PLUSIVO

Nano R3 Super Starter Kit

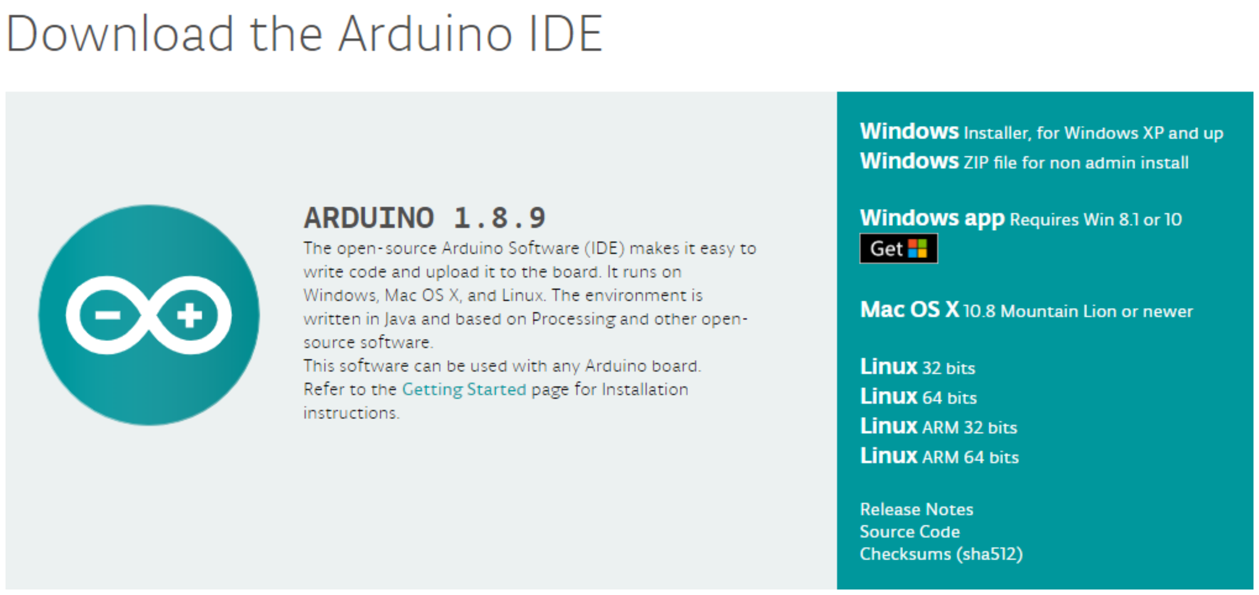
Table of Contents

Lesson 1: Installing the Arduino IDE

1.1 Introduction

The software of the Arduino device is called The Arduino Integrated Development Environment (IDE).

First, you will learn how to properly setup your computer to use your Arduino device and install the software. The IDE that will be used to program your Arduino is available for Windows, Linux and Mac, but the installation is not the same for all three.



Download the Arduino IDE

ARDUINO 1.8.9

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, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

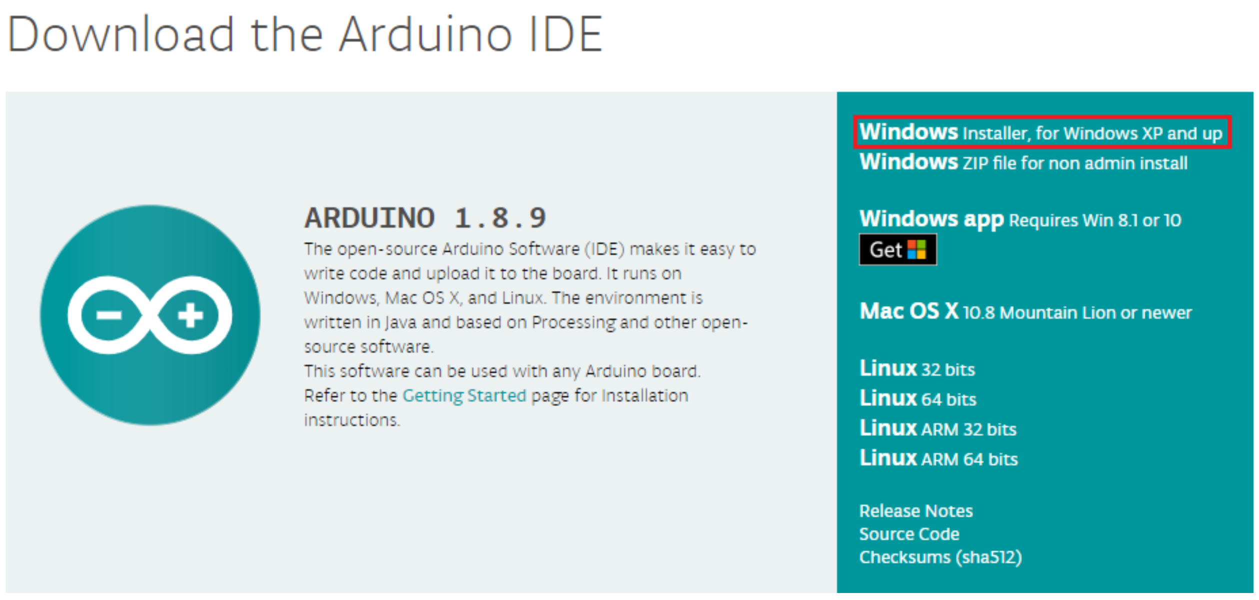
Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

1.2 Installing Arduino (Windows)

You can download it from this link:
https://www.arduino.cc/en/Main/Software
Click on **Windows installer, for windows XP and up.**



Download the Arduino IDE

ARDUINO 1.8.9

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, for Windows XP and up
Windows ZIP file for non admin install

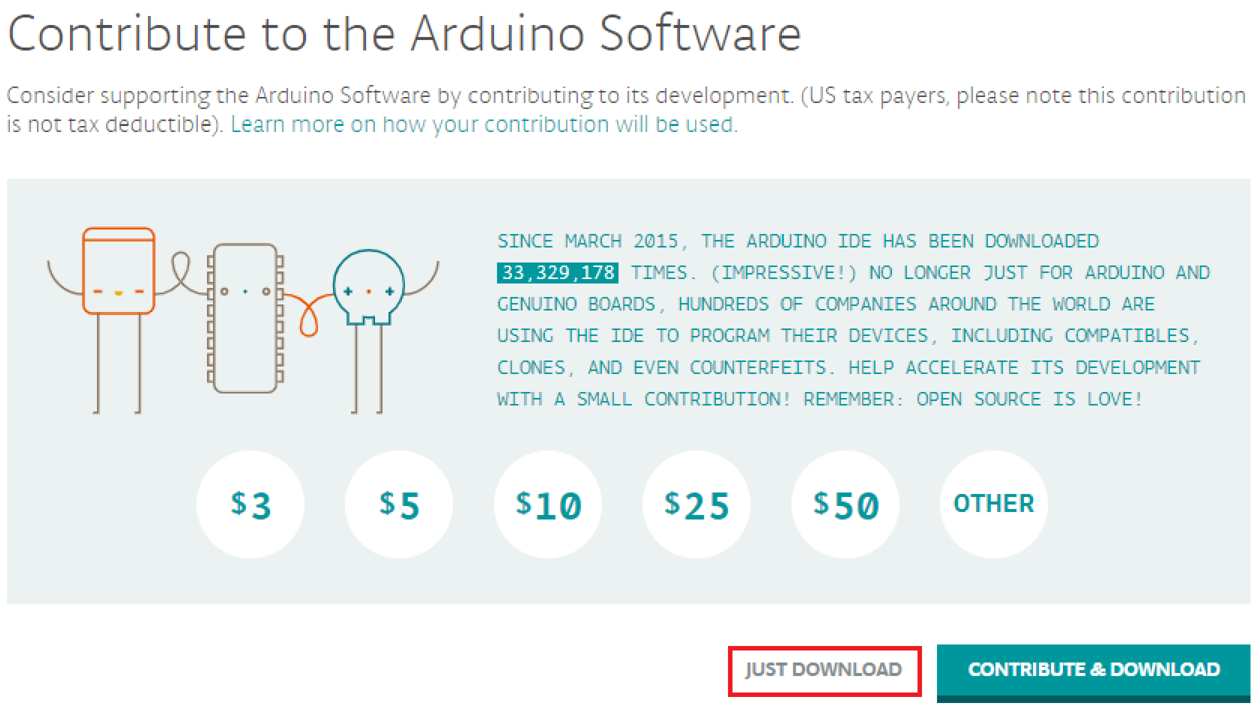
Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

Click on **JUST DOWNLOAD.**

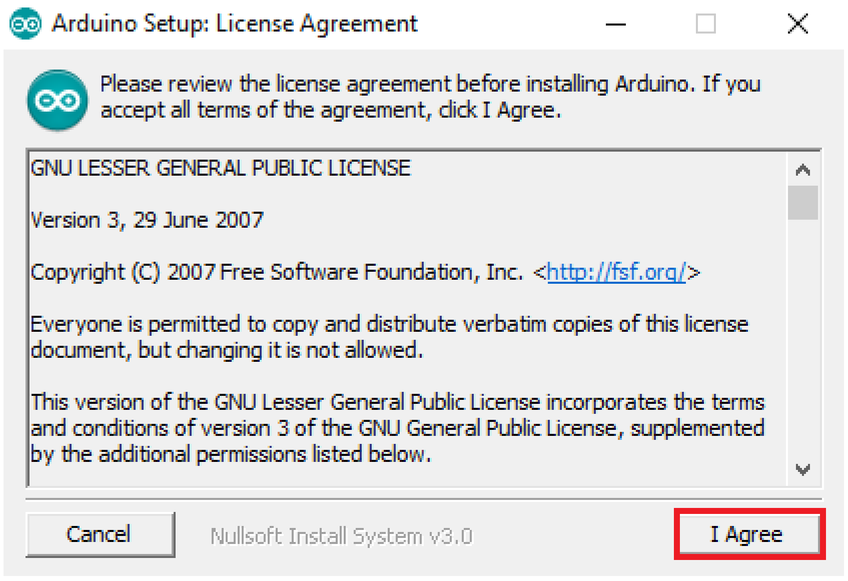


Contribute to the Arduino Software

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 33,329,178 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!

Find the file that you downloaded, open it.
After reading the agreement, click **I AGREE**.



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

GNU LESSER GENERAL PUBLIC LICENSE

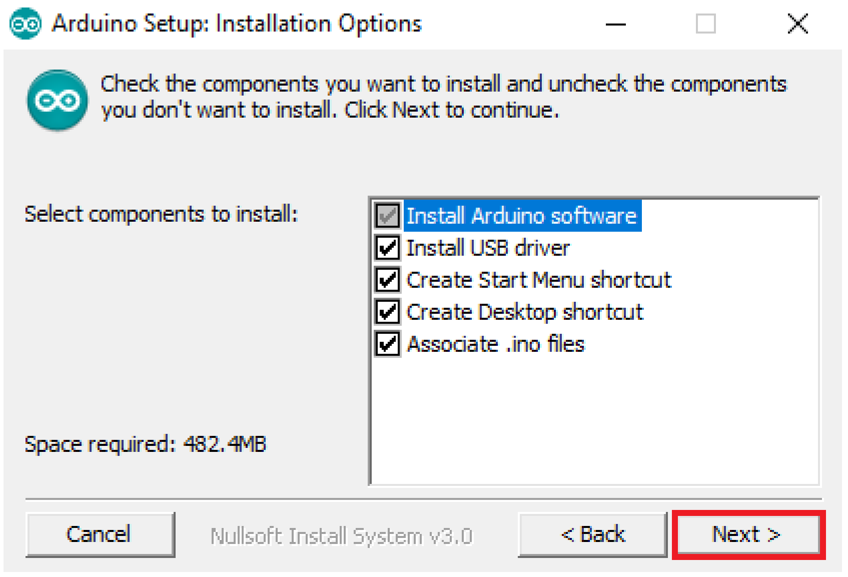
Version 3, 29 June 2007

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 on **Next.**


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.

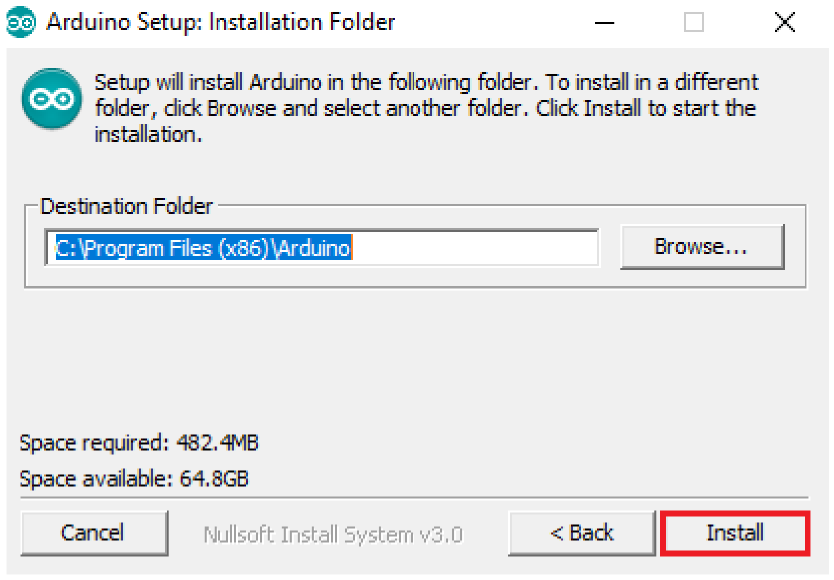
Select components to install:

- ☑ Install Arduino software
- ☑ Install USB driver
- ☑ Create Start Menu shortcut
- ☑ Create Desktop shortcut
- ☑ Associate .ino files

Space required: 482.4MB

Cancel
Nullsoft Install System v3.0
< Back
Next >

Click on **Install.**



Arduino Setup: Installation Folder

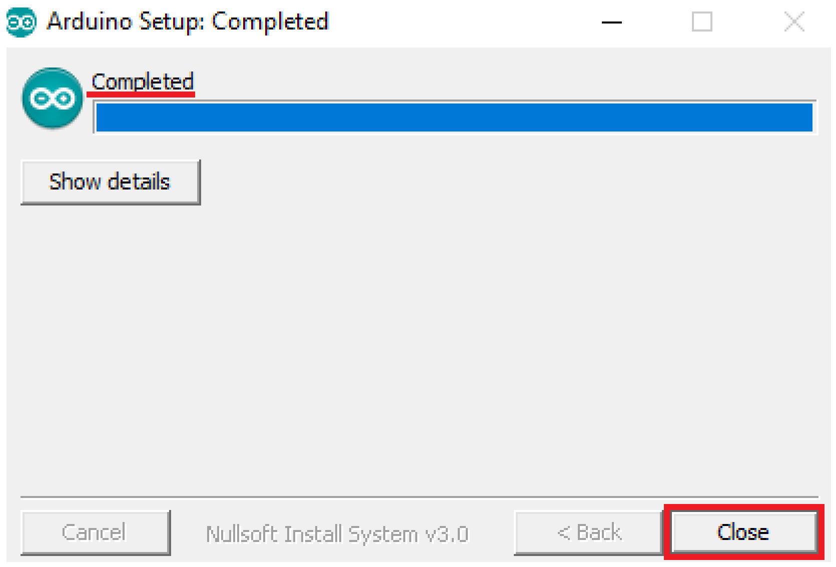
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

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

Space required: 482.4MB

Space available: 64.8GB

When the setup is finished, click on **Close.**



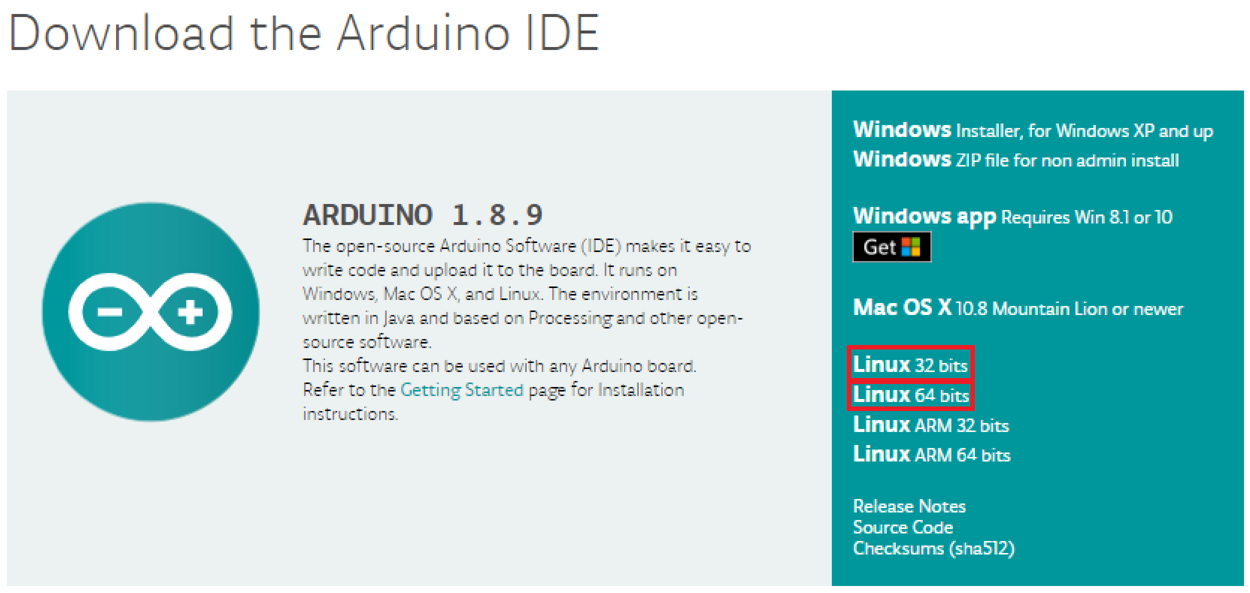
Arduino Setup: Completed

Completed

1.3 Installing Arduino (Linux)

You can download it from this link:
https://www.arduino.cc/en/Main/Software

Select **Linux 32 bits** or **Linux 64 bits** the one it is compatible with your Linux.



Download the Arduino IDE

ARDUINO 1.8.9

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, for Windows XP and up
Windows ZIP file for non admin install

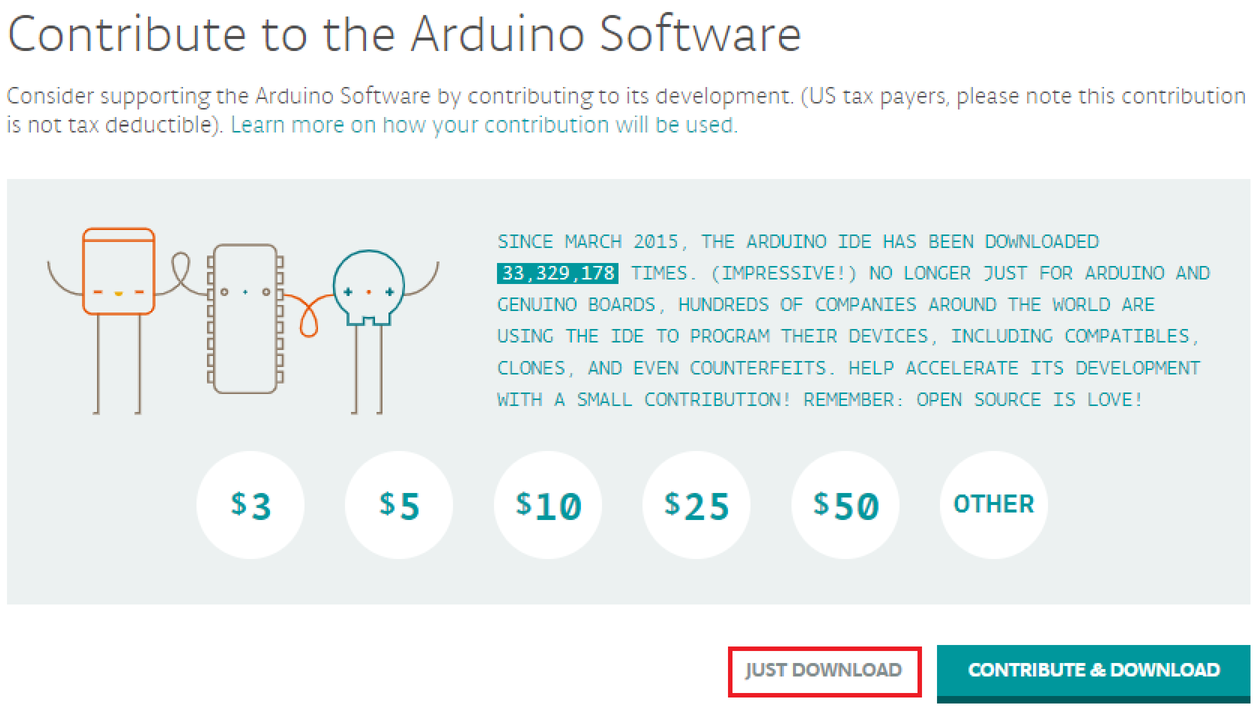
Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

Click on **JUST DOWNLOAD.**

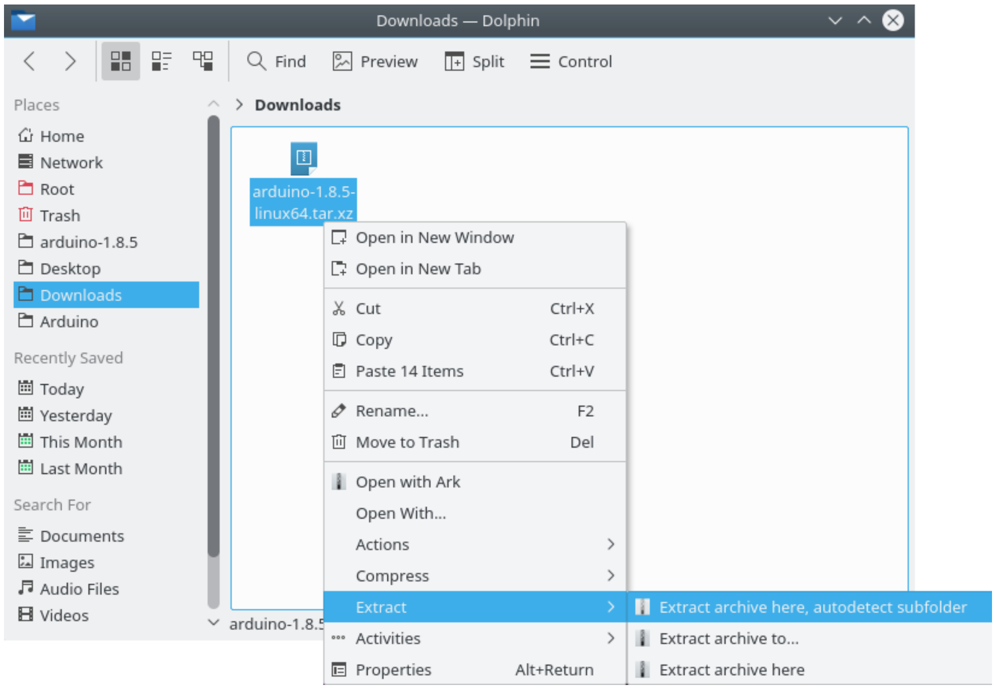


Contribute to the Arduino Software

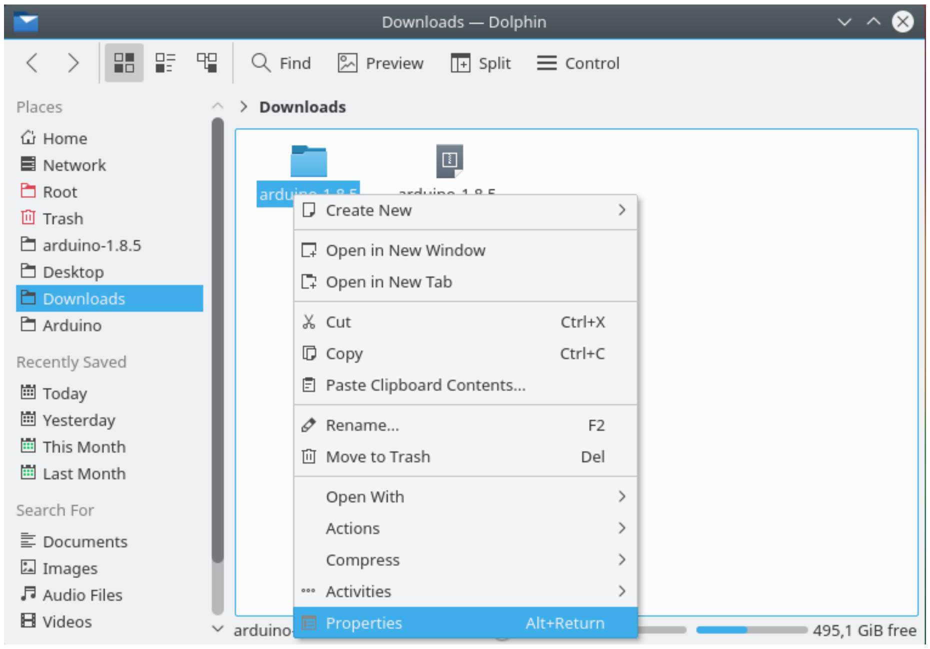
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 33,329,178 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!

