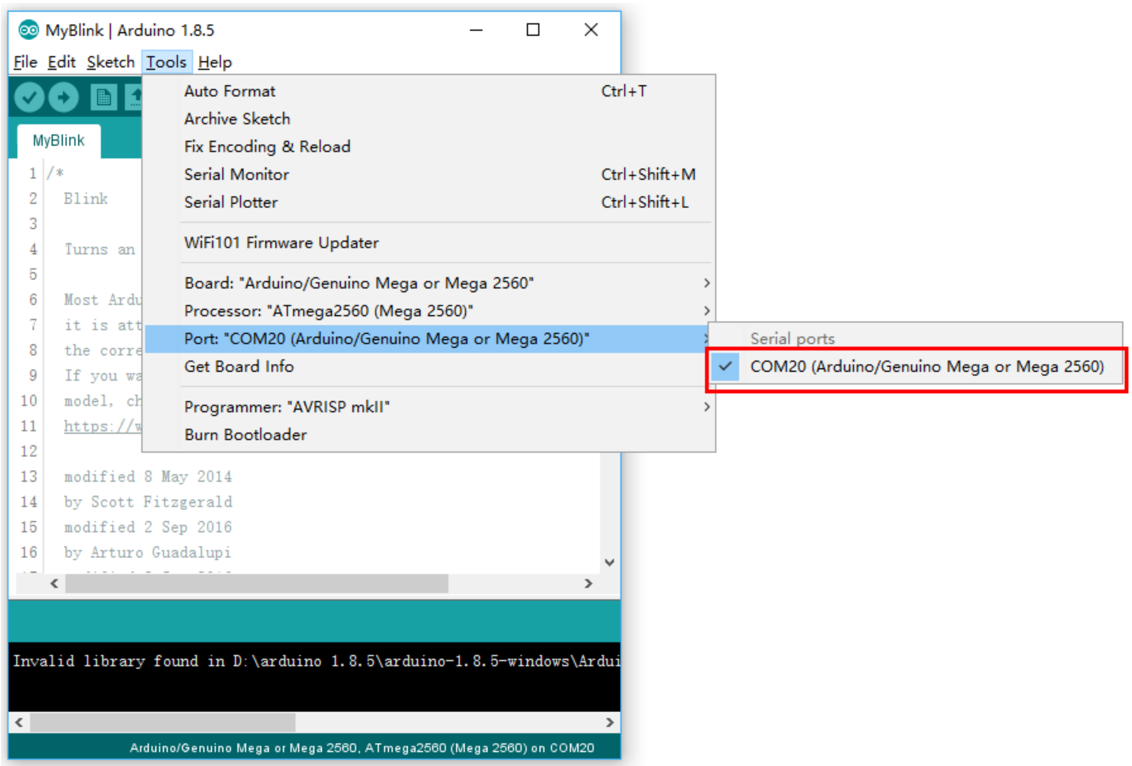
Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "COM20 (Arduino/Genuino Mega or Mega 2560)"
Programmer: "AVRISP mkII"

● Arduino/Genuino Mega or Mega 2560

```
/*
  Blink

  Turns an

  Most Ardu
  it is att
  the corre
  If you wa
  model, ch
  https://w

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
```

```
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ardui
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20



Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "COM20 (Arduino/Genuino Mega or Mega 2560)"
Programmer: "AVRISP mkII"

✓ COM20 (Arduino/Genuino Mega or Mega 2560)

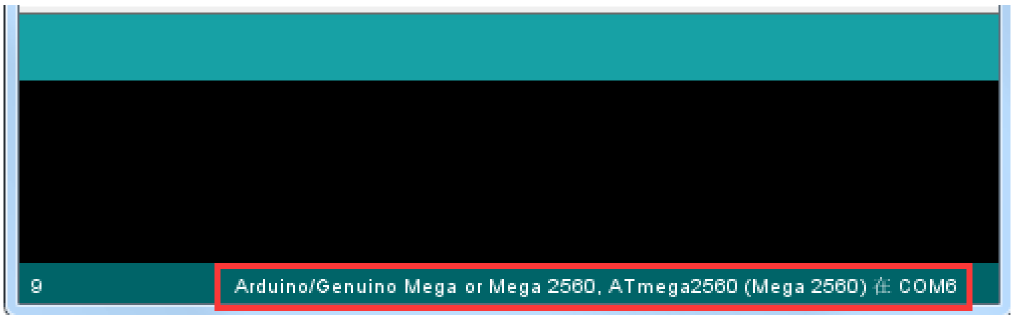
```
/*
  Blink

  Turns an

  Most Ardu
  it is att
  the corre
  If you wa
  model, ch
  https://w

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
```

```
Invalid library found in D:\arduino 1.8.5\arduino-1.8.5-windows\Ardui
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

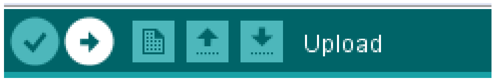
Note: The Board Type and Serial Port here are not necessarily the same as shown in picture. If you are using 2560, then you will have to choose Mega 2560 as the Board Type, other choices can be made in the same manner. And the Serial Port displayed for everyone is different, despite COM 20 chosen here, it could be COM3 or COM4 on your computer. A right COM port is supposed to be COMX (arduino XXX), which is by the certification criteria.

The Arduino IDE will show you the current settings for board at the bottom of the window.

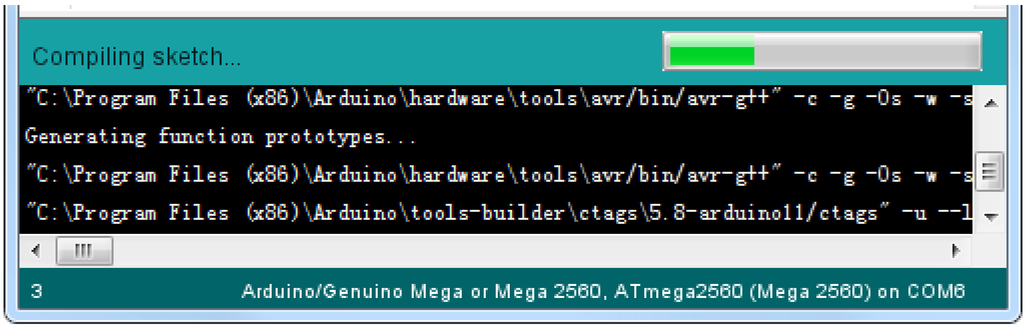


```
9          Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) 在 COM6
```

Click on the 'Upload' button. The second button from the left on the toolbar.

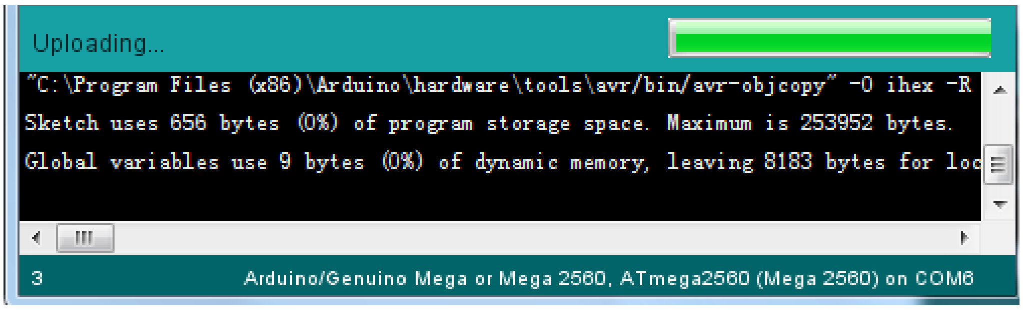


If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to theboard.



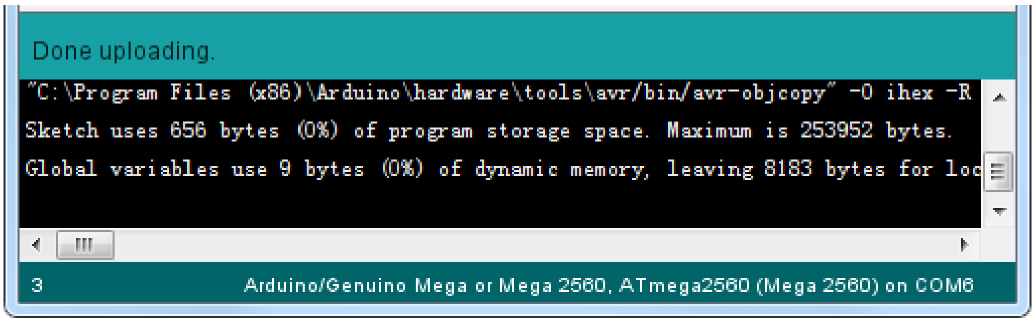
```
Compiling sketch...
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-g++" -c -g -Os -w -s
Generating function prototypes...
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-g++" -c -g -Os -w -s
"C:\Program Files (x86)\Arduino\tools-builder\ctags\5.8-arduino11/ctags" -u --l
3          Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6
```

Next, the status will change to 'Uploading'. At this point, the LEDs on the Arduino should start to flicker as the sketch is transferred.



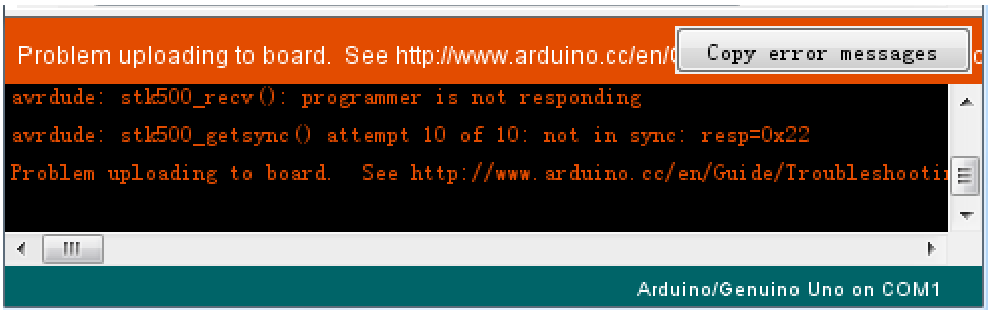
```
Uploading...
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-objcopy" -O ihex -R
Sketch uses 656 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for loc
3                    Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6
```

Finally, the staus will change to 'Done'.



```
Done uploading.
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-objcopy" -O ihex -R
Sketch uses 656 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for loc
3                    Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM6
```

The other message tells us that the sketch is using 928 bytes of the 32,256 bytes available. After the 'Compiling Sketch..' stage you could get the following error message:



```
Problem uploading to board.  See http://www.arduino.cc/en/(   Copy error messages
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x22
Problem uploading to board.  See http://www.arduino.cc/en/Guide/Troubleshooti
                                                   Arduino/Genuino Uno on COM1
```

It can mean that your board is not connected at all, or the drivers have not been installed (if necessary) or that the wrong serial port is selected.

If you encounter this, go back to Lesson 0 and check your installation.

Once the upload has completed, the board should restart and start blinking.

Open the code

Note that a huge part of this sketch is composed of comments. These are not actual program instructions; rather, they just explain how the program works. They are there for your benefit.

Everything between /* and */ at the top of the sketch is a block comment; it explains what the sketch is for.

Single line comments start with // and everything up until the end of that line is considered a comment.

The first line of code is:

```
int led = 13;
```

As the comment above it explains, this is giving a name to the pin that the LED is attached to. This is 13 on most Arduinos, including the MEGA 2560 and Leonardo.

Next, we have the 'setup' function. Again, as the comment says, this is executed when the reset button is pressed. It is also executed whenever the board resets for any reason, such as power first being applied to it, or after a sketch has been uploaded.

```
void setup() {
// initialize the digital pin as an output.
pinMode (led, OUTPUT);
}
```

Every Arduino sketch must have a 'setup' function, and the place where you might want to add instructions of your own is between the { and the }.

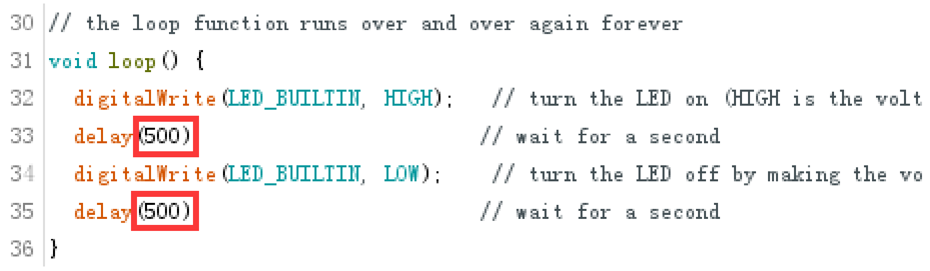
In this case, there is just one command there, which, as the comment states tells the Arduino board that we are going to use the LED pin as an output.

It is also mandatory for a sketch to have a 'loop' function. Unlike the 'setup' function that only runs once, after a reset, the 'loop' function will, after it has finished running its commands, immediately start again.

```
void loop() {
   digitalWrite(led, HIGH);     // turn the LED on (HIGH is the voltage level)
   delay(1000);                 // wait for a second
   digitalWrite(led, LOW);      // turn the LED off by making the voltage LOW
   delay(1000);                 // wait for a second
}
```

Inside the loop function, the commands first of all turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off and pause for another second.

You are now going to make your LED blink faster. As you might have guessed, the key to this lies in changing the parameter in () for the 'delay' command.



```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the volt
  delay(500)                        // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the vo
  delay(500)                        // wait for a second
}
```

This delay period is in milliseconds, so if you want the LED to blink twice as fast, change the value from 1000 to 500. This would then pause for half a second each delay rather than a wholesecond.

Upload the sketch again and you should see the LED start to blinkmore quickly.

Lesson 3 LED

Overview

In this lesson, you will learn how to change the brightness of an LED by using different values of resistor.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x 5mm red LED

(1) x 220 ohm resistor

(1) x 1k ohm resistor

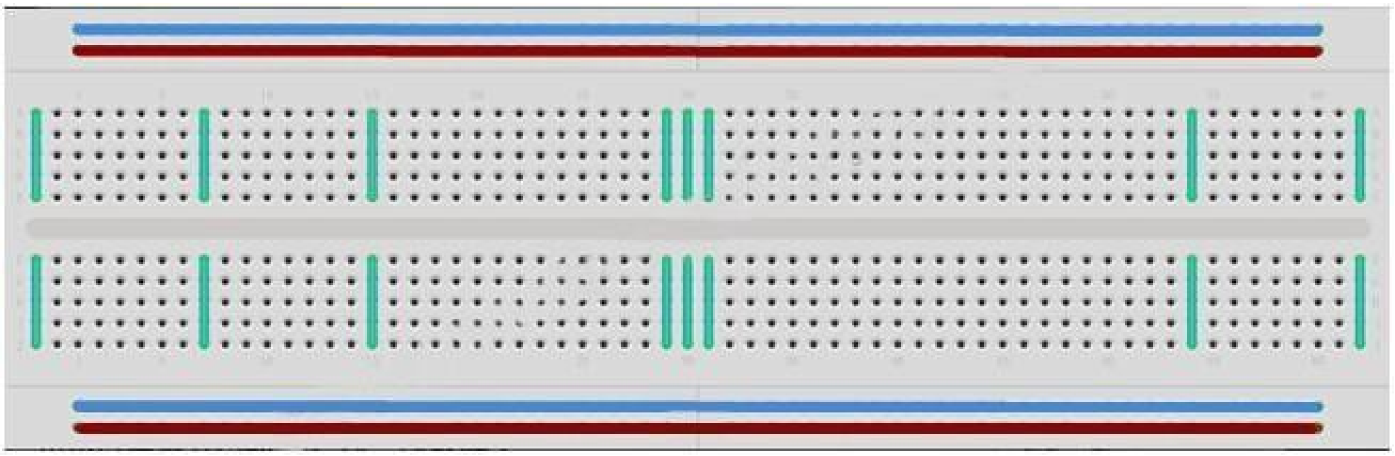
(1) x 10k ohm resistor

(2) x M-M wires (Male to Male jumper wires)

Component Introduction

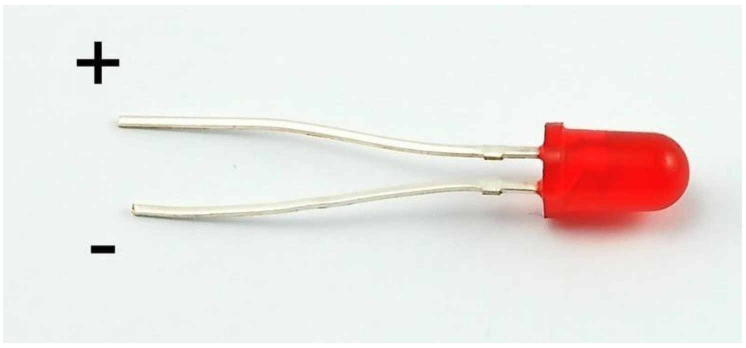
BREADBOARD MB-102：

A breadboard enables you to prototype circuits quickly, without having to solder the connections. Below is an example.



Breadboards come in various sizes and configurations. The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically. A deep channel running down the middle indicates that there is a break in connections there, meaning, you can push a chip in with the legs at either side of the channel without connecting them together. Some breadboards have two strips of holes running along the long edges of the board that are separated from the main grid. These have strips running down the length of the board inside and provide a way to connect a common voltage. They are usually in pairs for +5 volts and ground. These strips are referred to as rails and they enable you to connect power to many components or points in the board.
While breadboards are great for prototyping, they have some limitations. Because the connections are push-fit and temporary, they are not as reliable as soldered connections. If you are having intermittent problems with a circuit, it could be due to a poor connection on a breadboard.

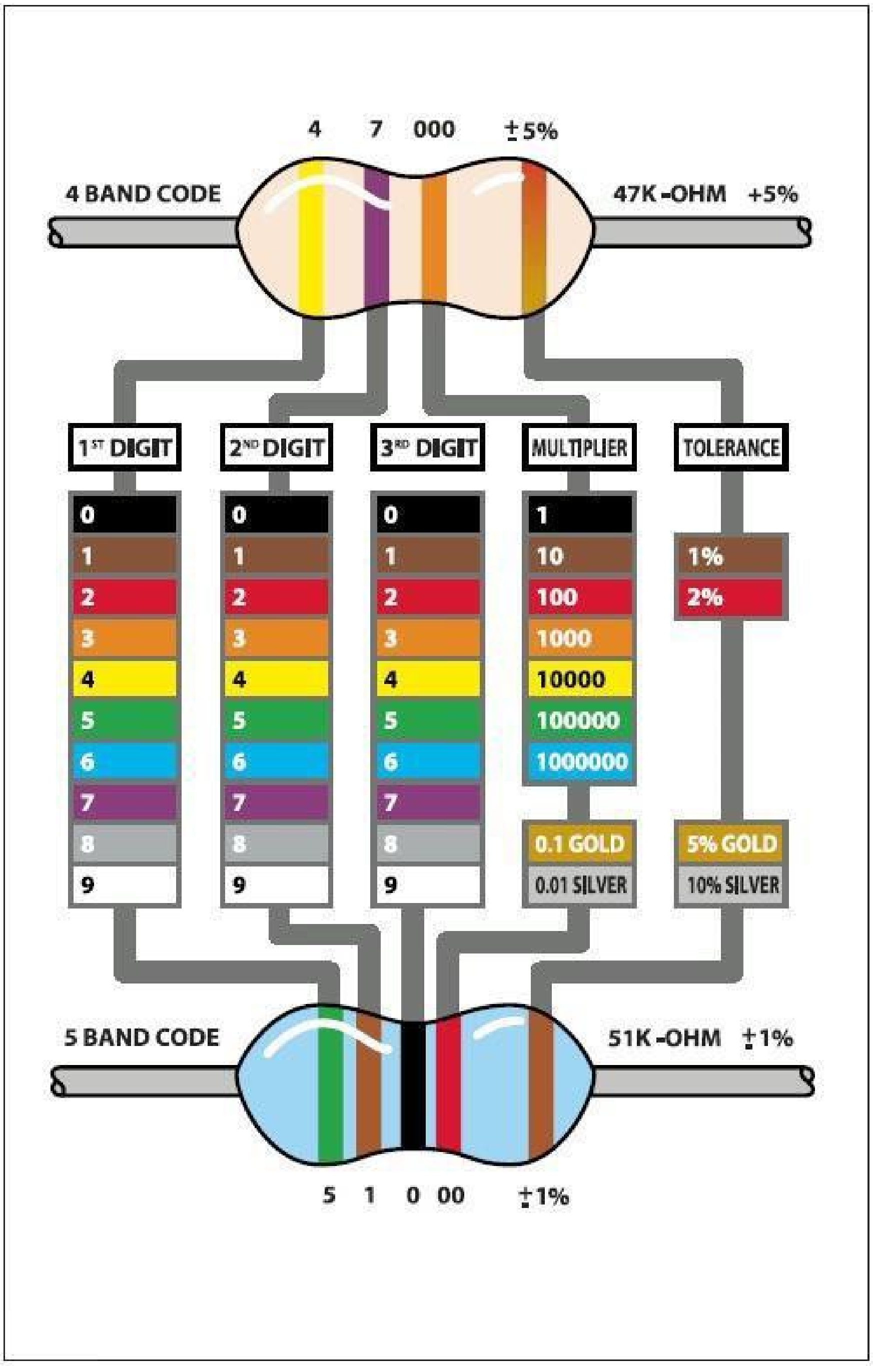
LED:
LEDs make great indicator lights. They use very little electricity and they pretty much last forever.
In this lesson, you will use perhaps the most common of all LEDs: a 5mm red LED.
5mm refers to the diameter of the LED. Other common sizes are 3mm and 10mm.
You cannot directly connect an LED to a battery or voltage source because 1) the LED has a positive and a negative lead and will not light if placed the wrong way and 2) an LED must be used with a resistor to limit or 'choke' the amount of current flowing through it; otherwise, it will burn out!



If you do not use a resistor with an LED, then it may well be destroyed almost immediately, as too much current will flow through, heating it and destroying the 'junction' where the light is produced.
There are two ways to tell which is the positive lead of the LED and which the negative.
Firstly, the positive lead is longer.
Secondly, where the negative lead enters the body of the LED, there is a flat edge to the case of the LED.
If you happen to have an LED that has a flat side next to the longer lead, you should assume that the longer lead is positive.

RESISTORS:
As the name suggests, resistors resist the flow of electricity. The higher the value of the resistor, the more it resists and the less electrical current will flow through it. We are going to use this to control how much electricity flows through the LED and therefore, how brightly it shines.

But first, more about resistors...
The unit of resistance is called the Ohm, which is usually shortened to Ω the Greek letter Omega. Because an Ohm is a low value of resistance (it doesn't resist much at all), we also denote the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω). These are called kilo-ohms and mega-ohms.
In this lesson, we are going to use three different values of resistor: 220Ω, 1kΩ and 10kΩ. These resistors all look the same, except that they have different colored stripes on them. These stripes tell you the value of the resistor.
The resistor color code has three colored stripes and then a gold stripe at one end.

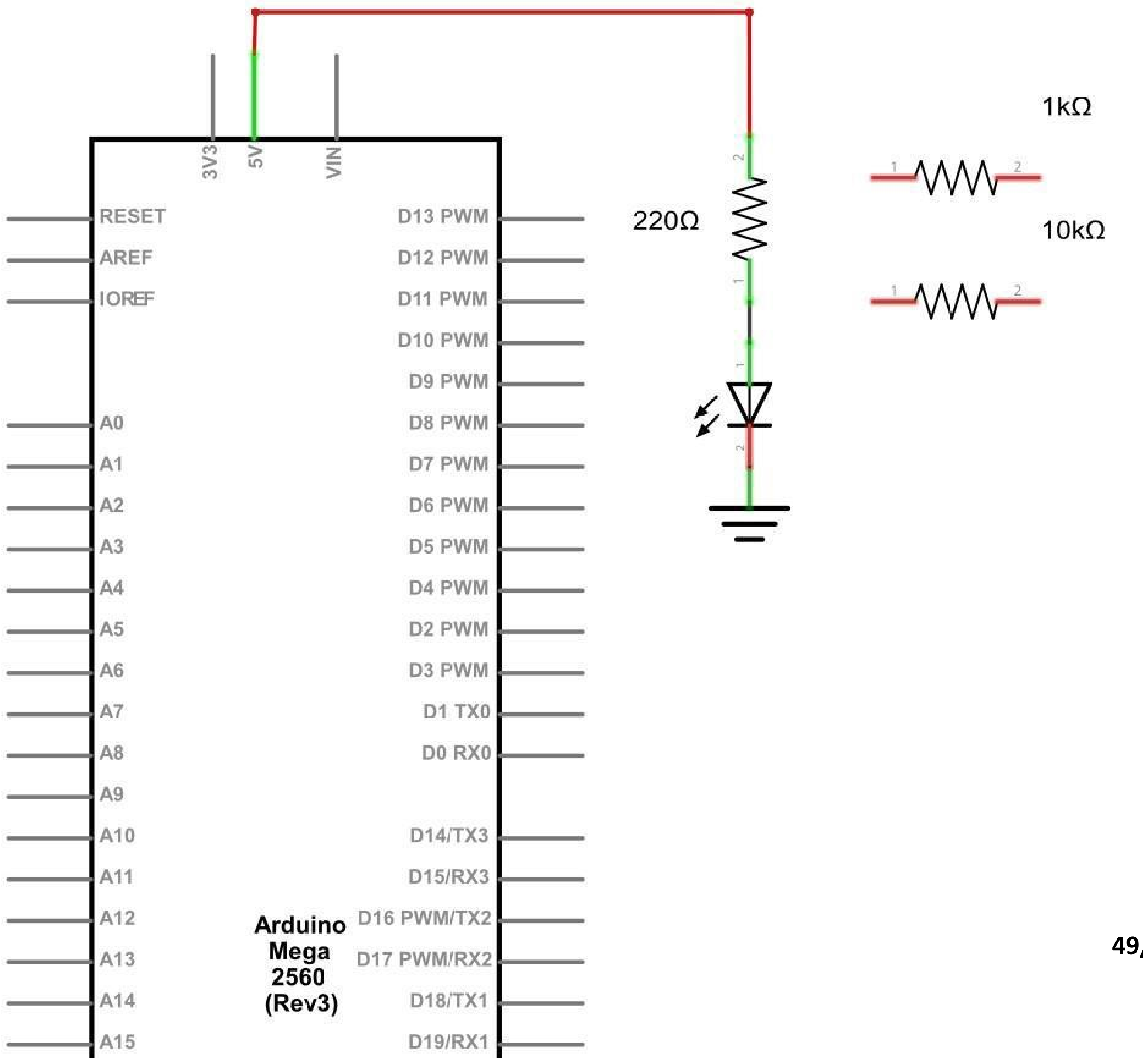


Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either way around.

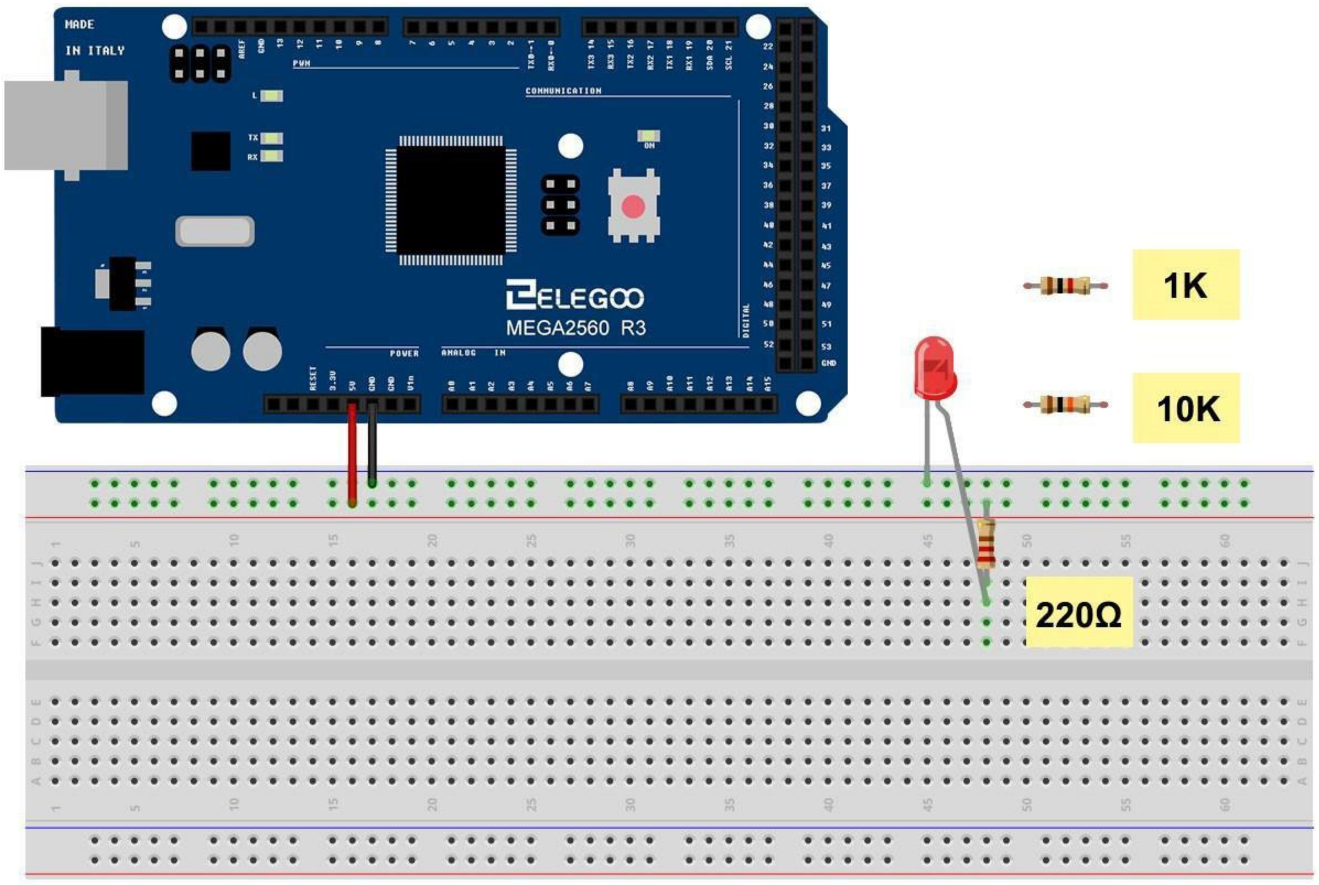
If you find this approach method too complicated, you can read the color ring flag on our resistors directly to determine its resistance value. Or you may use a digital multimeter instead.