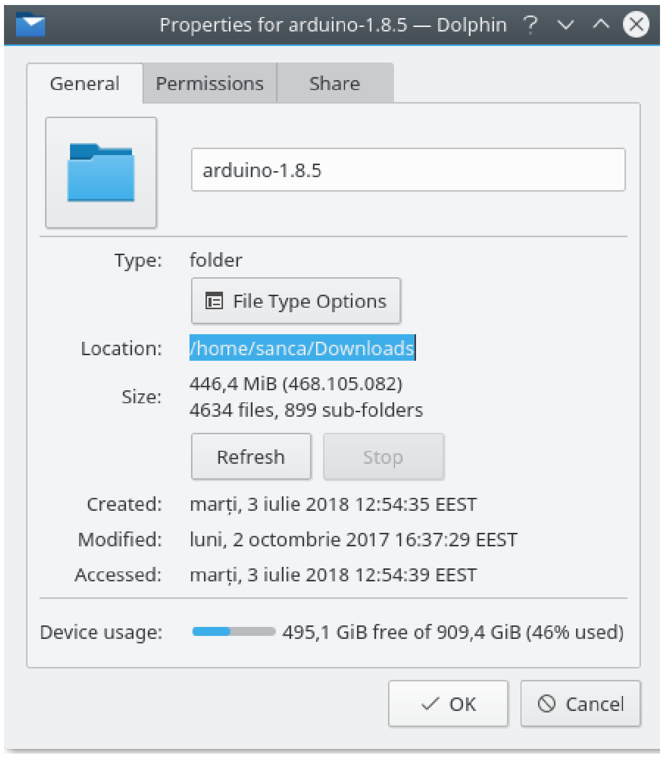
Find the file that you downloaded, Right Click on it, and extract the files in a folder of your choice.



Right click on the folder and containing the files extracted and go to Properties.



Copy the location of the folder.



arduino-1.8.5

Type: folder

Location: /home/sanca/Downloads

Size: 446,4 MiB (468.105.082)
4634 files, 899 sub-folders

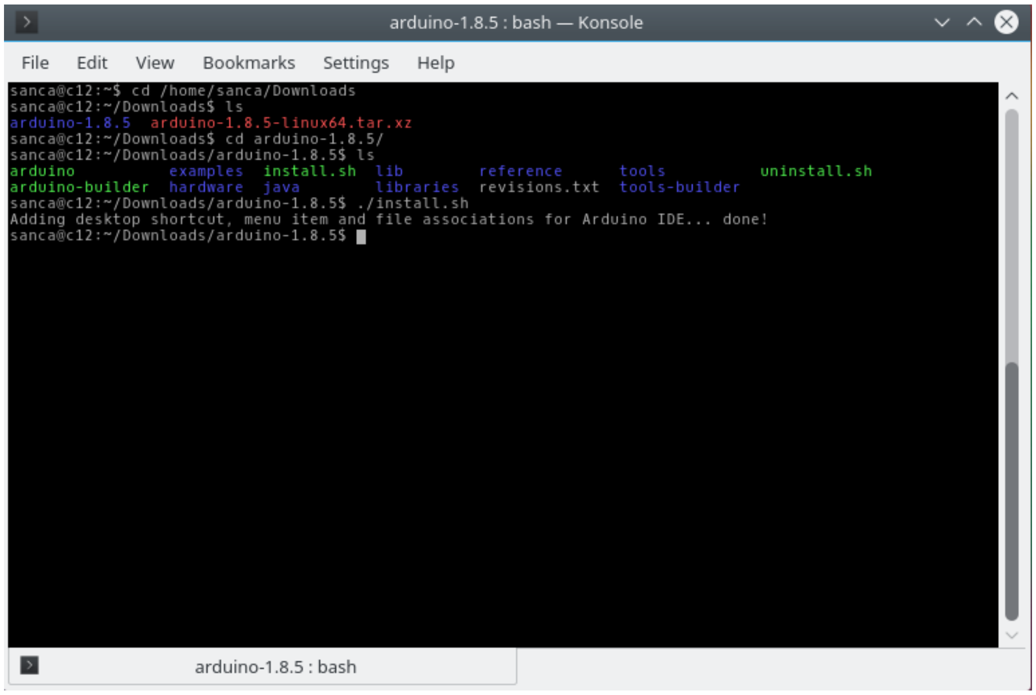
Created: marți, 3 iulie 2018 12:54:35 EEST

Modified: luni, 2 octombrie 2017 16:37:29 EEST

Accessed: marți, 3 iulie 2018 12:54:39 EEST

Device usage: 495,1 GiB free of 909,4 GiB (46% used)

Open a terminal and **cd** to that location, List all the files in that folder using **ls,** And **cd** to the folder containing the files extracted, And now just type **./install.sh** and press **Enter.**



```
sanca@c12:~$ cd /home/sanca/Downloads
sanca@c12:~/Downloads$ ls
arduino-1.8.5  arduino-1.8.5-linux64.tar.xz
sanca@c12:~/Downloads$ cd arduino-1.8.5/
sanca@c12:~/Downloads/arduino-1.8.5$ ls
arduino          examples  install.sh  lib        reference      tools          uninstall.sh
arduino-builder  hardware  java        libraries  revisions.txt  tools-builder
sanca@c12:~/Downloads/arduino-1.8.5$ ./install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE... done!
sanca@c12:~/Downloads/arduino-1.8.5$ 
```

You can close the terminal.

Lesson 2: Add Libraries and Open Serial Monitor

2.1 Installing Additional Arduino Libraries

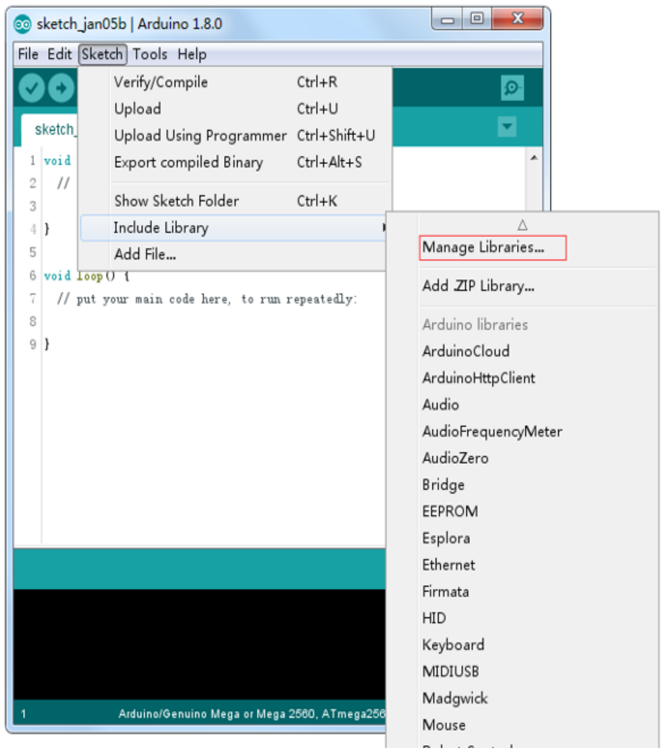
By now, you should be familiar with the Arduino software and you may want to explore the capabilities of your Arduino with additional libraries.

2.2 What are Libraries?

Libraries consist of lots of code, written to make it easier for you to connect to a sensor, display, module, etc. As an example, the built-in LiquidCrystal library makes it easy to use LCD displays. On the internet, there are hundreds of additional libraries available for download. You will need to install those additional libraries in order to use them.

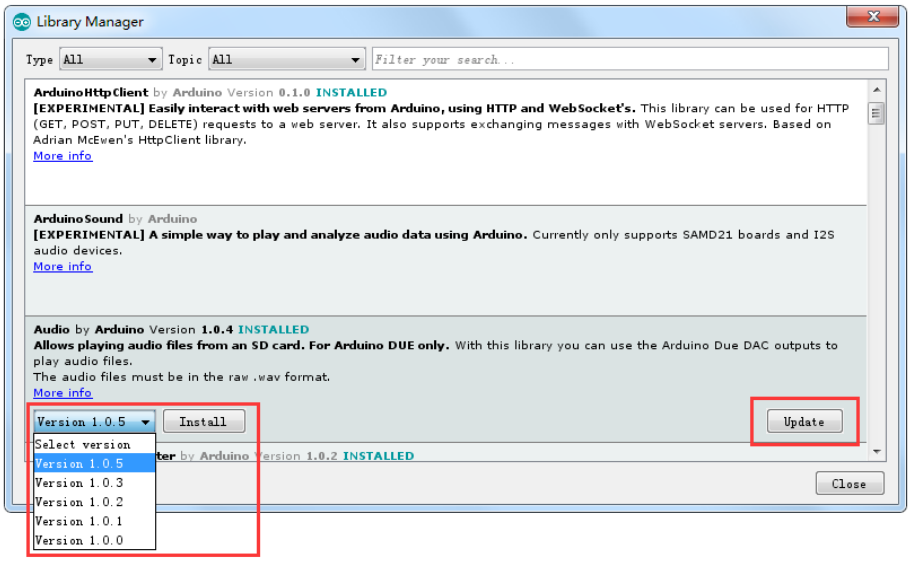
2.3 How to Install a Library

You can install a new library through the Library Manager (available from IDE version 1.8.0). Open the IDE and on the "Sketch" menu, choose Include Library > Manage Libraries.



```
void
  //

}

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

}
```

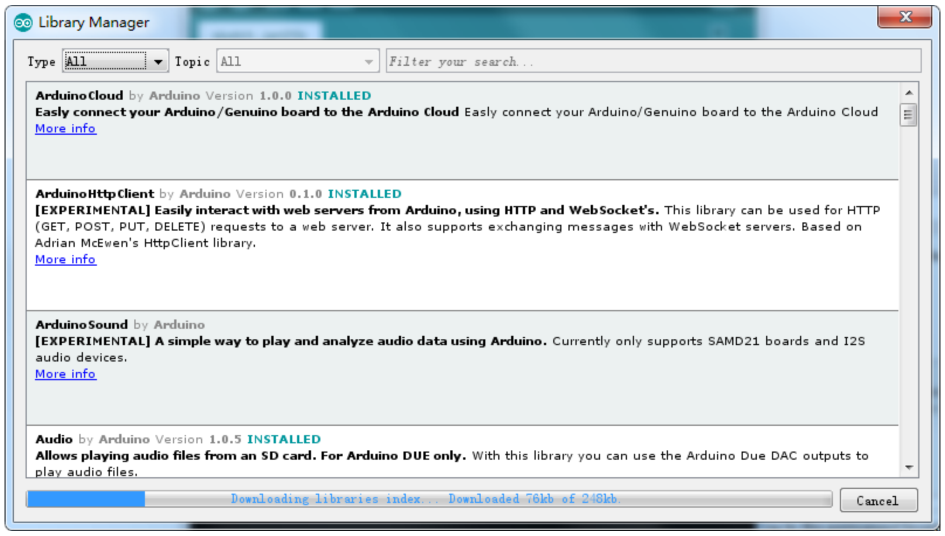
When the library manager opens, a list of installed libraries will be displayed as well as those libraries that are ready for installation. Take for example the installation of the Bridge library. Look for it in the list and then install the version of the library that you prefer. There are times that version selection menu does not appear, but that is normal.



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



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

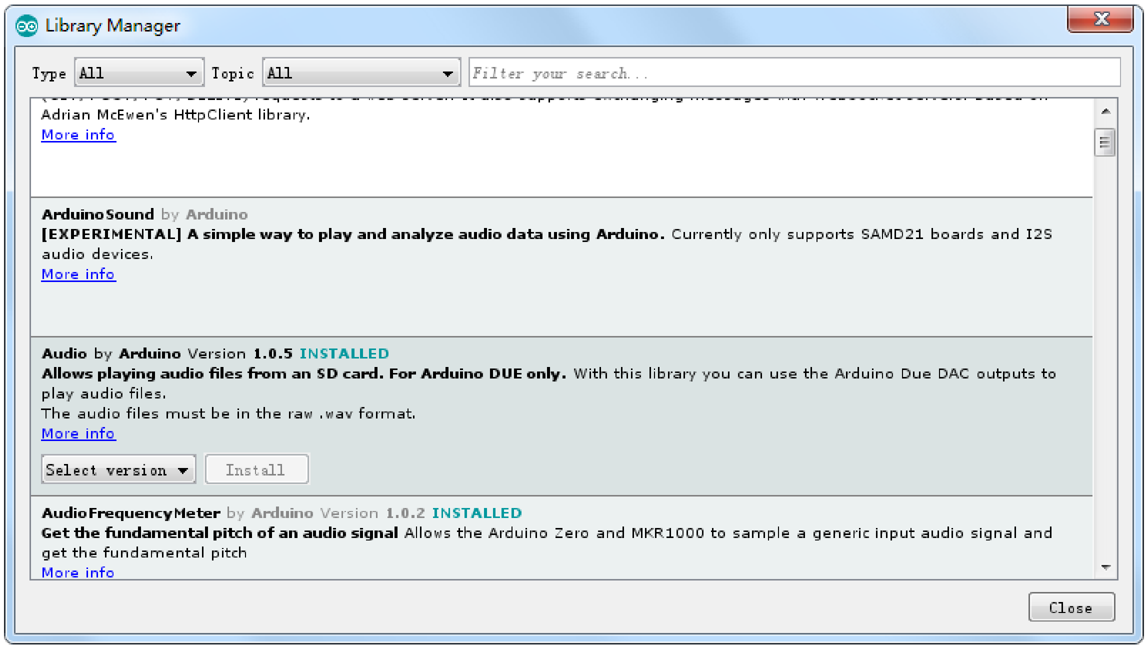
ArduinoHttpClient by Arduino Version 0.1.0 INSTALLED
[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's. This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian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

Click “Install” and wait for the new library to be installed. Once the process has finished, you can see an INSTALLED tag next to the Bridge library.



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

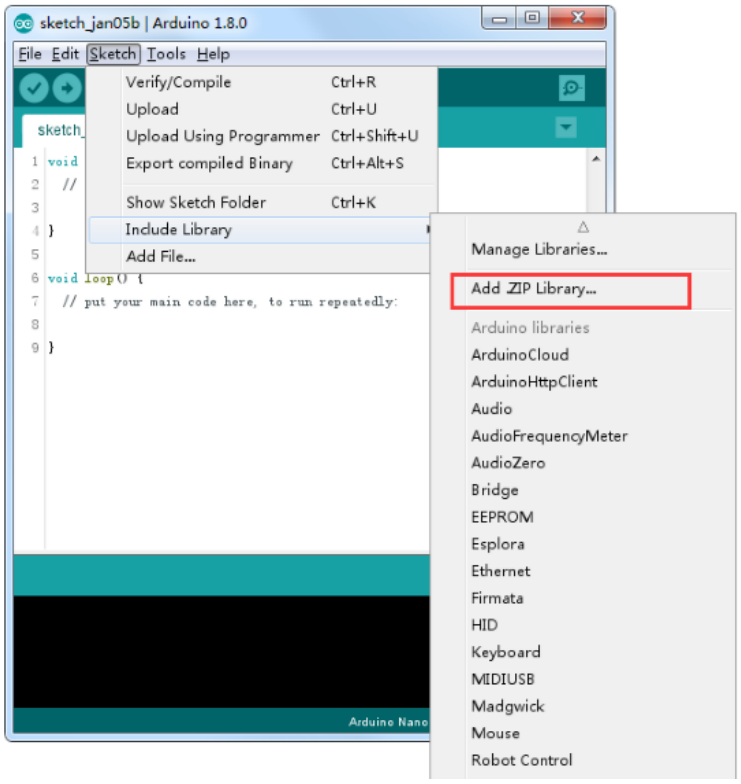
AudioFrequencyMeter by Arduino Version **1.0.2** INSTALLED
Get the fundamental pitch of an audio signal Allows the Arduino Zero and MKR1000 to sample a generic input audio signal and get the fundamental pitch
More info

The new library can now be located inside the Include Library menu. You can open a new issue on Github, if you wish to add your own library.

2.4 Importing a .zip Library

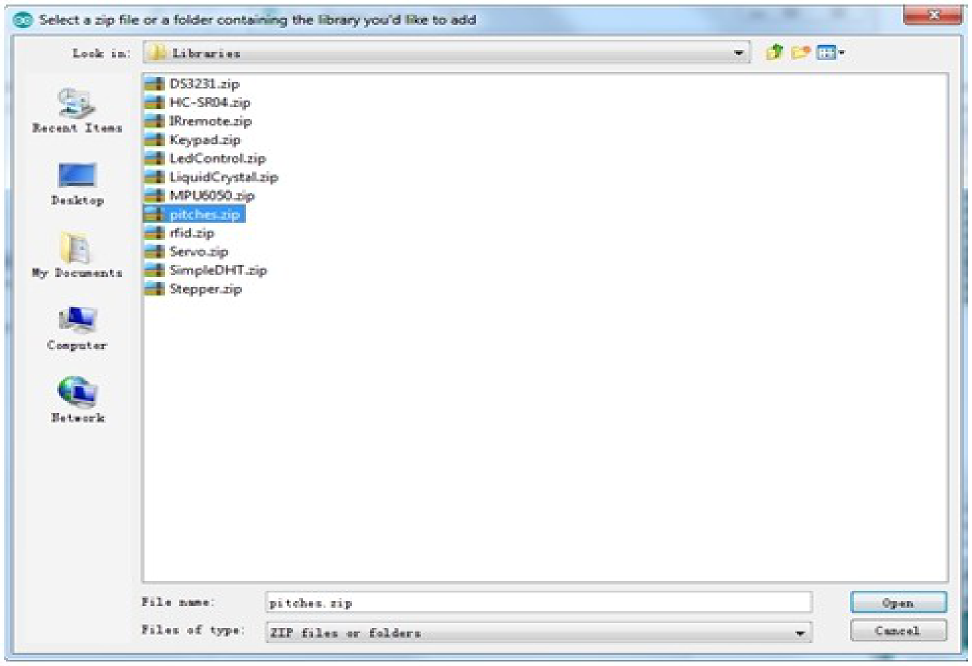
Libraries are most frequently distributed as a ZIP file or folder, with the name of the folder being the name of the library. Inside the folder, it will contain a .cpp file, a .h file and often a keywords.txt file, an examples folder, and any other files that are required by the library. Starting with version 1.0.5, 3rd party libraries can be installed in the IDE, without the need to unzip the downloaded library.

In the Arduino IDE, click Sketch menu > Include Library. Select "Add .ZIP Library".



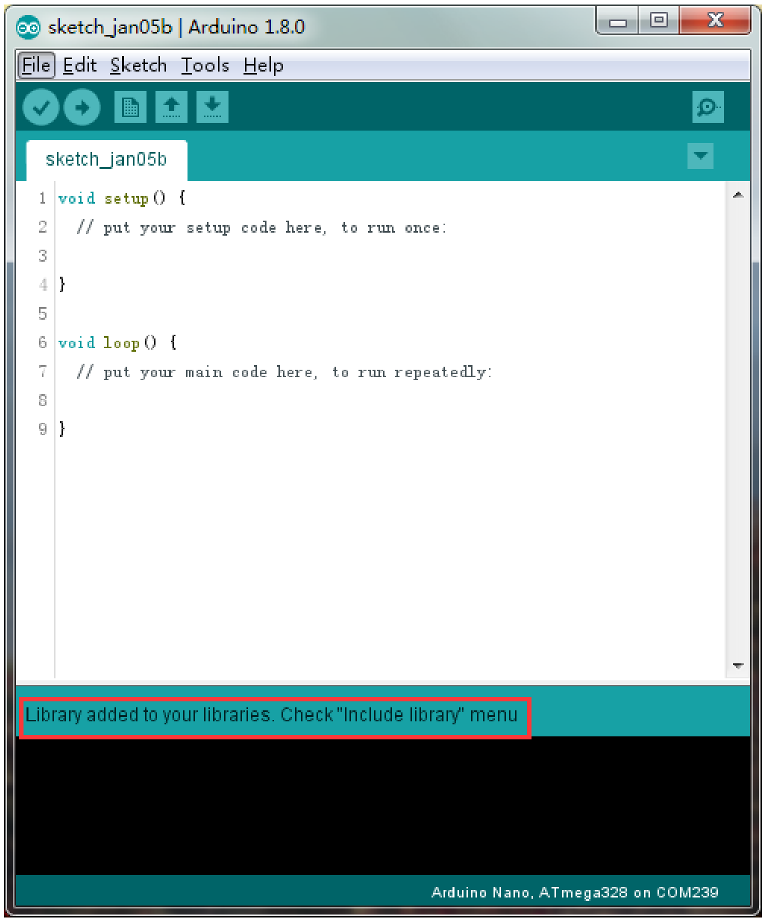
```
void
  //
}

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

}
```

Next step is you will need to select the library you would like to add. Go to the .zip file's location and open it.



File name: pitches.zip

Files of type: ZIP files or folders



```
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

Go back to the Sketch > Import Library menu. The library can now be seen at the bottom of the menu, indicating that it is ready to be used. The zip file will be present in the libraries folder in your Arduino sketches directory.

NB: The Library will be available to use in sketches, but examples will be displayed in the File > Examples only after the IDE restarts..

Those two are the most common method, with MAC and Linux systems being handled similarly. You can also proceed to manual installation, as presented below, but it is seldom used and you may skip it.

2.5 Manual installation

To install the library, first make sure the Arduino application is closed. Extract the files from the ZIP folder, you should now have a folder with the name of the library, containing files with the extensions .cpp and.h inside, with a few other additional ones. (In the case that these files are not there, you need to create them yourself).

Let’s take for example a library called ArduinoLove. Drag the ArduinoLove folder into your libraries folder. For Windows users, it usually appears as "My Documents\Arduino\libraries". For Mac and Linux users, it should be the ‘libraries’ folder where your Arduino documents are located.

Your Arduino library folder will be like this (on Windows):

My Documents\Arduino\libraries\ArduinoLove\ArduinoLove.cpp
My Documents\Arduino\libraries\ArduinoLove\ArduinoLove.h
My Documents\Arduino\libraries\ArduinoLove\examples

or like this (on Mac and Linux):

Documents/Arduino/libraries/ArduinoLove/ArduinoLove.cpp
Documents/Arduino/libraries/ArduinoLove/ArduinoLove.h
Documents/Arduino/libraries/ArduinoLove/examples

There are often more files other than the .cpp and .h, you just have to ensure that they can be found there (if you put the .cpp and .h files directly into the libraries folder, the library may not work or if they're inside an extra folder).

For example:

Documents\Arduino\libraries\ArduinoLove.cpp and
Documents\Arduino\libraries\ArduinoLove\ArduinoLove\ArduinoLove.cpp won't work.

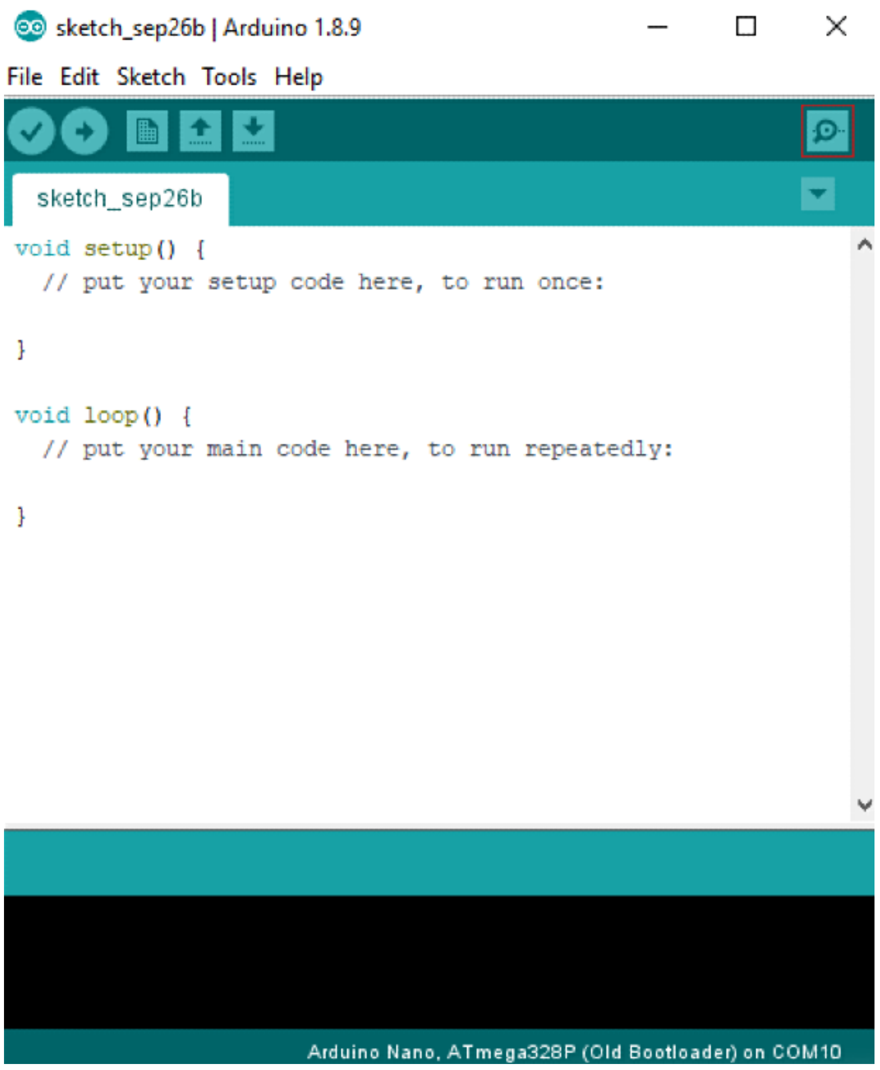
Restart the Arduino application. The new library should now appear in the Sketch->Import Library menu item of the software. Congratulations, you’ve installed a library!

2.6 Arduino Serial Monitor (Windows, Mac, Linux)

As mentioned in the beginning, the software side of the Arduino platform is called the Arduino Integrated Development Environment (IDE). In this software, a serial terminal is included because using such terminal is such a major part of working with an Arduino or other microcontrollers. The terminal is called the Serial Monitor.

2.7 Making a Connection

The serial monitor comes with all versions of the Arduino IDE. Click the Serial Monitor icon to open it.

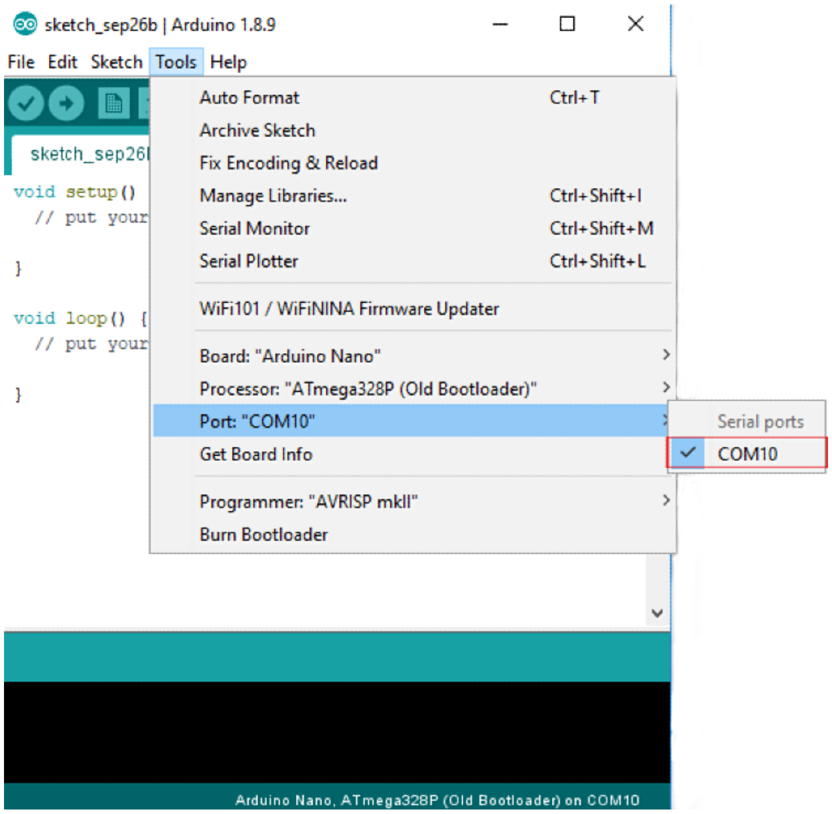


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Arduino Nano, ATmega328P (Old Bootloader) on COM10

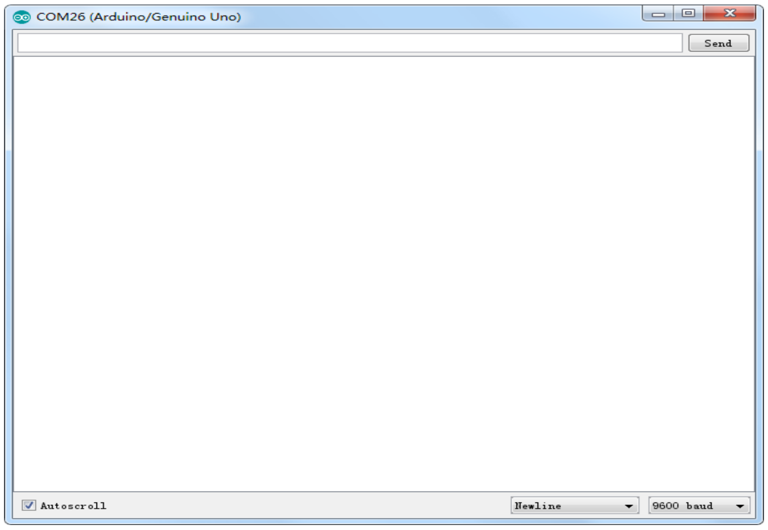
Choosing which port to open in the Serial Monitor is very similar to choosing a port for uploading your code. To do that, go to Tools -> Serial Port, and select the adequate port.

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


sketch_sep26b | Arduino 1.8.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port: "COM10"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM10
Arduino Nano, ATmega328P (Old Bootloader) on COM10

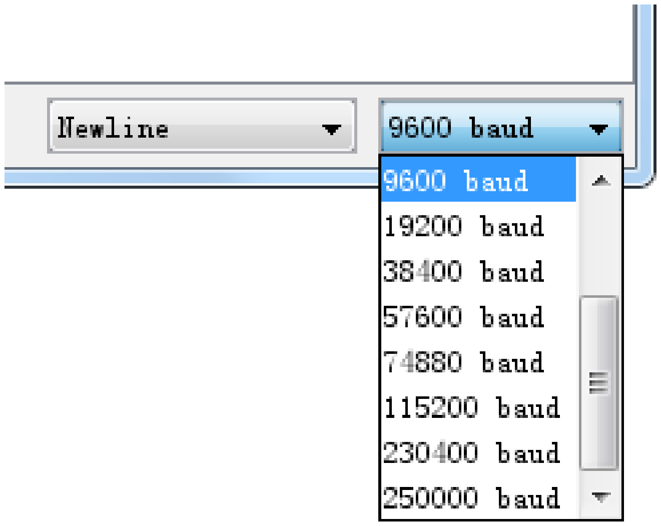
```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Once it is open, you should see something like below:

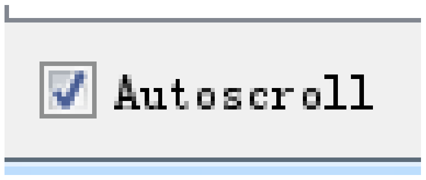


2.8 Settings

There are limited settings on the Serial Monitor, but enough to handle most of serial communication needs.. The baud rate is the first setting you can change. Choose the correct baud rate (9600 baud) using the drop-down menu on selecting baud rate.



You can also set the terminal to Autoscroll by checking the box beside the Autoscroll option.



Pros

The Serial Monitor is a quick and easy way to create a serial connection with your Arduino. But if you are already working in the Arduino IDE, there's no need to set up a separate terminal to display data.

Cons

There is a lack of settings in the Serial Monitor, so it is not suitable for advanced serial communications.

Lesson 3: Blink

3.1 Overview

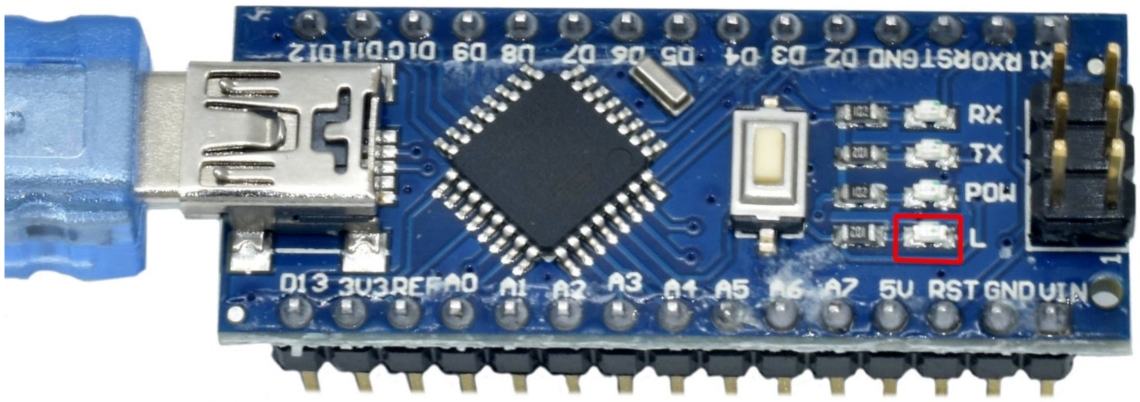
This chapter will teach you to program the Nano R3 controller board to blink the built-in LED in your Arduino and the fundamental steps in downloading programs.

3.2 Component Required

1 x Plusivo Nano R3 board

3.3 Principle

You can find in the Nano R3 board, along both sides, the rows of connectors. These are used to connect to several electronic devices and plug-in 'shields' that extend its capability. It also has a single LED, built onto the board, that you can control from your sketches,. This is often referred to as the 'L' LED, same as how it is labelled on the board.



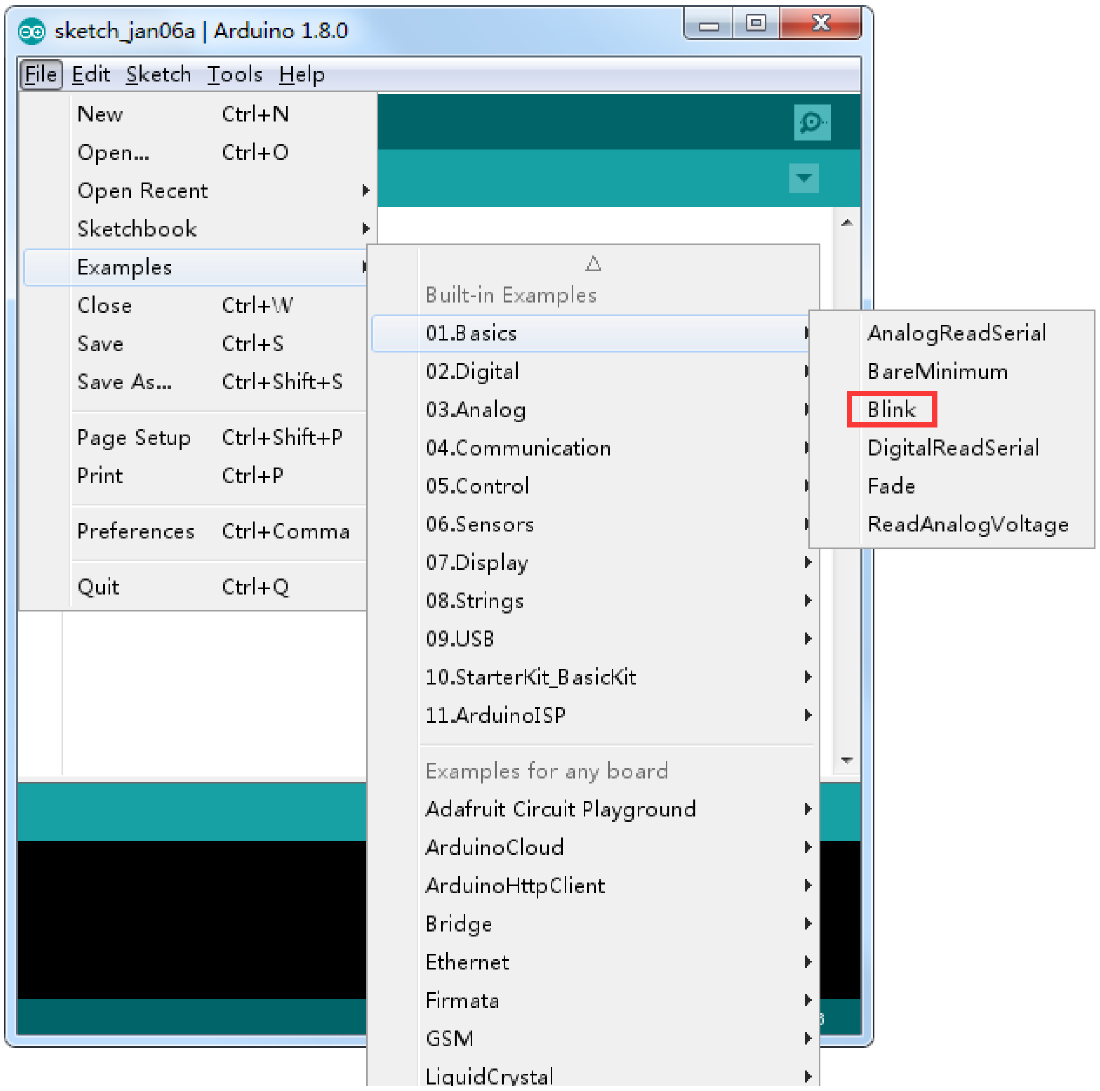
When you connect your board to a USB plug, you should see that the 'L' LED already blinks, as the 'Blink' sketch is generally pre-installed on the board.

In this chapter, we will rewrite our own Blink sketch and then change the rate at which it blinks.

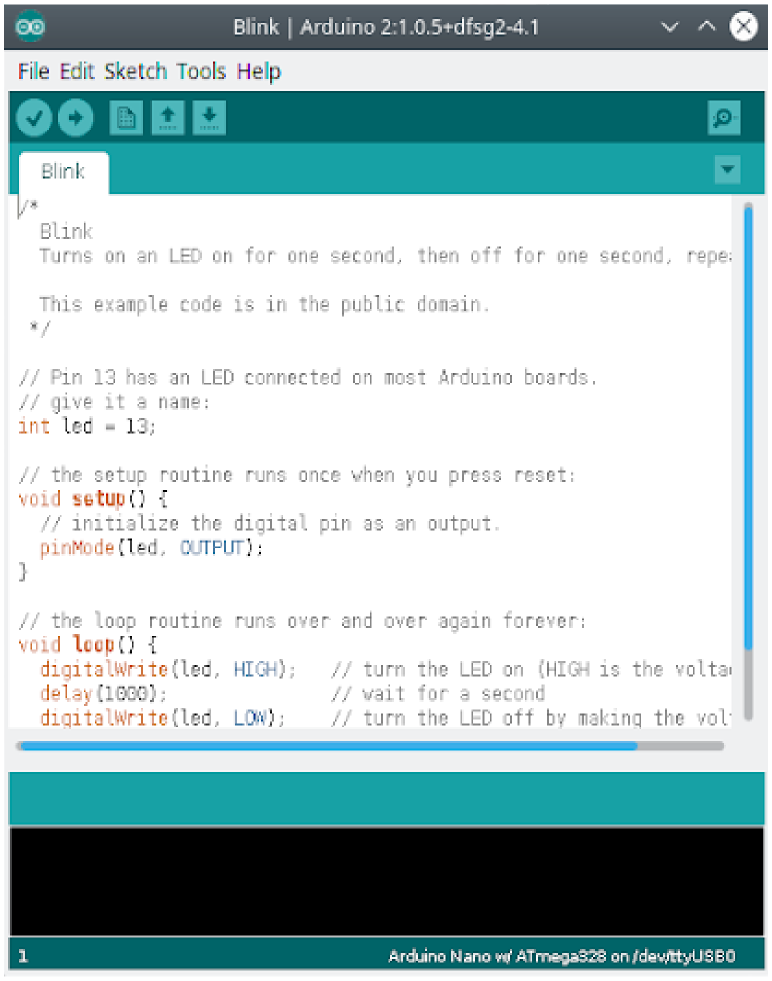
In the first lesson, we covered how to set up your Arduino IDE and ensure that you will be able to find the right serial port for it to connect to your Nano R3 board. Now it is time to test that connection and program your Nano R3 board.

The Arduino IDE consists of a large collection of example sketches that you can load up and use, including an example sketch on how to make the 'L' LED blink.

In the IDE’s menu system folder, go to File >Examples > 01.Basics and load the 'Blink' sketch.



Enlarge the sketch window to see the entire sketch.

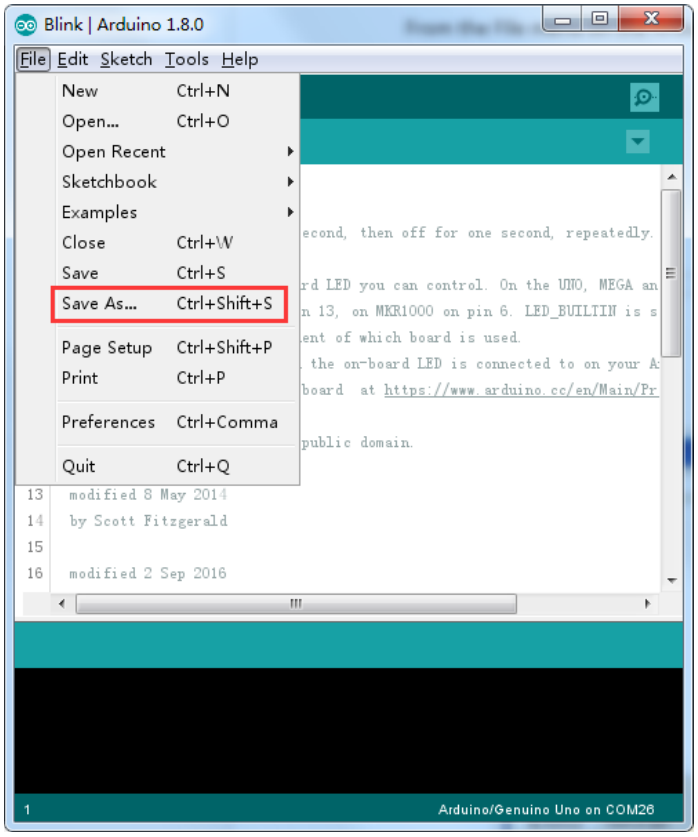

Blink | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the volta
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the vol
```

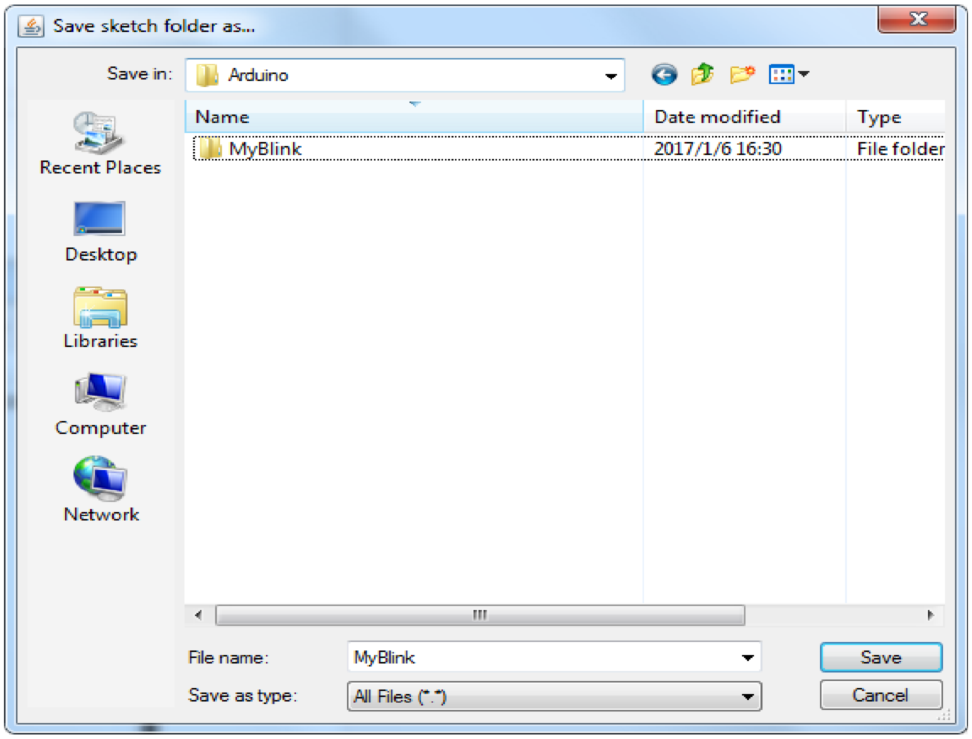
```
1                    Arduino Nano w/ ATmega328 on /dev/ttyUSB0
```

The example sketches are 'read-only', you can change them but cannot be saved as the same file. Thus, first you have to save your own copy as a new file since we are going to change this sketch.

On the Arduino IDE File menu, select 'Save As..', and save it with filename 'MyBlink'.



```
econd, then off for one second, repeatedly.

rd LED you can control. On the UNO, MEGA an
n 13, on MKR1000 on pin 6. LED_BUILTIN is s
ent of which board is used.
 the on-board LED is connected to on your A
board at https://www.arduino.cc/en/Main/Pr

public domain.
 modified 8 May 2014
 by Scott Fitzgerald

 modified 2 Sep 2016
```

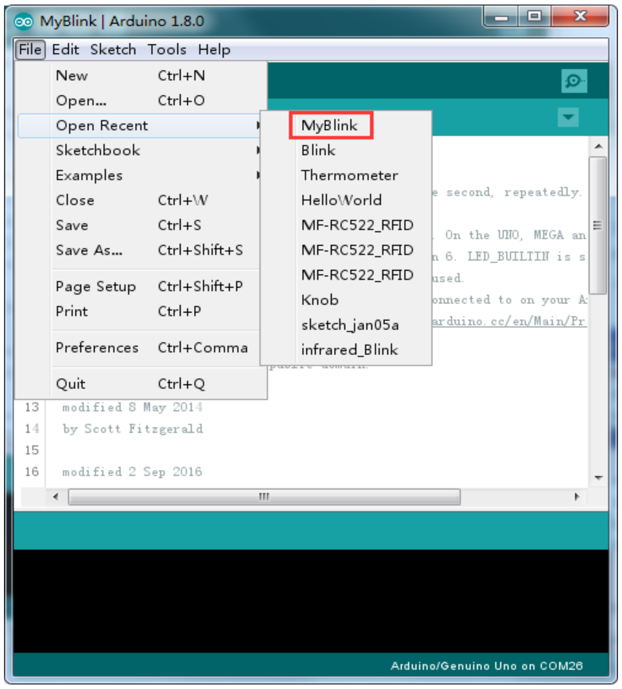


Name	Date modified	Type
MyBlink	2017/1/6 16:30	File folder

File name: MyBlink

Save as type: All Files (*.*)

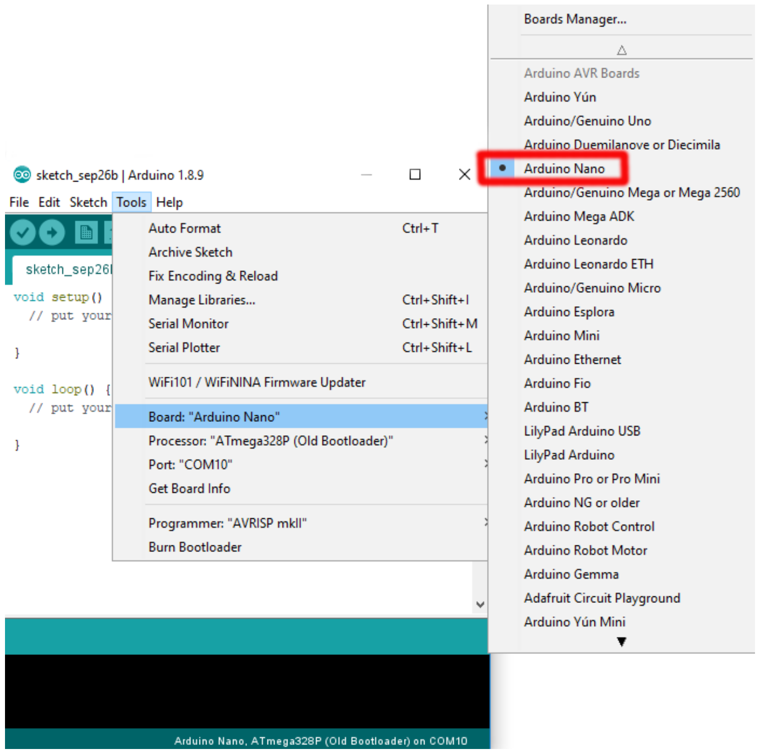
You have just saved your copy for blinking the ‘L’ LED as ‘MyBlink’ in your sketchbook, and when you want to use it you can just open it from the File > Sketchbook menu option.