Connection

Schematic



Wiring diagram



The MEGA 2560 is a convenient source of 5 volts, which we will use to provide power to the LED and the resistor. You do not need to do anything with your MEGA 2560, except to plug it into a USB cable.

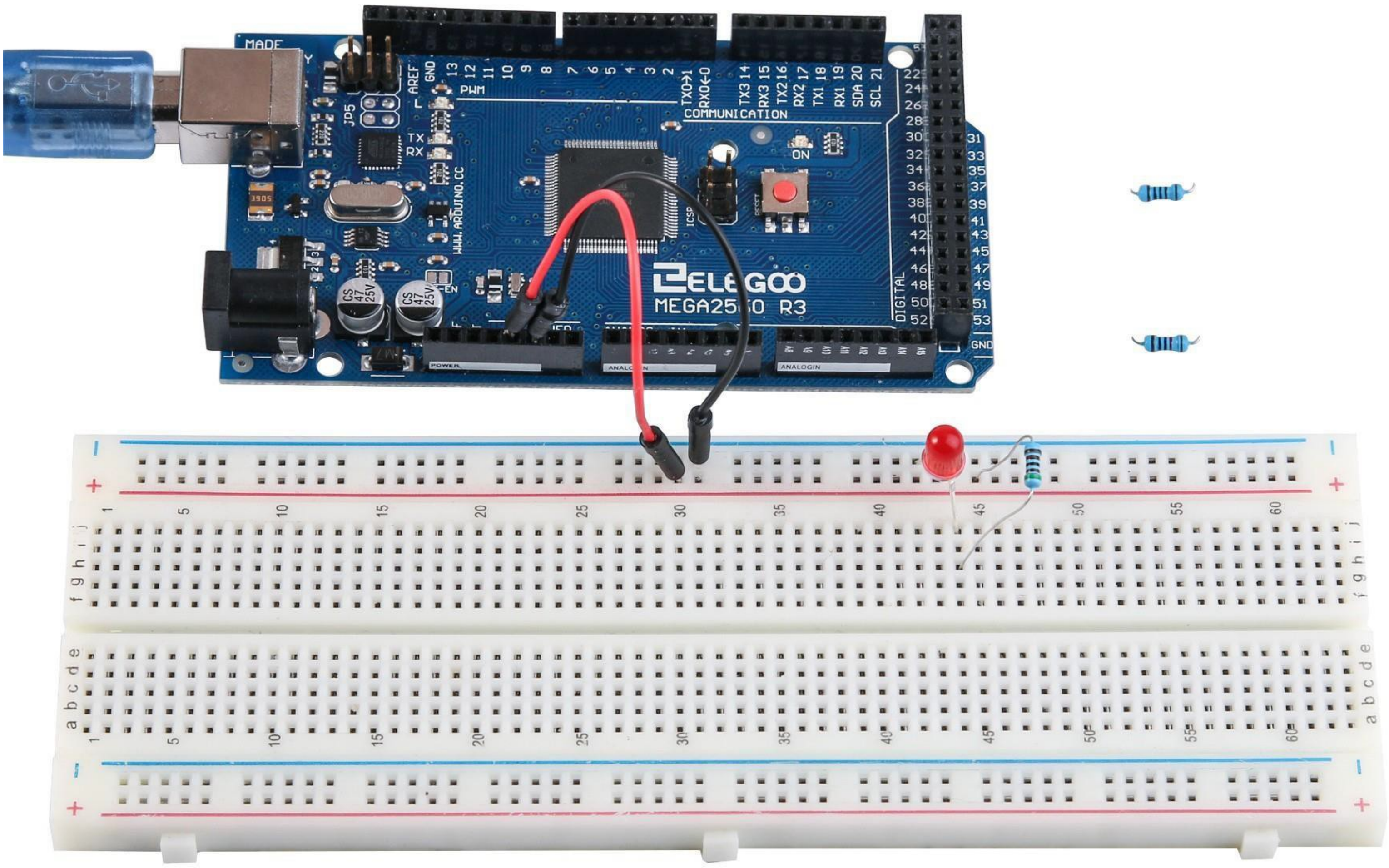
With the 220 Ω resistor in place, the LED should be quite bright. If you swap out the 220 Ω resistor for the 1kΩ resistor, then the LED will appear a little dimmer. Finally, with the 10 kΩ resistor in place, the LED will be just about visible. Pull the red jumper lead out of the breadboard and touch it into the hole and remove it, so that it acts like a switch. You should just be able to notice the difference.

At the moment, you have 5V going to one leg of the resistor, the other leg of the resistor going to the positive side of the LED and the other side of the LED going to GND. However, if we moved the resistor so that it came after the LED, as shown below, the LED will still light.

You will probably want to put the 220Ω resistor back in place.

It does not matter which side of the LED we put the resistor, as long as it is there somewhere

Examplepicture



Lesson 4 RGB LED

Overview

RGB LEDs are a fun and easy way to add some color to your projects. Since they are like 3 regular LEDs in one, how to use and connect them is not much different.
They come mostly in 2 versions: Common Anode or Common Cathode.
Common Anode uses 5V on the common pin, while Common Cathode connects to ground.
As with any LED, we need to connect some resistors inline (3 total) so we can limit the current being drawn.
In our sketch, we will start with the LED in the Red color state, then fade to Green, then fade to Blue and finally back to the Red color. By doing this we will cycle through most of the color that can be achieved.

Component Required:

(1) x Elegoo Mega2560 R3
(1) x 830 Tie Points Breadboard
(4) x M-M wires (Male to Male jumper wires)
(1) x RGB LED
(3) x 220 ohm resistors

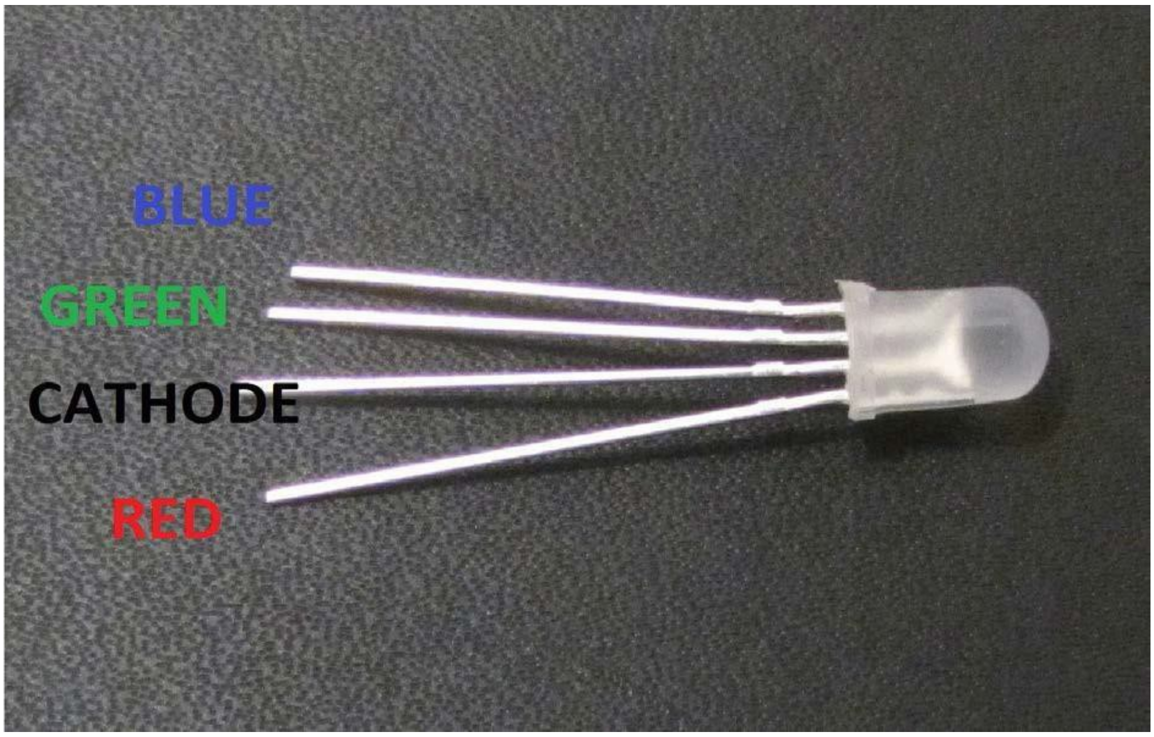
Component Introduction

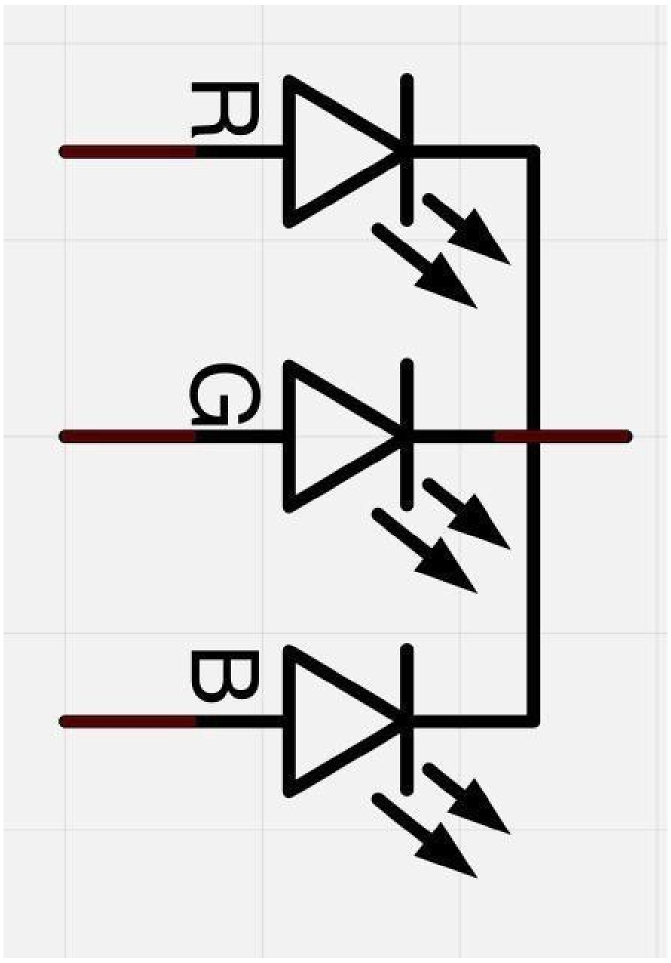
RGB:

At first glance, RGB (Red, Green and Blue) LEDs look just like regular LEDs. However, inside the usual LED package, there are actually three LEDs, one red, one green and yes, one blue. By controlling the brightness of each of the individual LEDs you can mix pretty much any color you want.

We mix colors the same way you would mix paint on a palette - by adjusting the brightness of each of the three LEDs. The hard way to do this would be to use different value resistors (or variable resistors) as we did with in Lesson 2, but that's a lot of work! Fortunately for us, MEGA 2560 R3 board has an analog Write function that you can use with pins marked with a ~ to output a variable amount of power to the appropriate LEDs.

The RGB LED has four leads. There is one lead going to the positive connection of each of the single LEDs within the package and a single lead that is connected to all three negative sides of the LEDs.





Here on the photographs you can see 4 electrode LED. Every separate pin for Green or Blue or Red color is called Anode. You will always connect “+” to it. Cathode goes to “-“ (ground). If you connect it other way round the LED will not light.

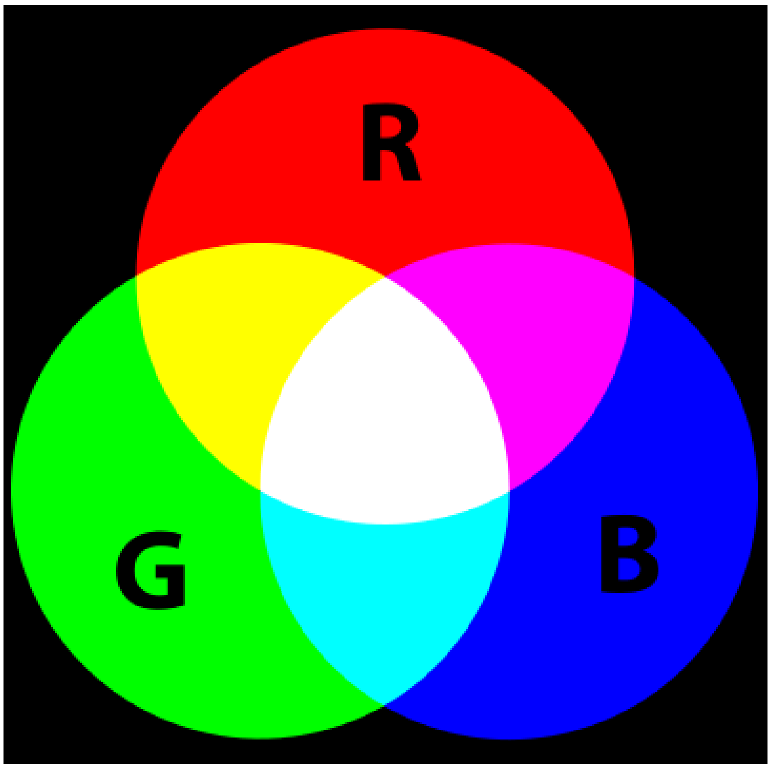
The common negative connection of the LED package is the second pin from the flat side. It is also the longest of the four leads and will be connected to the ground.

Each LED inside the package requires its own 220Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to MEGA 2560 output pins using these resistors.

COLOR:

The reason that you can mix any color you like by varying the quantities of red, green and blue light is that your eye has three types of light receptor in it (red, green and blue). Your eye and brain process the amounts of red, green and blue and convert it into a color of the spectrum.

In a way, by using the three LEDs, we are playing a trick on the eye. This same idea is used in TVs, where the LCD has red, green and blue color dots next to each other making up each pixel.



If we set the brightness of all three LEDs to be the same, then the overall color of the light will be white. If we turn off the blue LED, so that just the red and green LEDs are the same brightness, then the light will appear yellow.

We can control the brightness of each of the red, green and blue parts of the LED separately, making it possible to mix any color we like.

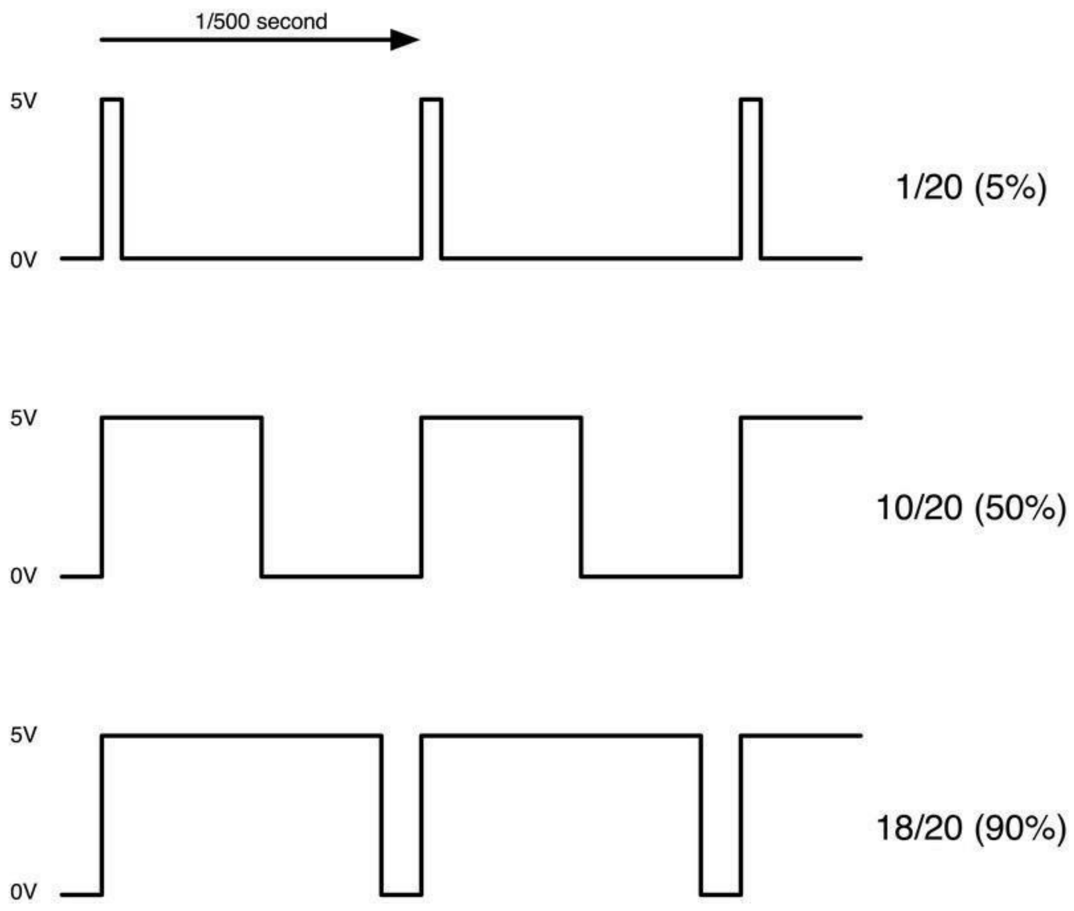
Black is not so much a color as an absence of light. Therefore, the closest we can come to black with our LED is to turn off all three colors.

Theory (PWM)

Pulse Width Modulation (PWM) is a technique for controlling power.

We also use it here to control the brightness of each of the LEDs.

The diagram below shows the signal from one of the PWM pins on the MEGA 2560.



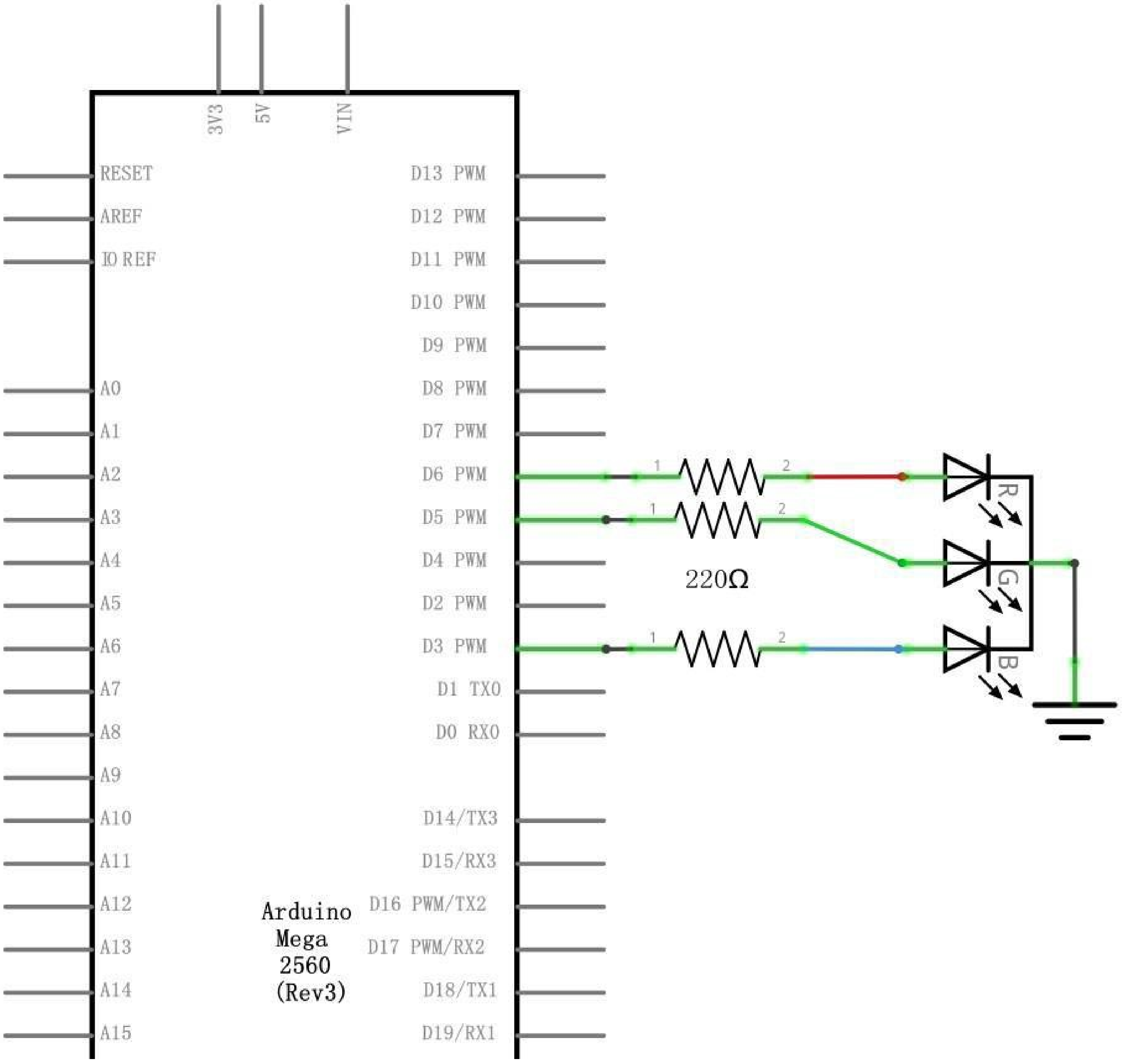
Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analog Write' function. So 'analog Write(0)' will not produce any pulse at all and 'analog Write(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the output is actually on all the time.

If we specify a value in the analog Write that is somewhere in between 0 and 255, then we will produce a pulse. If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power.

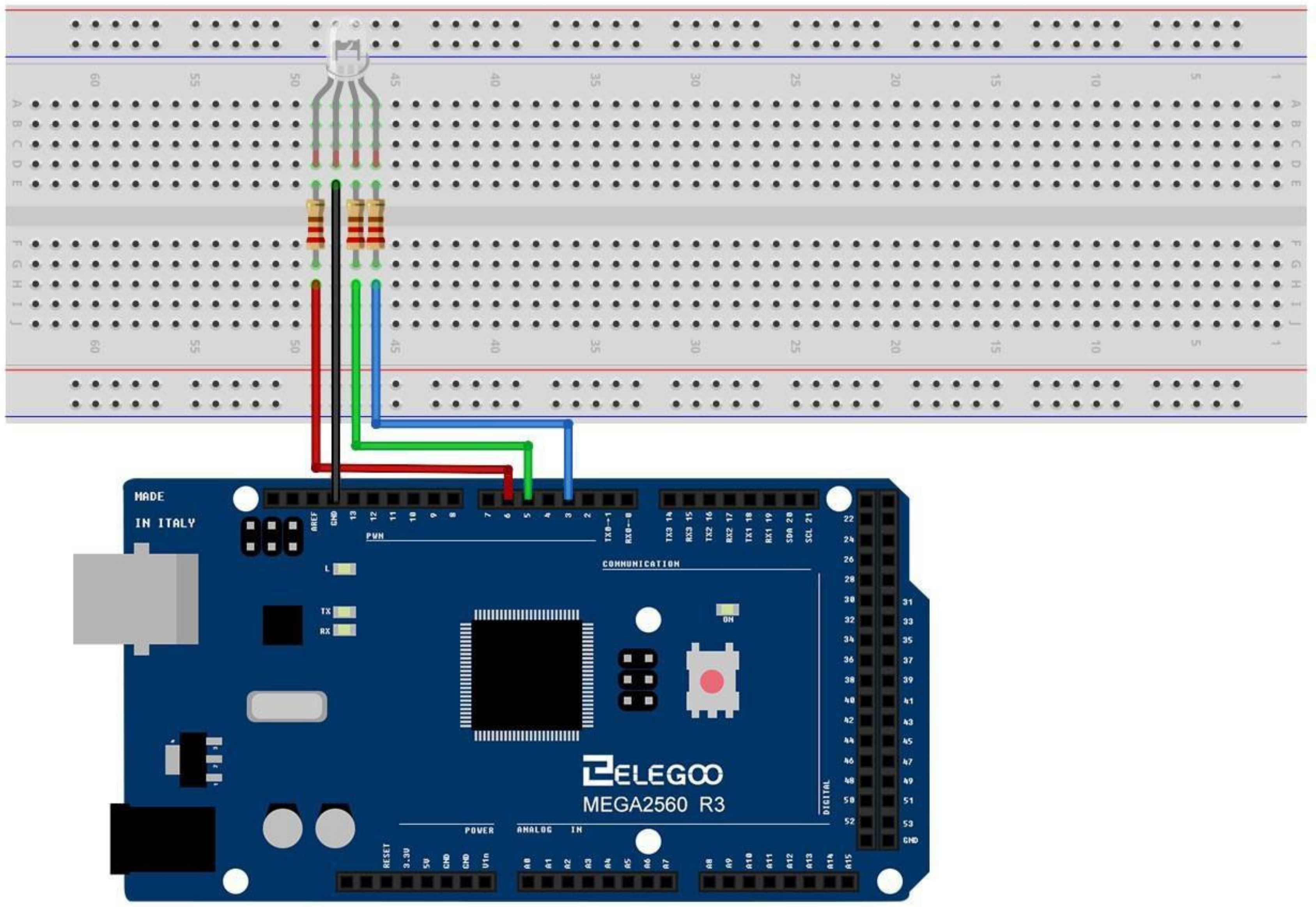
If, however, the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is changing.

Connection

Schematic

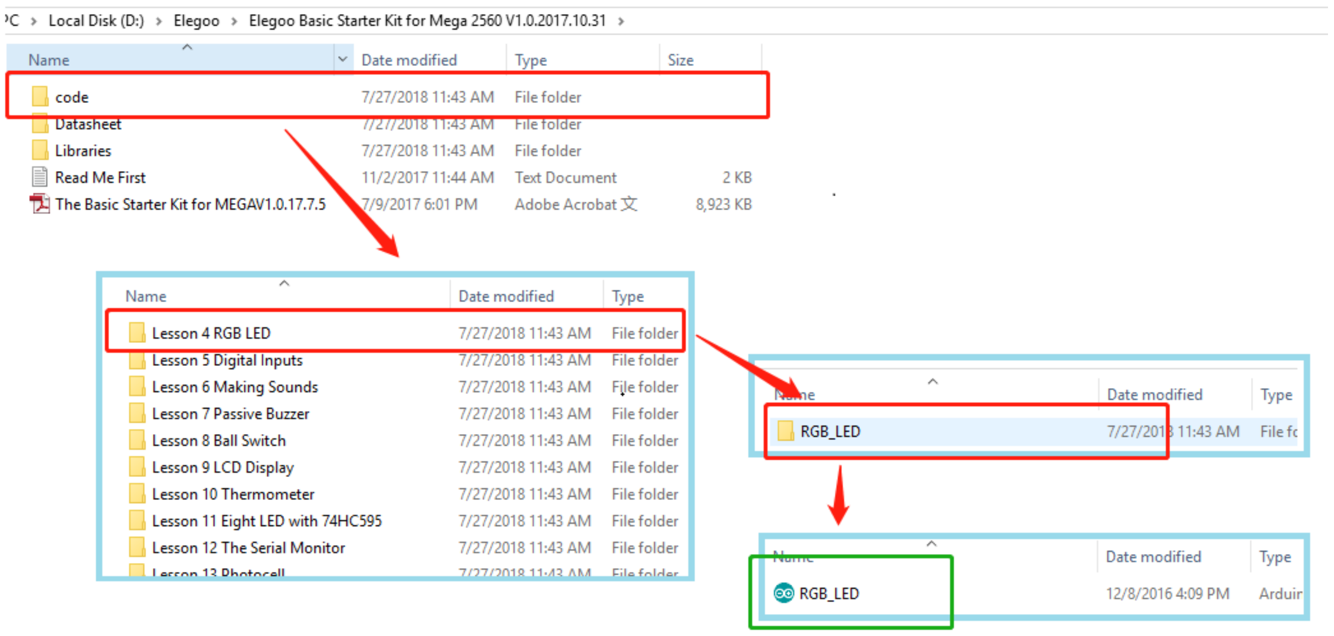


Wiring diagram



Code

After wiring, please open the Sketch in folder path: Tutorial > English > code > Lesson 4 RGB LED > RGB_LED, and click UPLOAD to upload the program.



›C › Local Disk (D:) › Elegoo › Elegoo Basic Starter Kit for Mega 2560 V1.0.2017.10.31 ›

Name	Date modified	Type	Size
code	7/27/2018 11:43 AM	File folder	
Datasheet	7/27/2018 11:43 AM	File folder	
Libraries	7/27/2018 11:43 AM	File folder	
Read Me First	11/2/2017 11:44 AM	Text Document	2 KB
The Basic Starter Kit for MEGAV1.0.17.7.5	7/9/2017 6:01 PM	Adobe Acrobat 文	8,923 KB

Name	Date modified	Type
Lesson 4 RGB LED	7/27/2018 11:43 AM	File folder
Lesson 5 Digital Inputs	7/27/2018 11:43 AM	File folder
Lesson 6 Making Sounds	7/27/2018 11:43 AM	File folder
Lesson 7 Passive Buzzer	7/27/2018 11:43 AM	File folder
Lesson 8 Ball Switch	7/27/2018 11:43 AM	File folder
Lesson 9 LCD Display	7/27/2018 11:43 AM	File folder
Lesson 10 Thermometer	7/27/2018 11:43 AM	File folder
Lesson 11 Eight LED with 74HC595	7/27/2018 11:43 AM	File folder
Lesson 12 The Serial Monitor	7/27/2018 11:43 AM	File folder
Lesson 13 Photocell	7/27/2018 11:43 AM	File folder

Name	Date modified	Type
RGB_LED	7/27/2018 11:43 AM	File fo

Name	Date modified	Type
RGB_LED	12/8/2016 4:09 PM	Arduir

See Lesson 2 for details about program uploading if there are any errors.