```
  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
```

Arduino/Genuino Uno on COM26

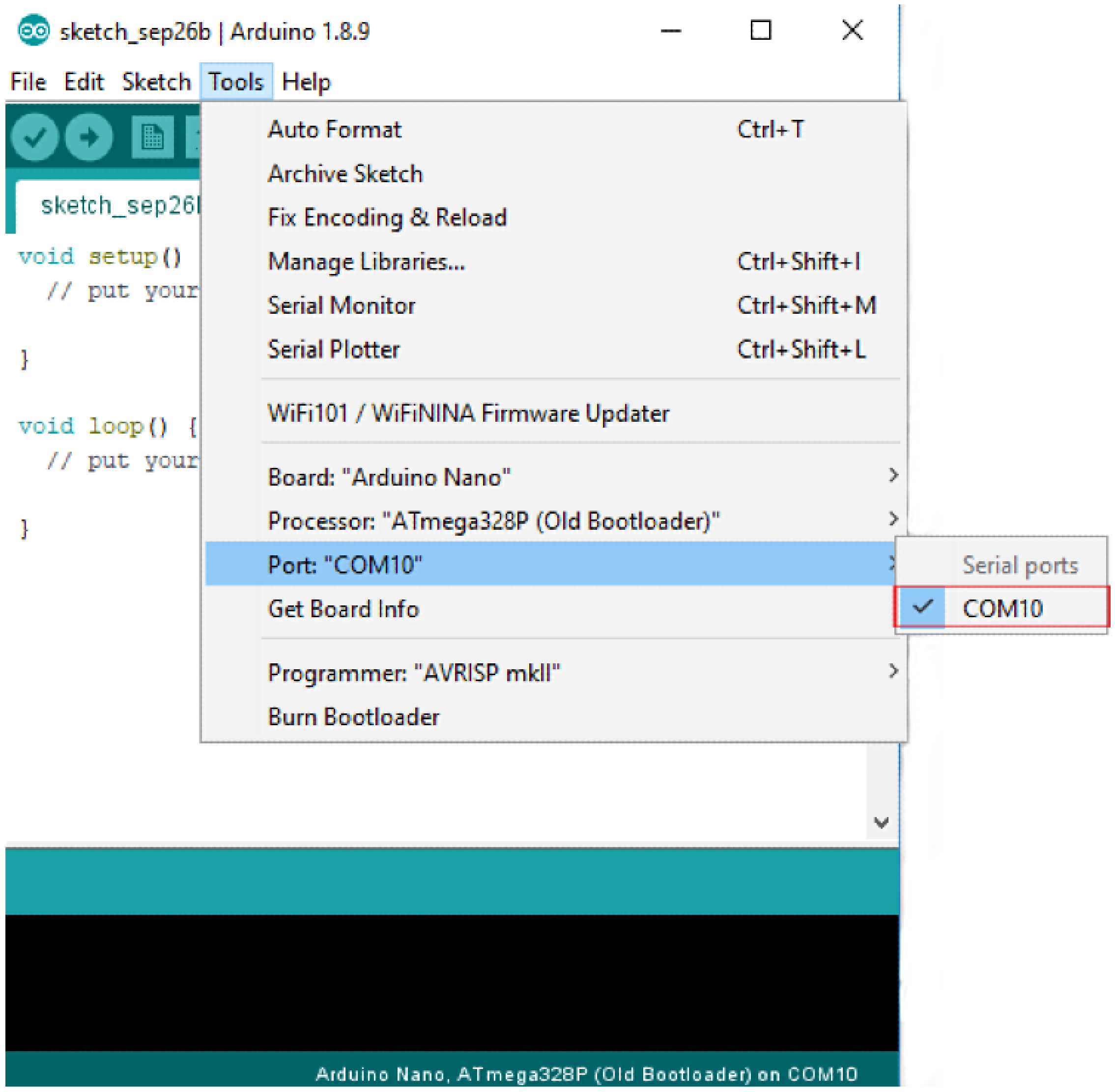
Using USB cable, connect your Arduino board to your computer and check that both the 'Board Type' and 'Serial Port' are set correctly.



```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port: "COM10"
Programmer: "AVRISP mkII"

Arduino Nano, ATmega328P (Old Bootloader) on COM10

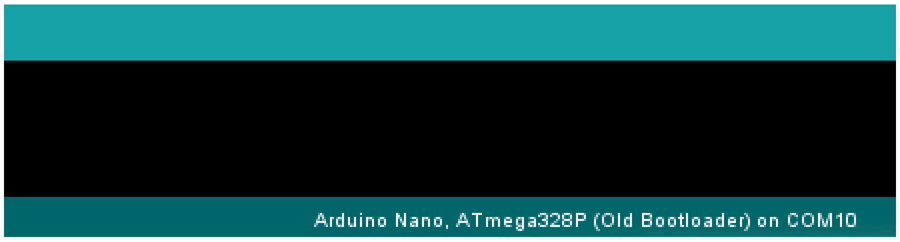

sketch_sep26b | Arduino 1.8.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port: "COM10"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM10
sketch_sep26b

```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

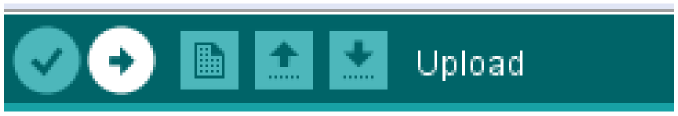
Arduino Nano, ATmega328P (Old Bootloader) on COM10

Note: The Board Type and Serial Port shown here may appear differently as the Serial Port displayed for everyone is different. COM110 that is shown here may be COM3 or COM4 on your computer. The right COM port should be COMX (arduino XXX), chosen according to the certification criteria. Also, if you are using 168, the Board type you will have to choose is Mega 168, or choose the type adequate to your situation.

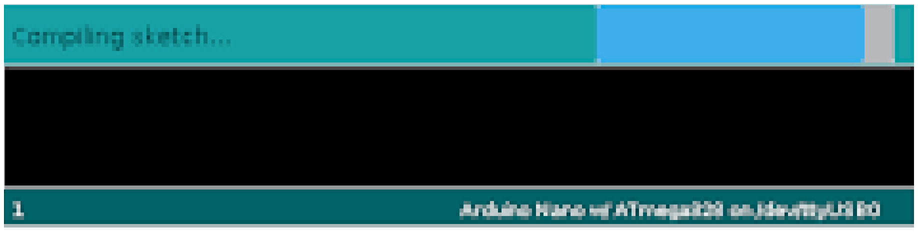
The current settings for the board are shown at the bottom corner of the window.


Arduino Nano, ATmega328P (Old Bootloader) on COM10

Click on the 'Upload' button, the second icon on the toolbar.



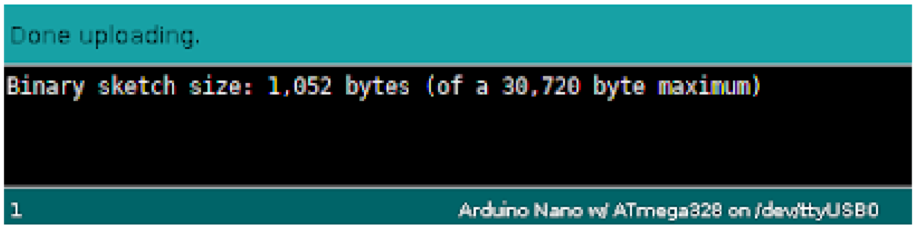
On the status area of the IDE, a progress bar and a series of messages will be shown. It will say 'Compiling Sketch...' in the beginning, indicating that it is converting the sketch into a suitable format for uploading to the board.


Compiling sketch...

1 Arduino Nano w/ ATmega328 on /dev/ttyUSB0

Next, the status will become 'Uploading'. During this time, the sketch is being transferred and it would cause the LEDs on the Arduino to start flickering

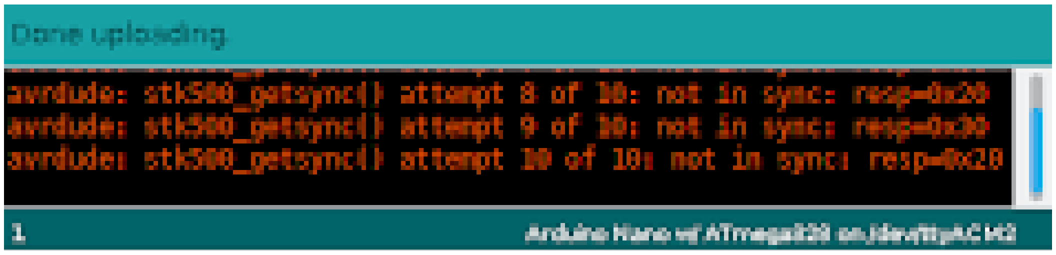
Lastly, the status will be 'Done uploading'.


Done uploading.

```
Binary sketch size: 1,052 bytes (of a 30,720 byte maximum)
```

1 Arduino Nano w/ ATmega328 on /dev/ttyUSB0

There is also a message that says that the sketch is using 1,052 bytes of the 30,720 bytes available. After the 'Compiling Sketch..' stage, the following error message may appear:



```
Done uploading
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x20
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x30
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x20
1                                   Arduino Nano w/ ATmega328 on /dev/ttyACM0
```

This can mean that your board is not properly connected, that the necessary drivers have not been installed or maybe you selected a wrong serial port. If you encounter this error, you should revisit Lesson 10. Once the upload is complete, the board should restart and start blinking.

Please note that a big part of this sketch is composed of comments. These are not actual program instructions, but an explanation on how the program works. A block comment is everything between /* and /*, and a single line comment is the one that starts with //. Everything up until the end of that line is considered a comment.

Below is the first line of code:

```
int led = 13;
```

This code is giving a name to the pin where the LED is attached to. On most Arduinos, including the Nano, UNO and Leonardo, this is pin 13. Then, we have the 'setup' function. This is executed when the reset button is pressed, and also whenever the board resets for any reason or after a sketch has been uploaded.

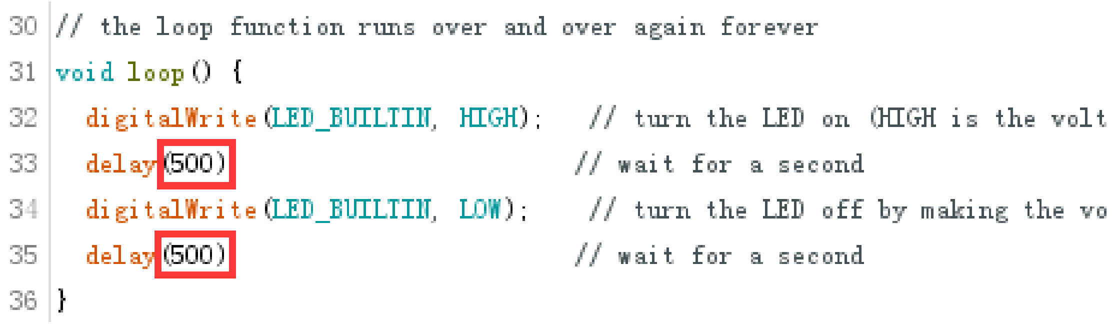
```
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
```

Every Arduino sketch must have a 'setup' function, and you can add other instructions between the { and the } brackets. In this case, there is only one command telling the Arduino that we are using the LED pin as an output. To have a ‘loop’ function is also mandatory for a sketch. After a reset, the 'loop' function will immediately start again after it has finished running its commands. This is unlike the 'setup' function that only runs once.

```
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 turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off (LOW) and pause for another second.

To make your LED blink faster, you need to change the parameter in the brackets () 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
}
```

If you want your LED to blink twice as fast, change the value from 1000 to 500 milliseconds. The setting would then pause for half a second each delay rather than a whole second.

Upload the sketch again and your LED should now start to blink more quickly.

Lesson 4: LED

4.1 Overview

Here, you will learn how to change the brightness of LED using different values of resistors.

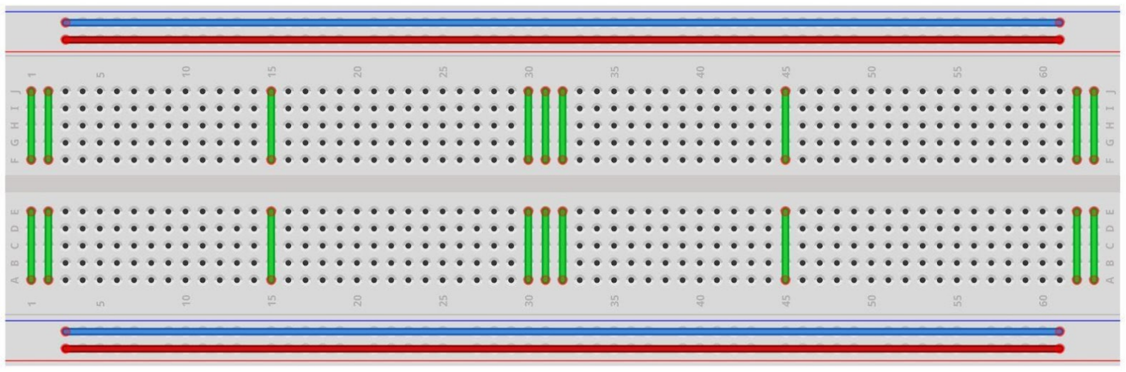
4.2 Components Required

1 x Nano R3
1 x BreadBoard 830p
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)

4.3 Component Introduction

BREADBOARD 830p

In a breadboard, you can prototype your circuits quickly without the need to solder the connections. You can see below an example of a breadboard.



These breadboards has various sizes and configurations. They simplest is just a grid of holes in a plastic block, where strips of metal inside provide electrical connection between the holes in the shorter rows. If you push the legs of two different components into the same row, they will be joined together electrically. The deep channel running down the middle indicates a break in connections there, which means placing a chip in with the legs at either side of the channel does not connect them together.

Some breadboards have two strips of holes (also called rails) running along the long edges of the board separated from the main grid, with strips running down the length of the board inside that enable you to connect at a common voltage. They are usually for +5 volts and ground.

While breadboards are great for prototyping, they have some limitations due to potential poor connections. Because the connections are temporary, they are not as reliable as soldered connections.

LED

LEDs are great for making indicator lights as they use very little electricity and they last longer. In this lesson, the most common of all LEDs will be used: a 5mm-diameter red LED (5mm LED). You cannot connect it directly to a battery or voltage source, as the LED has a positive and a negative lead and won't light if connected the wrong way and a resistor must be used with it to limit the current flowing through it so that it won't burn out.

Not using a resistor with LED will instantly destroyed it, as the excessive current flowing through will harm the 'junction' where the light is produced.

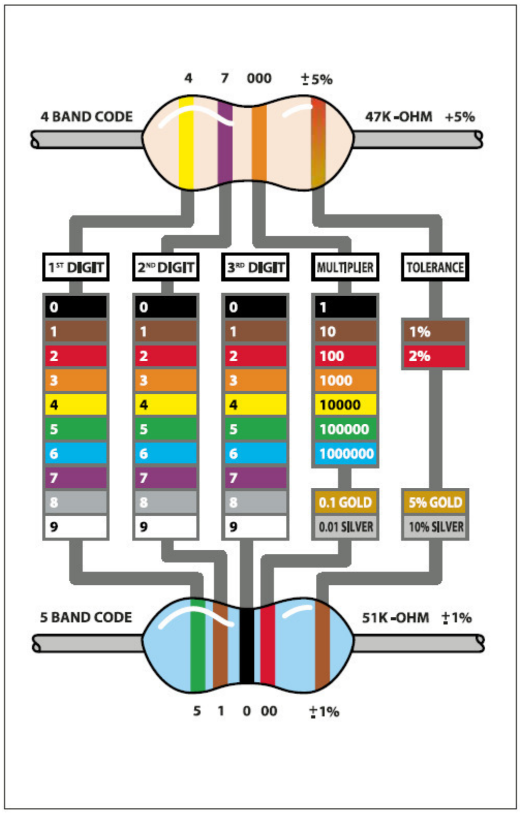
There are two ways to identify the two leads (positive and negative) of the LED. First, the longer lead is the positive lead. Second, you can see a flat edge to the case of the LED where the negative lead enters the body of the LED. If you have LED with flat side next to the longer lead, it can be assumed that the longer lead is positive.

RESISTORS

Resistors resist the flow of electricity, which means higher value of the resistor means that the more it can resist and less electrical current will flow through it. We use these to control how much electricity flows through the LED and eventually, how brightly the LED will shine.

The unit of resistance is Ohm with symbol Ω, from the Greek letter Omega. We also represent the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω), called kilo-ohms and mega-ohms.

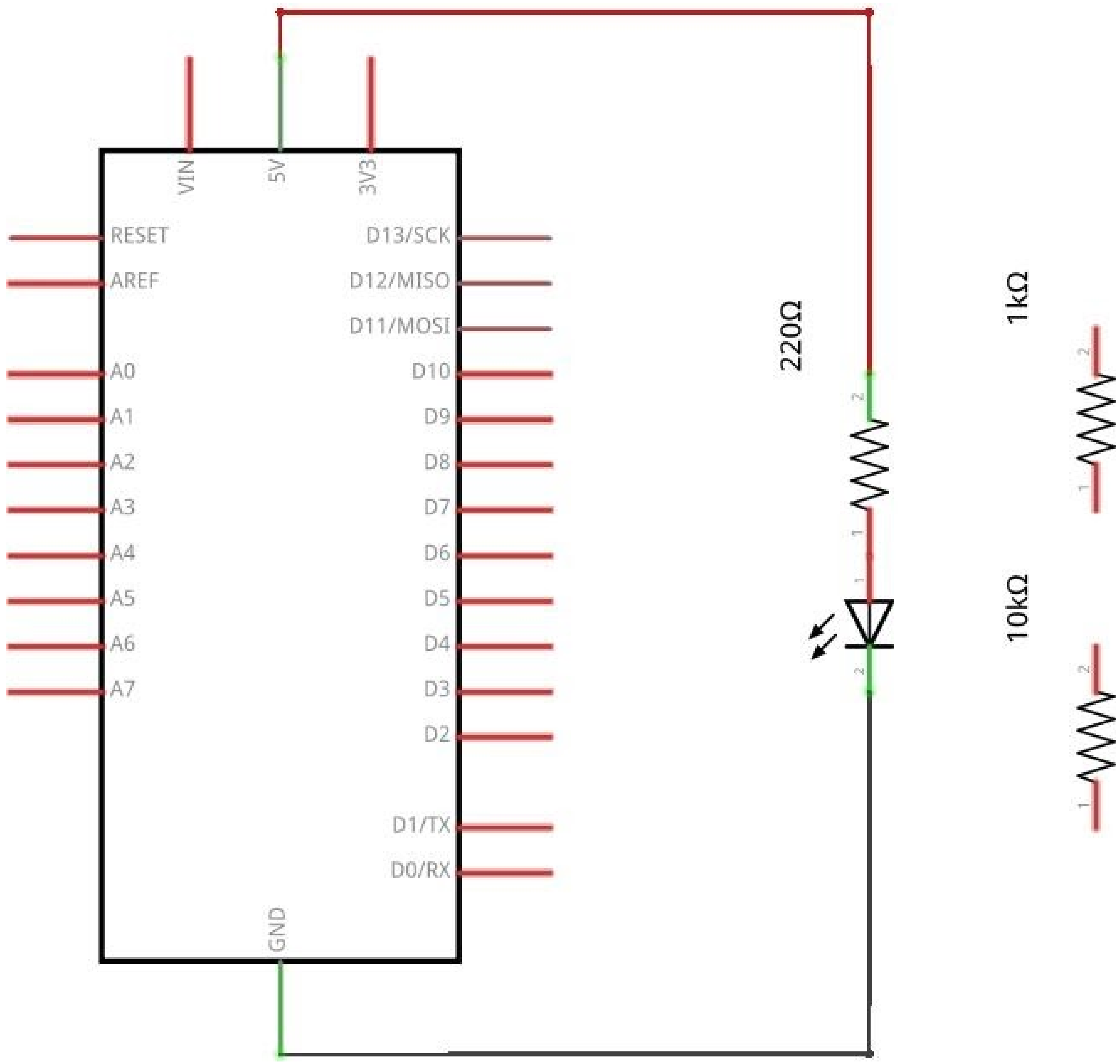
In this lesson, three different values of resistor will be used: 220Ω, 1kΩ and 10kΩ. These have the same appearance but have different colored stripes on them, which indicates the value of the resistor. The color code of a resistor is a three-colored stripes and a gold stripe at one end as seen below.



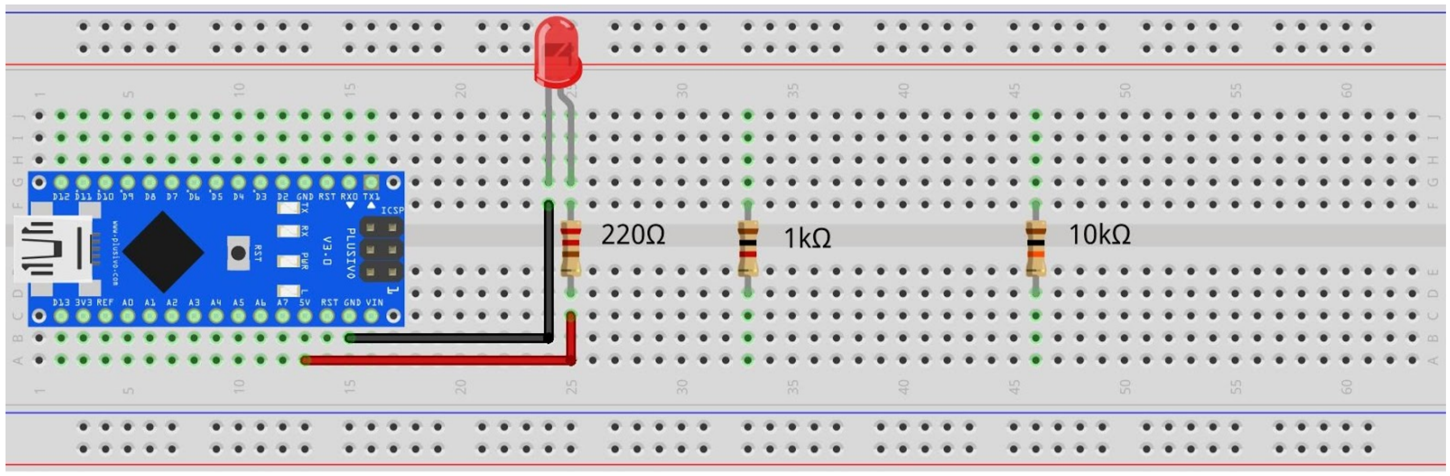
Resistors, unlike LEDs, can be connected either way as they don't have a positive and negative lead. If you find this colored lines method too complicated, you can use a digital multimeter to determine the resistance value of a resistor.

4.4 Connection

Schematic



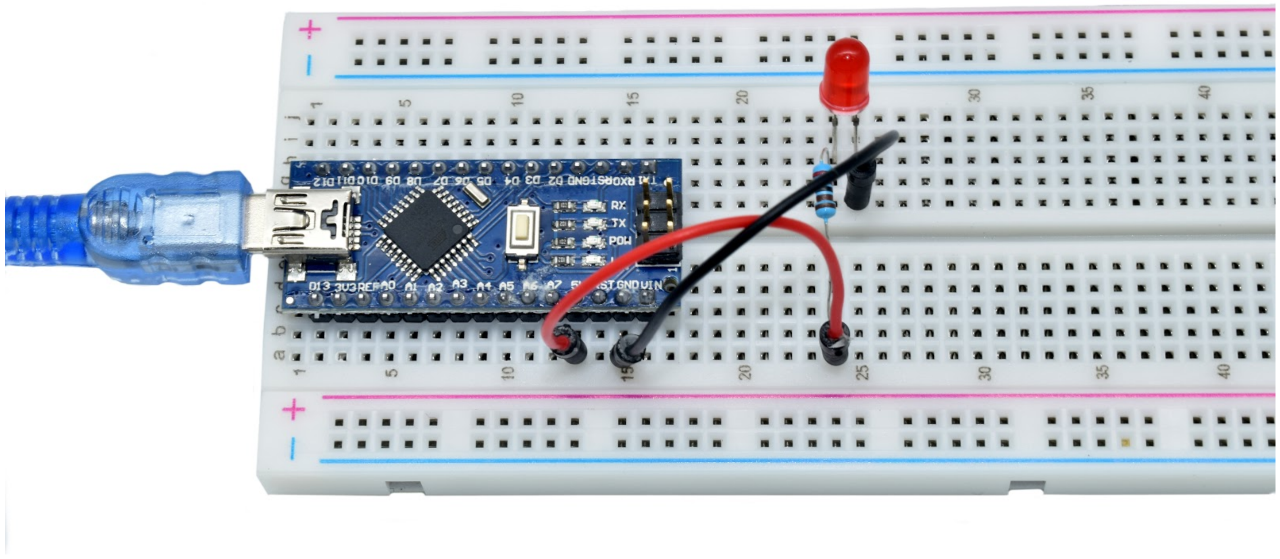
Wiring diagram



Plug your Nano (a convenient source of 5 volts) into your computer as this will be used to provide power to the LED and the resistor. The LED should be quite bright with the 220 Ω resistor in place. Alternatively, by using a 1kΩ resistor, the LED will appear a little dimmer. Finally, when you use the 10 kΩ resistor, the LED will be barely visible. To notice the difference, you can use the red jumber as a switch by pulling the red jumper lead out of the breadboard, touching it into the hole and immediately removing it.

Now, you have the 5V going to one leg of the resistor, then the other leg of the resistor going to the positive leg of the LED and the other leg of the LED going to GND. If we reposition the resistor and place it after the LED, as shown below, the LED will still light, because it won't matter where we put the resistor, on either side of the LED, as long as it is there.

4.5 Example picture



Lesson 5: RGB LED

5.1 Overview

These RGB LEDs are a fun and easy way to add some color to your projects. Using them is easy and connecting them is pretty much the same because they are just like 3 regular LEDs in one and they mostly come in Common Anode or Common Cathode versions.

The Common Anode connects to the 5 V on the common pin and Common Cathode connects to ground. Just like with any LED, we need to limit the current being drawn, so we need to connect some resistors inline (3 total).

In the sketch that we will do, we will start in Red color state of the LED, then will fade to Green, then fade to Blue and finally back to the Red color state. Through this, we will be able to cycle through most of the colors.

5.2 Components Required

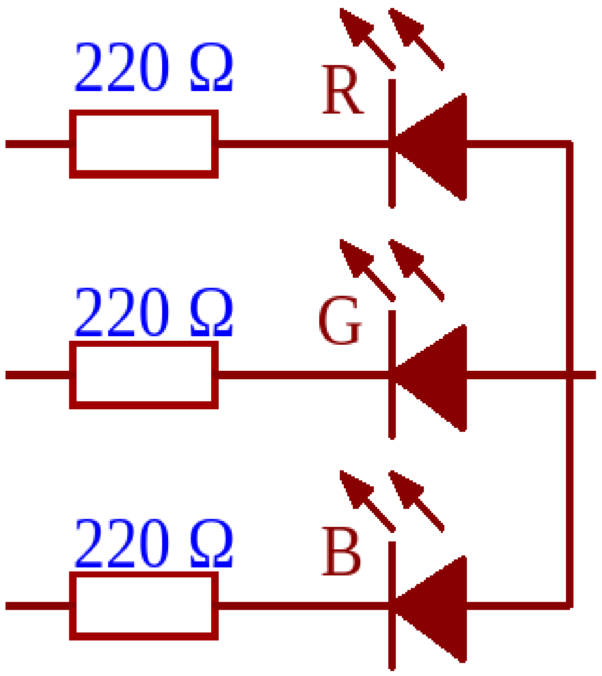
1 x Nano R3
1 x 830p Breadboard
4 x M-M wires (Male to Male jumper wires)
1 x RGB LED
3 x 220 ohm resistors

5.3 Component Introduction

RGB

If you take a look at RGB (Red, Green and Blue) LEDs, they look just like regular LEDs. But there are actually three LEDs inside the usual LED package, one of each of the primary colours (red, green, blue). You can mix any color that you want by controlling the brightness of each of the individual LEDs.

How we mix colors of the LED is by adjusting the brightness of each of the three LEDs, just like how you mix paint on a palette. Or to use different value resistors or variable resistors, which is the harder way and a lot of work. Fortunately, the analogWrite function of Nano R3 board can be used to output a variable amount of power to the appropriate LEDs.

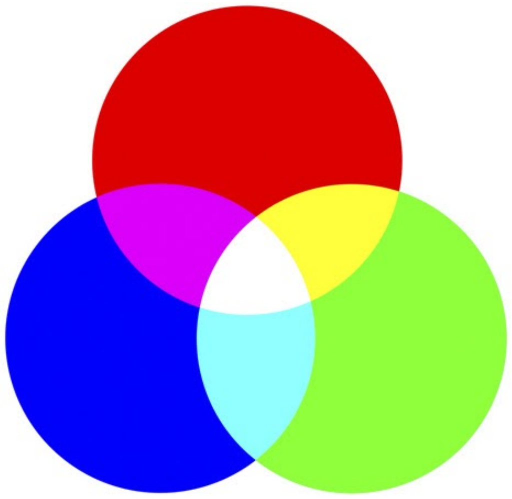


In the picture above, you can see 4 electrode LEDs with each pin for the Green, Blue or Red color called Anode. The Anode will always be connected to "+" (power), while Cathode goes to "-" (ground). The LED won't light if you connect it the other way. The second pin from the flat side is the common negative connection of the LED package. You can easily see it because it is also the longest among the four leads that will be connected to the ground. It is required that each LED inside the package will have its own 220Ω resistor in order to prevent too much current flowing through it. These resistors are used when you connect the three positive leads of the LEDs (one red, one green and one blue) to Nano output pins.

COLOR

By varying the quantities of red, green and blue light, you can mix any color that you like. Your eye has three types of light receptors (red, green and blue) and together with your brain, they can process the amounts of red, green and blue and convert it into a color of the spectrum.

In a way, we are playing a trick on the eye by using the three LEDs. This is also the mechanism used in TVs, where the red, green and blue color dots (in LCD) next to each other make a pixel.



For example, the overall color of the light will be white when we set the brightness of all three LEDs to be the same. By turning off the blue LED, the light will be yellow since there are just the red and green LEDs with the same brightness.

We can mix any color we like just by controlling the brightness of each of the red, green and blue parts of the LED separately.

Black is the absence of light, thus we can have black when we turn off all three colors of LED.

5.4 Common Anode vs Common Cathode

These two types of RGB LEDs do not connect in the same way even though they look the same. The Nano R3 can control both types of LEDs due to its symmetrical ability to source and sink exactly the same amount of current.

Common Cathode is straightforward, meaning, having a **higher** current means a brighter corresponding LED. Here, the current is flowing from the board to the LED and is called **Current Sourcing.**

Common Anode is a bit different, which means, if the current is **lower**, it will result in a brighter corresponding LED. Here, the current is flowing from the LED to the board. and is called **Current Sinking**.

5.5 How to determine if the LED is common anode or common cathode

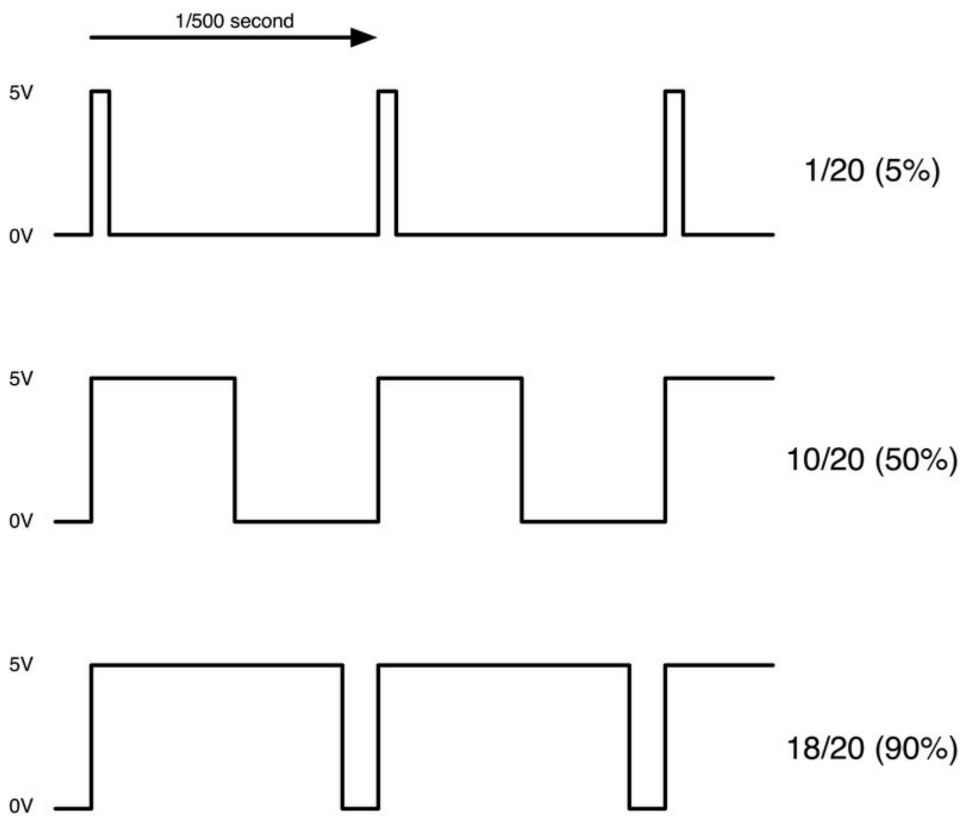
There is no visible difference between the common anode and common cathode, thus, to determine which is which is by testing it.

You may follow these steps to identify if it is a common anode or common cathode:

● Power up the development board.
● Have the longest leg of the RGB LED connected to **GND.**
● Using a 220 Ω resistor in series, connect **QUICKLY (a fraction of a second)** one of the legs remaining to the **5 V**. It is better to mount all of them on a breadboard.
● It is a **COMMON CATHODE** if the LED lights**,** if not, it is **COMMON ANODE.**

5.6 Theory (PWM)

The technique for controlling power is called Pulse Width Modulation (PWM). Here, we also use it to control the brightness of each of the LEDs. Shown in the diagram below the signal from one of the PWM pins on the Nano.



The PWM output will produce a pulse at roughly every 1/500 of a second. The 'analogWrite' function controls the length of this pulse, thus, specifying a value between 0 and 255 won't produce any pulse at all at 'analogWrite(0)' and a pulse that lasts all the way until the next pulse is due will be at 'analogWrite(255)'.

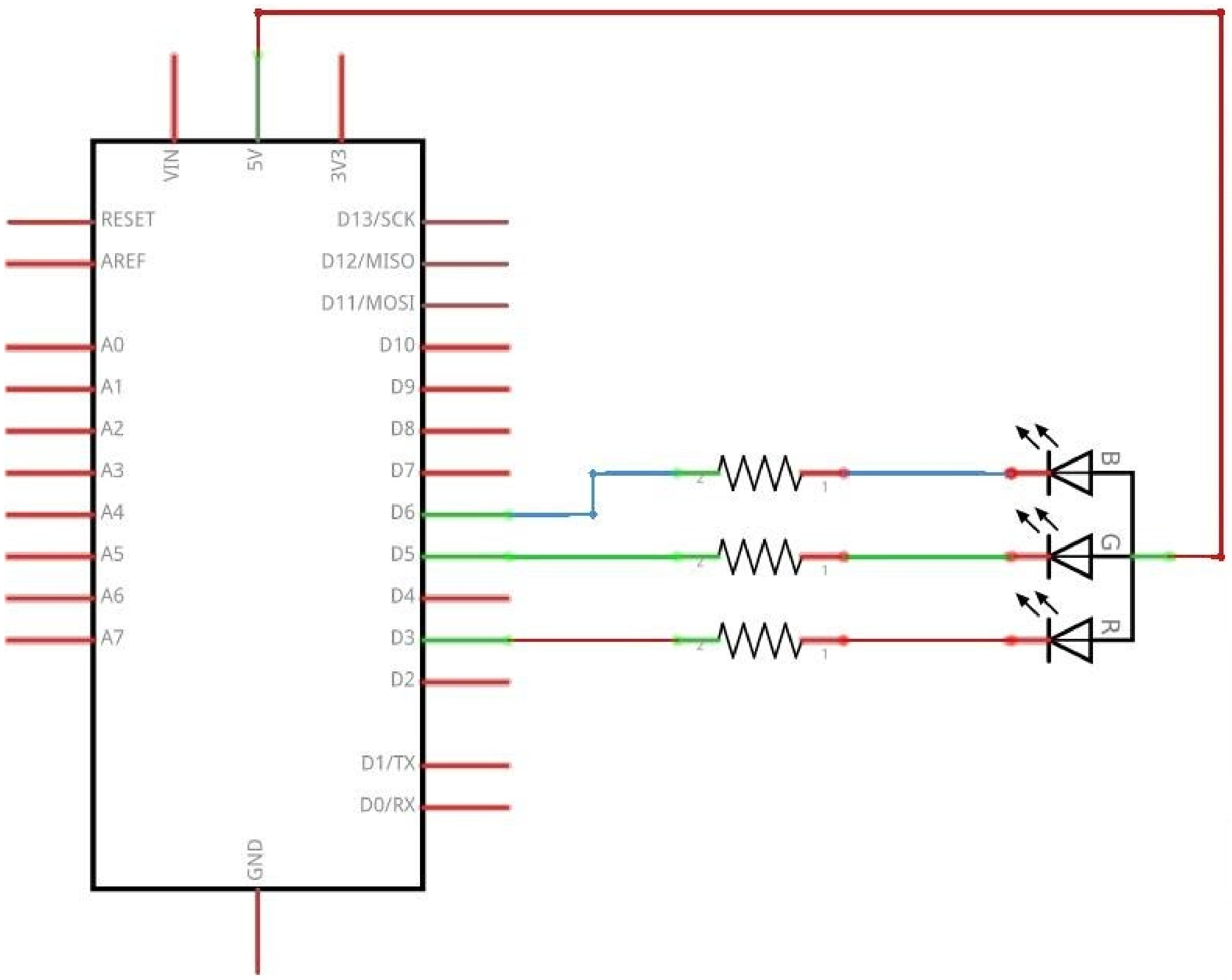
If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power. However, in the case that the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs rapidly turning on and off, so to the human eye, it just looks like the brightness is changing.

5.7 Connection

Schematic

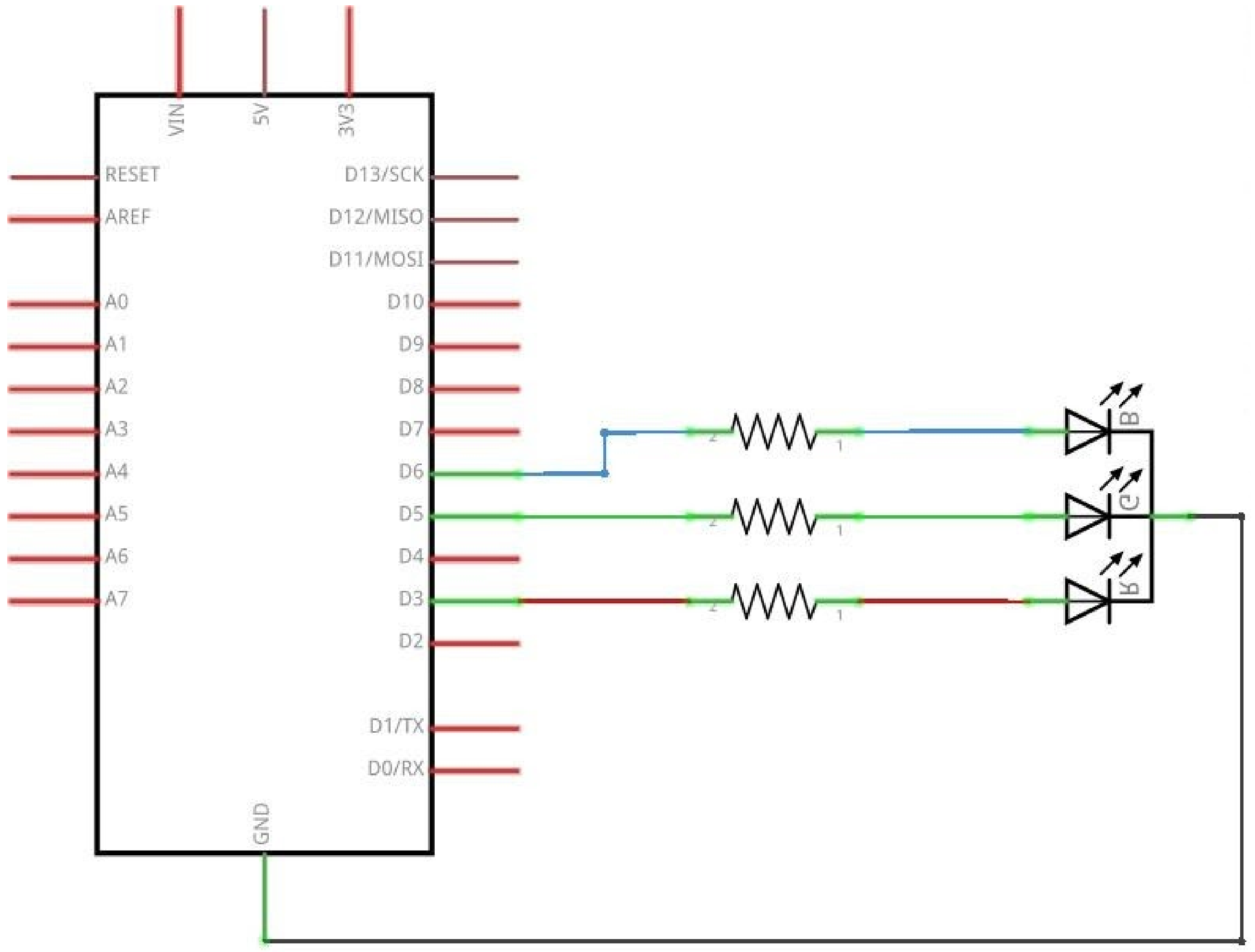
Common Anode

You can see from the diagram that the longest leg directly connects to the **5 V** and the other pins connects (in series with a 220Ω resistor each) to 3 digital pins (capable of PWM).



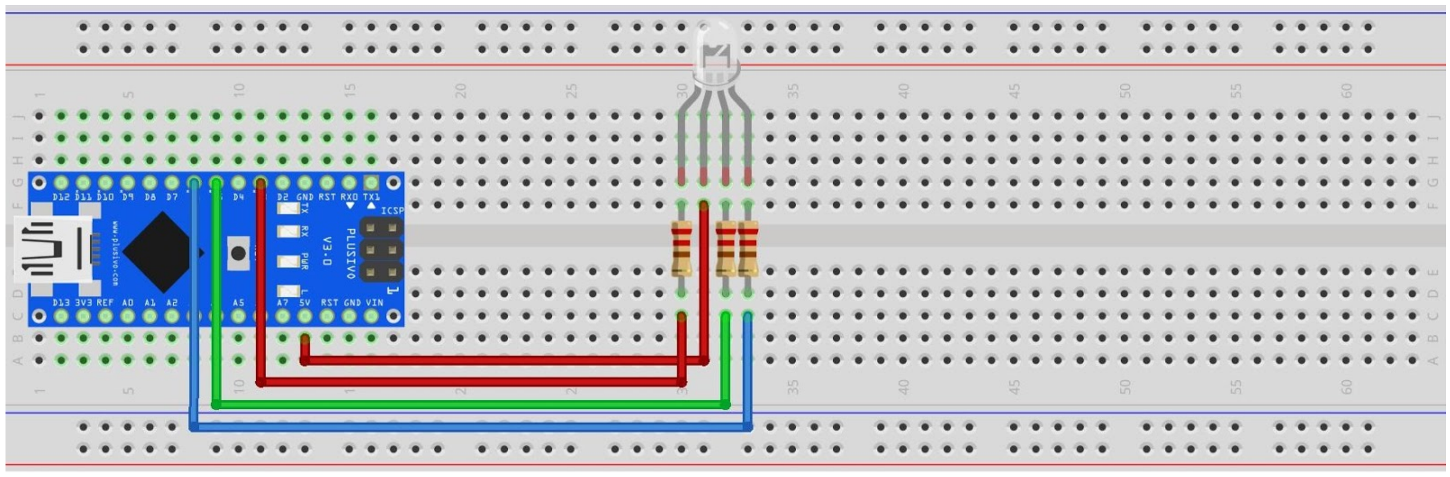
Common Cathode

You can see from the diagram that the longest leg directly connects to the **GND** and the other pins connects (in series with a 220Ω resistor each) to 3 digital pins (capable of PWM).

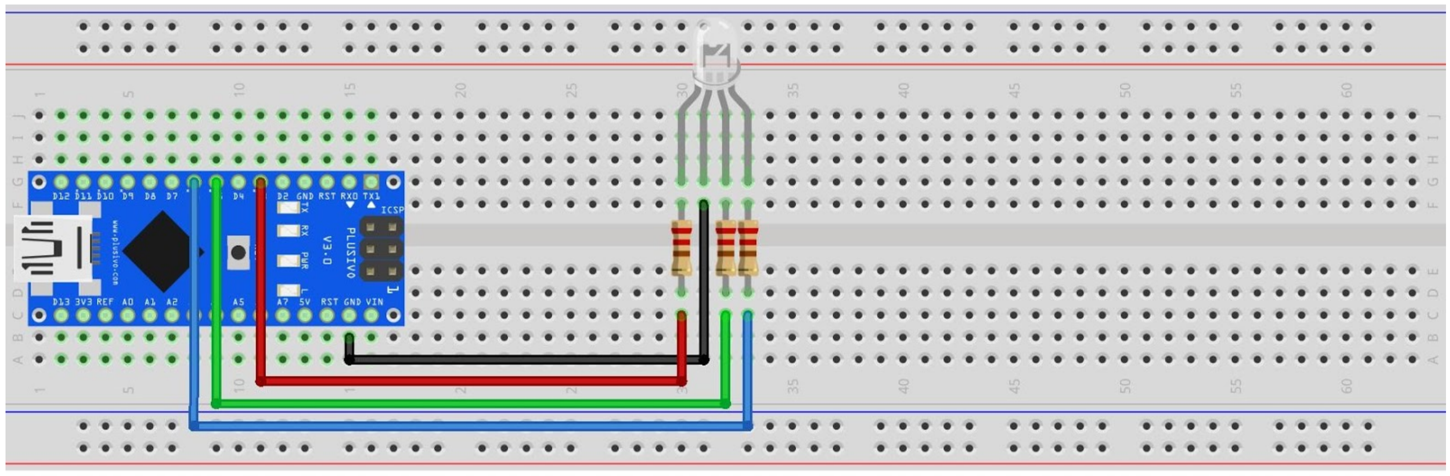


Wiring diagram

Common Anode



Common Cathode



5.8 Code

After wiring, find and open the program located in the folder - Lesson 5 RGB LED, and UPLOAD to upload the code. If there are any errors, see Lesson 3 for details about program uploading

FOR loops will be used in our code to be able to cycle through the colors. The first will go from RED to GREEN, then GREEN to BLUE and the third is from BLUE to RED.

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

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

Then we will write the 'setup' function. This function only runs once after the Arduino has reset as shown in the previous lesson. Here, the function just need to 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);
}
```

Let’s have a look at the last function in the sketch and define variables.

```
redValue = 255; // choose a value between 1 and 255 to change the color.
greenValue = 0;
blueValue = 0;
```

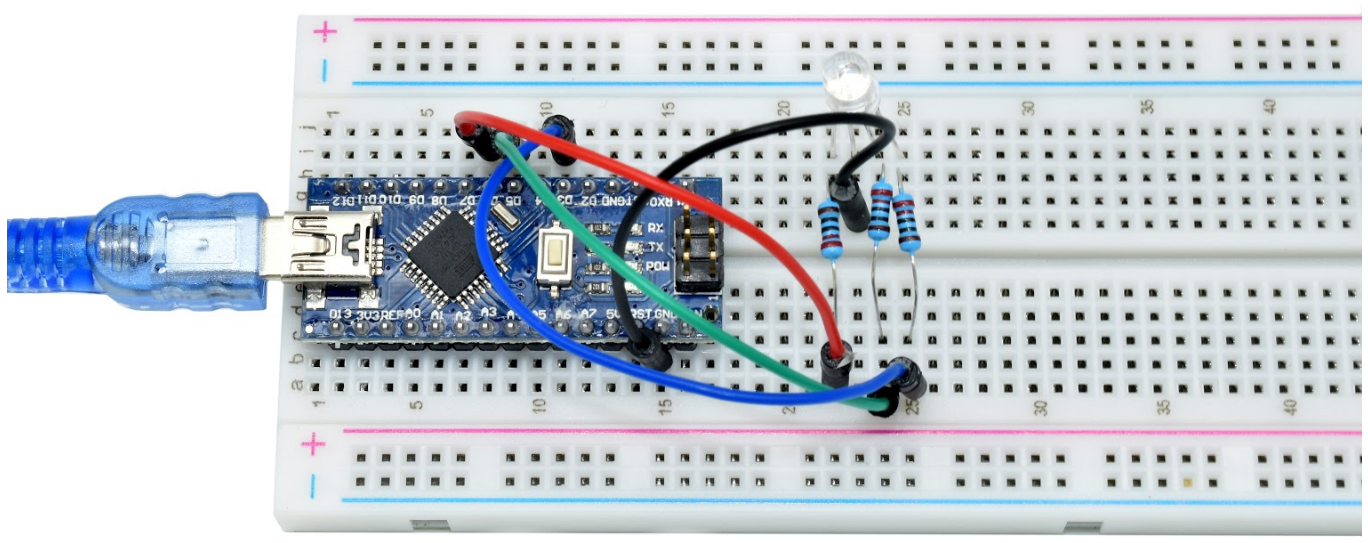
There are three arguments in this function, one argument is for the brightness of each of the red, green and blue LEDs, with values between 0 (means off) and 255 (maximum brightness). The 'analogWrite' argument of the function will then be used to set the brightness of each LED.

Looking at the 'loop' function, it can be seen that we are specifying the amount of red, green and blue light that we want to produce, then pause for a second before moving on to the next color.

```
#define delayTime 10 // fading time between colors Delay(delayTime);
```

Now, try to add few colors of your own to the sketch and observe the effect on your LED.

5.9 Example picture



Lesson 6: Digital Inputs

6.1 Overview

In this lesson, it will be discussed how to use push buttons containing digital inputs to turn the LED on and off. Pressing the button will turn the LED on and off.