The sketch starts by specifying which pins are going to be used for each of the colors:

```
// Define Pins
#define BLUE 3
#define GREEN 5
#define RED 6
```

The next step is to write the 'setup' function. As we have learnt in earlier lessons, the setup function runs just once after the Arduino has reset. In this case, all it has to do is define the three pins we are using as being outputs.

```
void setup()
{
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
digitalWrite(RED, HIGH);
digitalWrite(GREEN, LOW);
digitalWrite(BLUE, LOW);
}
```

Before we take a look at the 'loop' function, let’s look at the last function in the sketch.

The define variables

redValue = 255; // choose a value between 1 and 255 to change the color.

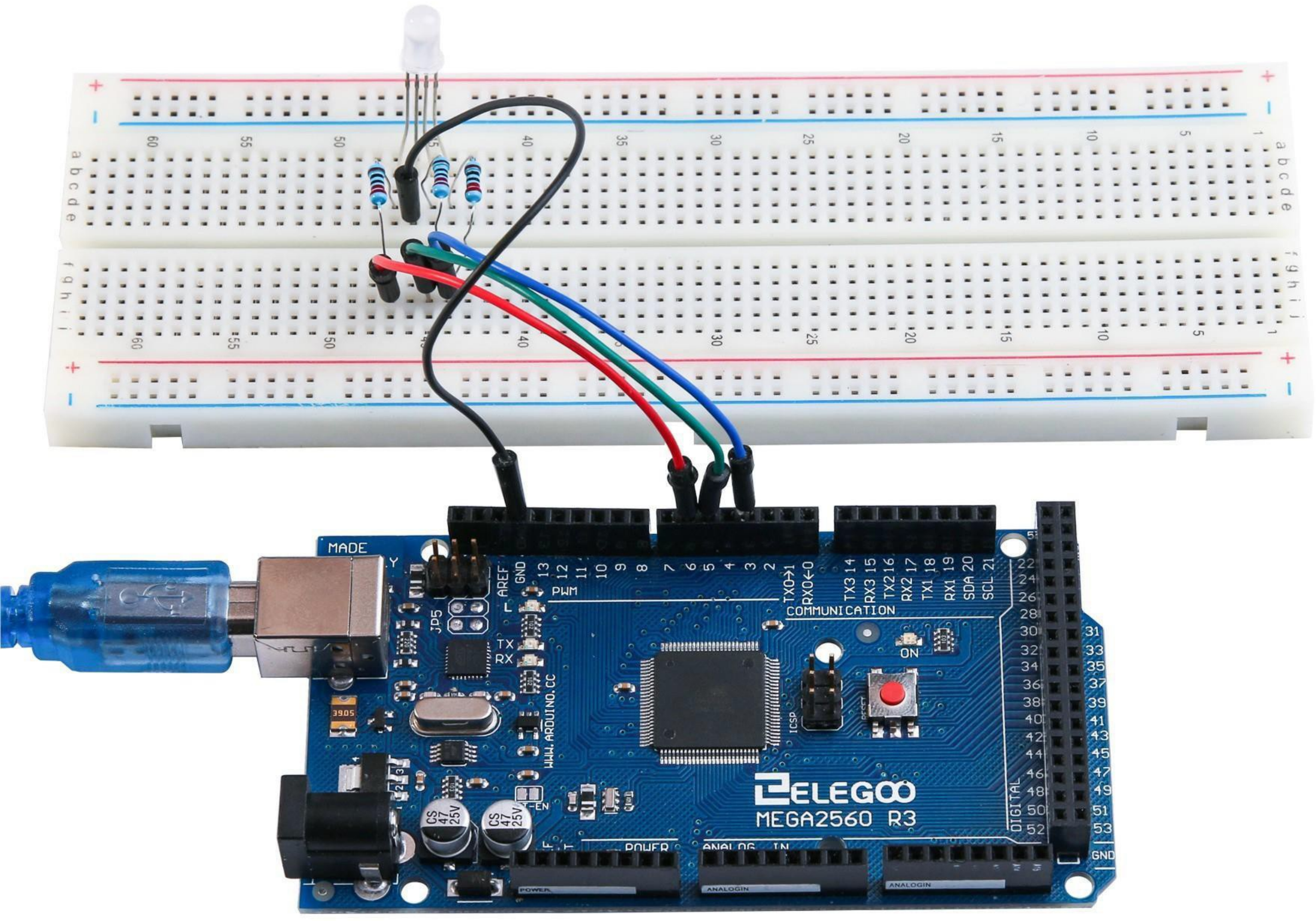
greenValue = 0;

blueValue = 0;

This function takes three arguments, one for the brightness of the red, green and blue LEDs. In each case the number will be in the range 0 to 255, where 0 means off and 255 means maximum brightness. The function then calls 'analogWrite' to set the brightness of each LED.

Try adding a few colors of your own to the sketch and watch the effect on your LED.

Example picture



Lesson 5 Digital Inputs

Overview

In this lesson, you will learn to use push buttons with digital inputs to turn an LED on and off.

Pressing the button will turn the LED on; pressing the other button will turn the LED off.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x 830 Tie-points Breadboard

(1) x 5mm red LED

(1) x 220 ohm resistor

(2) x push switches

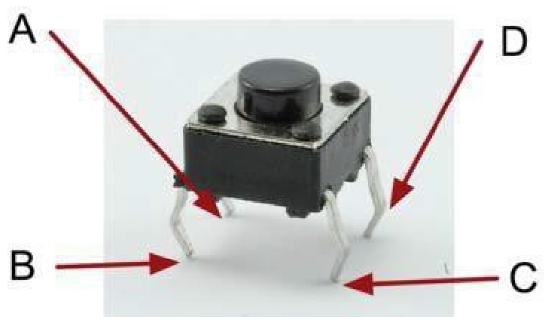
(7) x M-M wires (Male to Male jumper wires)

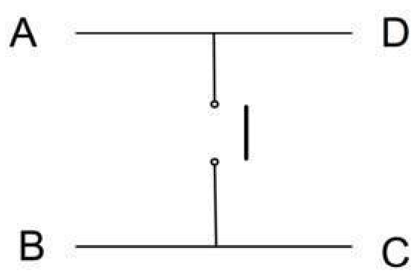
Component Introduction

PUSH SWITCHES:

Switches are really simple components. When you press a button or flip a lever, they connect two contacts together so that electricity can flow through them.

The little tactile switches that are used in this lesson have four connections, which can be a little confusing.

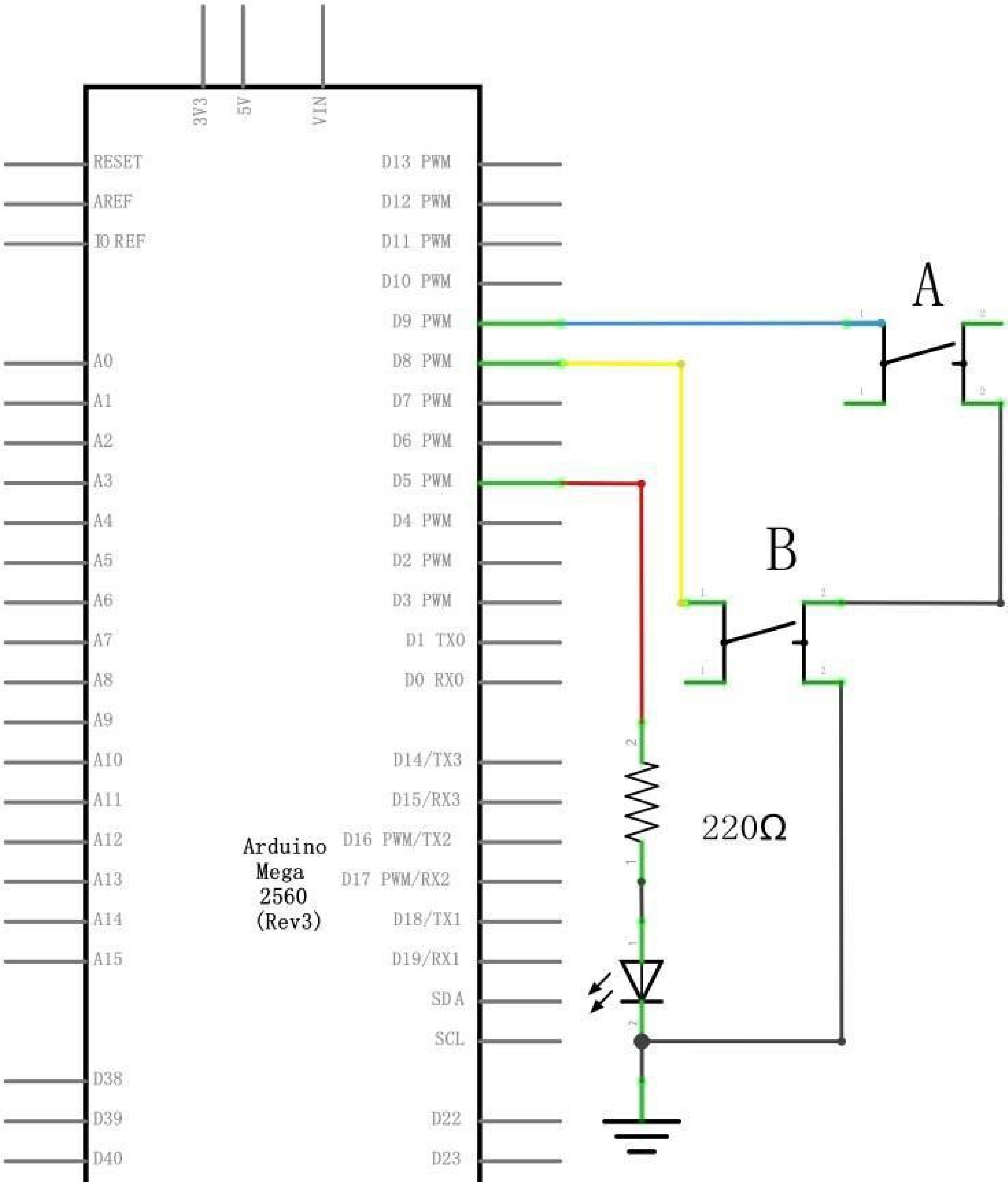




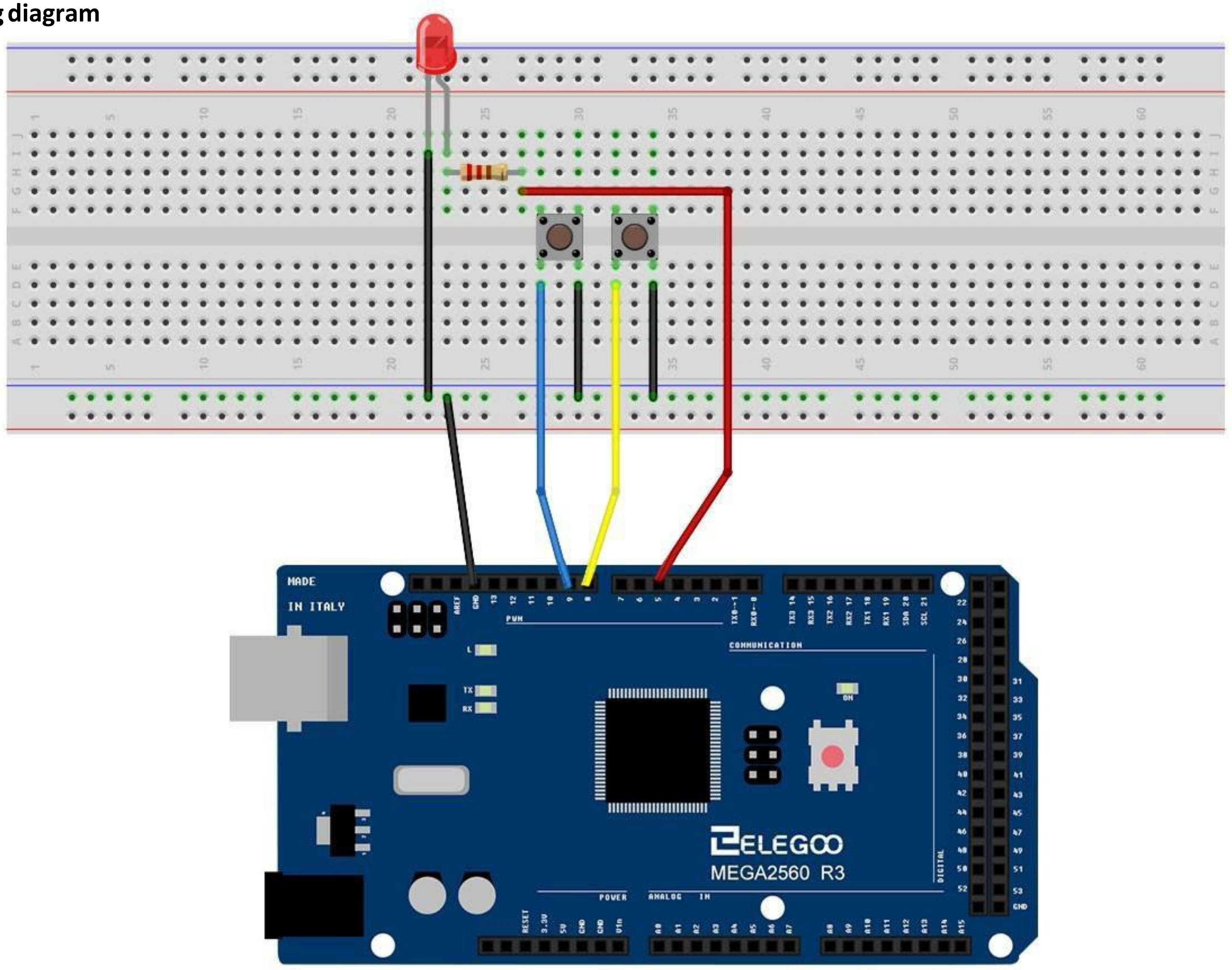
Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

Connection

Schematic



Wiring diagram



Although the bodies of the switches are square, the pins protrude from opposite sides of the switch. This means that the pins will only be far enough apart when they are placed correctly on the breadboard.

Remember that the LED has to have the shorter negative lead to the left.

Code

After wiring please open program in the code folder- Lesson 5 Digital Inputs, and press UPLOAD to upload the program. If errors are prompted, see Lesson 2 for details about the tutorial on program upload.

Load the sketch onto your MEGA2560 board. Pressing the left button will turn the LED on while pressing the right button will turn it off.

The first part of the sketch defines three variables for the three pins that are to be used. The 'ledPin' is the output pin and 'buttonApin' will refer to the switch nearer the top of the breadboard and 'buttonBpin' to the other switch.

The 'setup' function defines the ledPin as being an OUTPUT as normal, but now we have the two inputs to deal with. In this case, we use the set the pinMode to be 'INPUT_PULLUP' like this:

```
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
```

The pin mode of INPUT_PULLUP means that the pin is to be used as an input, but that if nothing else is connected to the input, it should be 'pulled up' to HIGH. In other words, the default value for the input is HIGH, unless it is pulled LOW by the action of pressing the button.

This is why the switches are connected to GND. When a switch is pressed, it connects the input pin to GND, so that it is no longer HIGH.

Since the input is normally HIGH and only goes LOW when the button is pressed, the logic is a little upside down. We will handle this in the 'loop' function.

```
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
```

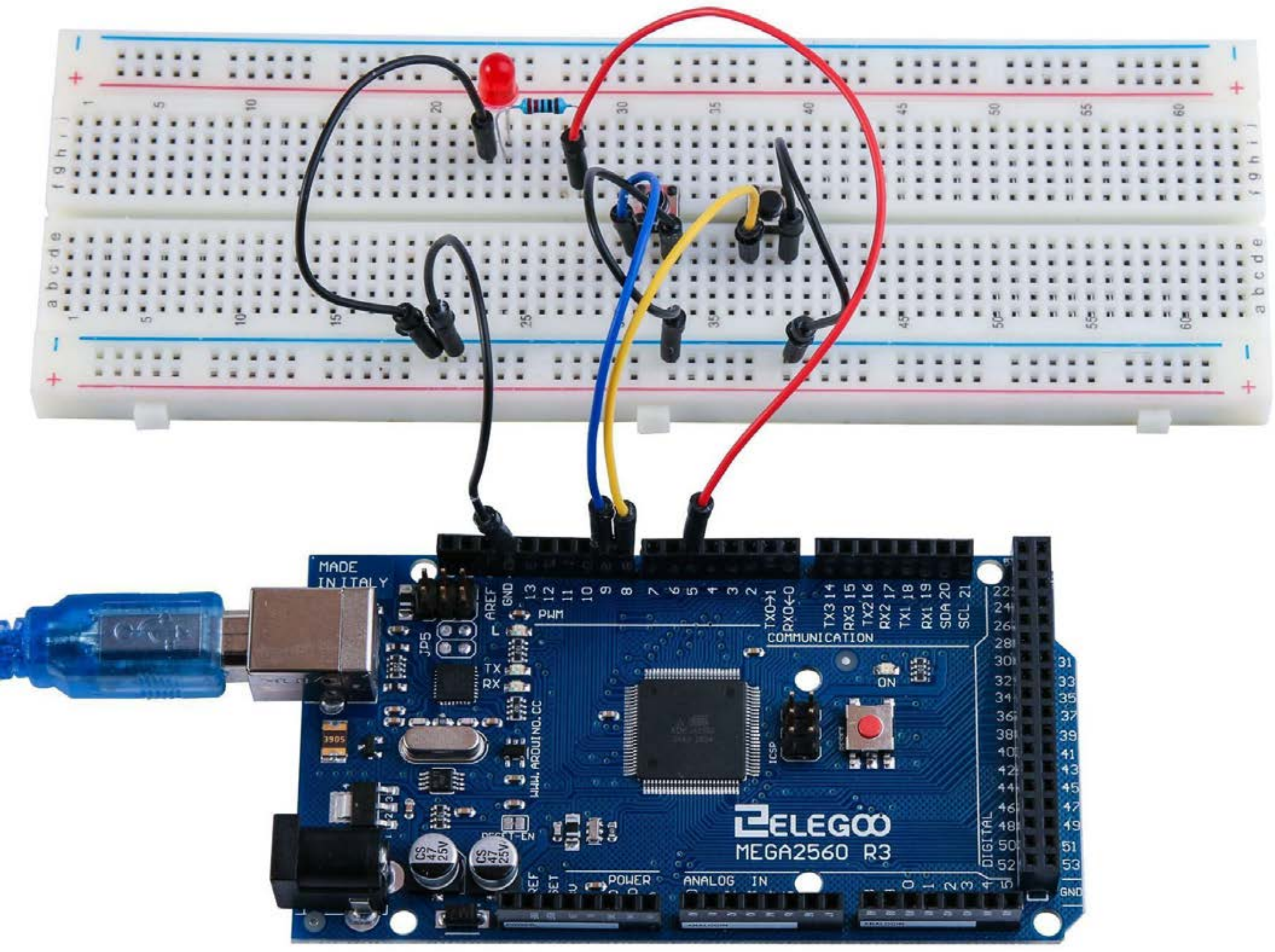
```
    {
        digitalWrite(ledPin, LOW);
    }
}
```

In the 'loop' function there are two 'if' statements. One for each button. Each does an 'digitalRead' on the appropriate input.

Remember that if the button is pressed, the corresponding input will be LOW, if button A is low, then a 'digitalWrite' on the ledPin turns it on.

Similarly, if button B is pressed, a LOW is written to the ledPin.

Example picture



Lesson 6 Active buzzer

Overview

In this lesson, you will learn how to generate a sound with an active buzzer.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x Active buzzer

(2) x F-M wires (Female to Male DuPont wires)

Component Introduction

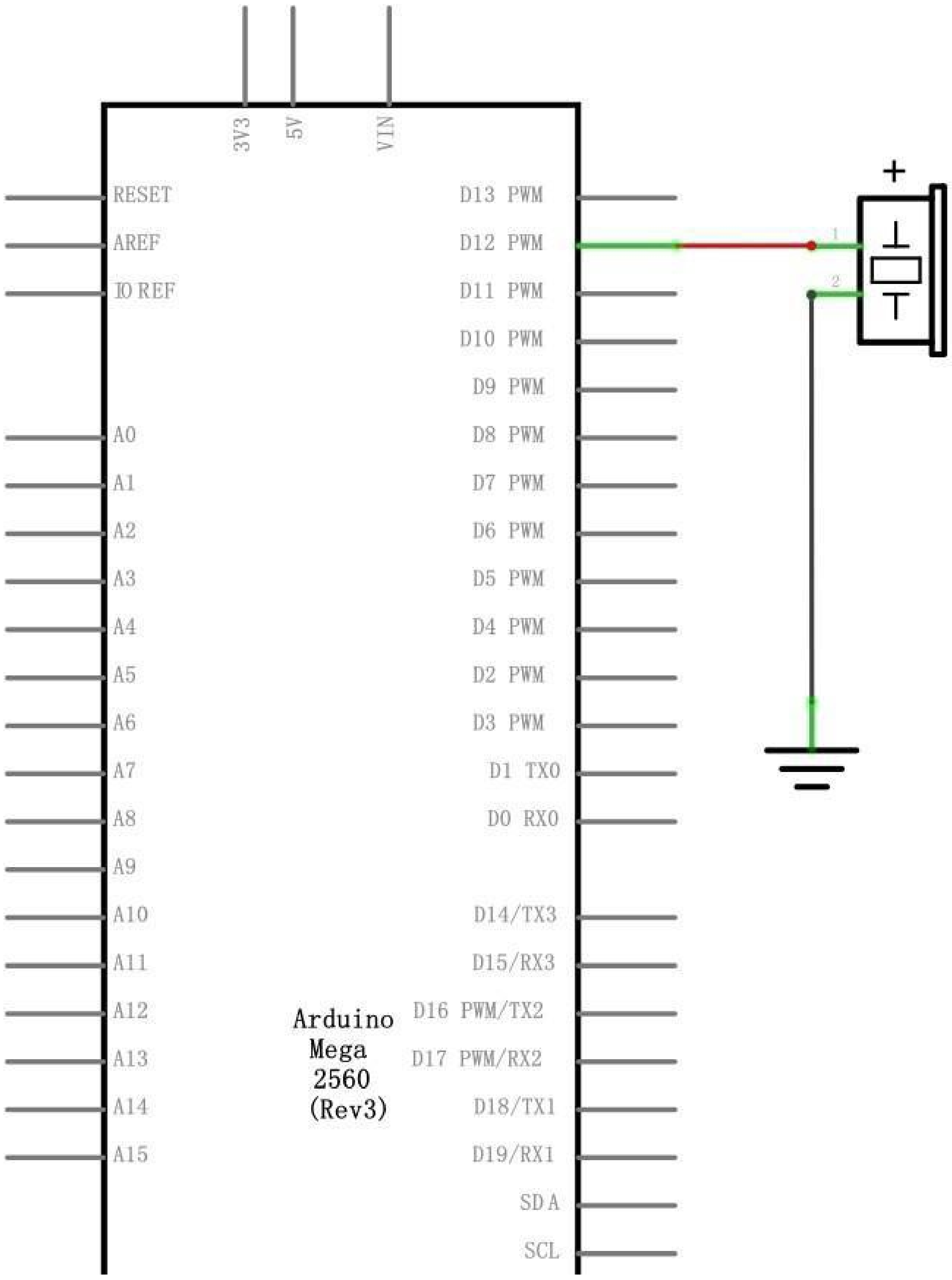
BUZZER:

Electronic buzzers are DC-powered and equipped with an integrated circuit. They are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive ones. Turn the pins of two buzzers face up. The one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

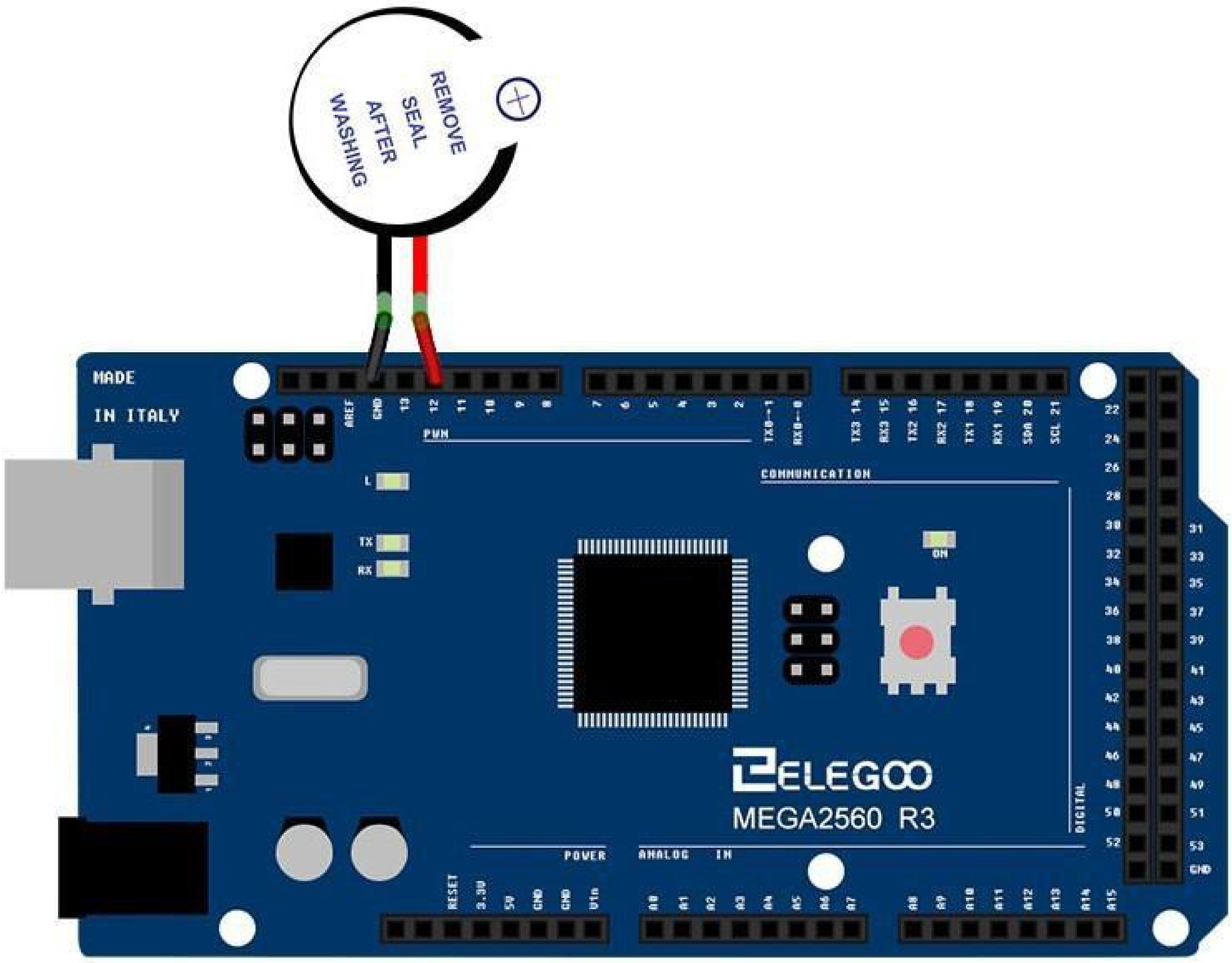
The difference between the two is that an active buzzer has a built-in oscillating source, so it will generate a sound when electrified. A passive buzzer does not have such a source so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

Connection

Schematic



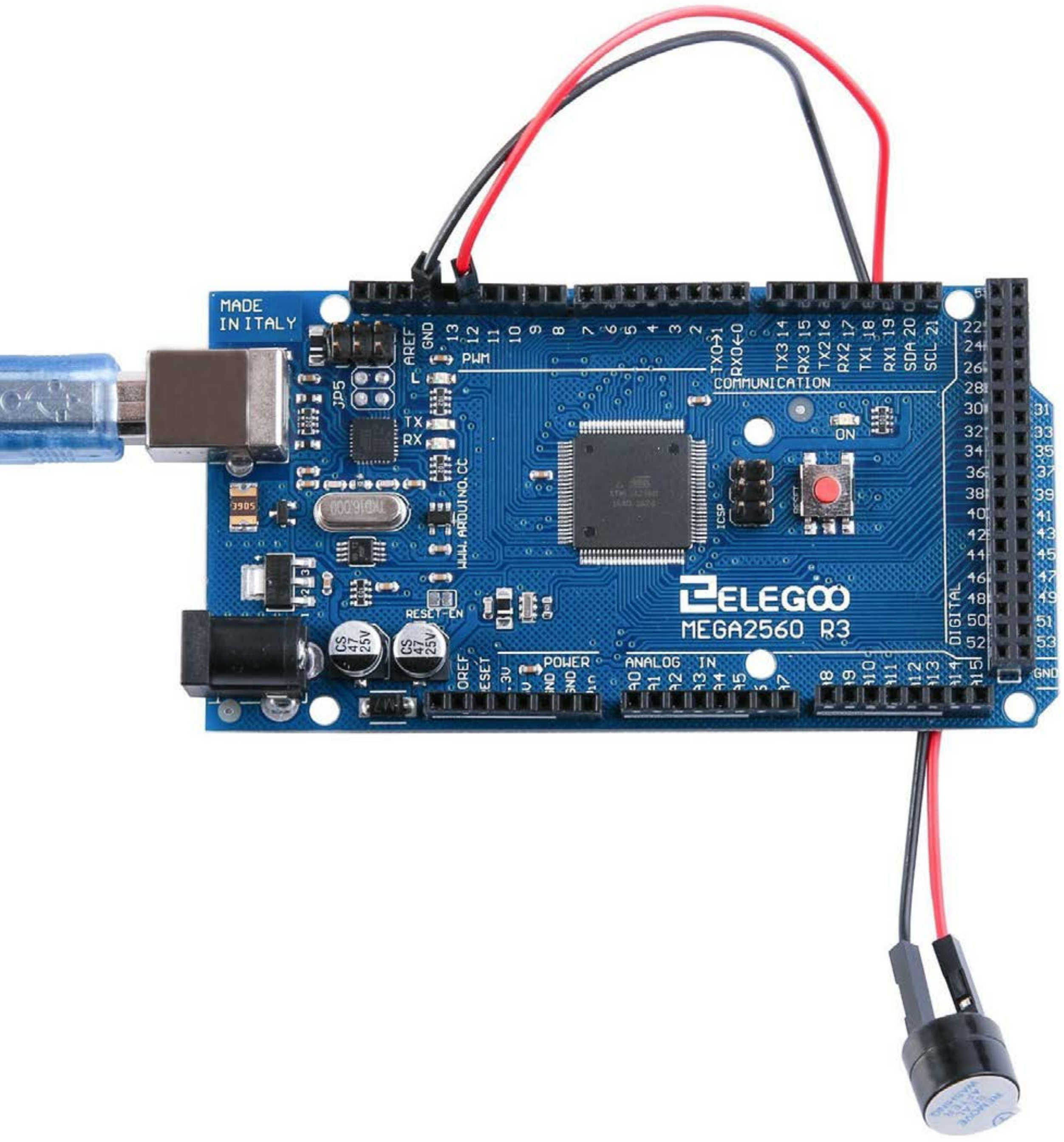
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 6 Making Sounds and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 7 Passive Buzzer

Overview

In this lesson, you will learn how to use a passivebuzzer.