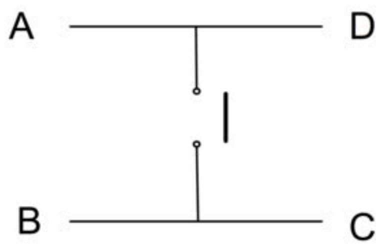
6.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x 5mm red LED
1 x 220 ohm resistor
2 x push Button
7 x M-M wires (Male to Male jumper wires)

6.3 Component Introduction

PUSH Button

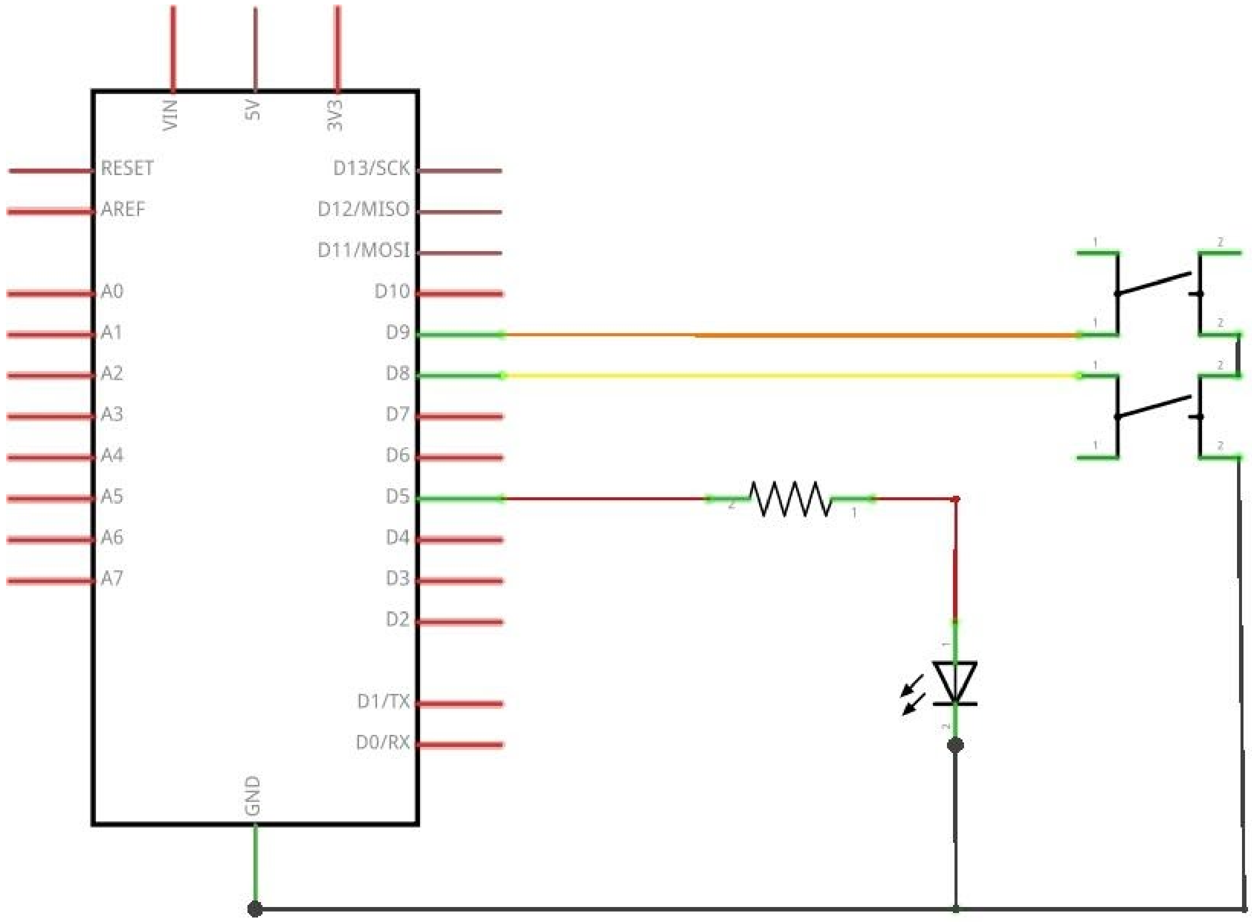
Push buttons are simple components that works in the following manner: when a button is pressed or a lever is flipped, it causes two contacts to connected together resulting to flow of electricity through them. The push buttons in this lesson have four connections.



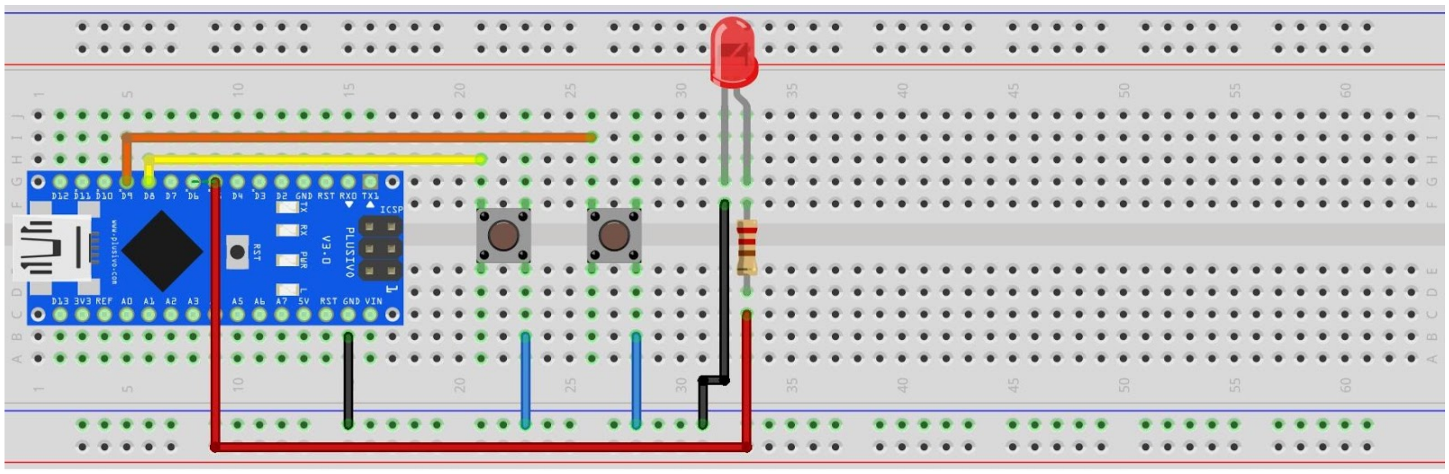
There are only two electrical connections inside the switch package: pins B – C and A - D that are connected together.

6.4 Connection

Schematic



Wiring diagram



The pins of switches protrude from opposite sides of the square bodies of the component. Thus the distance between the pins will only be enough when placed correctly on the breadboard. Keep in mind that the LED has to have the shorter negative lead to the left.

6.5 Code

After wiring, find and open the program located in the folder - Lesson 6 Digital Inputs, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

Load the sketch. The code tells us the LED will turn ON when left button is pressed, and it will turn OFF when the right button is pressed.

In the sketch, three variables are defined for the three pins that will be used. The output is the 'ledPin', the 'buttonApin' is the switch closer to the top of the breadboard and other switch is 'buttonBpin'.

The ledPin is the OUTPUT defined by the 'setup' function, but we have the two inputs to set up. In this case, we use the pinMode as 'INPUT_PULLUP' as below:

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

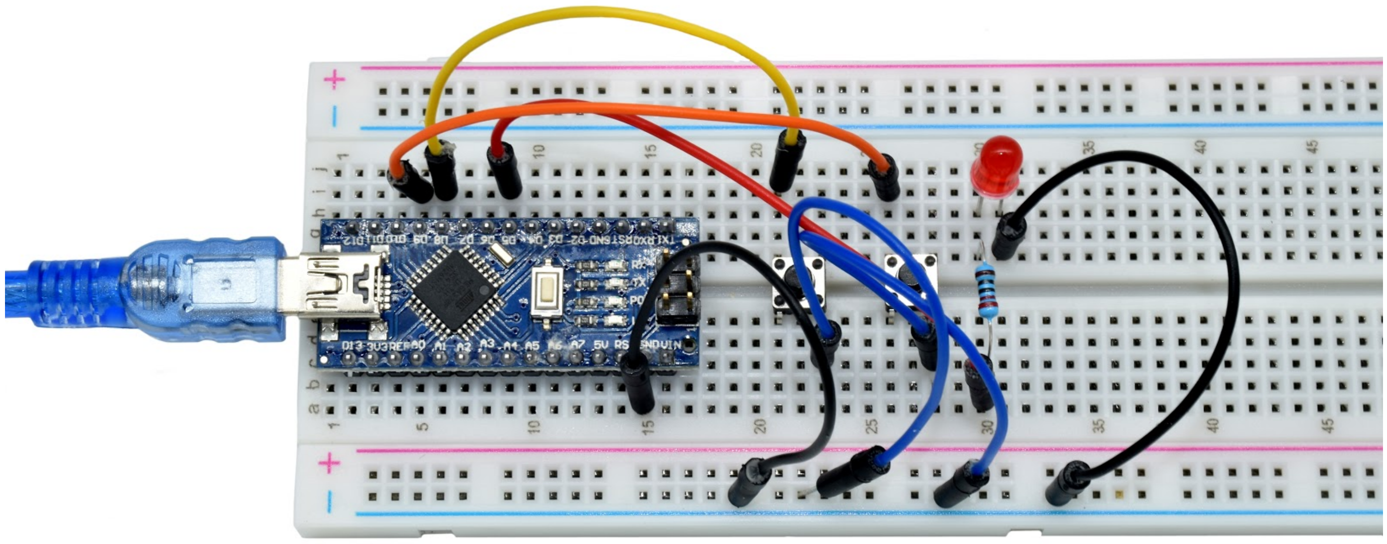
The INPUT_PULLUP pin mode tells us that the pin is used as an input, and it should then be 'pulled up' to HIGH if nothing else is connected. That basically means that the default value is HIGH for the input, unless it is pulled LOW by a press of the button. Thus, switches are usually connected to GND. Pressing a switch will connect the input pin to GND, and it will no longer be HIGH.

Since default value is HIGH unless it is pulled LOW by a press of the button, the logic is a bit backwards. This is resolved in the 'loop' function.

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

We have two 'if' statements in the 'loop' function, one for each button. Each of the statements does a 'digitalRead', verifying the appropriate input. Keep in mind that pressing a button will result in LOW input. Thus, if we have button A that is low, the 'digitalWrite' on the ledPin will turn it on. Likewise, a LOW is written to the ledPin if button B is pressed.

6.6 Example picture



Lesson 7: Passive Buzzer

7.1 Overview

This lesson will teach you the basics about using a passive buzzer. The goal of this chapter is to produce eight different sounds of 0.5 seconds each: from Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz).

7.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x Passive buzzer
2 x M-M wires (Male to Male jumper wires)

7.3 Component Introduction

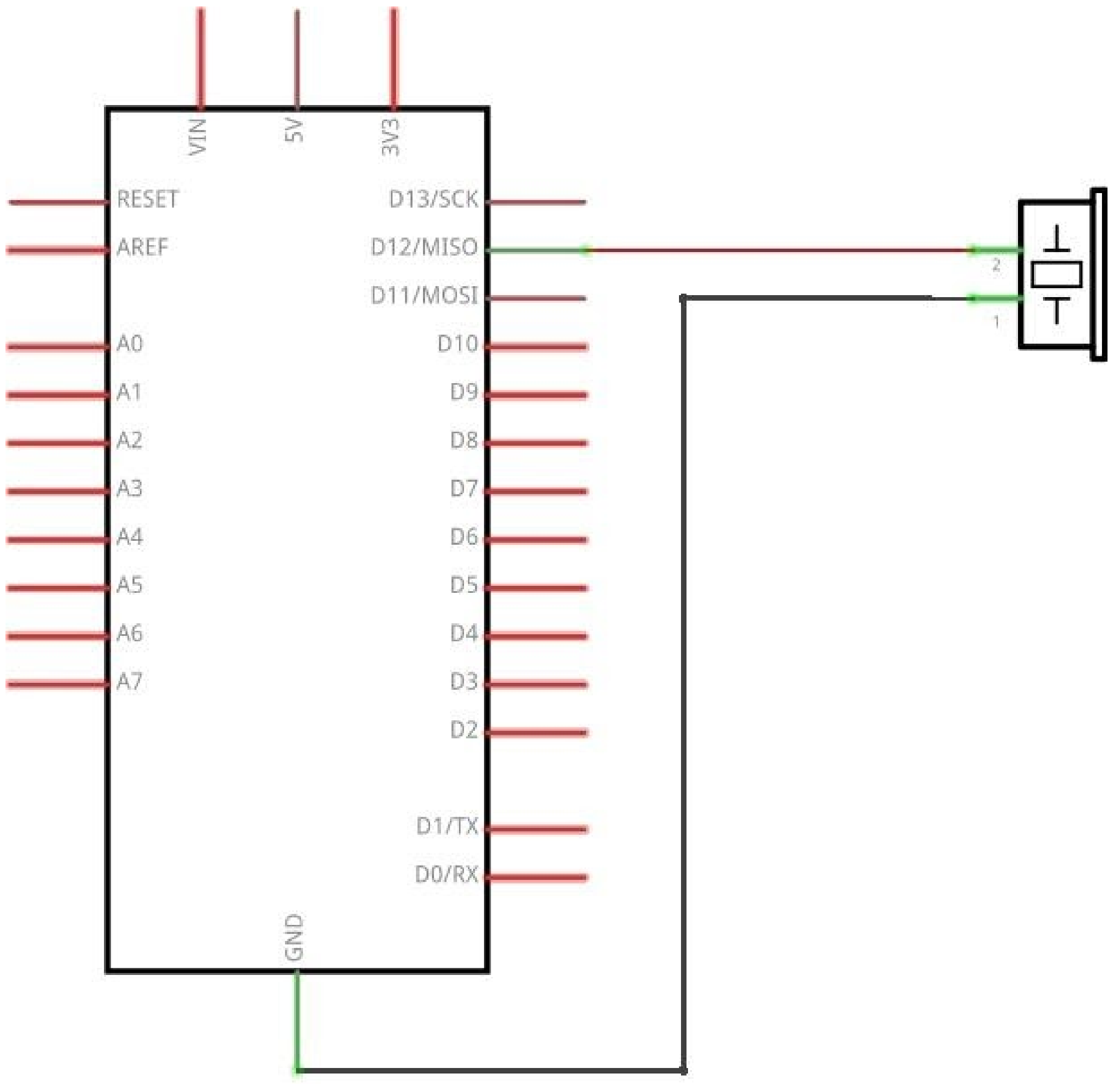
Passive Buzzer

How a passive buzzer works is by using PWM generating audio to make the air vibrate and generate different sounds. For instance, sending a pulse of 523Hz generates Alto Do, a pulse of 587Hz generates midrange Re, and so on. Thus, you can even play a song using the buzzer.

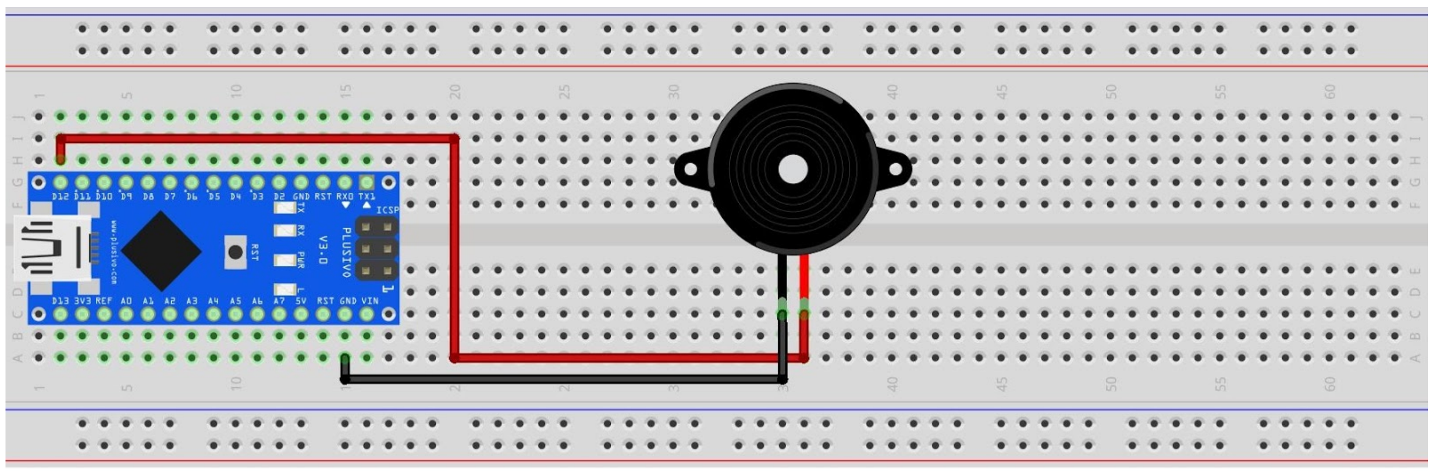
This time we do not use the Arduino board analog Write () function to generate a pulse to the buzzer, as the function will always output 500Hz.

7.4 Connection

Schematic



Wiring diagram

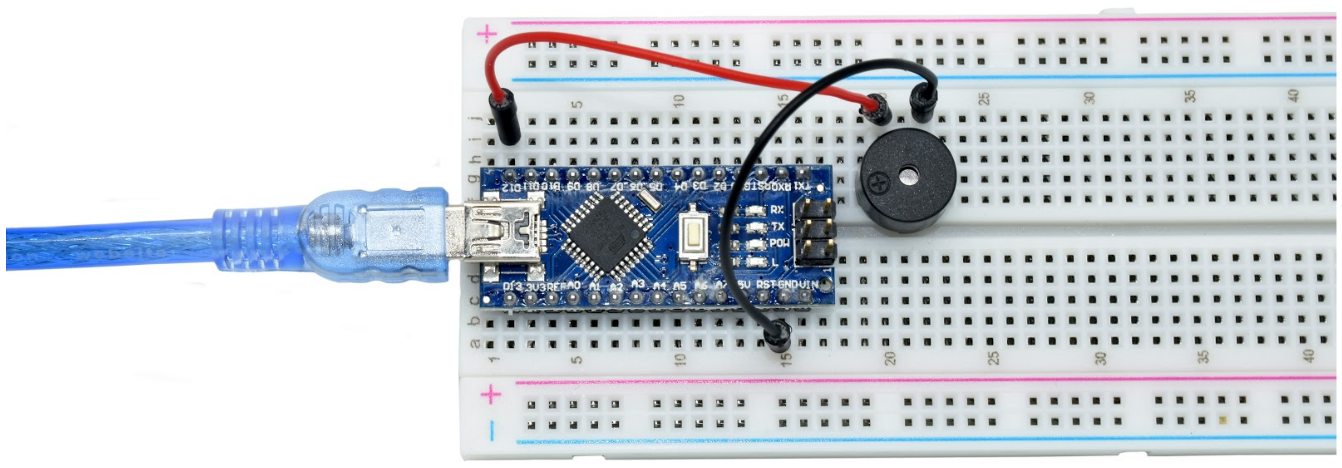


In wiring the buzzer connected to the Arduino board, we put the red wire (+) going to the pin 12 and the black wire (-) going to the GND.

7.5 Code

After wiring, find and open the program located in the folder – Lesson 7 Passive Buzzer, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

7.6 Example picture



Lesson 8: Tilt Ball Switch

8.1 Overview

This lesson will teach you the basics on using a tilt ball switch to identify a small inclination angle.

8.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x Tilt Ball switch
2 x M-M wires (Male to Male jumper wires)

8.3 Component Introduction

Tilt sensor

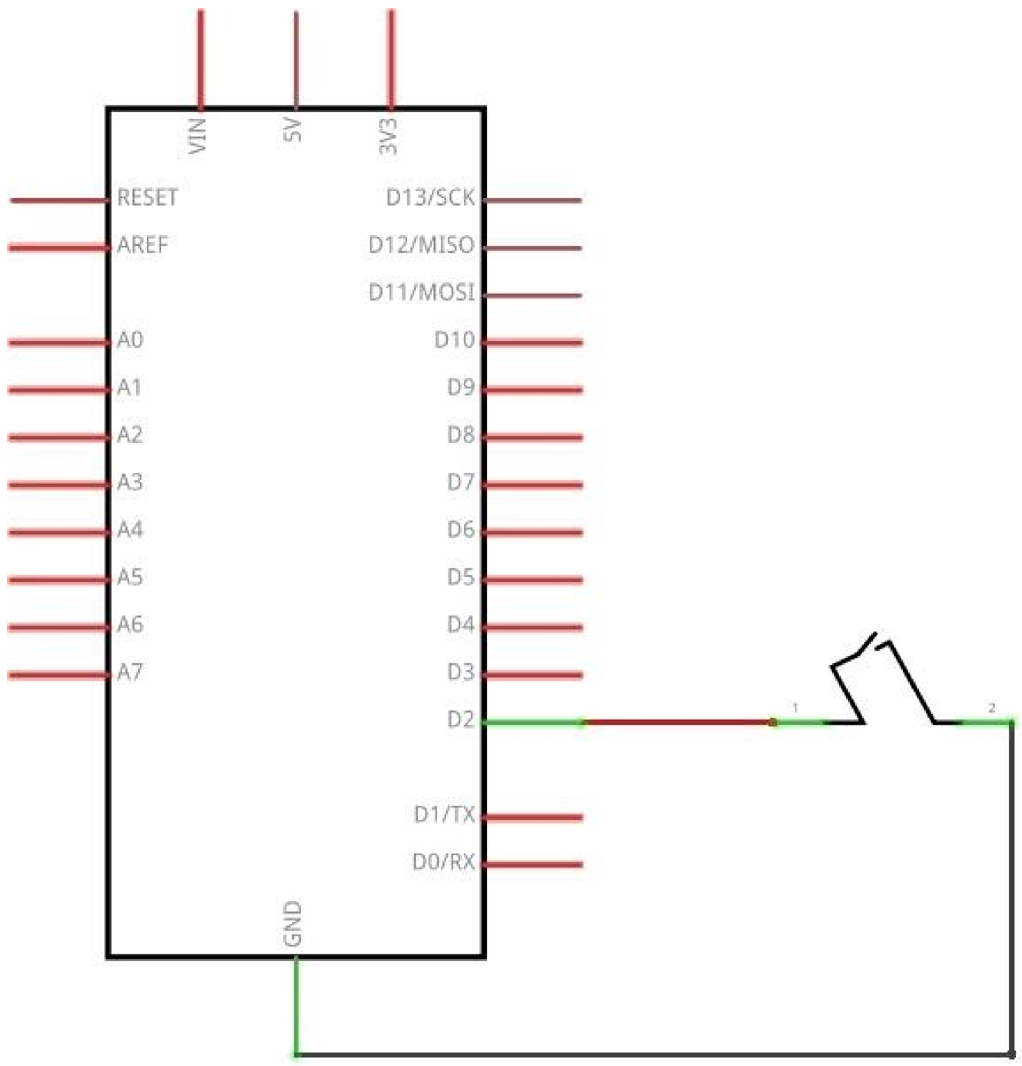
Tilt sensors are used to detect inclination or orientation. They are reliable, low-power, long-lasting and very inexpensive. Because they are so simple, they are quite popular for appliances, gadgets and toys. They are also known as "mercury switches", "tilt switches" or "rolling ball sensors".

They usually consist of a cavity (most often of cylindrical shape) with a conductive free mass inside, like a blob of mercury or rolling ball. When the end of the cavity, which contains two poles, is pointed downwards, the sensor can act as a switch throw as the mass rolls onto the poles and shorts them.

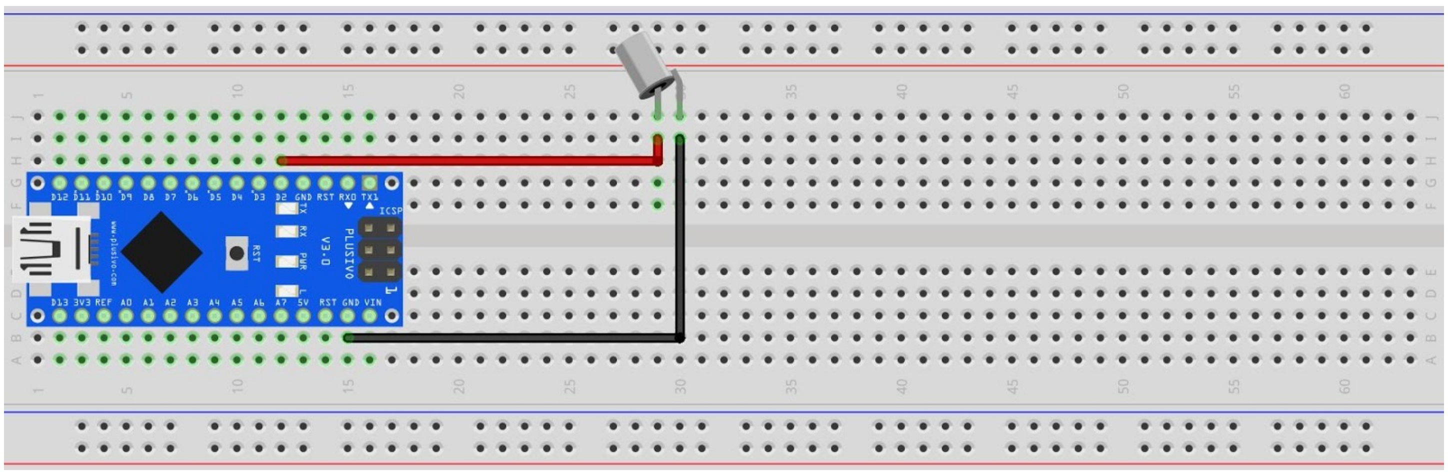
Tilt sensors can detect motion or orientation, but they are not as accurate or flexible as a full accelerometer. The big ones have the capacity of switching power independently, whereas accelerometers require extra circuit components for evaluating output digital or analog voltage.

8.4 Connection

Schematic



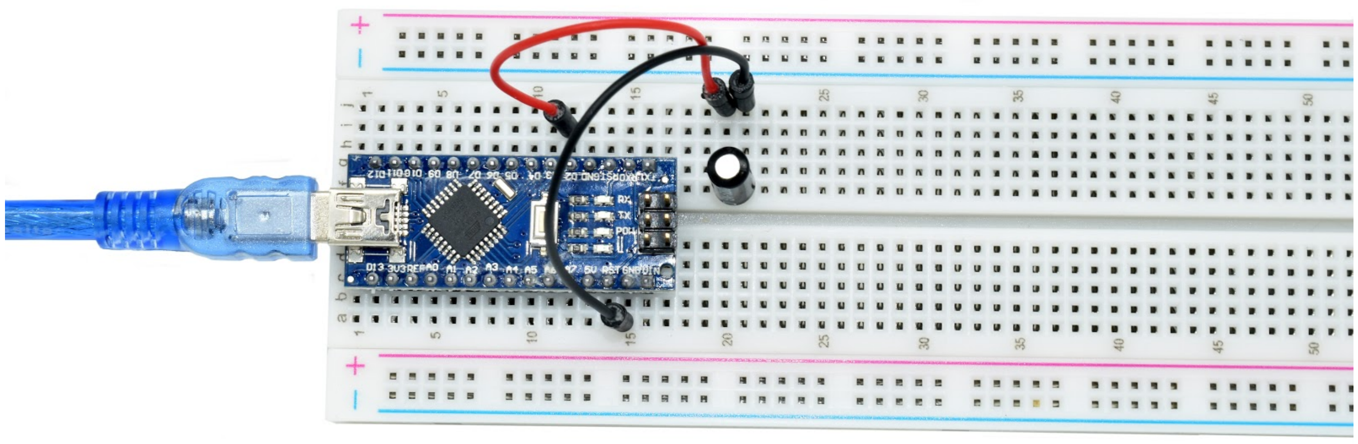
Wiring diagram



8.5 Code

After wiring, find and open the program located in the folder – Lesson 8 Ball Switch, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

8.6 Example picture



Lesson 9: Ultrasonic Sensor Module

9.1 Overview

The Ultrasonic sensor is a great component for all kinds of projects that need distance measurements such as avoiding obstacles.

In this lesson, a Library designed for HC-SR04 sensor, which is a cheap and easy to use sensor, will be used.

9.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x Ultrasonic sensor module
4 x M-M wires (Male to Male jumper wires)

9.3 Component Introduction

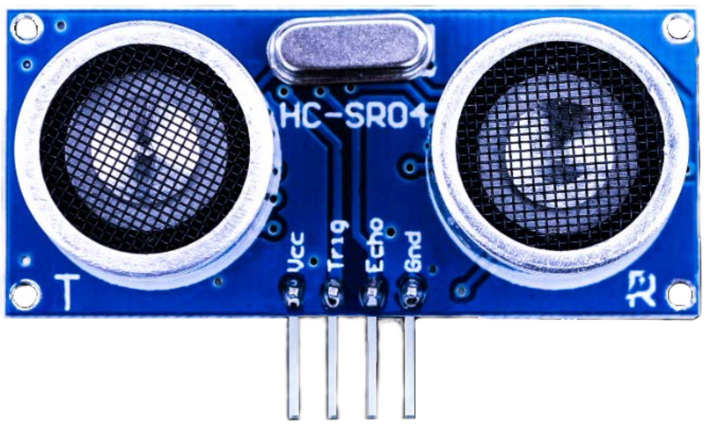
Ultrasonic sensor

The ultrasonic sensor module HC-SR04 has a capacity of 2cm to 400cm indirect measurement, with a ranging precision of up to 3mm. Ultrasonic transmitters, receivers and control circuits are included in the module. The fundamental working formula is:

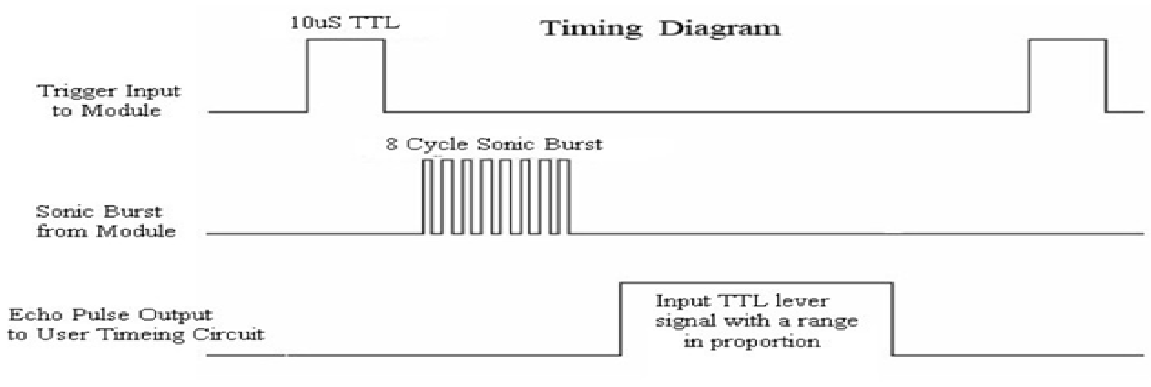
Test distance = (high level time × velocity of sound (340m/s))/2

And the basic principle of work are the following:

(1) Using IO trigger for at least 10ms high level signal
(2) The module automatically sends eight 40 kHz and detect whether there is a pulse signal back
(3) If the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

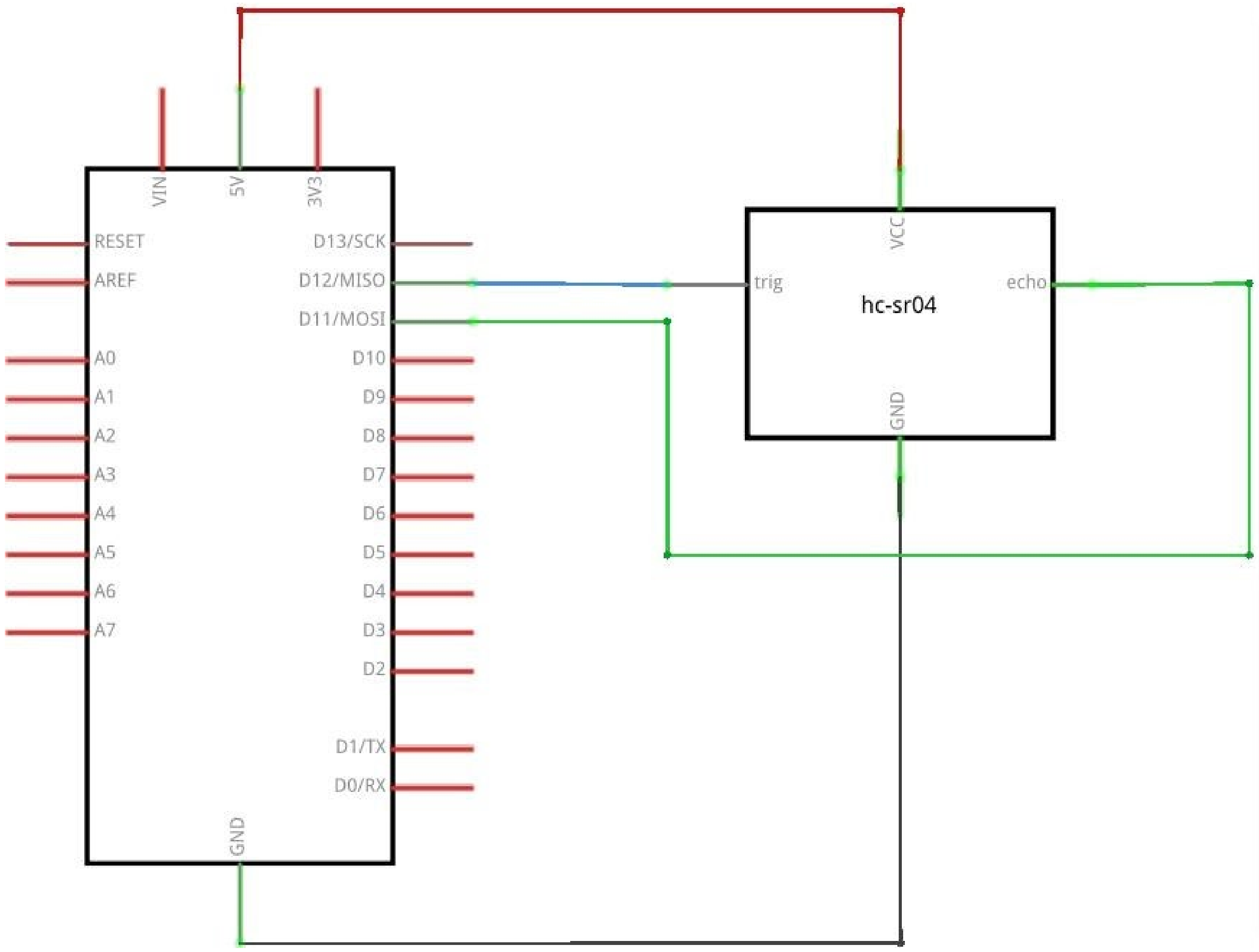


You can observe from the Timing Diagram below. As you can see, only a short 10ms pulse is necessary to trigger the device to start the ranging. Afterwards, the module will send out 8-cycle bursts of ultrasound at 40 kHz and raise its echo, thus the range through the time interval between sending trigger signal and receiving echo signal can be measured. The formula is: ms/58 in centimeters or ms/148 in inches or the range = high level time * velocity (340M/S) / 2. We recommend using measurement cycles over 60ms.

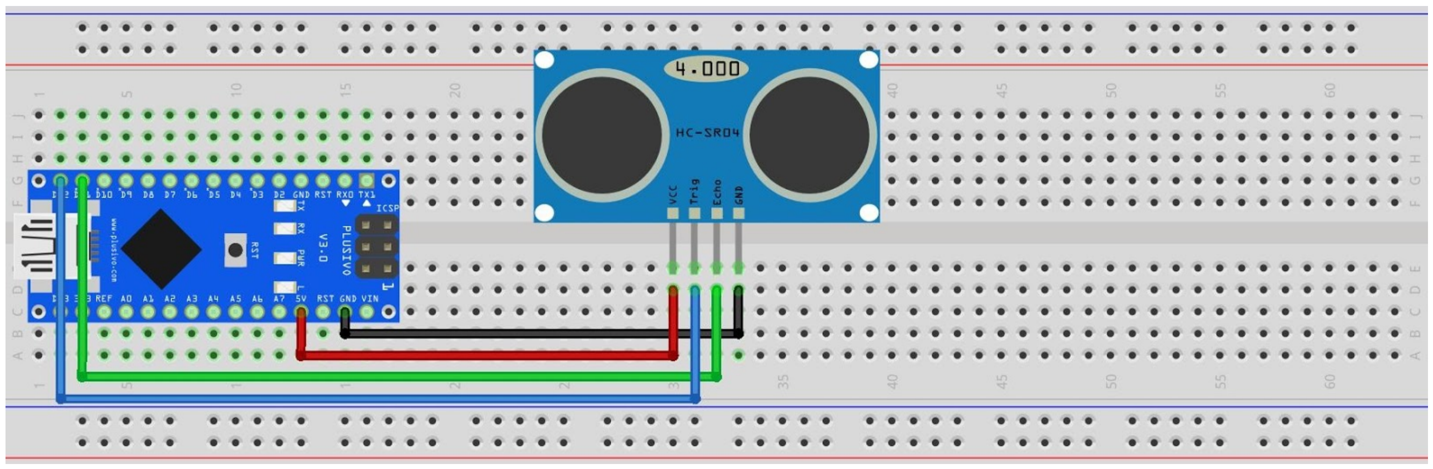

10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

9.4 Connection

Schematic



Wiring diagram

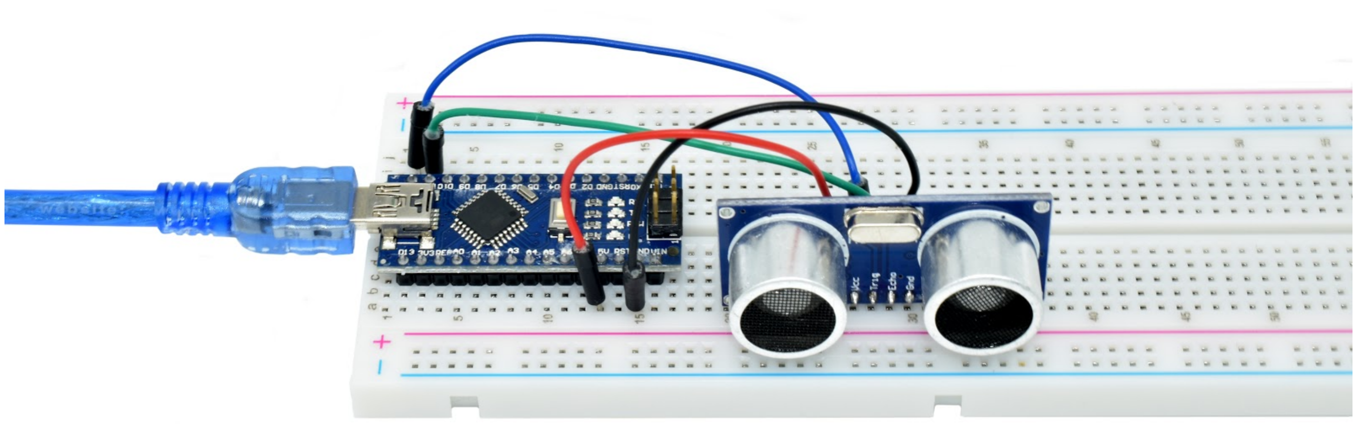


9.5 Code

Our code will be short and simple if we use a Library designed for these sensors. The Library will be included at the start of the code and by using simple commands we can control the behavior of the sensor.

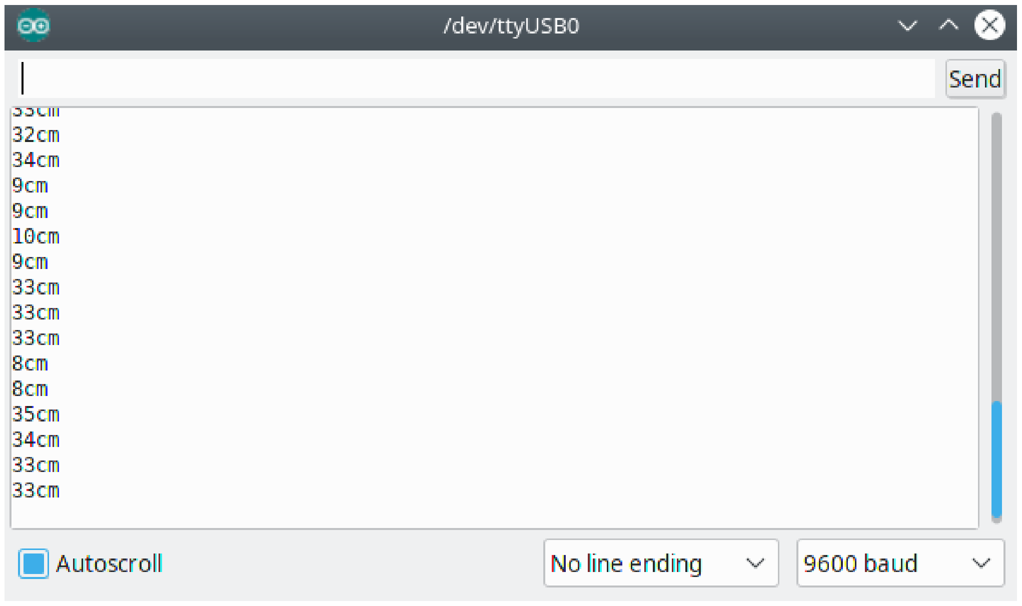
After wiring, find and open the program located in the folder – Lesson 11 Ultrasonic Sensor Module,, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading. To be able to run this, have the <HC-SR04> library installed or re-install it, if necessary, for the code to work. For details about loading the library file, see Lesson 2.

9.6 Example picture



Open the monitor so we can see the following values:

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 2.



```
33cm
32cm
34cm
9cm
9cm
10cm
9cm
33cm
33cm
33cm
8cm
8cm
35cm
34cm
33cm
33cm
```

Lesson 10: IR Receiver Module

10.1 Overview

Wireless control on your projects can be easily achieved using an Infrared (IR) Remote, as they are simple to understand and use. In this lesson we will be learning to program the IR receiver using a Library designed specifically for it.

In our code, we will have access to all the IR Hexadecimal codes available on this remote, we will check if the code was identified and whether we are pressing a key.

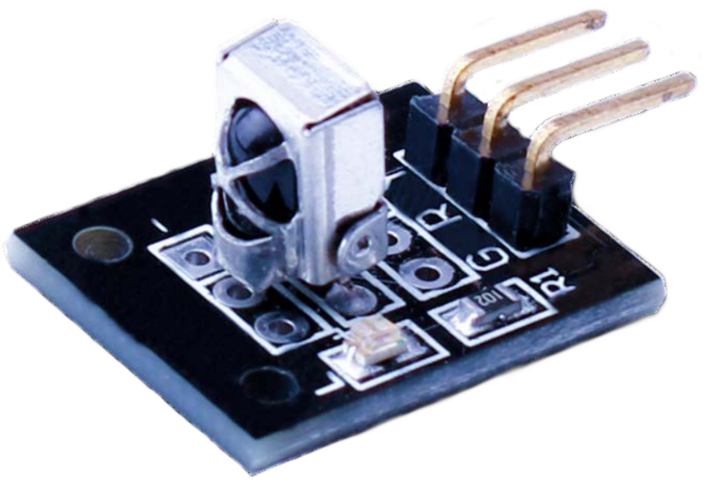
10.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x IR receiver module
1 x IR remote
3 x M-M wires (Male to Male jumper wires)

10.3 Component Introduction

IR RECEIVER SENSOR

IR detectors are essentially small microchips that contains a photocell. This photocell's function is to detect infrared light. These detectors are commonly used for remote control detection, especially in TVs and DVD players. TV commands are given by the infrared light inside the remote control: to power on, change the volume, etc. This light can not be seen by the human eye, which makes it harder for us to test a setup.



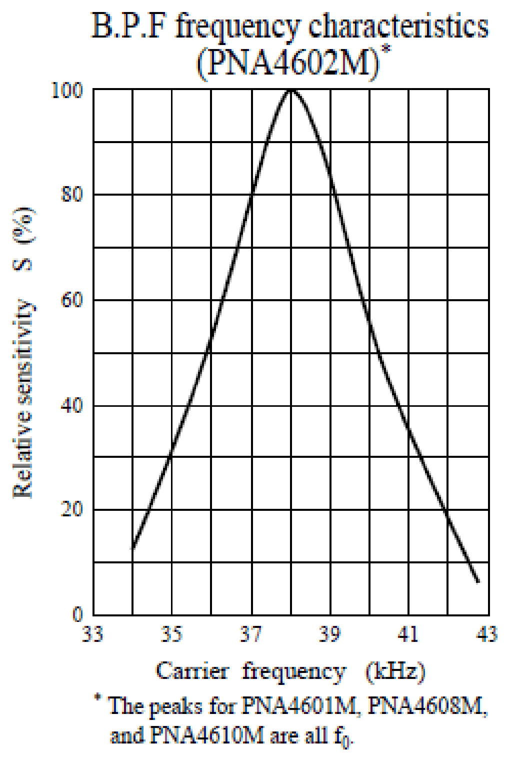
We can spot some differences between these and photocells:

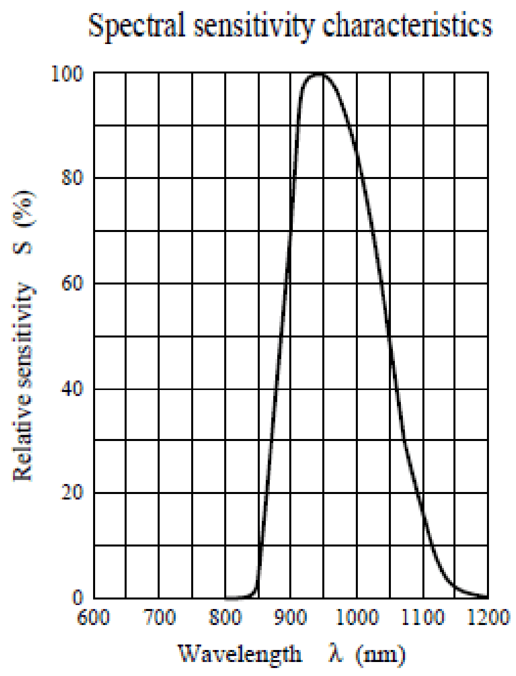
IR detectors are not suitable for visible light, as they are especially created for IR light. On the other hand, photocells are the opposite: good at detecting yellow/green light, but not for IR light.

Photocells do not have demodulators and they can distinguish any frequency within the speed of the photocell's response (which is about 1KHz). On the contrary, IR detectors have a demodulator inside that searches for modulated IR at 38 KHz.

IR detectors are straightforward - they either detect a 38KHz IR signal and output low (0V) or they do not find any and output high (5V). Photocells behave similar to resistors, the value changing according to the intensity of light that they are exposed to.

What You Can Measure

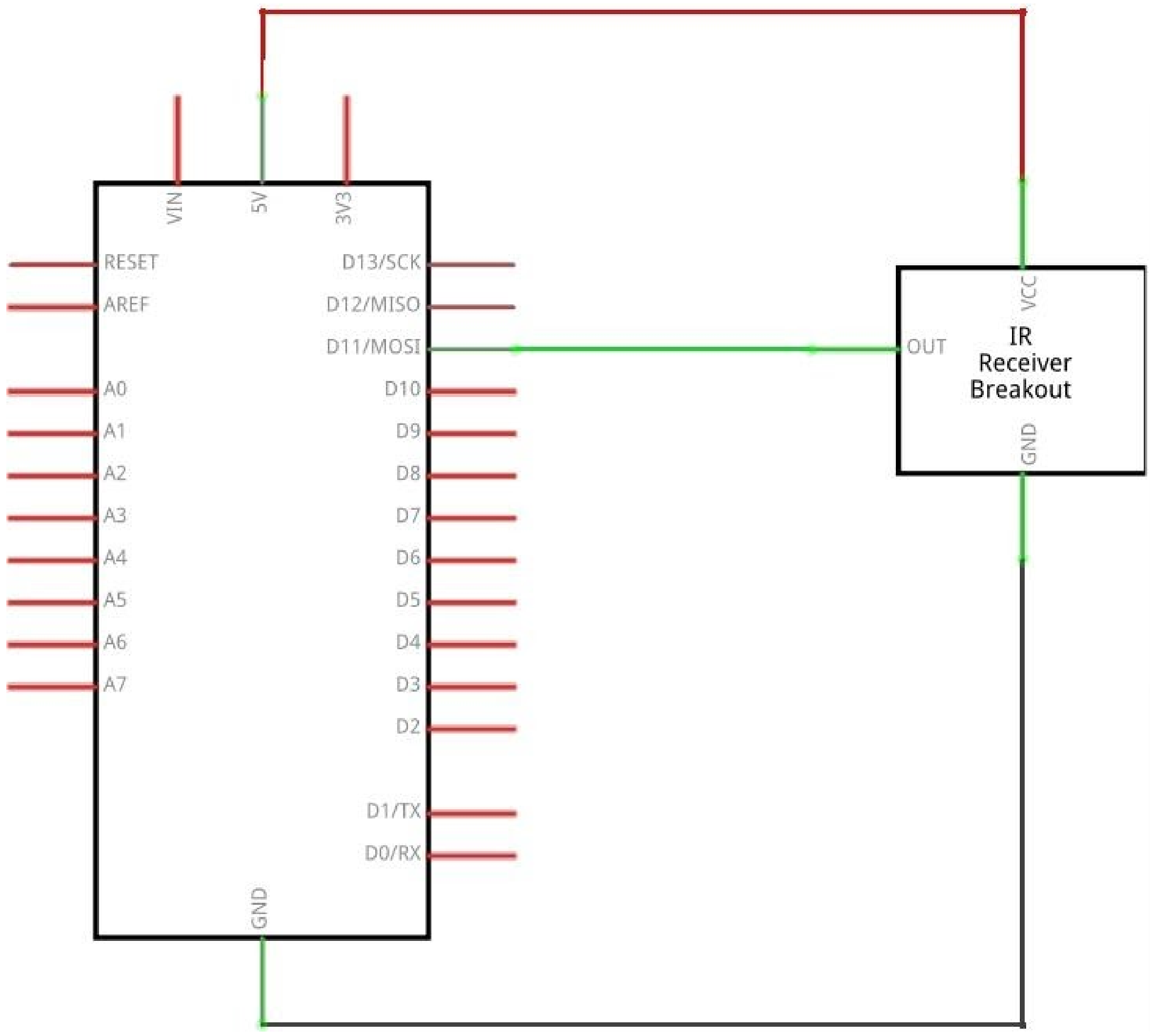




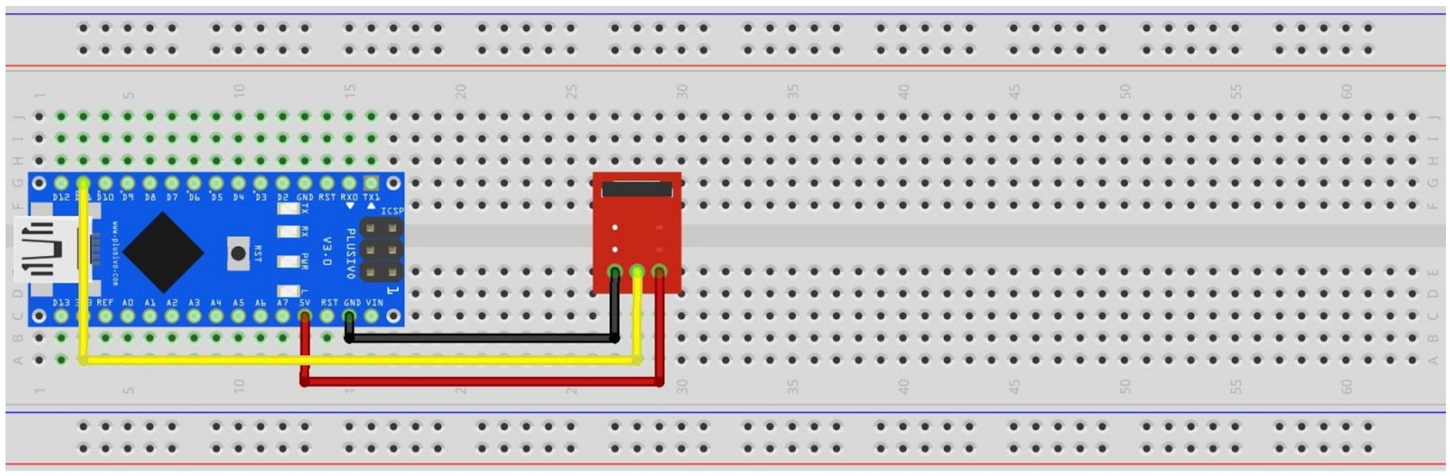
These graphs show us that the peak frequency detection is at 38
KHz and the peak LED color is 940 nm. You can use a frequency of about 35 to 41 KHz, but it won't identify as accurately from a bigger distance. Remember to look at the datasheet for your IR LED to check the wavelength and get the correct LED, preferably a 940nm as this is not a visible light.