The purpose of the experiment is to generate eight different sounds, each sound lasting 0.5 seconds: from Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz).

Component Required:

(1) x Elegoo Mega2560 R3

(1) x Passive buzzer

(2) x F-M wires (Female to Male DuPont wires)

Component Introduction

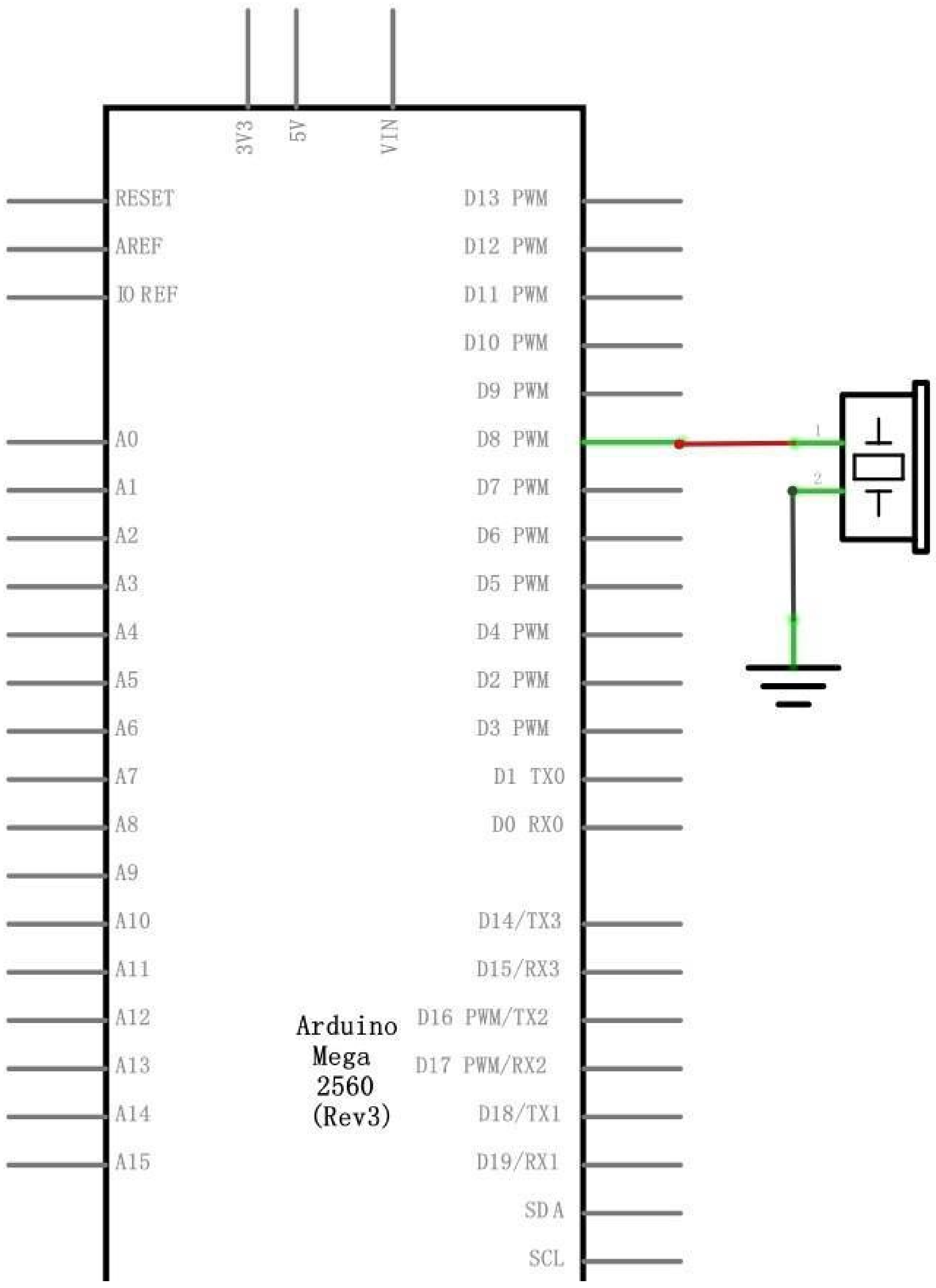
Passive Buzzer:

The working principle of passive buzzer is using PWM generating audio to make the air to vibrate. Appropriately changed as long as the vibration frequency, it can generate different sounds. For example, sending a pulse of 523Hz, it can generate Alto Do, pulse of 587Hz, it can generate midrange Re, pulse of 659Hz, it can produce midrange Mi. By the buzzer, you can play a song.

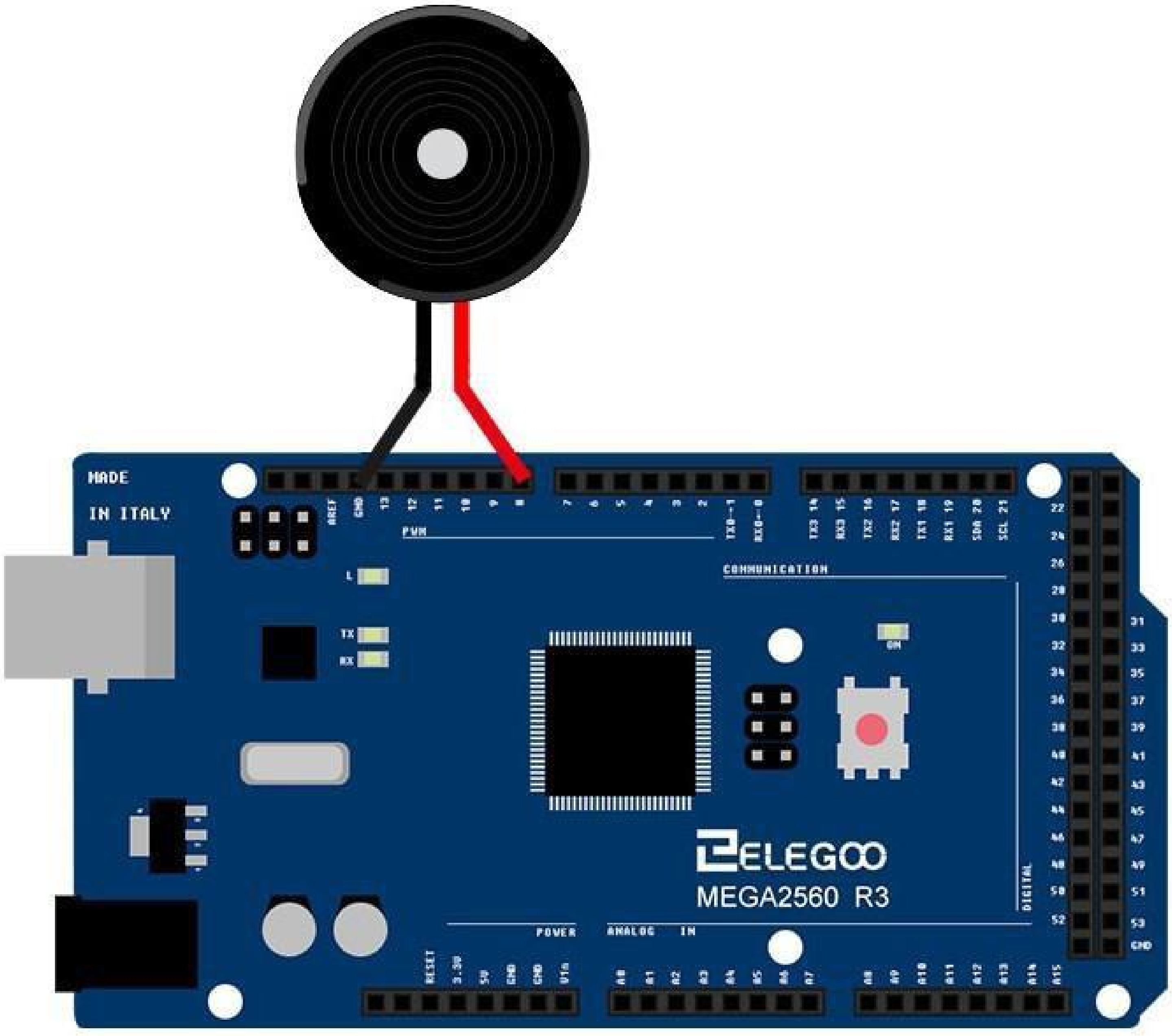
We should be careful not to use the MEGA 2560 R3 board analog Write () function to generate a pulse to the buzzer, because the pulse output of analog Write () is

Connection

Schematic



Wiring diagram



Wiring the buzzer connected to the MEGA 2560 R3 board, the red (positive) to the pin8, black wire (negative) to the GND.

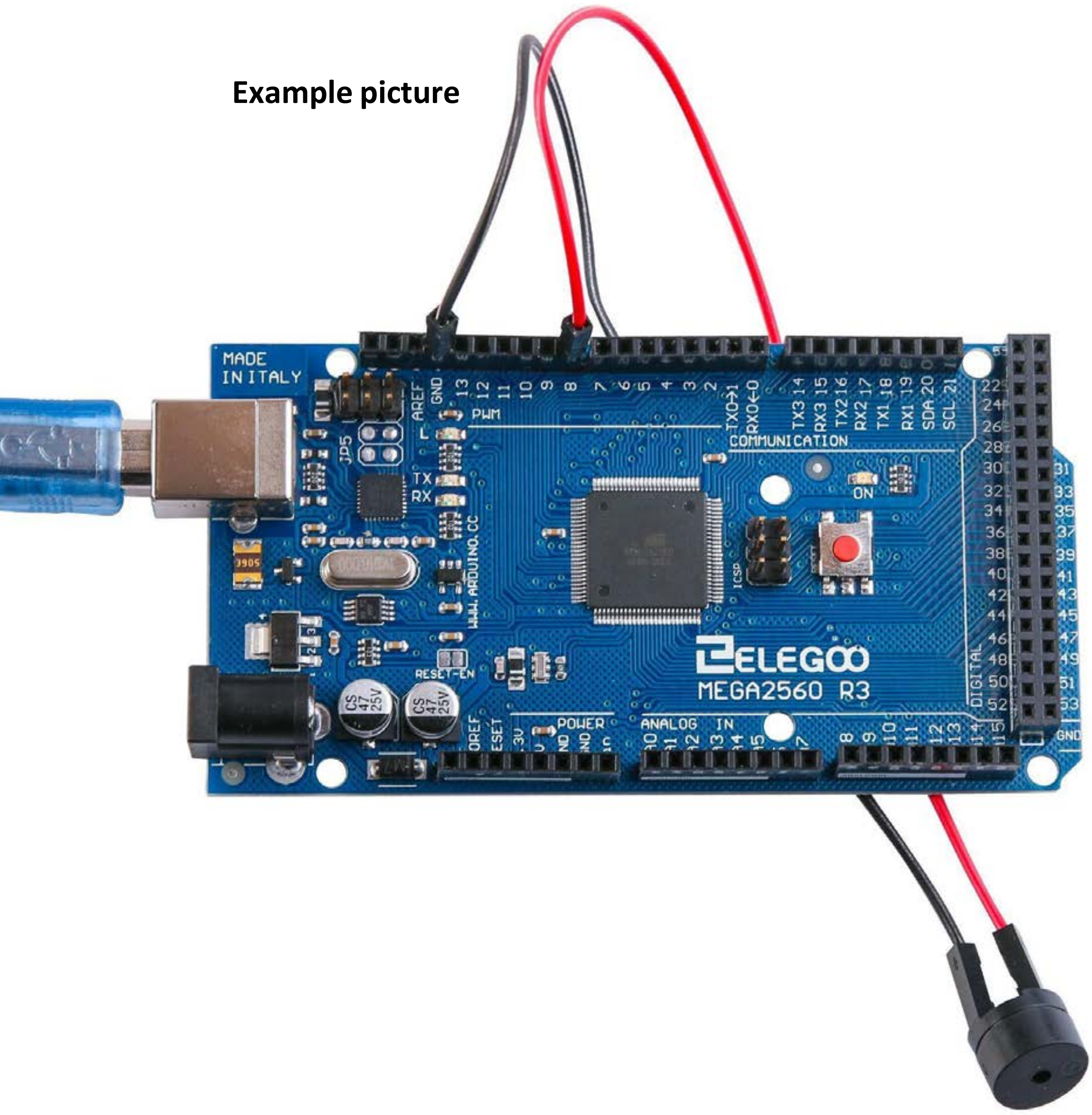
Code

After wiring, please open the program in the code folder- Lesson 7 Passive Buzzer and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the <pitches> library or re-install it, if necessary. Otherwise, your code won't work.

For details about loading the library file, see Lesson 1.

Example picture



Lesson 8 Tilt Ball Switch

Overview

In this lesson, you will learn how to use a tilt ball switch in order to detect small angle of inclination.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x Tilt Ball switch

(2) x F-M wires (Female to Male DuPont wires)

Component Introduction

Tilt sensor:

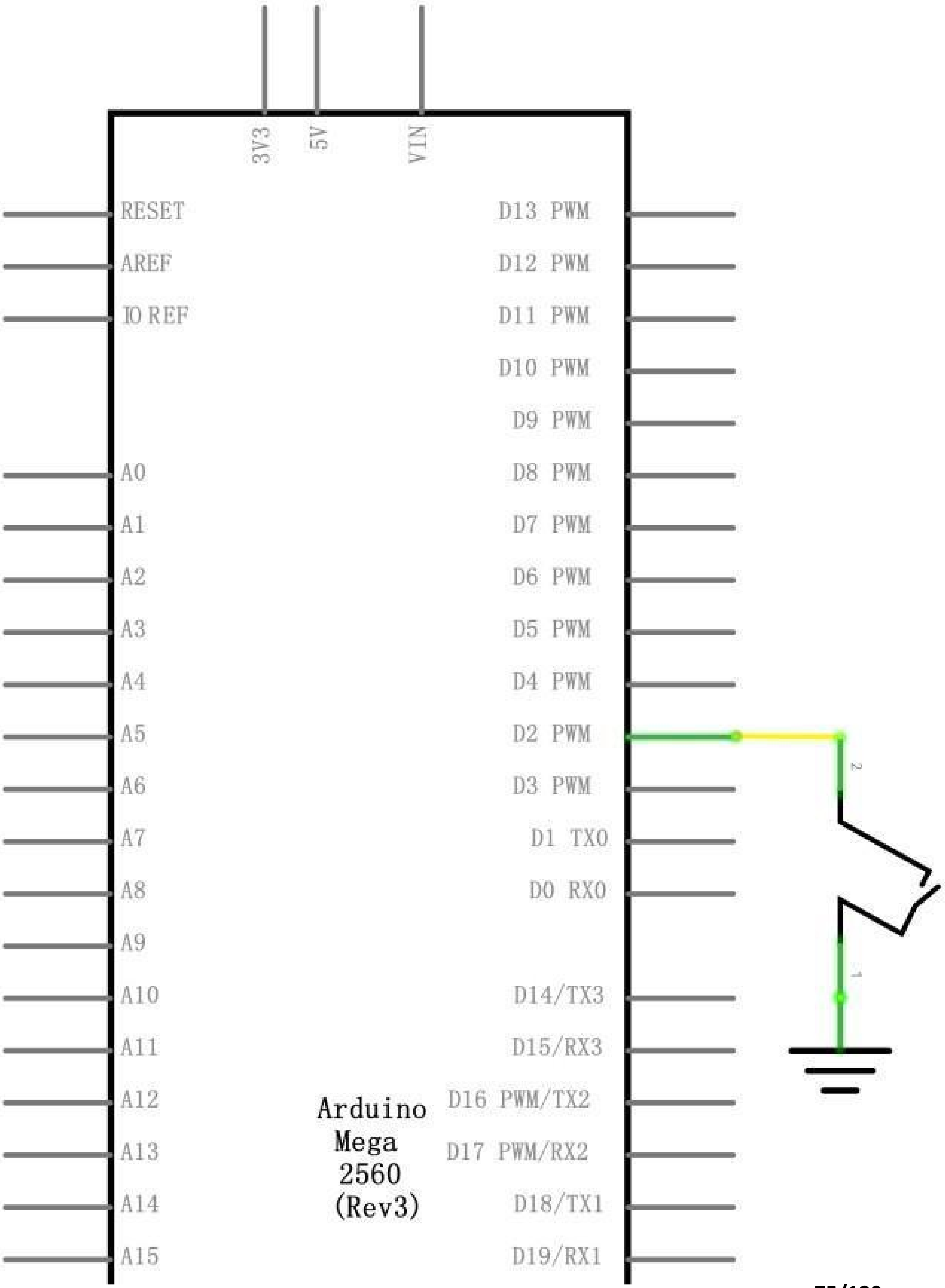
Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out. Their simplicity makes them popular for toys, gadgets and appliances. Sometimes, they are referred to as "mercury switches", "tilt switches" or "rolling ball sensors" for obvious reasons.

They are usually made up of a cavity of some sort (cylindrical is popular, although not always) with a conductive free mass inside, such as a blob of mercury or rolling ball. One end of the cavity has two conductive elements (poles). When the sensor is oriented so that that end is downwards, the mass rolls onto the poles and shorts them, acting as a switch throw.

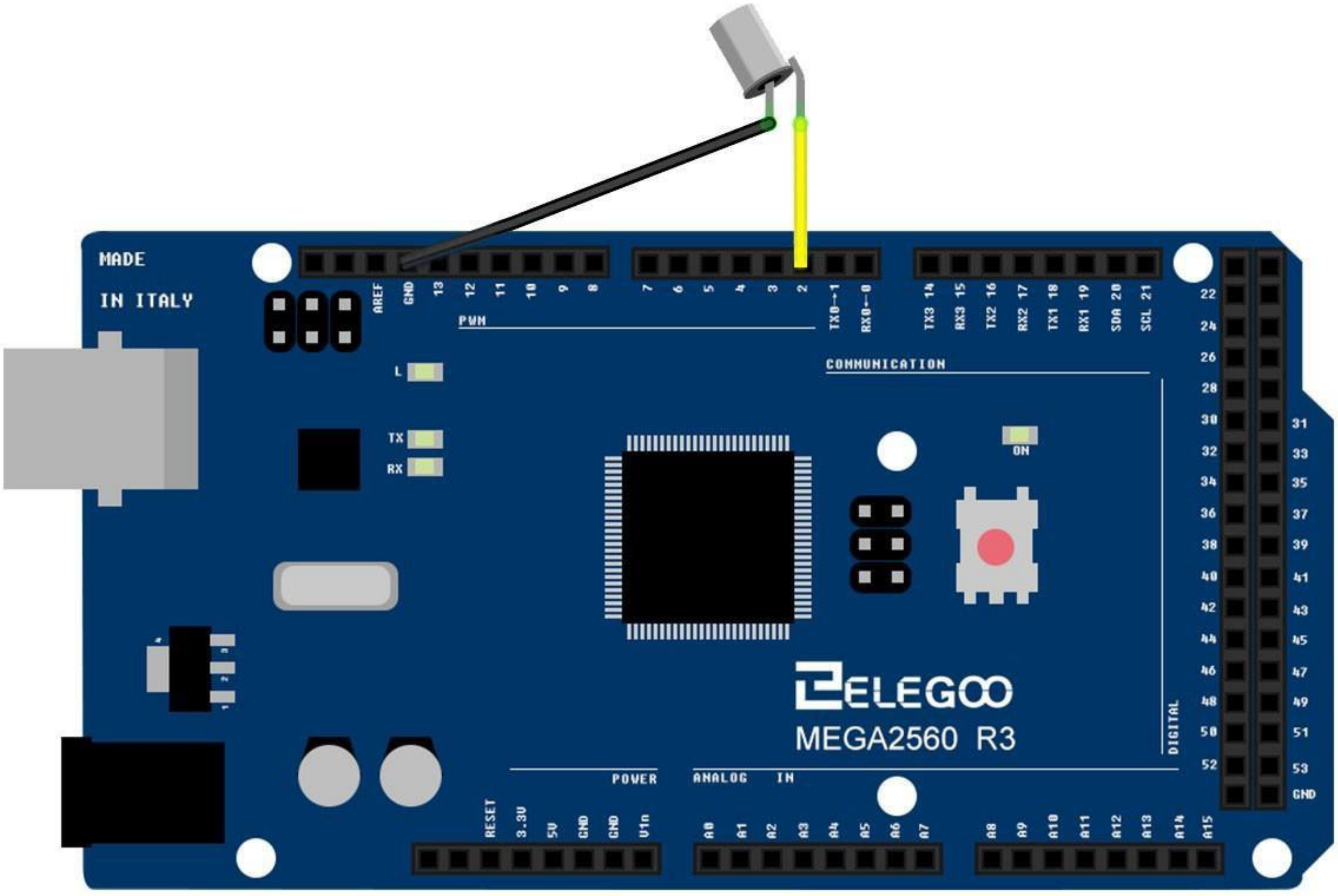
While not as precise or flexible as a full accelerometer, tilt switches can detect motion or orientation. Another benefit is that the big ones can switch power on their own. Accelerometers, on the other hand, output digital or analog voltage that must then be analyzed using extra circuitry.

Connection

Schematic



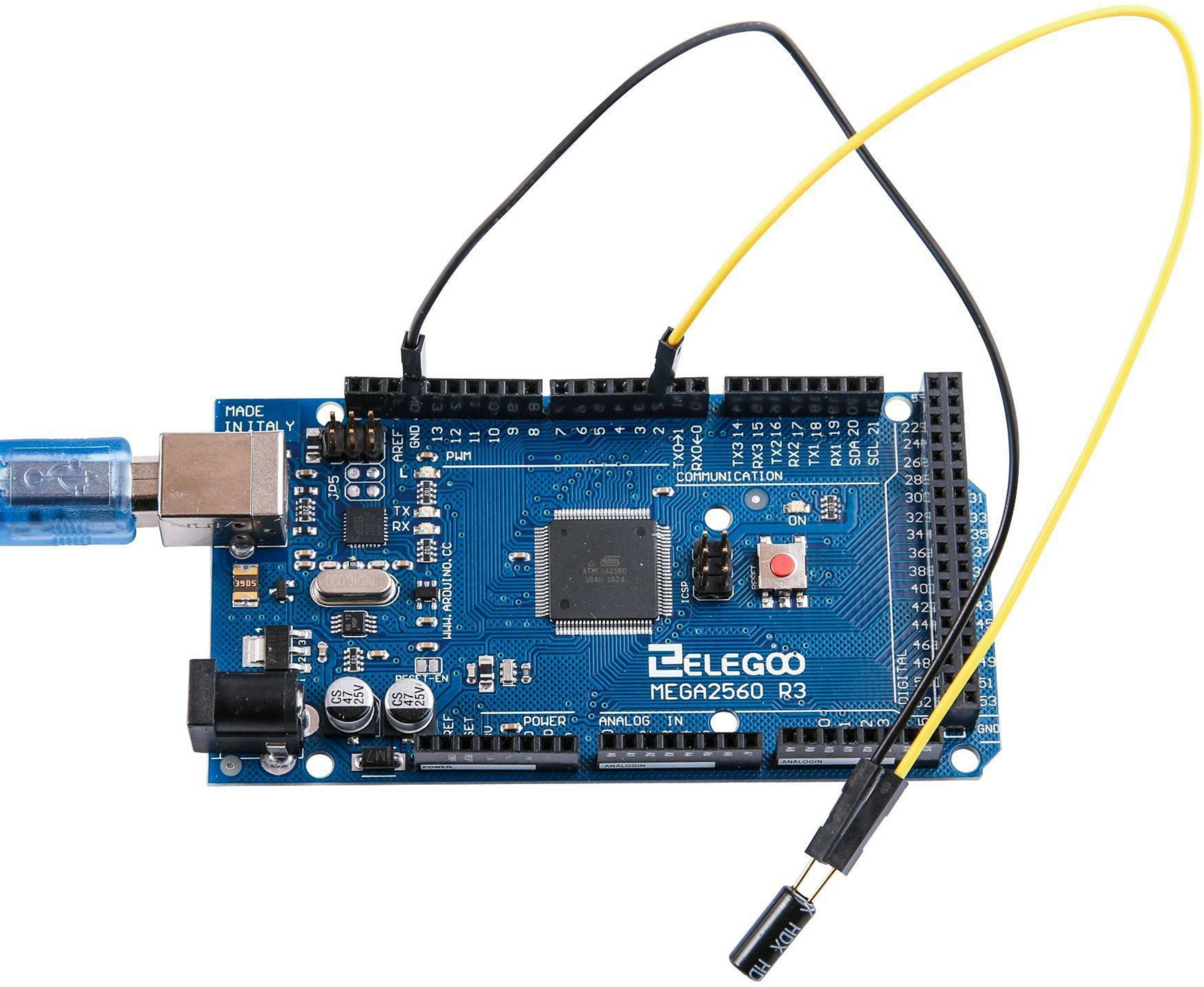
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 8 Ball Switch and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture

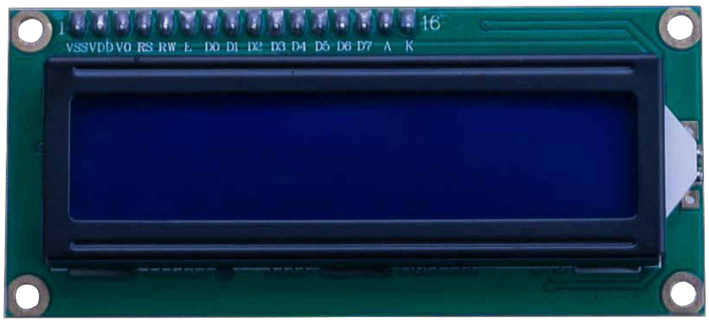


Lesson 9 LCD Display

Overview

In this lesson, you will learn how to wire up and use an alphanumeric LCD display. The display has an LED backlight and can display two rows with up to 16 characters on each row. You can see the rectangles for each character on the display and the pixels that make up each character. The display is just white on blue and is intended for showing text.

In this lesson, we will run the Arduino example program for the LCD library, but in the next lesson, we will get our display to show the temperature, using sensors.



Component Required:

(1) x Elegoo Mega 2560 R3
(1) x LCD1602 module
(1) x Potentiometer (10k)
(1) x 830 tie-points Breadboard
(16) x M-M wires (Male to Male jumper wires)

Component Introduction

LCD1602

Introduction to the pins of LCD1602:

VSS: A pin that connects to ground

VDD: A pin that connects to a +5V power supply

VO: A pin that adjust the contrast of LCD1602

RS: A register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

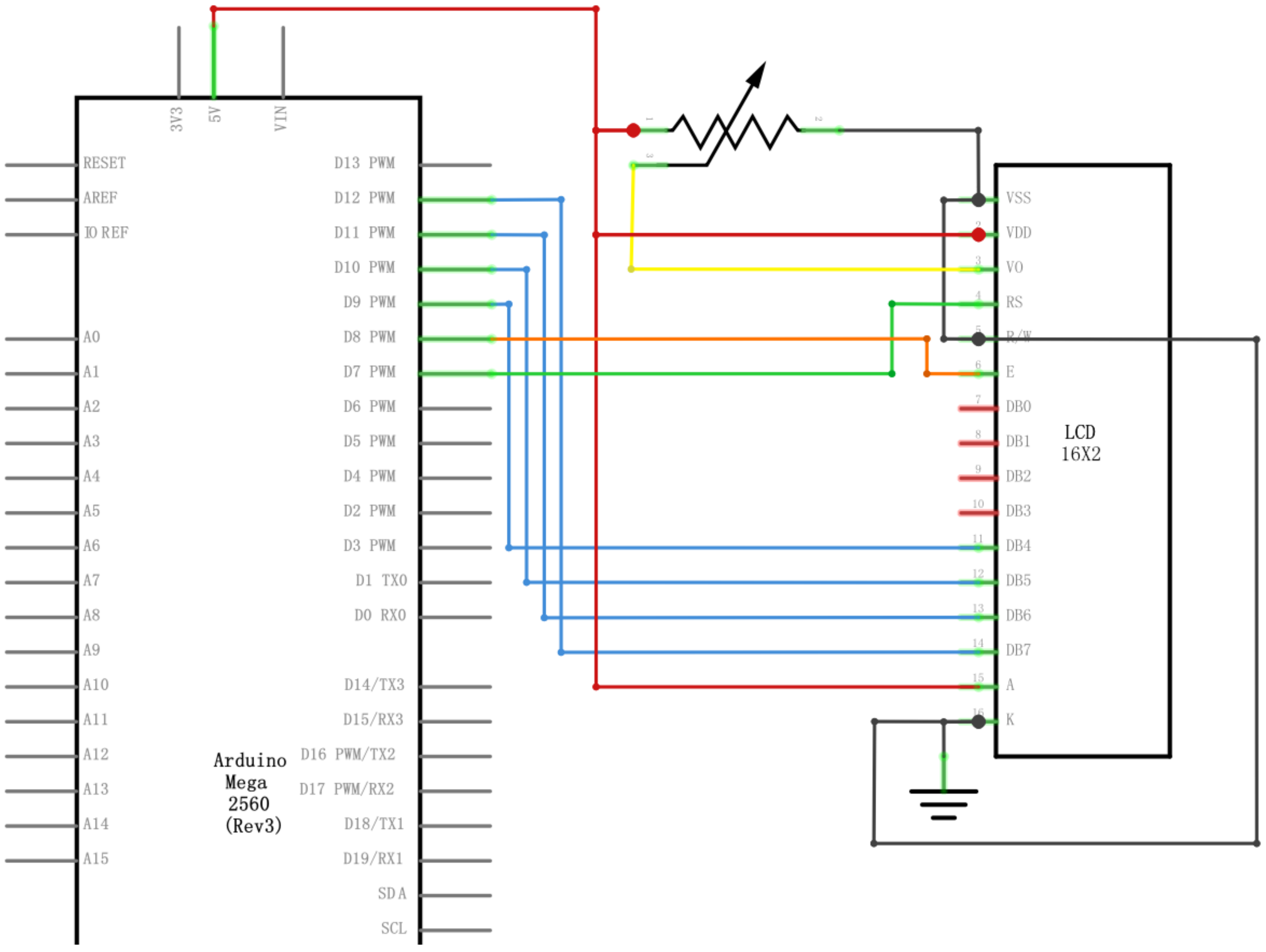
R/W: A Read/Write pin that selects reading mode or writing mode

E: An enabling pin that, when supplied with low-level energy, causes the LDC module to execute relevant instructions.

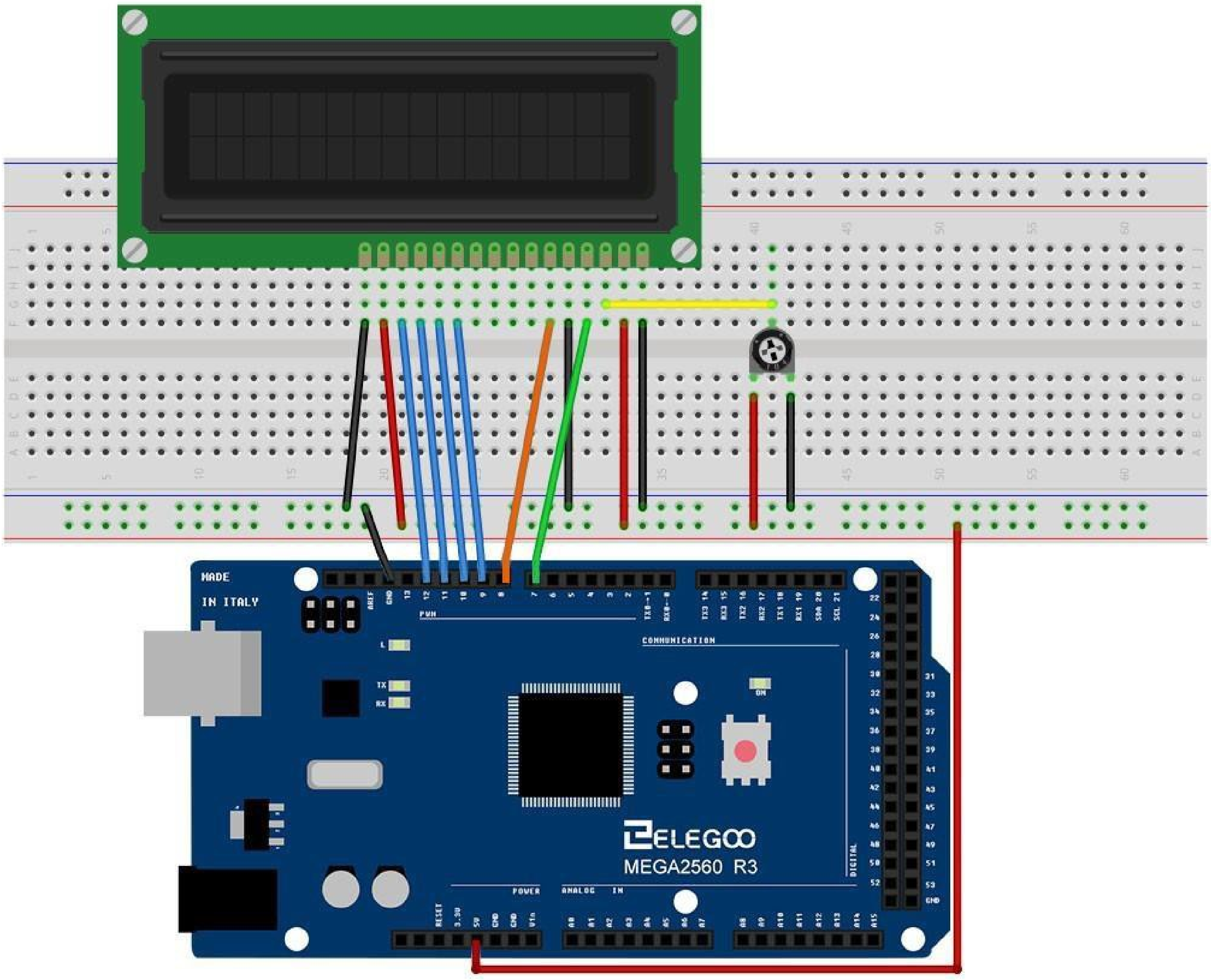
D0-D7： Pins that read and write data

A and K: Pins that control the LED backlight

Connection
Schematic



Wiring diagram



The LCD display needs six Arduino pins, all set to be digital outputs. It also needs 5V and GND connections.
There are a number of connections to be made. Lining up the display with the top of the breadboard helps to identify its pins without too much counting, especially if the breadboard has its rows numbered with row 1 as the top row of the board. Do not forget, the long yellow lead that links the slider of the pot to pin 3 of the display. The 'pot' is used to control the contrast of the display.
You may find that your display is supplied without header pins attached to it. If so, follow the instructions in the next section.

Code

After wiring, please open the program in the code folder- Lesson 9 LCD Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.
Before you can run this, make sure that you have installed the < Liquid Crystal > library or re-install it, if necessary. Otherwise, your code won't work.
For details about loading the library file, see Lesson 1.
Upload the code to your Arduino board and you should see the message 'hello, world' displayed, followed by a number that counts up from zero.
The first thing of note in the sketch is the line:

```
#include <LiquidCrystal.h>
```

This tells Arduino that we wish to use the Liquid Crystal library.
Next we have the line that we had to modify. This defines which pins of the Arduino are to be connected to which pins of the display.

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

After uploading this code, make sure the backlight is lit up, and adjust the potentiometer all the way around until you see the text message
In the 'setup' function, we have two commands:

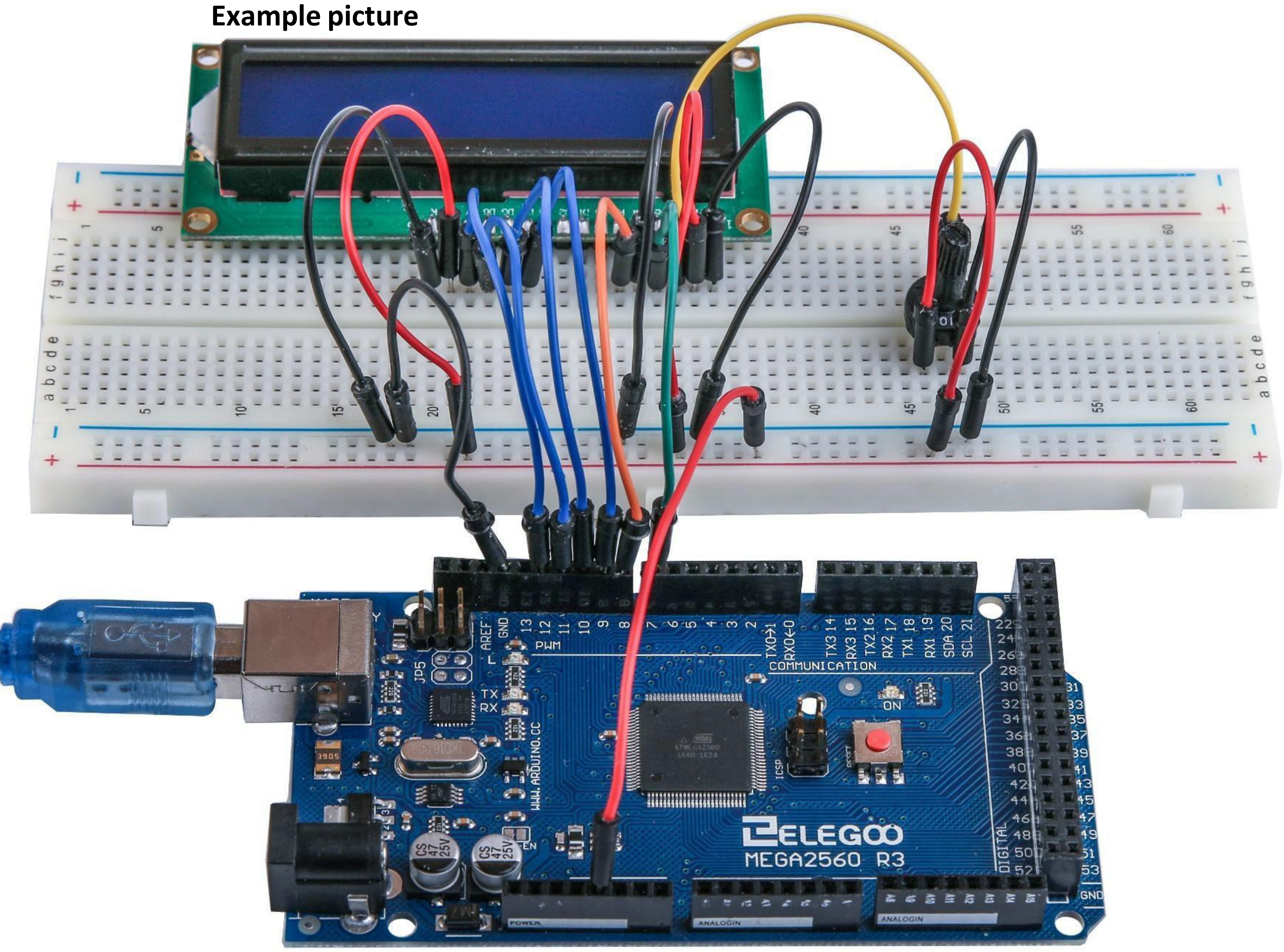
```
lcd.begin(16, 2);
lcd.print("Hello, World!");
```

The first tells the Liquid Crystal library how many columns and rows the display has. The second line displays the message that we see on the first line of the screen.
In the 'loop' function, we aso have two commands:

```
lcd.setCursor(0, 1);
lcd.print(millis()/1000);
```

The first sets the cursor position (where the next text will appear) to column 0 & row 1. Both column and row numbers start at 0 rather than 1.
The second line displays the number of milliseconds since the Arduino was reset.

Example picture



Lesson 10 Thermometer

Overview

In this lesson, you will use an LCD display to show the temperature.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x LCD1602 Module

(1) x 10k ohm resistor

(1) x Thermistor

(1) x Potentiometer

(1) x 830 tie-points Breadboard

(18) x M-M wires (Male to Male jumperwires)

Component Introduction

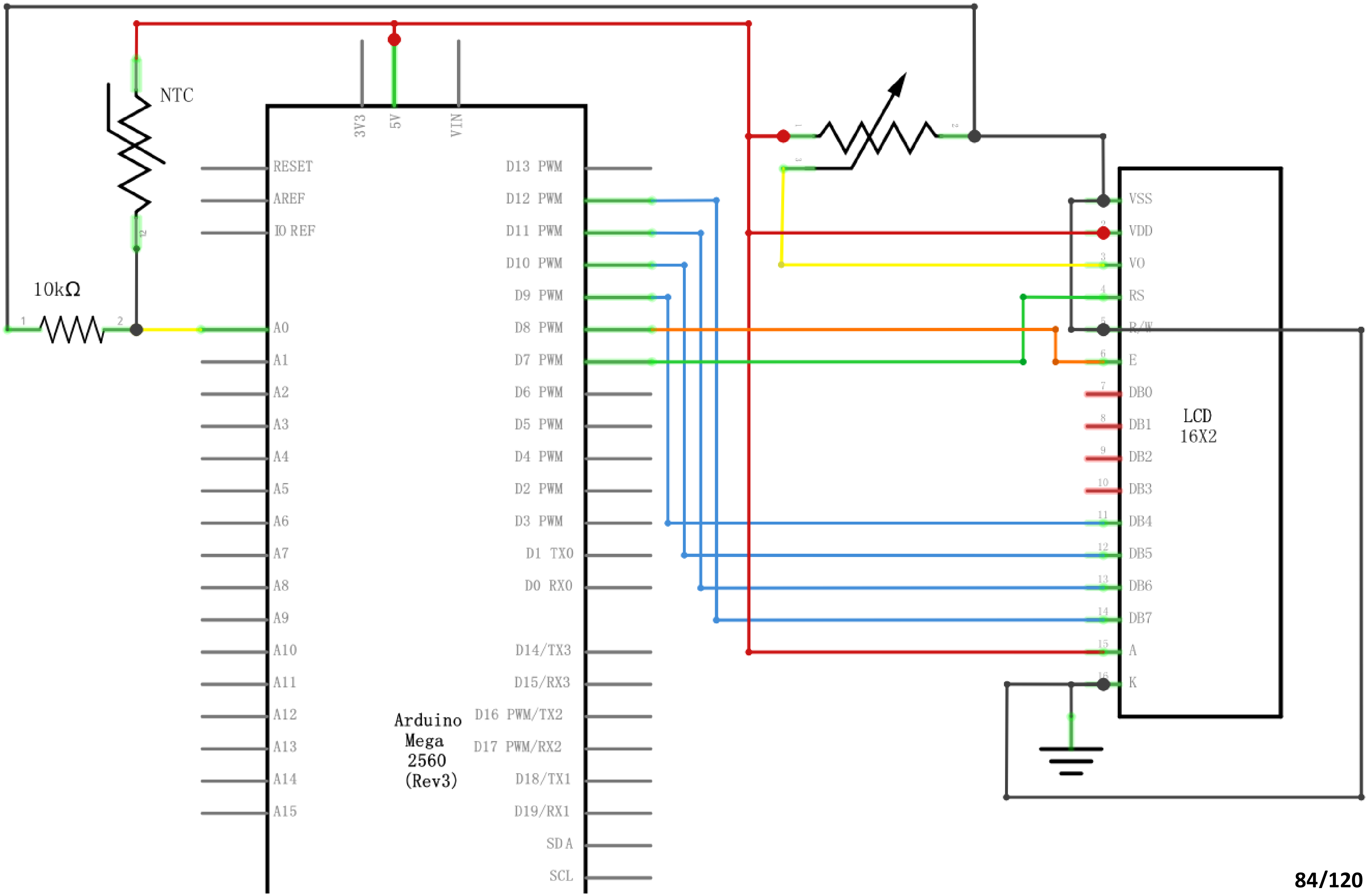
Thermistor

A thermistor is a thermal resistor - a resistor that changes its resistance with temperature. Technically, all resistors are thermistors - their resistance changes slightly with temperature - but the change is usually very small and difficult to measure. Thermistors are made so that the resistance changes drastically with temperature so that it can be 100 ohms or more of change per degree!

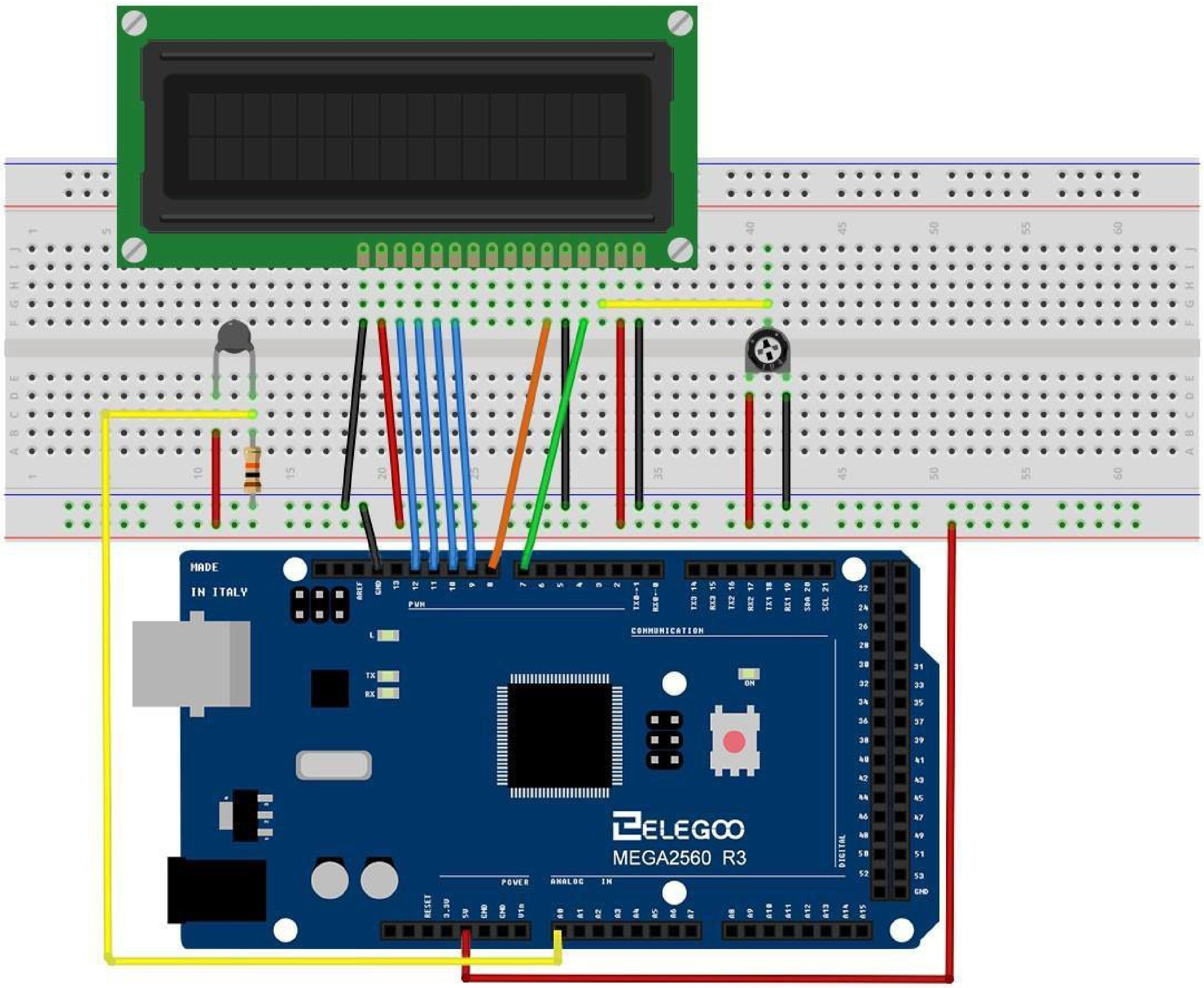
There are two kinds of thermistors, NTC (negative temperature coefficient) and PTC (positive temperature coefficient). In general, you will see NTC sensors used for temperature measurement. PTC's are often used as resettable fuses - an increase in temperature increases the resistance which means that as more current passes thru them, they heat up and 'choke back' the current, quite handy for protectingcircuits

Connection

Schematic



Wiring diagram



The breadboard layout is based on the layout from Lesson 22, so it will simplify things if you still have this on the breadboard.

There are a few jumper wires near the pot that have been moved slightly on this layout.

The 10 kΩ resistor and thermistor are all new additions to the board.

Code

After wiring, please open the program in the code folder- Lesson 10 Thermometer and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < LiquidCrystal > library or re-install it, if necessary. Otherwise, your code won't work.

For details about loading the library file, see Lesson 1.

The sketch for this is based on that of lesson 22. Load it up onto your Arduino and you should find that warming the temperature sensor by putting your finger on it will increase the temperature reading.

I find it useful to put a comment line above the 'lcd' command.

```
// BS   E   D4 D5   D6  D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

This makes things easier if you decide to change which pins you use.

In the 'loop' function there are now two interesting things going on. Firstly we have to convert the analog from the temperature sensor into an actual temperature, and secondly we have to work out how to display them.

First of all, let's look at calculating the temperature.

```
  int tempReading = analogRead(tempPin);
  double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
 tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK * tempK ))
  * tempK );
  float tempC = tempK - 273.15;
  float tempF = (tempC * 9.0)/ 5.0 + 32.0;
```

Displaying changing readings on an LCD display can be tricky. The main problem is that the reading may not always be the same number of digits. So, if the temperature changed from 101.50 to 99.00 then the extra digit from the old reading is in danger of being left on the display.

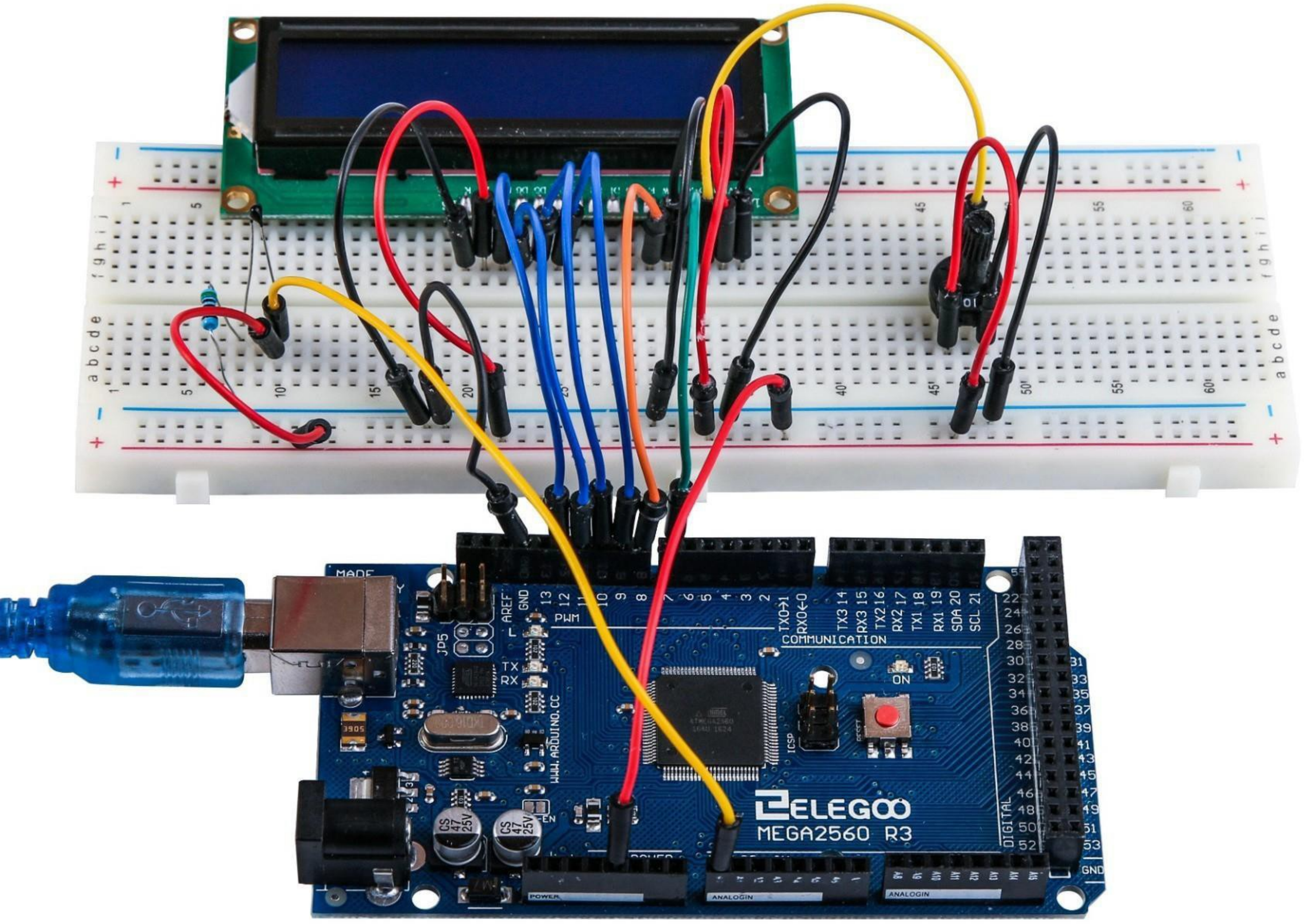
To avoid this, write the whole line of the LCD each time around the loop.

```
  lcd.setCursor(0, 0);
 lcd.print("Temp         C  ");
 lcd.setCursor(6, 0);
 lcd.print(tempF);
```

The rather strange comment serves to remind you of the 16 columns of the display. You can then print a string of that length with spaces where the actual reading will go.

To fill in the blanks, set the cursor position for where the reading should appear and then print it.

Example picture



Lesson 11 Eight LED with 74HC595

Overview

In this lesson, you will learn how to use eight large red LEDs with an MEGA2560 without needing to give up 8 output pins!

Although you could wire up eight LEDs each with a resistor to an MEGA2560 pin you would rapidly start to run out of pins on your MEGA2560. If you don't have a lot of stuff connected to your MEGA2560. It's OK to do so - but often times we want buttons, sensors, servos, etc. and before you know it you've got no pins left. So, instead of doing that, you are going to use a chip called the 74HC595 Serial to Parallel Converter. This chip has eight outputs (perfect) and three inputs that you use to feed data into it a bit at a time.

This chip makes it a little slower to drive the LEDs (you can only change the LEDs about 500,000 times a second instead of 8,000,000 a second) but it's still really fast, way faster than humans can detect, so it's worth it!

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x 830 tie-points breadboard

(8) x leds

(8) x 220 ohm resistors

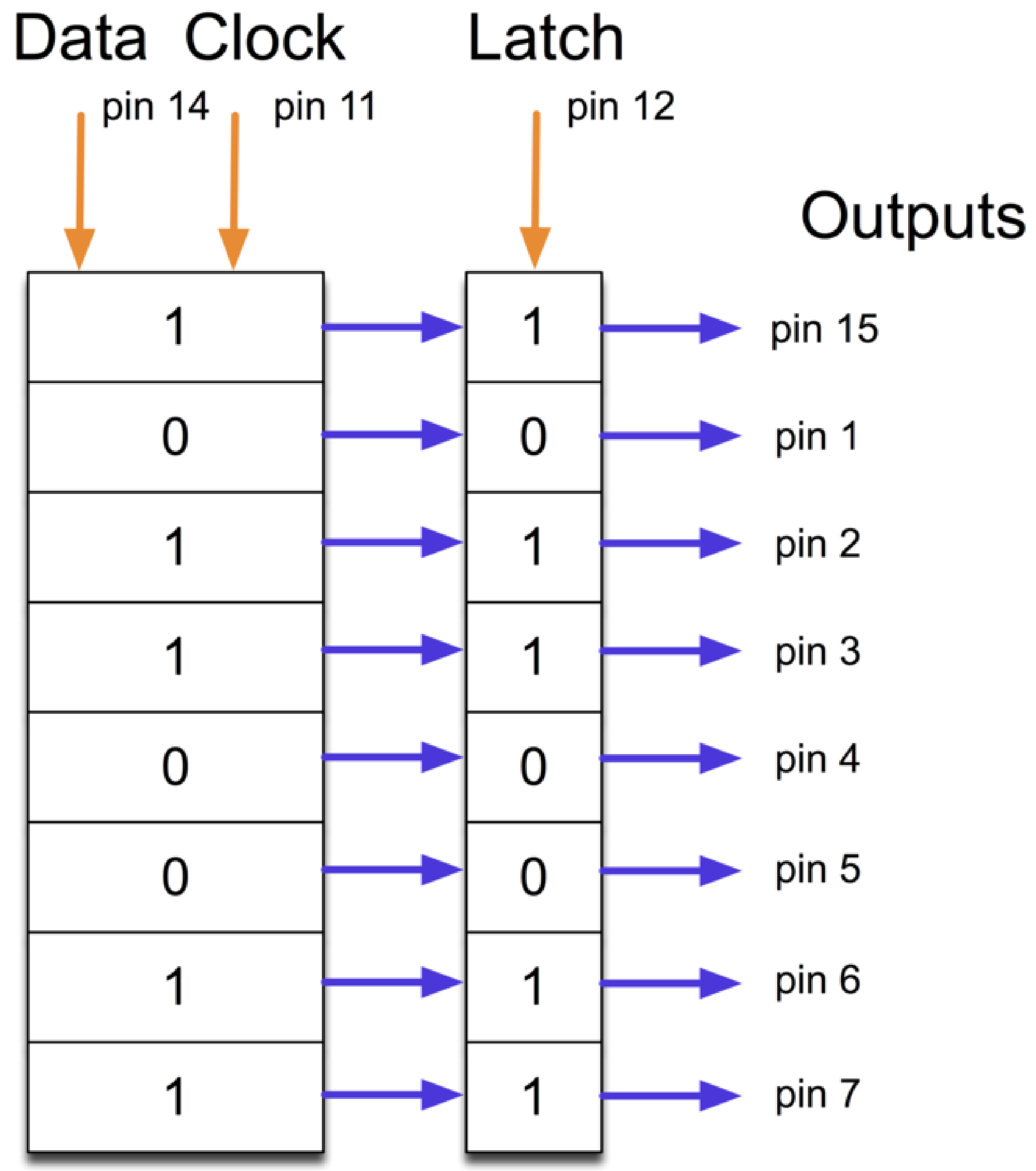
(1) x 74hc595 IC

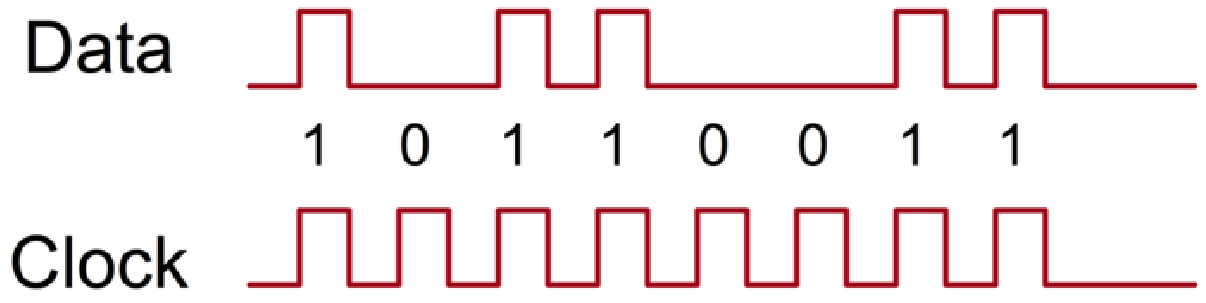
(14) x M-M wires (Male to Male jumper wires)

Component Introduction

74HC595 Shift Register:

The shift register is a type of chip that holds what can be thought of as eight memory locations, each of which can either be a 1 or a 0. To set each of these values on or off, we feed in the data using the 'Data' and 'Clock' pins of the chip.


Data
Clock
Latch
pin 14
pin 11
pin 12
Outputs
1
0
1
1
0
0
1
1
1
0
1
1
0
0
1
1
pin 15
pin 1
pin 2
pin 3
pin 4
pin 5
pin 6
pin 7

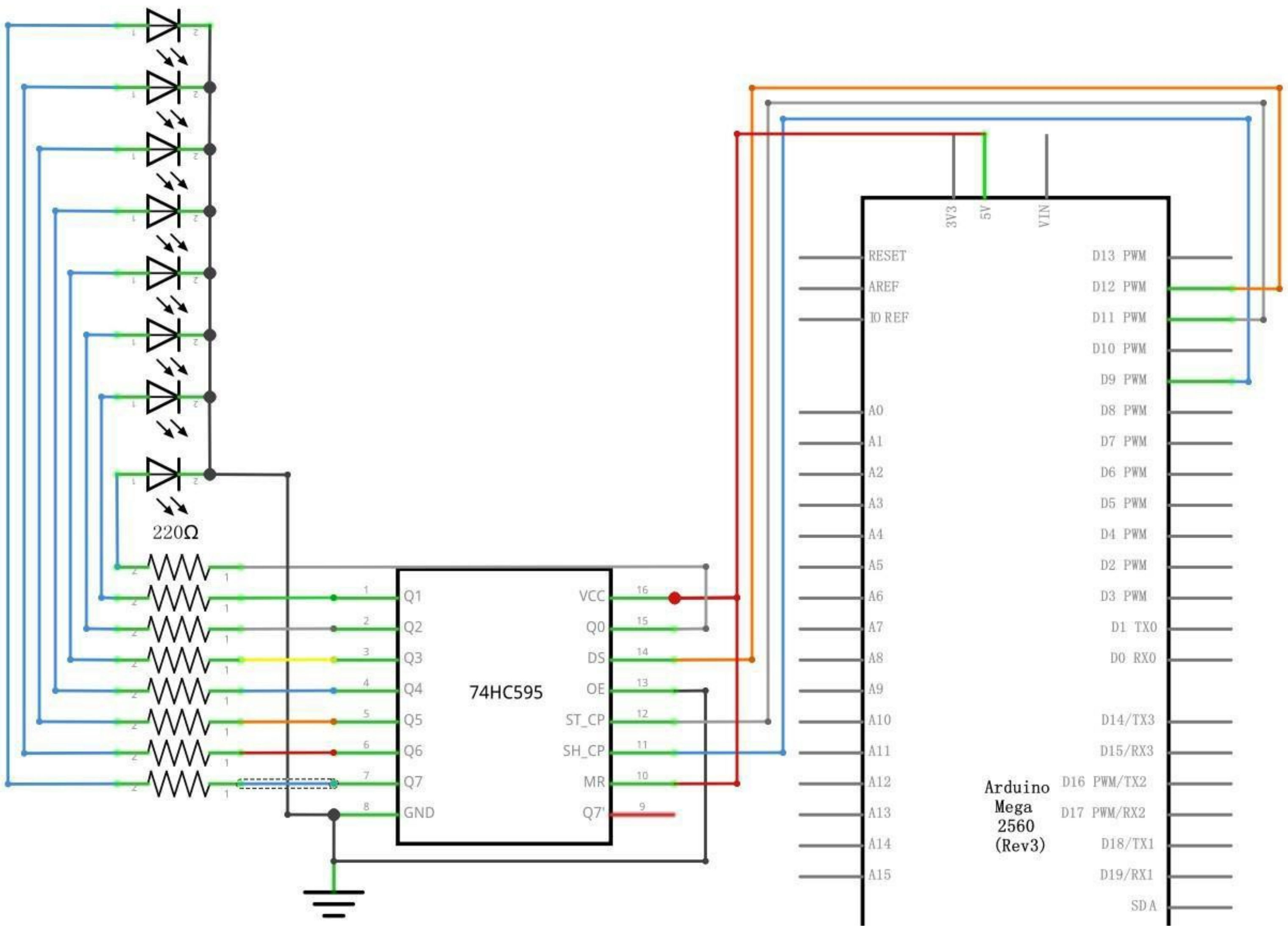

Data
1 0 1 1 0 0 1 1
Clock

The clock pin needs to receive eight pulses. At each pulse, if the data pin is high, then a 1 gets pushed into the shift register; otherwise, a 0. When all eight pulses have been received, enabling the 'Latch' pin copies those eight values to the latch register. This is necessary; otherwise, the wrong LEDs would flicker as the data is being loaded into the shift register.

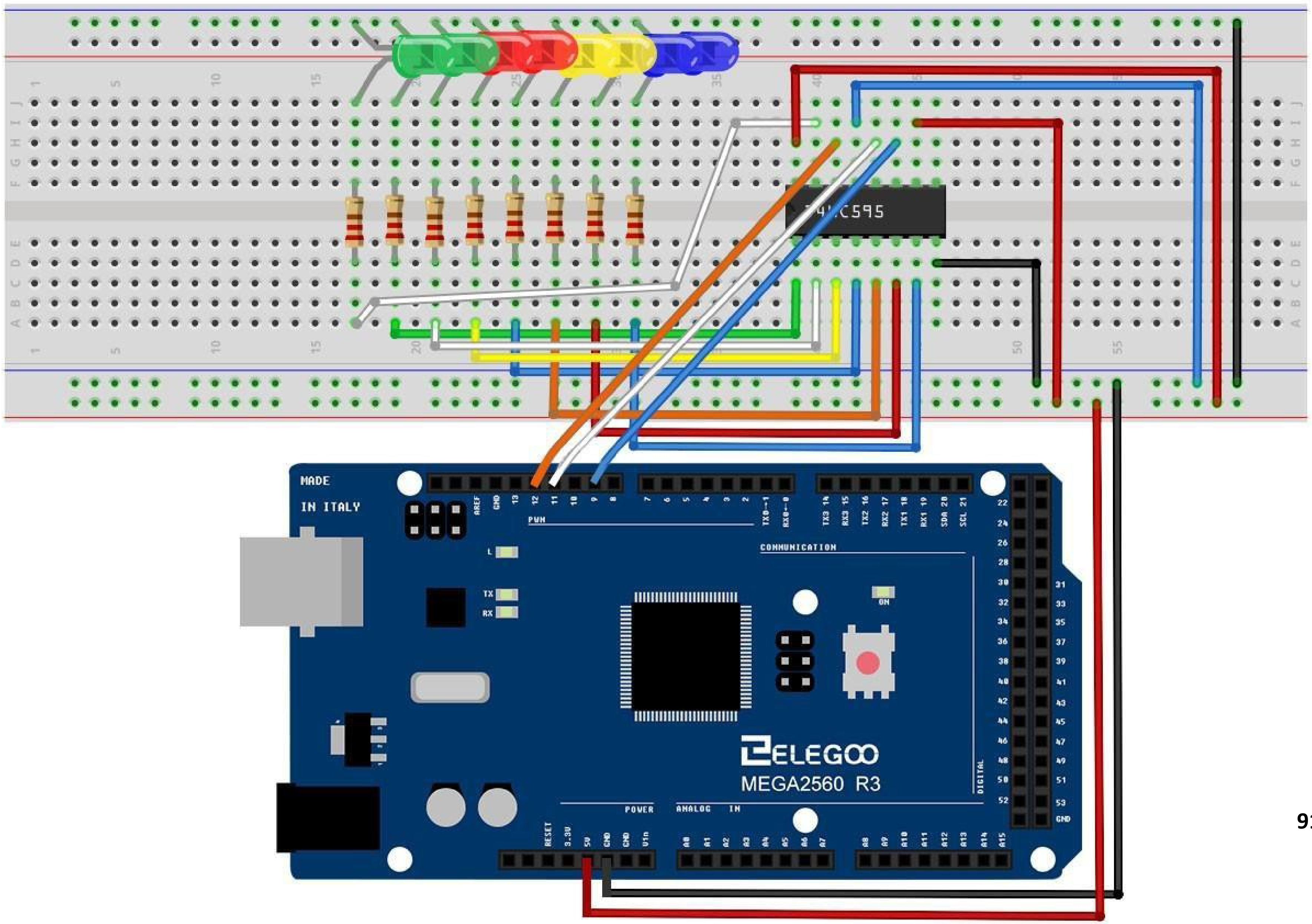
The chip also has an output enable (OE) pin, which is used to enable or disable the outputs all at once. You could attach this to a PWM-capable MEGA 2560 pin and use 'analog Write' to control the brightness of the LEDs. This pin is active low, so we tie it to GND.

Connection

Schematic



Wiring diagram



As we have eight LEDs and eight resistors to connect, there are actually quite a few connections to be made.

It is probably easiest to put the 74HC595 chip in first, as pretty much everything else connects to it. Put it so that the little U-shaped notch is towards the top of the breadboard. Pin 1 of the chip is to the left of this notch.

Digital 12 from the MEGA2560 goes to pin #14 of the shift register

Digital 11 from the MEGA2560 goes to pin #12 of the shift register

Digital 9 from the MEGA2560 goes to pin #11 of the shift register

All but one of the outputs from the IC is on the left side of the chip. Hence, for ease of connection, that is where the LEDs are, too.

After the chip, put the resistors in place. You need to be careful that none of the leads of the resistors are touching each other. You should check this again before you connect the power to your MEGA2560. If you find it difficult to arrange the resistors without their leads touching, then it helps to shorten the leads so that they are lying closer to the surface of thebreadboard.

Next, place the LEDs on the breadboard. The longer positive LED leads must all be towards the chip, whichever side of the breadboard they are on.

Attach the jumper leads as shown above. Do not forget the one that goes from pin 8 of the IC to the GND column of thebreadboard.

Load up the sketch listed a bit later and try it out. Each LED should light in turn until all the LEDs are on, and then they all go off and the cycle repeats.

Code

After wiring, please open the program in the code folder- Lesson 11 Eight LED with 74HC595 and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are anyerrors.

The first thing we do is define the three pins we are going to use. These are the MEGA2560 digital outputs that will be connected to the latch, clock and data pins of the 74HC595.

```
int latchPin = 11;
int clockPin = 9;
int dataPin = 12;
```

Next, a variable called 'leds' is defined. This will be used to hold the pattern of which LEDs are currently turned on or off. Data of type 'byte' represents numbers using eight bits. Each bit can be either on or off, so this is perfect for keeping track of which of our eight LEDs are on or off.

```
byte leds = 0;
```

The 'setup' function just sets the three pins we are using to be digital outputs.

```
void setup()
{
   pinMode(latchPin,  OUTPUT);
   pinMode(dataPin,   OUTPUT);
   pinMode(clockPin, OUTPUT);
}
```

The 'loop' function initially turns all the LEDs off, by giving the variable 'leds' the value 0. It then calls 'updateShiftRegister' that will send the 'leds' pattern to the shift register so that all the LEDs turn off. We will deal with how 'updateShiftRegister' works later.

The loop function pauses for half a second and then begins to count from 0 to 7 using the 'for' loop and the variable 'i'. Each time, it uses the Arduino function 'bitSet' to set the bit that controls that LED in the variable 'leds'. It then also calls 'updateShiftRegister' so that the leds update to reflect what is in the variable'leds'. There is then a half second delay before 'i' is incremented and the next LED is lit.

```
void loop()
{
   leds = 0;
   updateShiftRegister();
   delay(500);
   for (int i = 0; i < 8; i++)
   {
      bitSet(leds, i);
      updateShiftRegister();
      delay(500);
   }
}
```

The function 'updateShiftRegister', first of all sets the latchPin to low, then calls the MEGA2560 function 'shiftOut' before putting the 'latchPin' high again. This takes four parameters, the first two are the pins to use for Data and Clock respectively. The third parameter specifies which end of the data you want to start at. We are going to start with the right most bit, which is referred to as the 'Least Significant

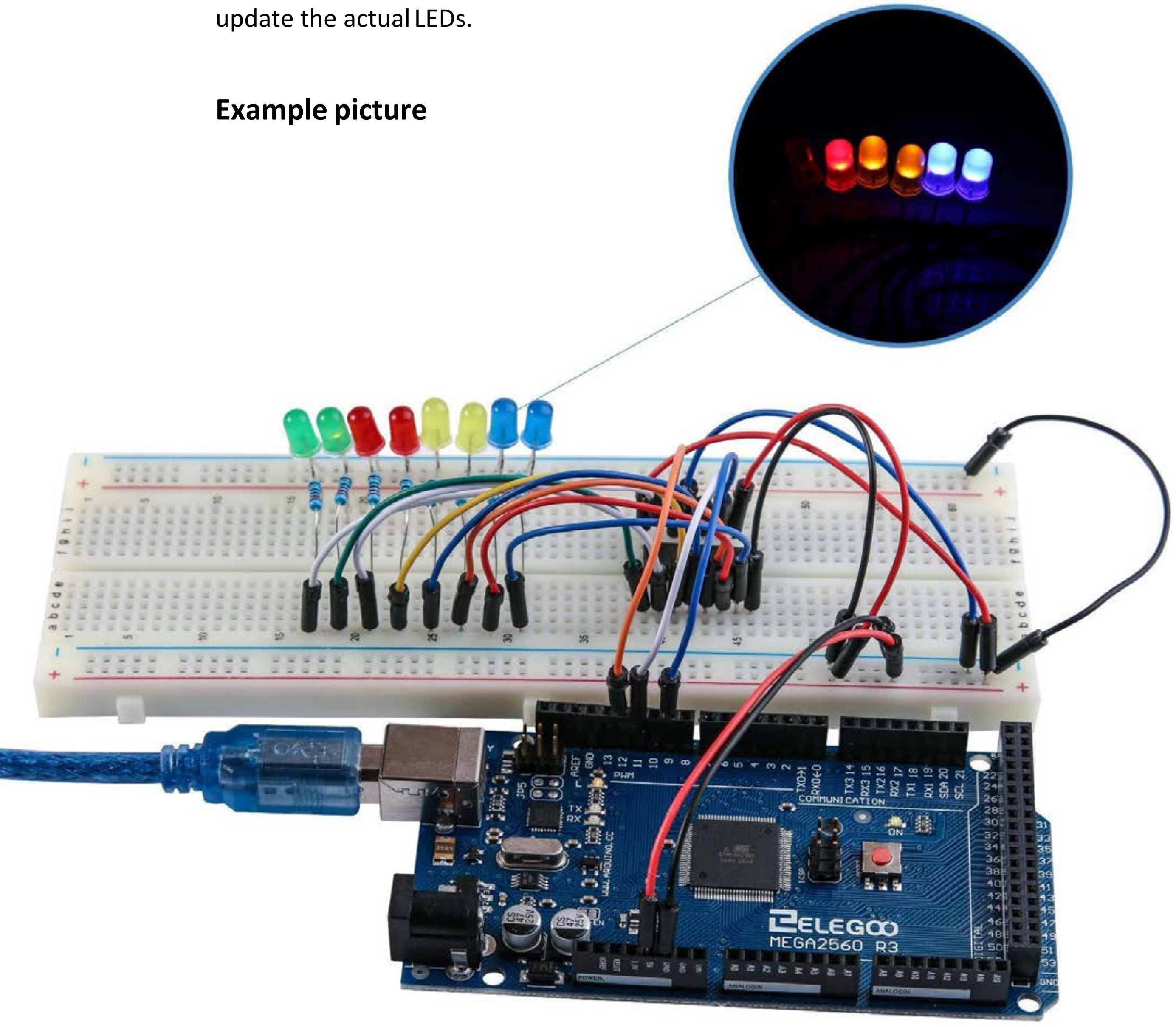
Bit' (LSB).

The last parameter is the actual data to be shifted into the shift register, which in this case is 'leds'.

```
void updateShiftRegister()
{
      digitalWrite(latchPin, LOW);
      shiftOut(dataPin, clockPin, LSBFIRST, leds);
      digitalWrite(latchPin, HIGH);
}
```

If you wanted to turn one of the LEDs off rather than on, you would call a similar Arduino function (bitClear) with the 'leds' variable. This will set that bit of 'leds' to be 0 and you would then just need to follow it with a call to 'updateShiftRegister' to update the actual LEDs.

Example picture



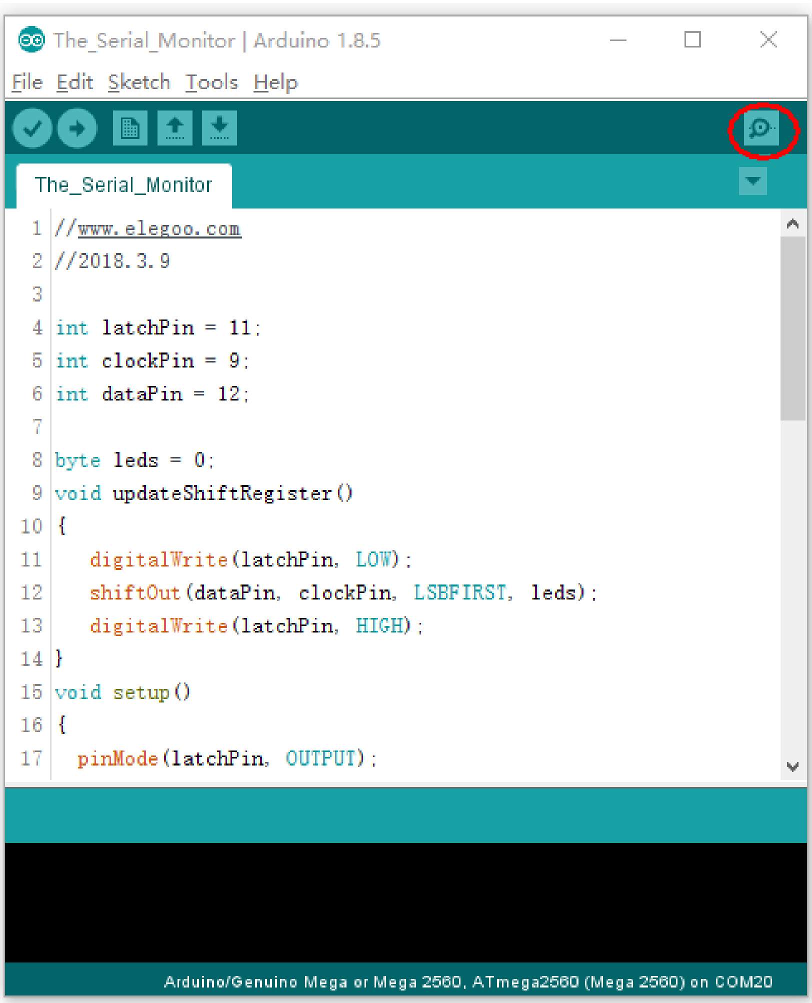
Lesson 12 The Serial Monitor

Overview

In this lesson, you will build on Lesson 11, adding the facility to control the LEDs from your computer using the Arduino Serial Monitor. The serial monitor is the 'tether' between the computer and your MEGA2560. It lets you send and receive text messages, handy for debugging and also controlling the MEGA2560 from a keyboard! For example, you will be able to send commands from your computer to turn on LEDs.

In this lesson, you will use exactly the same parts and a similar breadboard layout as Lesson 11. So, if you have not already done so, follow Lesson 11 now.

Steps taken

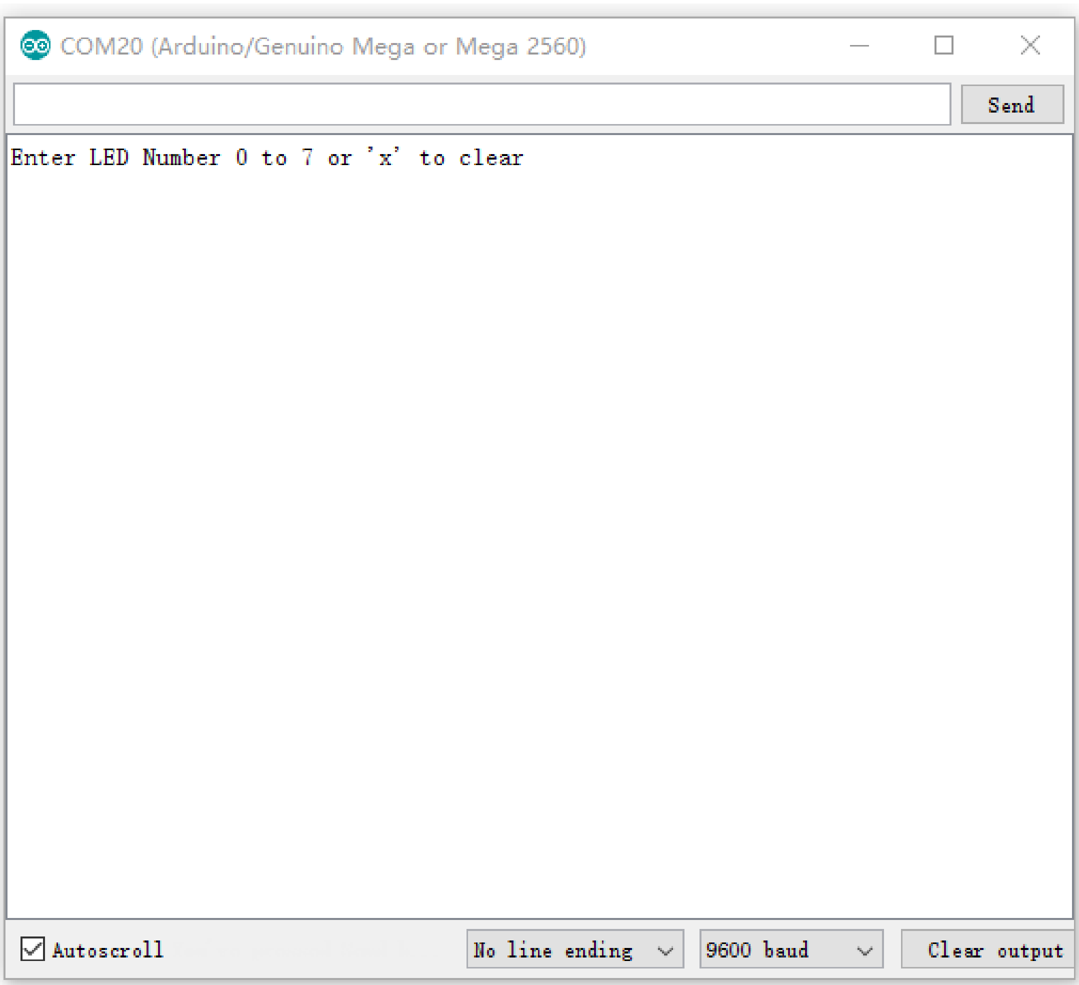

The_Serial_Monitor | Arduino 1.8.5
File Edit Sketch Tools Help
The_Serial_Monitor
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM20

```
//www.elegoo.com
//2018.3.9

int latchPin = 11;
int clockPin = 9;
int dataPin = 12;

byte leds = 0;
void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
void setup()
{
  pinMode(latchPin, OUTPUT);
```

After you have uploaded this sketch onto your MEGA2560, click on the right-most button on the toolbar in the Arduino IDE. The button is circled below.

The following window will open.

Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 1.



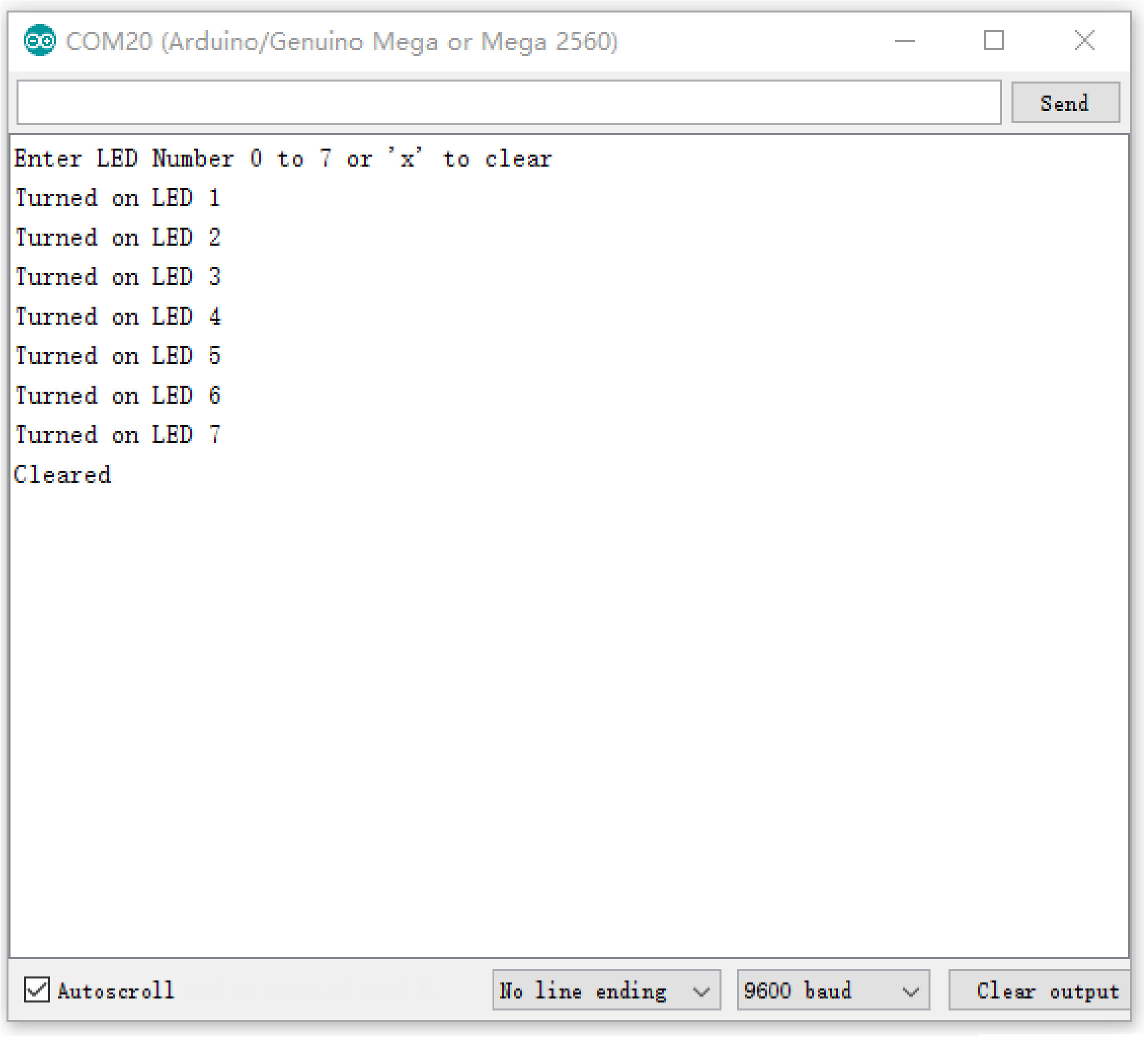
```
Enter LED Number 0 to 7 or 'x' to clear
```

This window is called the Serial Monitor and it is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an MEGA2560 board (over USB) and also to receive messages from the MEGA2560.

The message “Enter LED Number 0 to 7 or 'x' to clear” has been sent by the Arduino. It is telling us what commands we can send to the Arduino: either send the 'x' (to turn all the LEDs off) or the number of the LED you want to turn on (where 0 is the bottom LED, 1 is the next one up, all the way to 7 for the top LED).

Try typing the following commands into the top area of the Serial Monitor that is level with the 'Send' button. Press 'Send', after typing each of these characters: x 0 3 5

Typing x will have no effect if the LEDs are already all off, but as you enter each number, the corresponding LED should light and you will get a confirmation message from the MEGA2560 board. The Serial Monitor will appear as shown below.



```
Enter LED Number 0 to 7 or 'x' to clear
Turned on LED 1
Turned on LED 2
Turned on LED 3
Turned on LED 4
Turned on LED 5
Turned on LED 6
Turned on LED 7
Cleared
```

Type x again and press ‘Send’ to turn off all LEDs.

Code

After wiring, please open program in the code folder- Lesson 12 The Serial Monitor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

As you might expect, the sketch is based on the sketch used in Lesson 24. So, we will just cover the new bits here. You will find it useful to refer to the full sketch in your Arduino IDE.

In the 'setup' function, there are three new lines at the end:

```
void setup()
{
   pinMode(latchPin, OUTPUT);
   pinMode(dataPin, OUTPUT);
   pinMode(clockPin, OUTPUT);
   updateShiftRegister();
   Serial.begin(9600);

   while (! Serial); // Wait until Serial is ready -Leonardo
   Serial.println("Enter LED Number 0 to 7 or 'x' toclear");
}
```

Firstly, we have the command 'Serial.begin(9600)'. This starts serial communication, so that the MEGA2560 can send out commands through the USB connection. The value 9600 is called the 'baud rate' of the connection. This is how fast the data is to be sent. You can change this to a higher value, but you will also have to change the Arduino Serial monitor to the same value. We will discuss this later; for now, leave it at 9600.

The line beginning with 'while' ensures that there is something at the other end of the USB connection for the Arduino to talk to before it starts sending messages. Otherwise, the message might be sent, but not displayed. This line is actually only necessary if you are using an Arduino Leonardo because the Arduino MEGA2560 automatically resets the Arduino board when you open the Serial Monitor, whereas this does not happen with the Leonardo.

The last of the new lines in 'setup' sends out the message that we see at the top of the Serial Monitor.

The 'loop' function is where all the action happens:

```
void loop()
{
    if (Serial.available())
    {
        char ch = Serial.read();
        if (ch >= '0' && ch <= '7')
        {
            int led = ch - '0';
            bitSet(leds, led);
            updateShiftRegister();
            Serial.print("Turned on LED ");
            Serial.println(led);
        }
        if (ch == 'x')
        {
            leds = 0; updateShiftRegister();

            Serial.println("Cleared");
        }
    }
}
```

Everything that happens inside the loop is contained within an 'if' statement. So unless the call to the built-in Arduino function 'Serial.available()' is 'true' then nothing else will happen.

Serial.available() will return 'true' if data has been send to the MEGA2560 and is there ready to be processed. Incoming messages are held in what is called a buffer and Serial.available() returns true if that buffer is Not empty.

If a message has been received, then its on to the next line of code:

```
char ch = Serial.read();
```

This reads the next character from the buffer, and removes it from the buffer. It also assigns it to the variable 'ch'. The variable 'ch' is of type 'char' which stands for 'character' and as the name suggests, holds a single character.

If you have followed the instructions in the prompt at the top of the Serial Monitor, then this character will either be a single digit number between 0 and 7 or the letter 'x'.

The 'if' statement on the next line checks to see if it is a single digit by seeing if 'ch' is greater than or equal to the character '0' and less than or equal to the character '7'. It looks a little strange comparing characters in this way, but is perfectly acceptable.

Each character is represented by a unique number, called its ASCII value. This means that when we compare characters using <= and >= it is actually the ASCII values that were being compared.

If the test passes, then we come to the next line:

```
int led = ch – '0';
```

Now we are performing arithmetic on characters! We are subtracting the digit '0' from whatever digit was entered. So, if you typed '0' then '0' – '0' will equal 0. If you typed '7' then '7' – '0' will equal the number 7 because it is actually the ASCII values that are being used in the subtraction.

Since that we know the number of the LED that we want to turn on, we just need to set that bit in the variable 'leds' and update the shift register.

```
bitSet(leds, led);
updateShiftRegister();
```

The next two lines write back a confirmation message to the Serial Monitor.

```
Serial.print("Turned on LED ");
Serial.println(led);
```

The first line uses Serial.print rather than Serial.println. The different between the two is that Serial.print does not start a new line after printing whatever is in its parameter. We use this in the first line, because we are printing the message in two parts. Firstly the general bit: 'Turned on LED ' and then the number of the LED.

The number of the LED is held in an 'int' variable rather than being a text string. Serial.print can take either a text string enclosed in double-quotes, or an 'int' or for that matter pretty much any type of variable.

After the 'if' statement that handles the case, when a single digit has been handled, there is a second 'if' statement that checks to see if 'ch' is the letter 'x'.

```
if (ch == 'x')
{
   leds = 0; updateShiftRegister();
   Serial.println("Cleared");
}
```

If it is, then it clears all the LEDs and sends a confirmation message

Lesson 13 Photocell

Overview

In this lesson, you will learn how to measure light intensity using an Analog Input. You will build on lesson 12 and use the level of light to control the number of LEDs to be lit.

The photocell is at the bottom of the breadboard, where the pot was above.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x 830 tie-points breadboard

(8) x leds

(8) x 220 ohm resistors

(1) x 1k ohm resistor

(1) x 74hc595 IC

(1) x Photoresistor (Photocell)

(16) x M-M wires (Male to Male jumperwires)

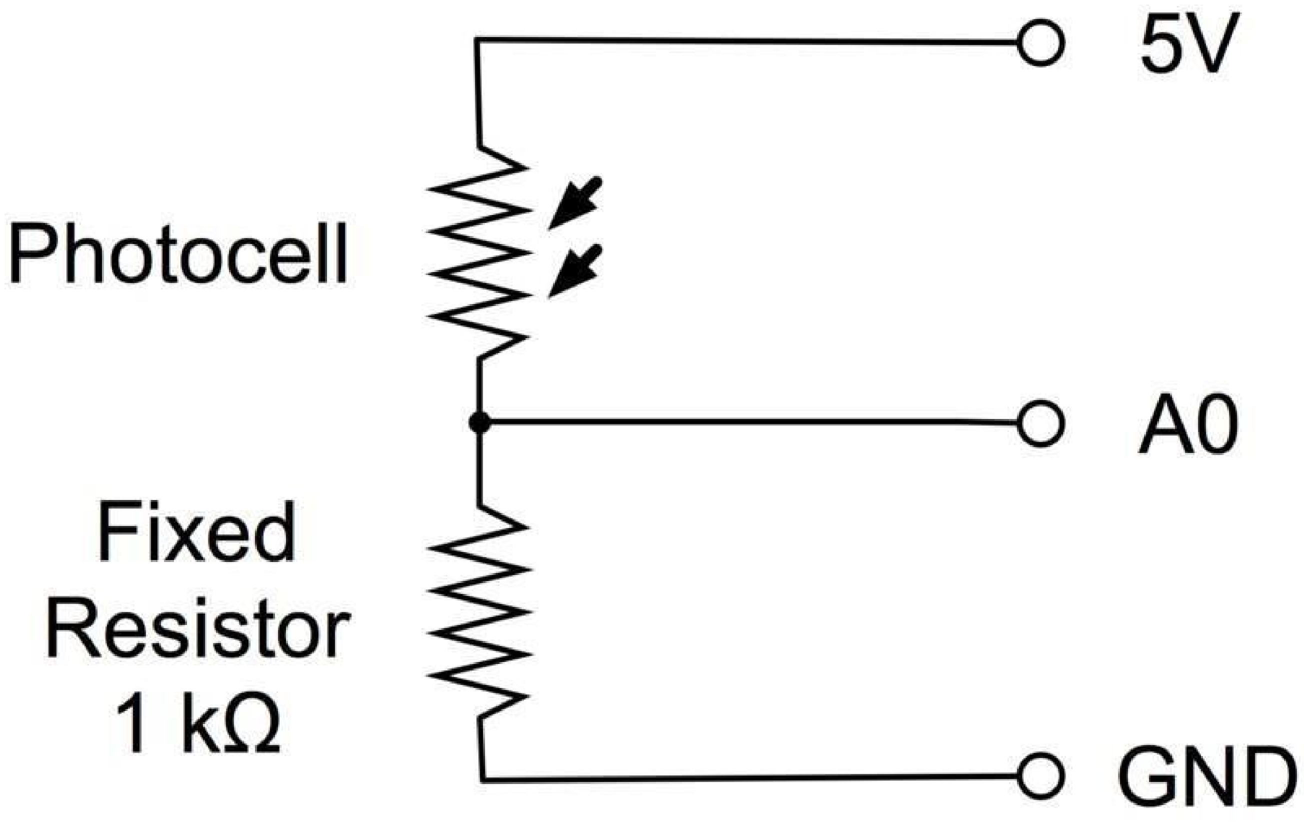
Component Introduction

PHOTOCELL:

The photocell used is of a type called a light dependent resistor, sometimes called an LDR. As the name suggests, these components act just like a resistor, except that the resistance changes in response to how much light is falling on them.

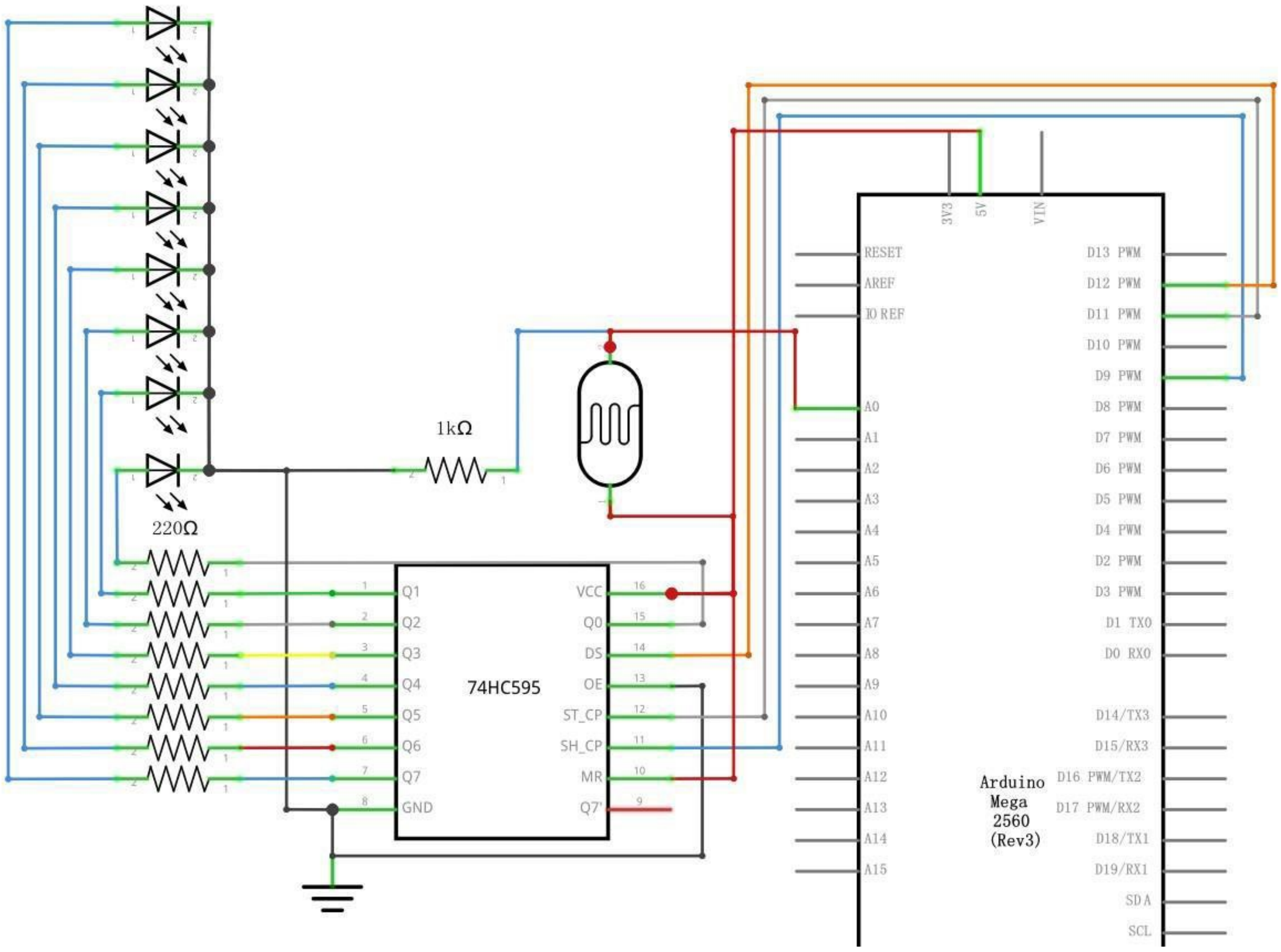
This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light. To convert this varying value of resistance into something we can measure on an MEGA 2560 R3 board's analog input, it needs to be converted into a voltage.

The simplest way to do that is to combine it with a fixed resistor.

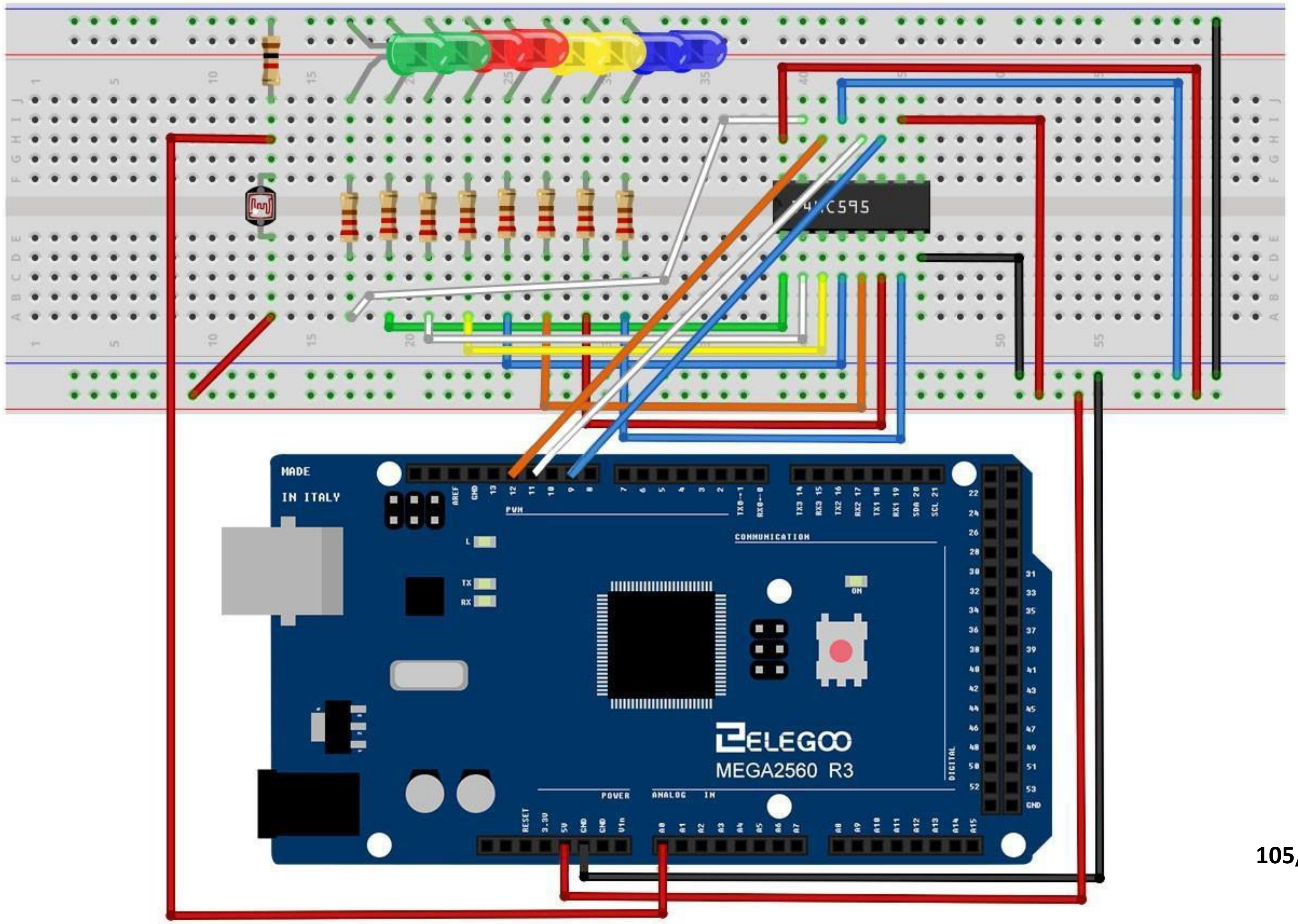


The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.
When the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND.
Load up the sketch given in the next section and try covering the photocell with your finger, and then holding it near a light source.

Connection
Schematic



Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 13 Photocell and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

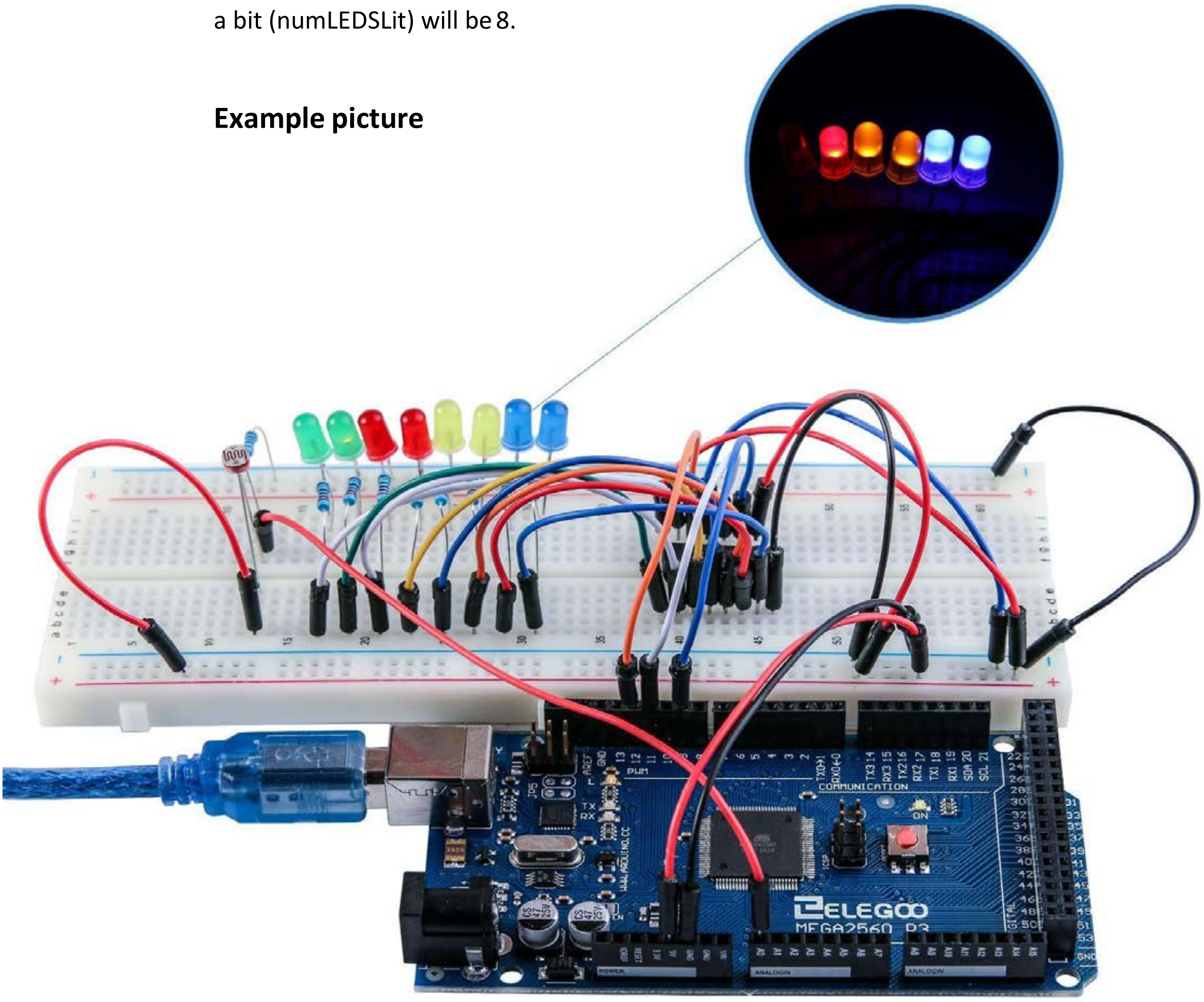
The first thing to note is that we have changed the name of the analog pin to be 'lightPin' rather than 'potPin' since we no longer have a pot connected.

The only other substantial change to the sketch is the line that calculates how many of the LEDs to light:

```
int numLEDSLit = reading / 57;    // all LEDs lit at 1k
```

This time, we divide the raw reading by 57 rather than 114. In other words, we divide it by half as much as we did with the pot to split it into nine zones, from no LEDs lit to all eight lit. This extra factor is to account for the fixed 1 kΩ resistor. This means that when the photocell has a resistance of 1 kΩ (the same as the fixed resistor), the raw reading will be 1023 / 2 = 511. This will equate to all the LEDs being lit and then a bit (numLEDSLit) will be 8.

Example picture



Lesson 14 Four Digital Seven SegmentDisplay

Overview

In this lesson, you will learn how to use a 4-digit 7-segment display.

When using 1-digit 7-segment display please notice that if it is common anode, the common anode pin connects to the power source; if it is common cathode, the common cathode pin connects to theGND.

When using 4-digit 7-segment display, the common anode or common cathode pin is used to control which digit is displayed. Even though there is only one digit working, the principle of Persistence of Vision enables you to see all numbers displayed because each the scanning speed is so fast that you hardly notice the intervals.

Component Required:

(1) x Elegoo Mega 2560 R3

(1) x 830 tie-pointsbreadboard

(1) x 74HC595 IC

(1) x 4 Digit 7-Segment Display

(4) x 220 ohm resistors

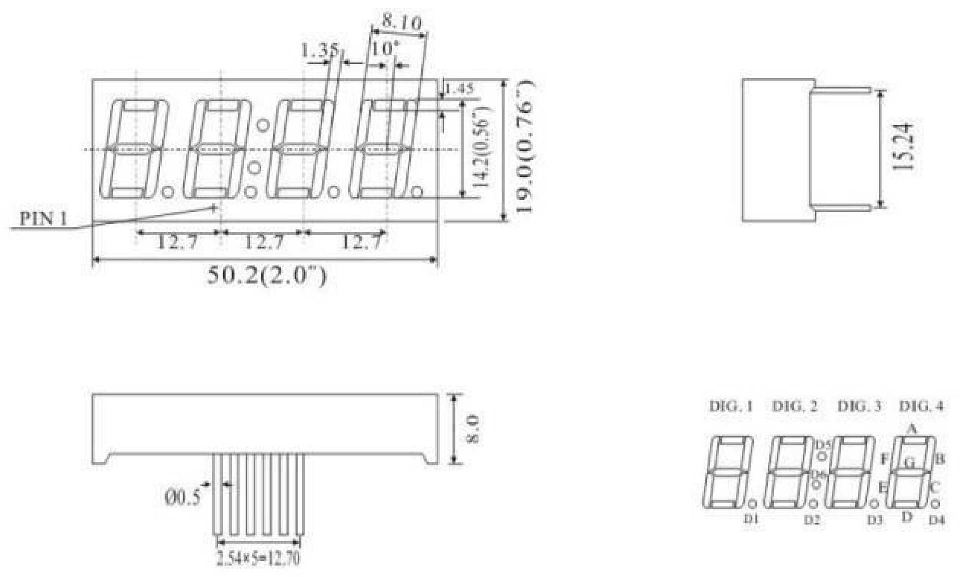
(23) x M-M wires (Male to Male jumper wires)

Component Introduction

Four Digital Seven segment display

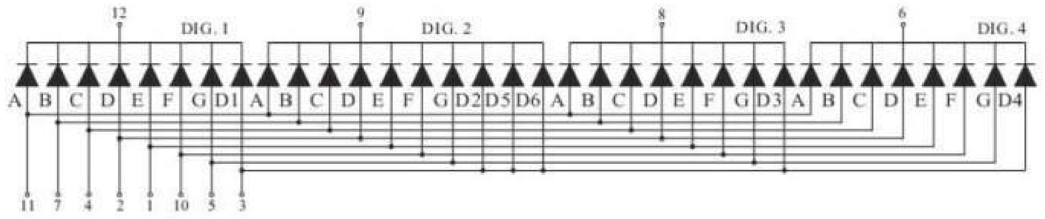
Package Dimensions

CPS05643AB

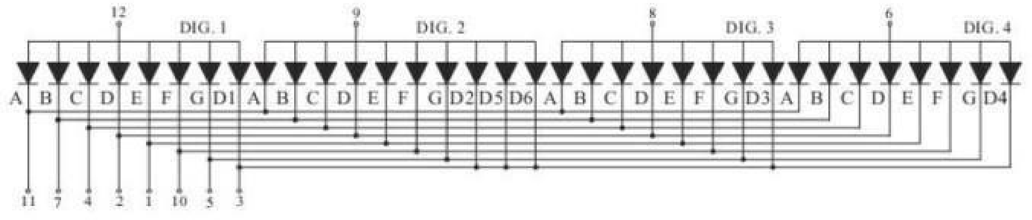


UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

Internal Circuit Diagram


12
DIG. 1
9
DIG. 2
8
DIG. 3
6
DIG. 4
A B C D E F G D1 A B C D E F G D2 D5 D6 A B C D E F G D3 A B C D E F G D4
11 7 4 2 1 10 5 3

5643A

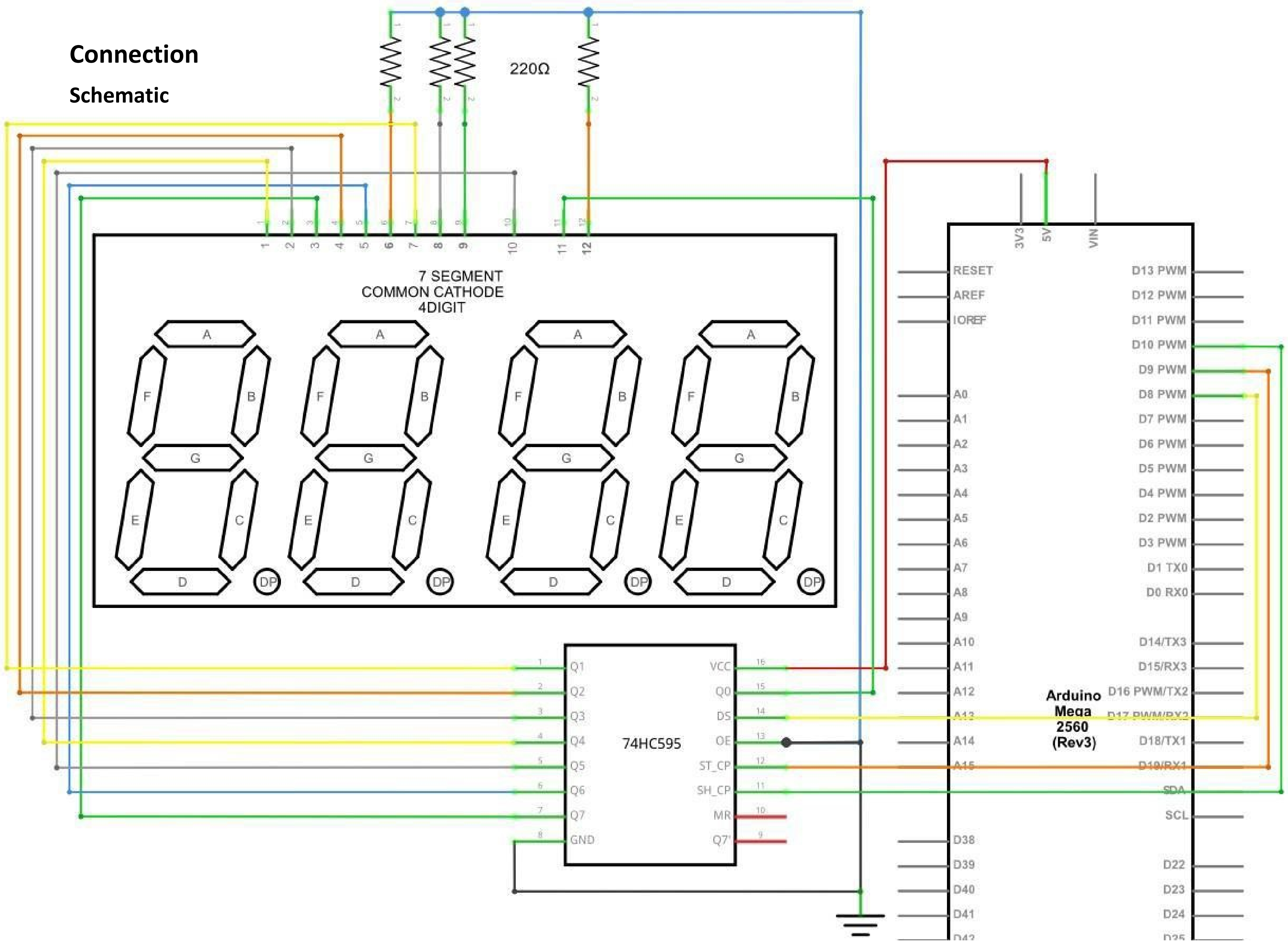

12
DIG. 1
9
DIG. 2
8
DIG. 3
6
DIG. 4
A B C D E F G D1 A B C D E F G D2 D5 D6 A B C D E F G D3 A B C D E F G D4
11 7 4 2 1 10 5 3

5643B

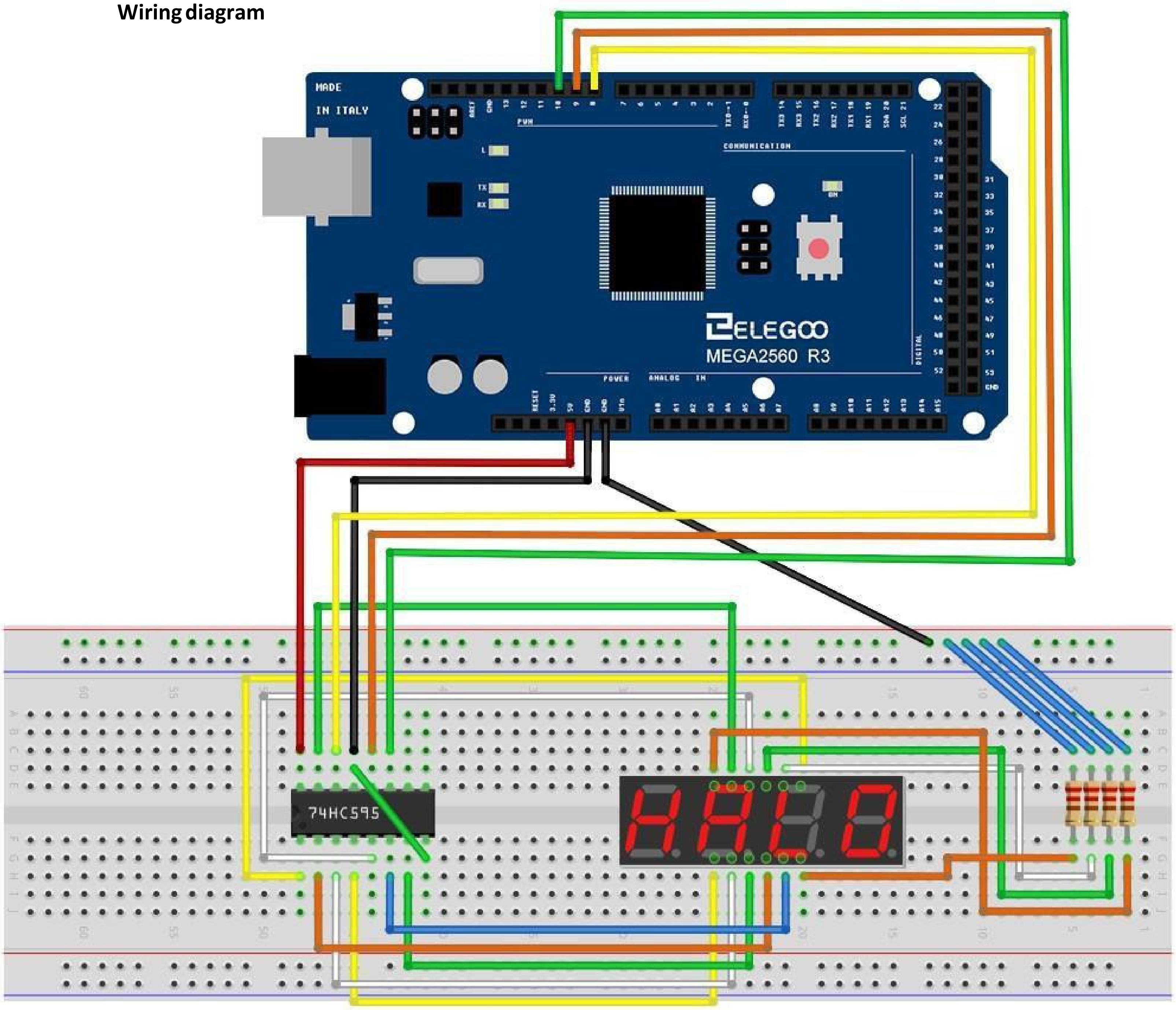
Four Digits Displays Series

Connection

Schematic


220Ω
7 SEGMENT
COMMON CATHODE
4DIGIT
1 2 3 4 5 6 7 8 9 10 11 12
A B C D E F G DP
74HC595
Q1 Q2 Q3 Q4 Q5 Q6 Q7 GND
VCC Q0 DS OE ST_CP SH_CP MR Q7'
Arduino Mega 2560 (Rev3)
3V3 5V VIN
RESET AREF IOREF
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
D38 D39 D40 D41
D13 PWM D12 PWM D11 PWM D10 PWM D9 PWM D8 PWM D7 PWM D6 PWM D5 PWM D4 PWM D2 PWM D3 PWM
D1 TX0 D0 RX0
D14/TX3 D15/RX3 D16 PWM/TX2 D17 PWM/RX2 D18/TX1 D19/RX1
SDA SCL
D22 D23 D24

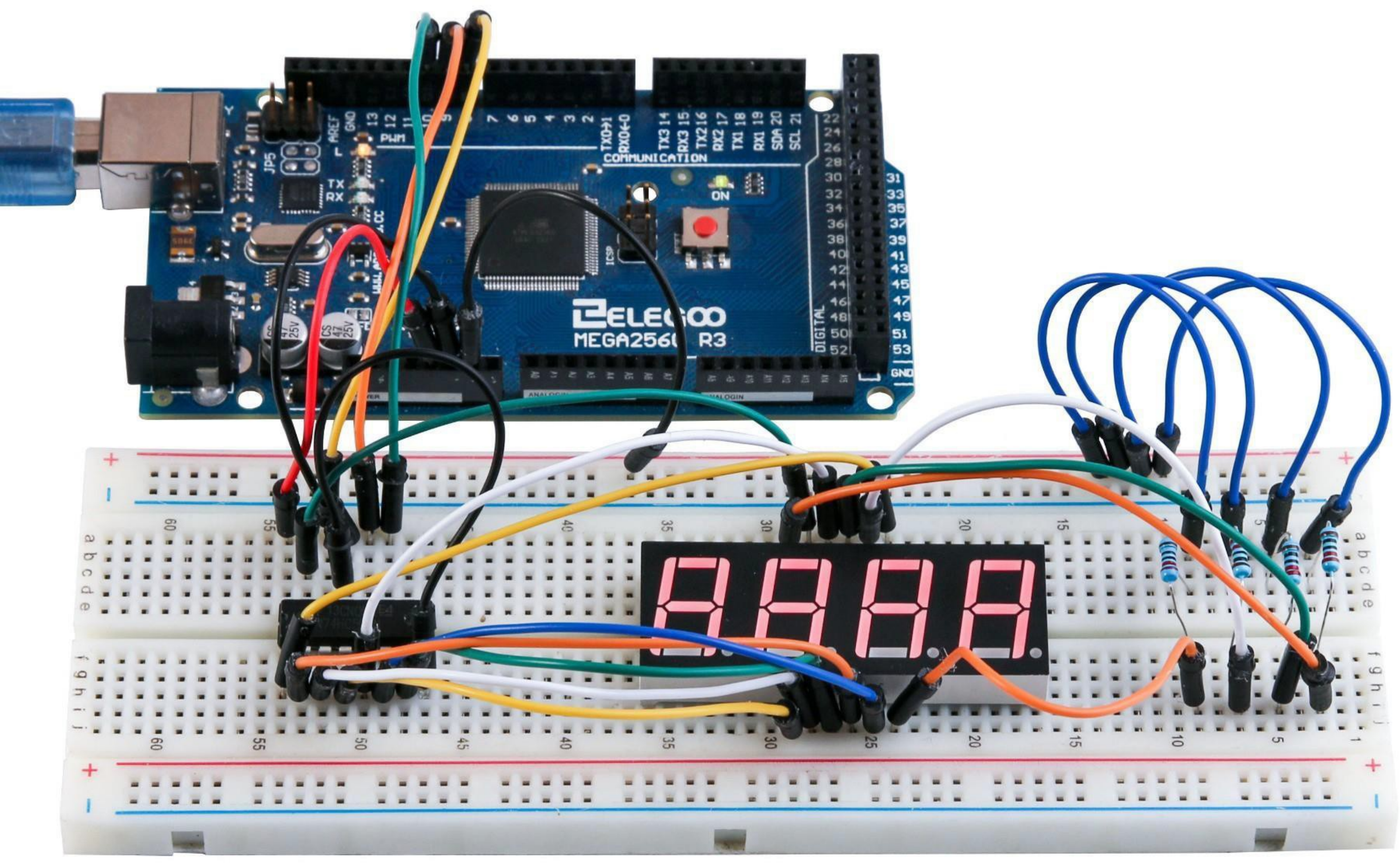
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 14 Four Digital Seven Segment Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 15 DC Motors

Overview

In this lesson, you will learn how to control a small DC motor using an MEGA 2560 and a transistor.

Component Required:

(1) x Elegoo MEGA 2560

(1) x Breadboard

(1) x 6v dc motor

(1) x PN2222

(1) x 1N4007

(1) x 220 ohm resistor

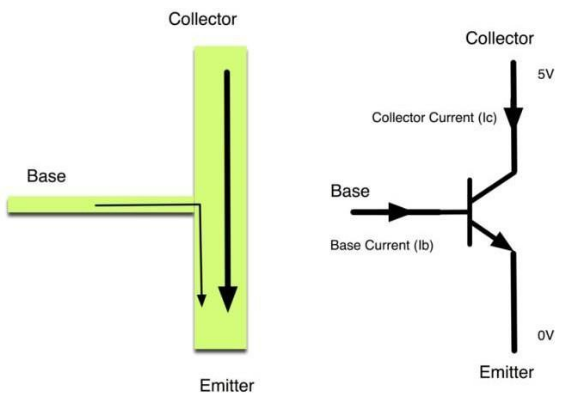
(3) x M-M wires

Component Introduction

TRANSISTORS

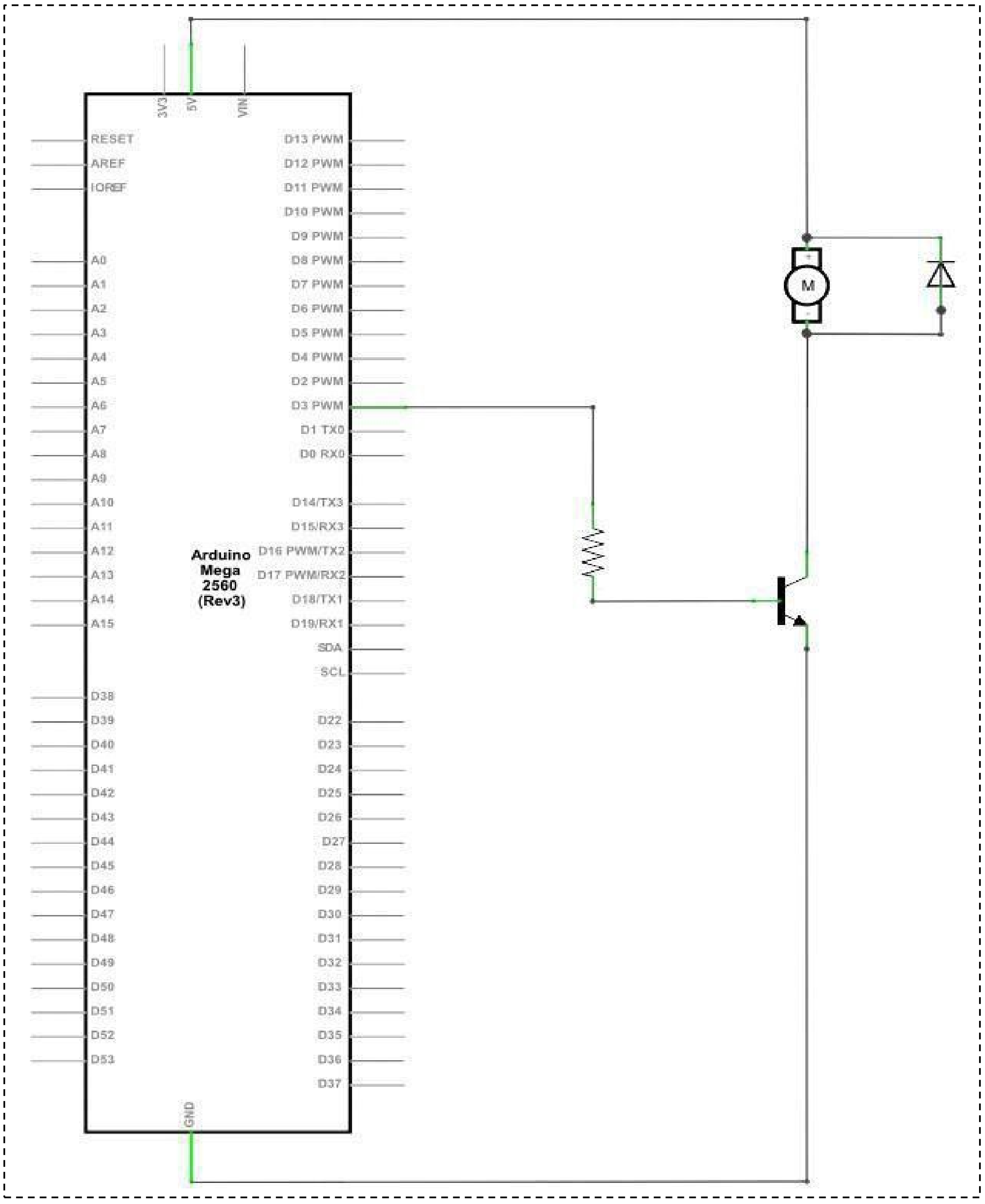
The small DC motor is likely to use more power than an MEGA 2560 board digital output can handle directly. If we tried to connect the motor straight to an MEGA 2560 board pin, there is a good chance that it could damage the MEGA 2560board.

A small transistor like the PN2222 can be used as a switch that uses just a little current from the MEGA 2560 board digital output to control the much bigger current of themotor.



The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the MEGA 2560 board digital output.

Below is called a schematic diagram of the. Like a breadboard layout, it is a way of showing how the parts of an electronic project are connected together.



The pin D3 of the MEGA2560 board is connected to the resistor. Just like when using an LED, this limits the current flowing into the transistor through thebase.

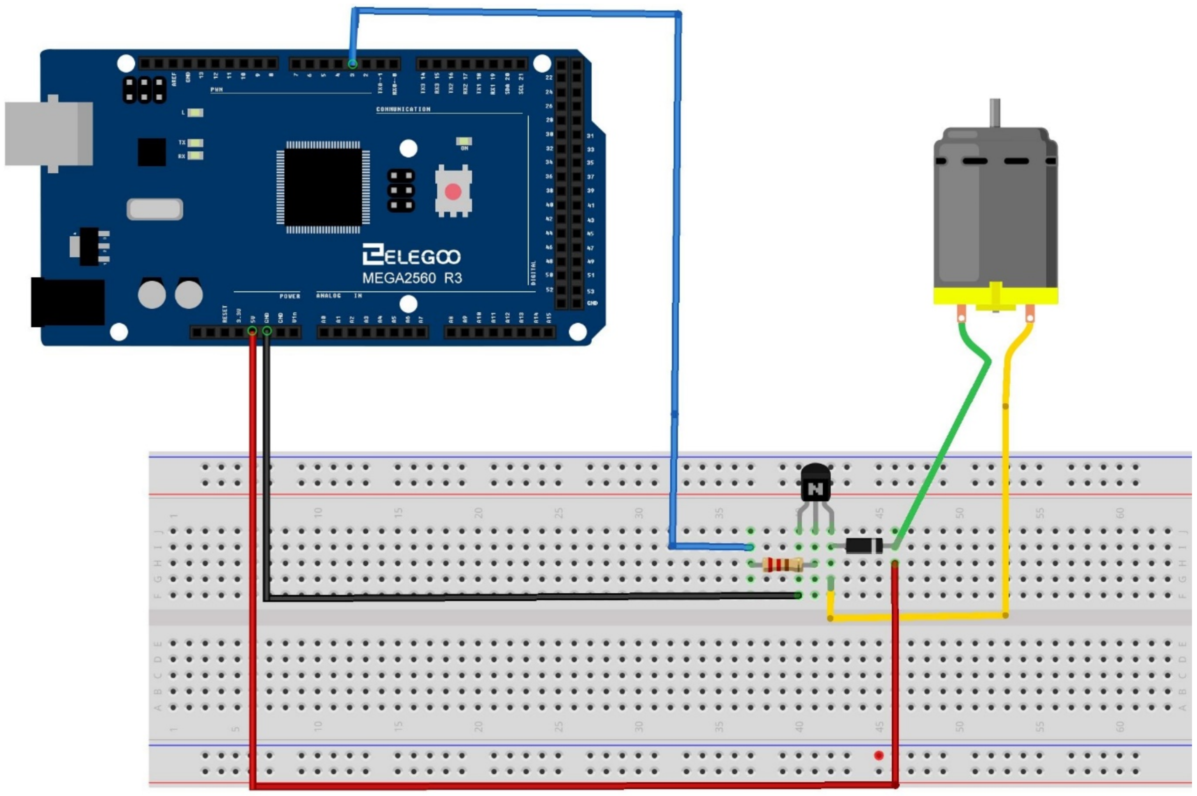
There is a diode connected across the connections of the motor. Diodes only allow electricity to flow in one direction (the direction of theirarrow).

When you turn the power off to a motor, you get a negative spike of voltage, that can damage your MEGA2560 board or the transistor. The diode protects against this, by shorting out any such reverse current from the motor.

Through this experiment, you've learned how to drive LCD1602. Now you can create your own messages to display! You can also try letting your LCD1602 display numbers.

Connection

wiring diagram



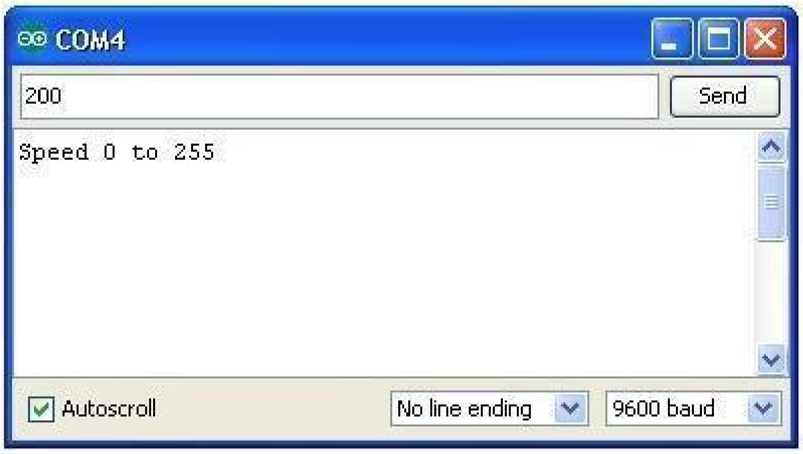
You will use an MEGA2560 board analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.

When you put together the breadboard, there are two things to look outfor.

First, make sure that the flat side of the transistor is on the right-hand side of thebreadboard.

Second, the striped end of the diode should be towards the +5V power line - see the image below! The motor that comes with the kits does not draw more than 250mA but if you have a different motor, it could easily draw 1000mA, more than a USB port can handle! If you aren't sure of a motor's current draw, power the MEGA 2560 board from a wall adapter, not just USB. The transistor acts like a switch, controlling the power to the motor, MEGA 2560 board pin 3 is used to turn the transistor on and off and is given the name 'motor Pin' in the sketch.

When the sketch starts, it prompts to remind you that to control the speed of the motor, you need to enter a value between 0 and 255 in the Serial Monitor.

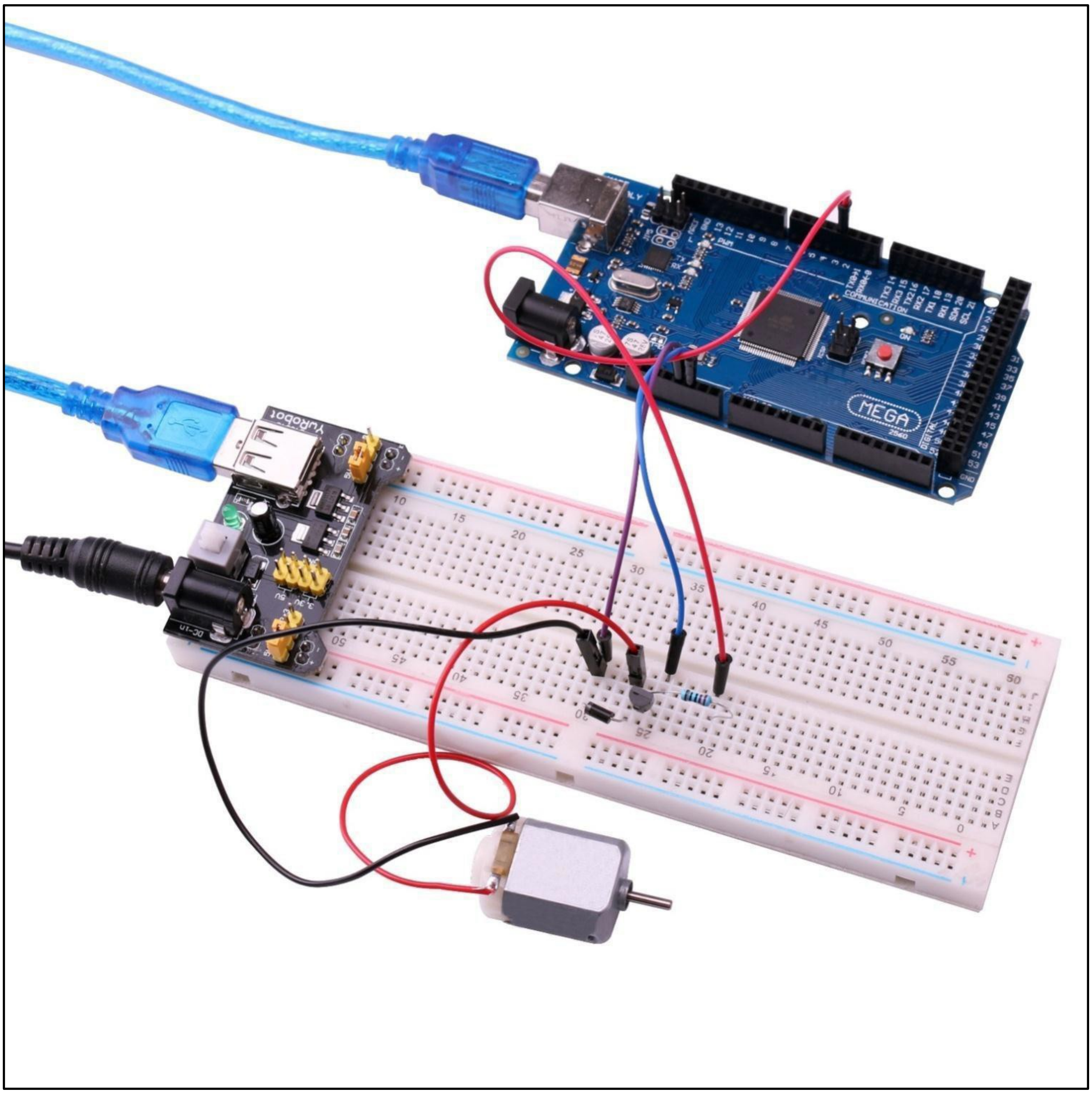


```
Speed 0 to 255
```

In the 'loop' function, the command 'Serial.parseInt' is used to read the number entered as text in the Serial Monitor and convert it into an'int'.

You could type any number here, so the 'if' statement on the next line only does an analog write with this number if the number is between 0 and255.

Example picture



Lesson 16 Relay

Overview

In this lesson, you will learn how to use a relay.

Component Required:

(1) x Elegoo MEGA 2560

(1) x Breadboard

(1) x PN2222

(1) x 1N4007

(1) x 220 ohm resistor

(1) x 6v dc motor

(1) x Relay

(1) x Breadboard power supply module

(1) x 9v power supply adapter

(8) x M-M wires

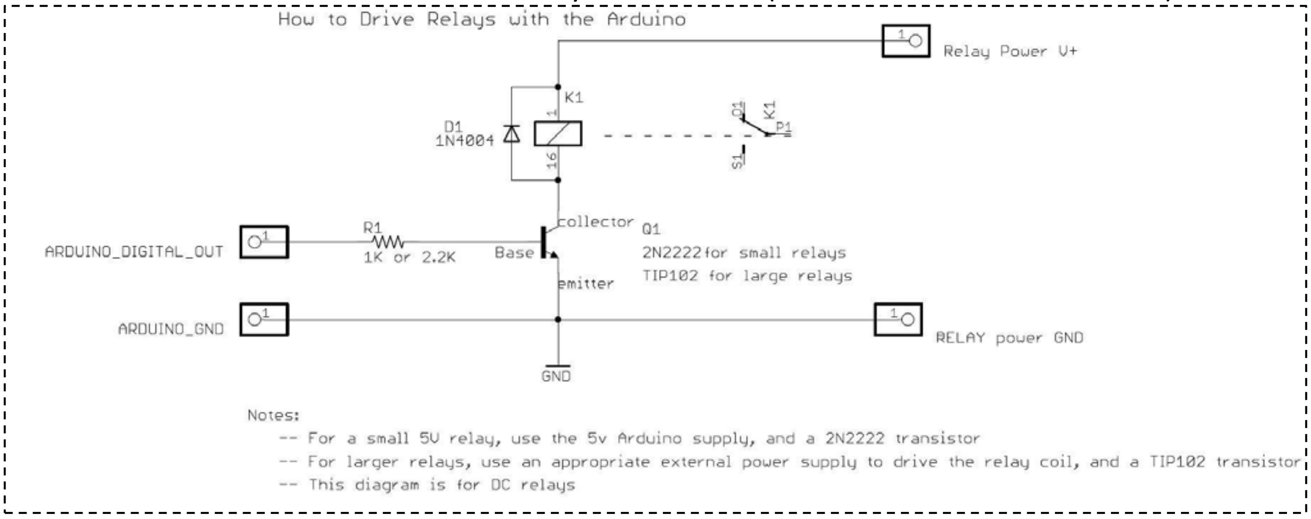
Component Introduction

Relay:

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used as in solid-state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long-distance telegraph circuits as amplifiers. They repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

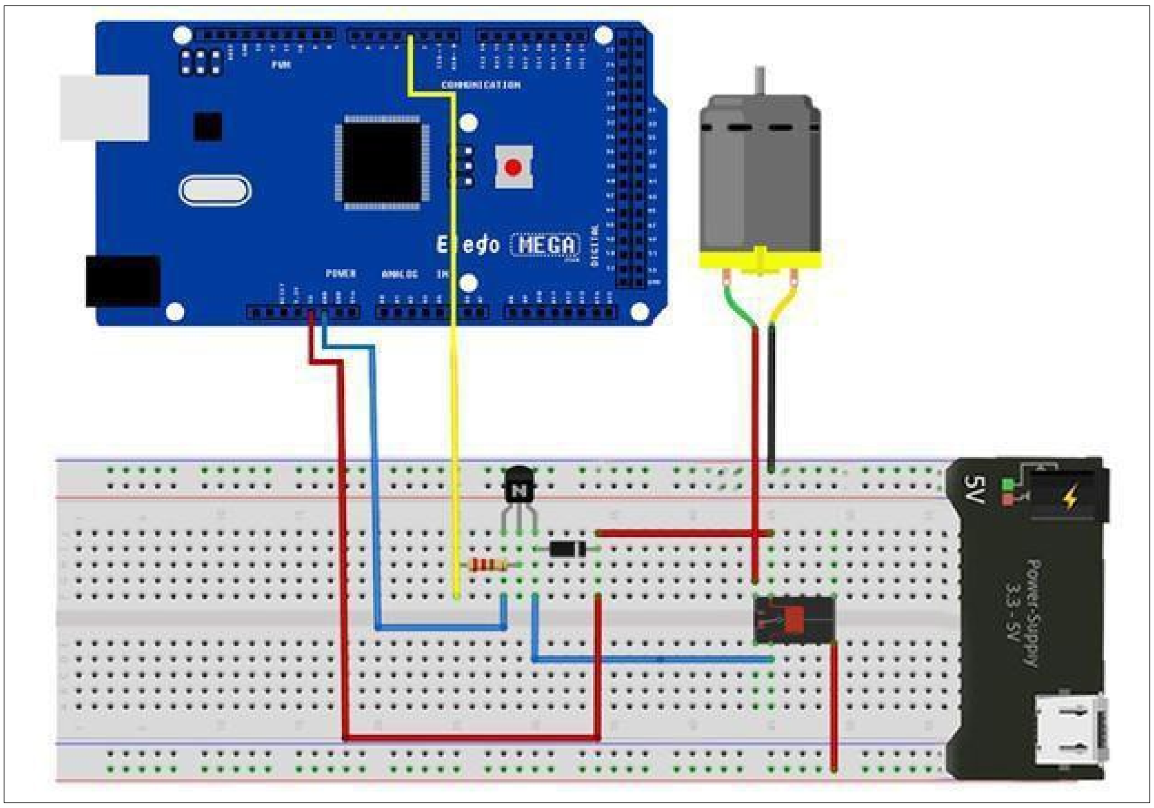
A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform the switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults. In modern electric power systems, these functions are performed by digital instruments called "protectiverelays".

Below is the schematic of how to drive relay with Arduino (download from the arduino.cc)


How to Drive Relays with the Arduino
Relay Power V+
K1
D1
1N4004
1
16
Q1
K1
P1
S1
collector
Q1
R1
1K or 2.2K
Base
2N2222 for small relays
TIP102 for large relays
emitter
ARDUINO_DIGITAL_OUT
ARDUINO_GND
RELAY power GND
GND
Notes:
-- For a small 5V relay, use the 5v Arduino supply, and a 2N2222 transistor
-- For larger relays, use an appropriate external power supply to drive the relay coil, and a TIP102 transistor
-- This diagram is for DC relays

Connection

wiring diagram



Code

See the code file.