10.4 Connection

Schematic



Wiring diagram



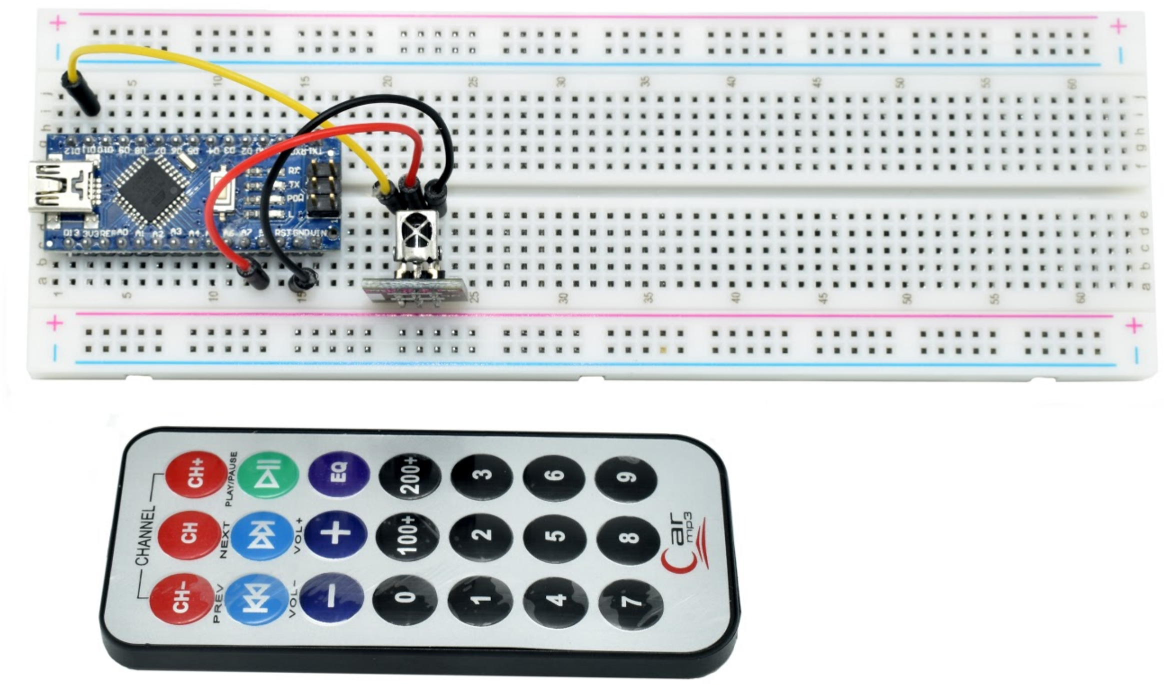
There are 3 connections to the IR Receiver: Signal, Voltage and Ground. The “-” is the Ground, “S” is signal, and middle pin is Voltage 5V.

10.5 Code

After wiring, find and open the program located in the folder – Lesson 10 IR Receiver Module, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading. To be able to run this, have the < IRremote > library installed or re-install it, if necessary, for the code to work. For details about loading the library file, see Lesson 2.

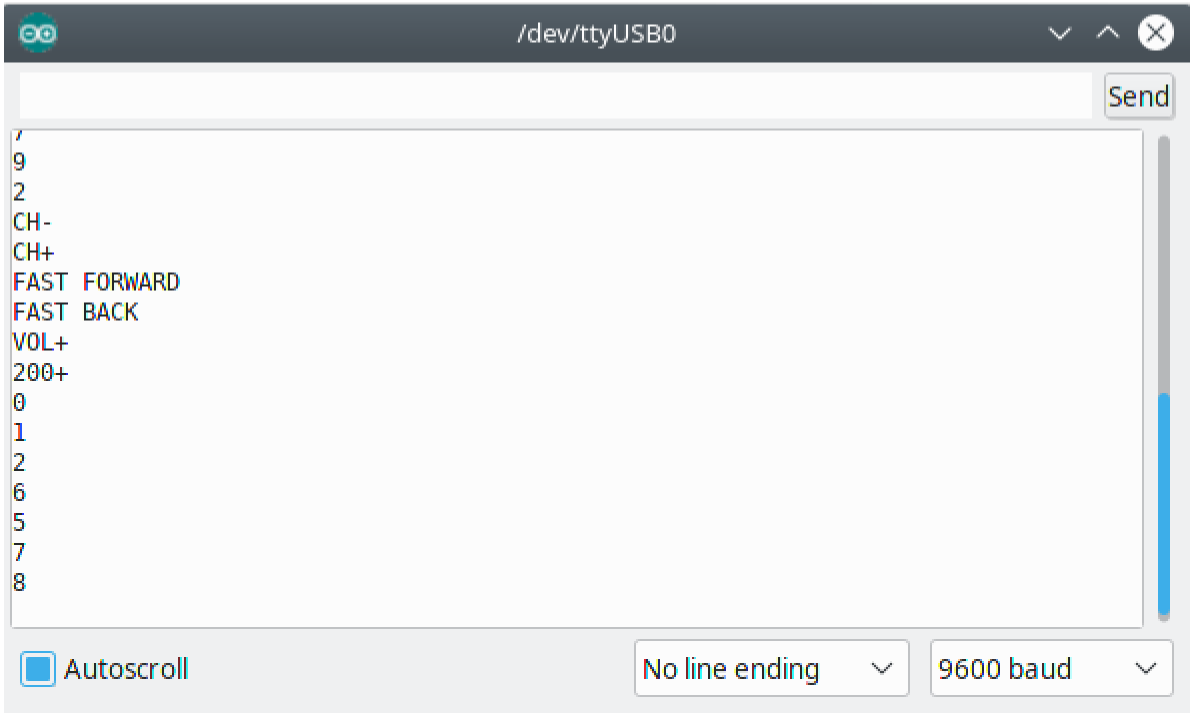
Now, we will move the <RobotIRremote> out of the Library folder, to avoid problems the one we are going to be utilizing. Just put it back after you are done with this lesson. After installing the library, restart your IDE.

10.6 Example picture



Open the monitor, so you can see the following figure:

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 2.



```
9
2
CH-
CH+
FAST FORWARD
FAST BACK
VOL+
200+
0
1
2
6
5
7
8
```

Lesson 11: Controlling LEDs with the remote

11.1 Overview

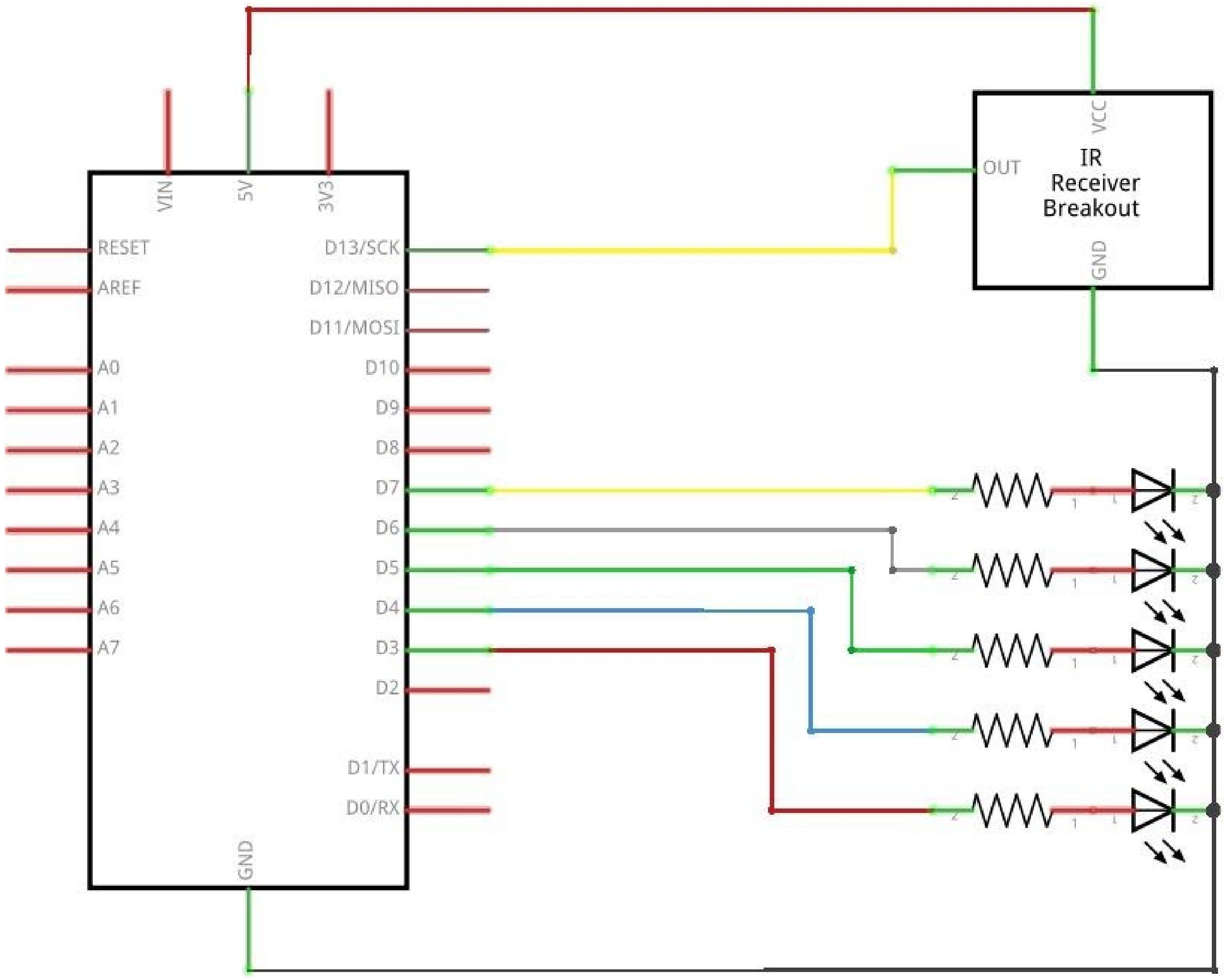
In the previous lesson, we used the IR remote to see which button is pressed on the serial monitor, in this lesson we will learn how to use it to control 5 LEDs. we will use the numbers on the remote from 0 to 9 to control them.

11.2 Components Required

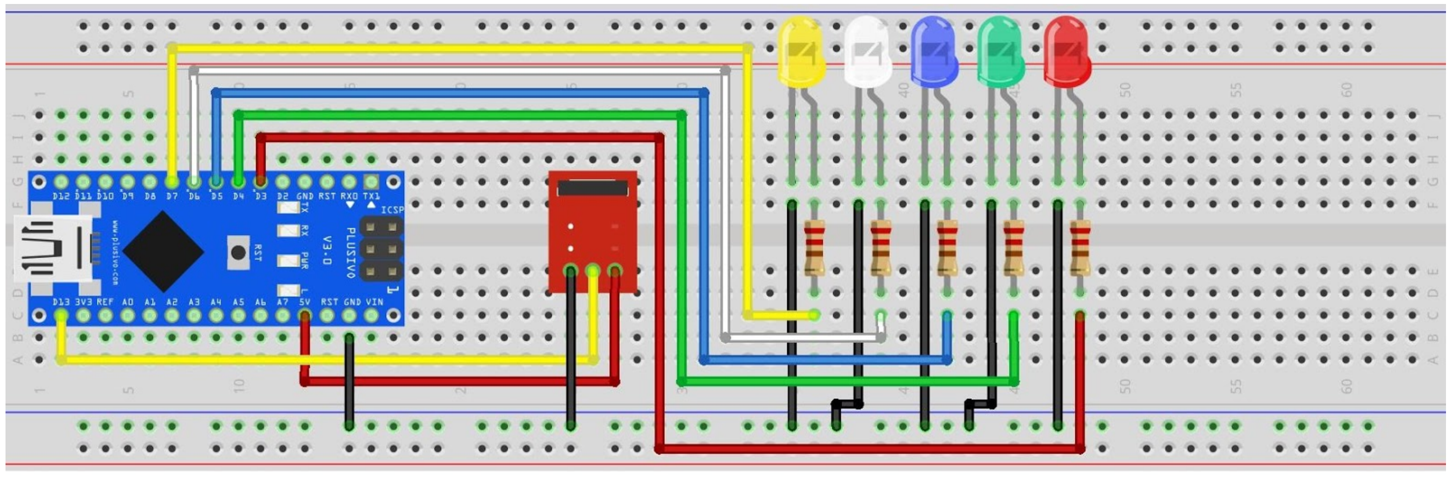
1 x Nano R3
1 x Breadboard 830p
1 x IR receiver module
1 x IR remote
1 x 5mm red LED
1 x 5mm green LED
1 x 5mm blue LED
1 x 5mm white LED
1 x 5mm yellow LED
5 x 220 ohm resistor
14 x M-M wires (Male to Male jumper wires)

11.3 Connection

Schematic



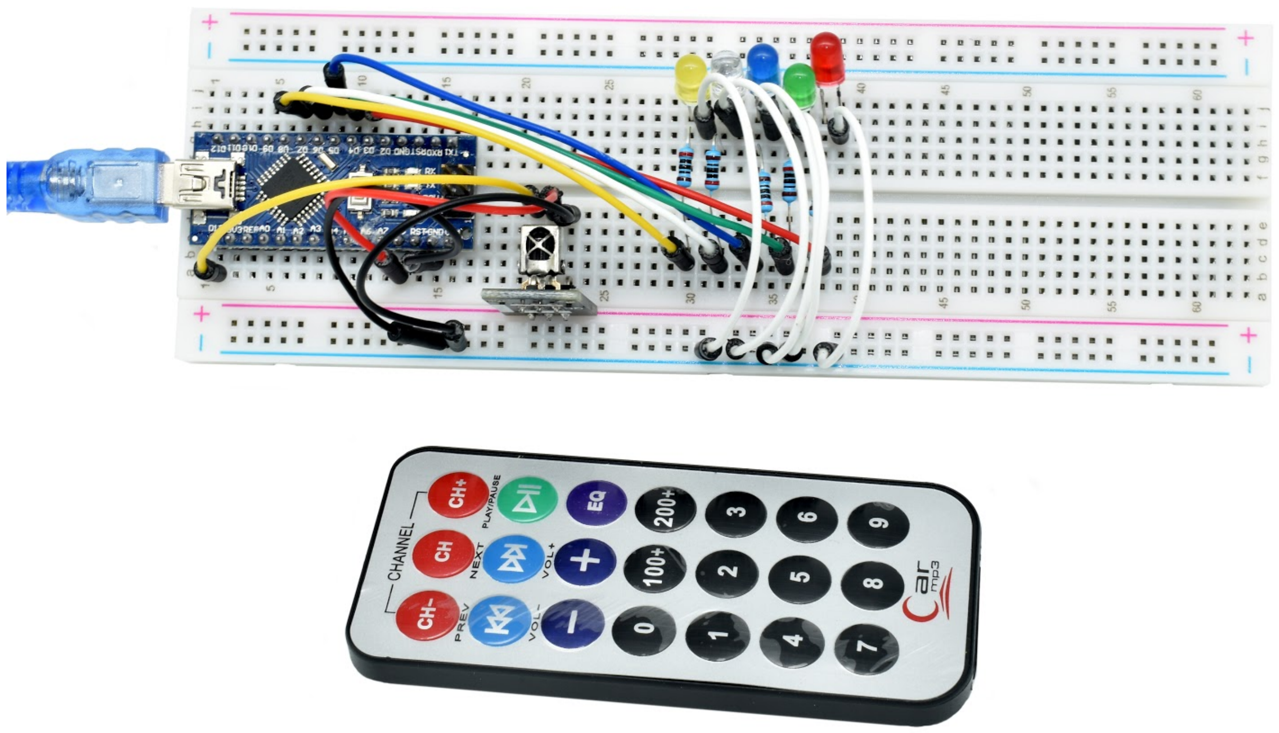
Wiring diagram



11.4 Code

After wiring, find and open the program located in the folder – Lesson 14 IR Receiver Module, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading. To be able to run this, have the < IRremote > library installed or re-install it, if necessary, for the code to work. For details about loading the library file, see Lesson 2.

11.5 Example picture



Lesson 12: DC Motors

12.1 Overview

This lesson will be teaching you the basics on how to control a small DC motor using L293D IC and an Arduino.

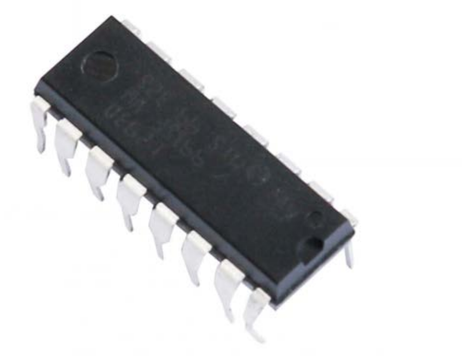
12.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x L293D IC
1 x Fan blade and 3-6v motor
8 x M-M wires (Male to Male jumper wires)

12.3 Component Introduction

L293D

This is quite a handy chip, as it has the ability to control two motors individually. We'll be using half of it in this particular case. The pins on the right hand side of the chip are mostly for managing a second motor.



Product Specifications

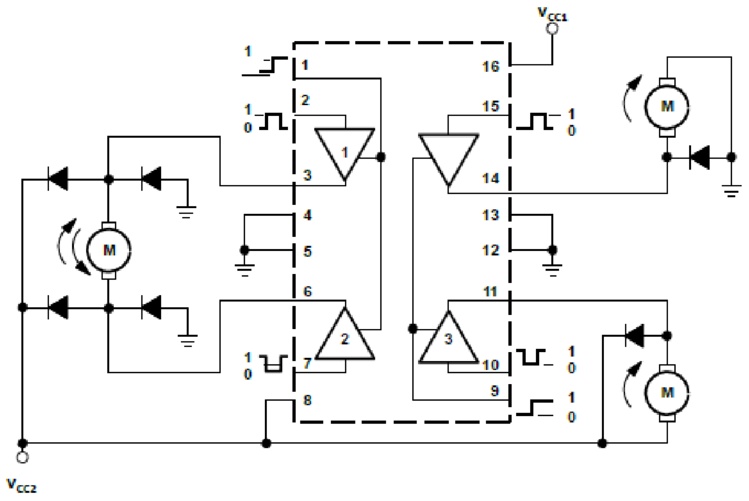
- Features Unitrode L293 and L293D Products From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

Description/ordering information

The purpose of the L293 is to supply bidirectional currents of a maximum of 1A, at voltages between 4.5V and 36V, whereas the L293D can handle a bidirectional current of up to 600mA using the same voltage range. Both of them are called quadruple high-current half-H drivers, and they are both devised with the purpose of driving inductive parts: DC and bipolar stepping motors, relays, solenoids, and any other high-current and voltage loads in positive-supply operations.

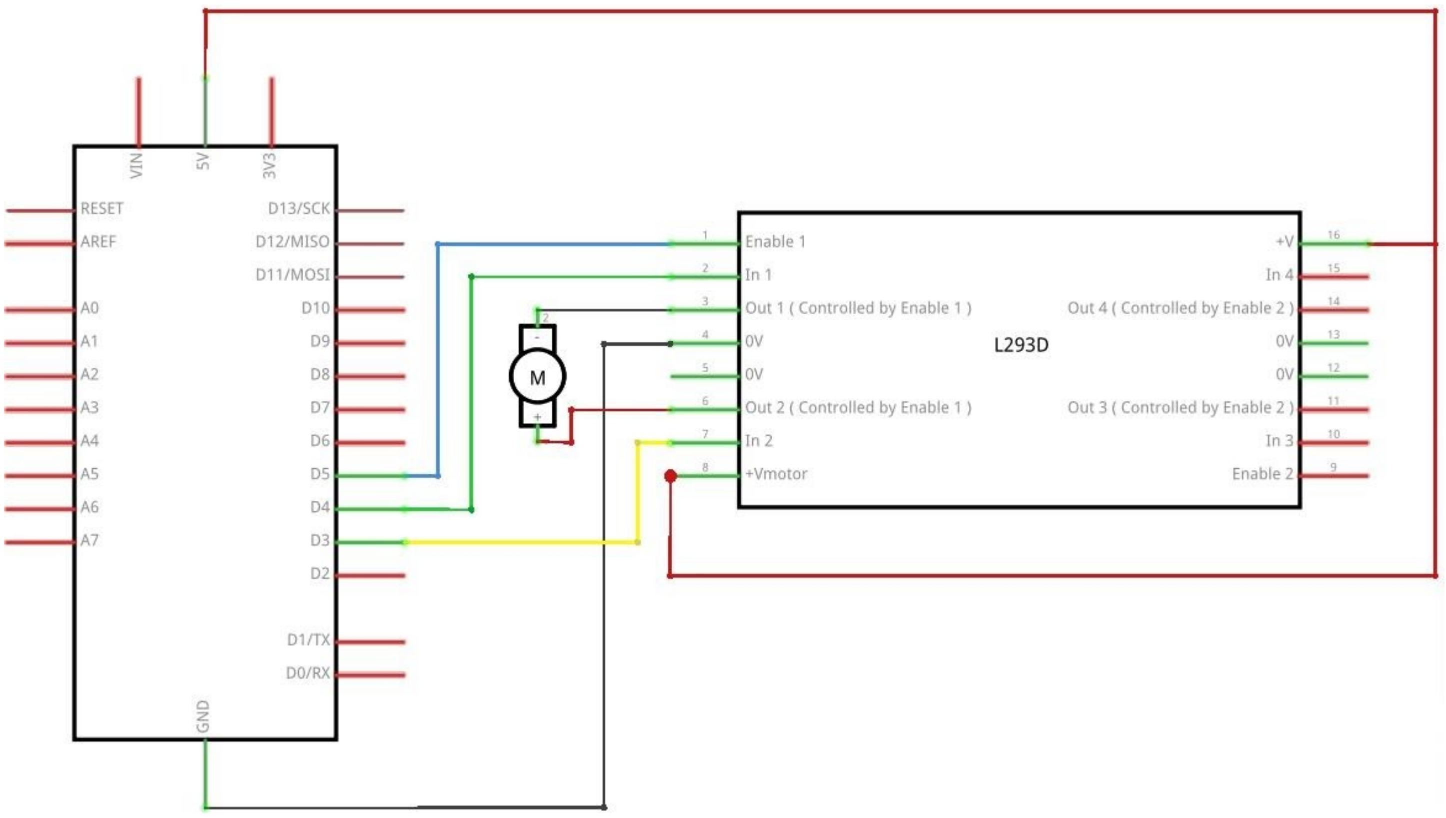
Each and every one of the outputs are complete totem-pole drive circuits, with a Darlington transistor sink and a pseudo-Darlington source, while all the inputs are TTL compatible. Drivers 1 and 2 are enabled by 1,2EN, while drivers 3 and 4 are enabled by 3,4EN, as they are set up in pairs. They are enabled when an enable input is high, and the corresponding outputs are active and in phase with their inputs. Alternatively, those drivers are disabled when the enable input is low, and the corresponding outputs are off and have high impedance. Each pair of drivers creates a full-H, also called a bridge, reversible drive suited for solenoid or motor usage with the appropriate data inputs.

Block diagram

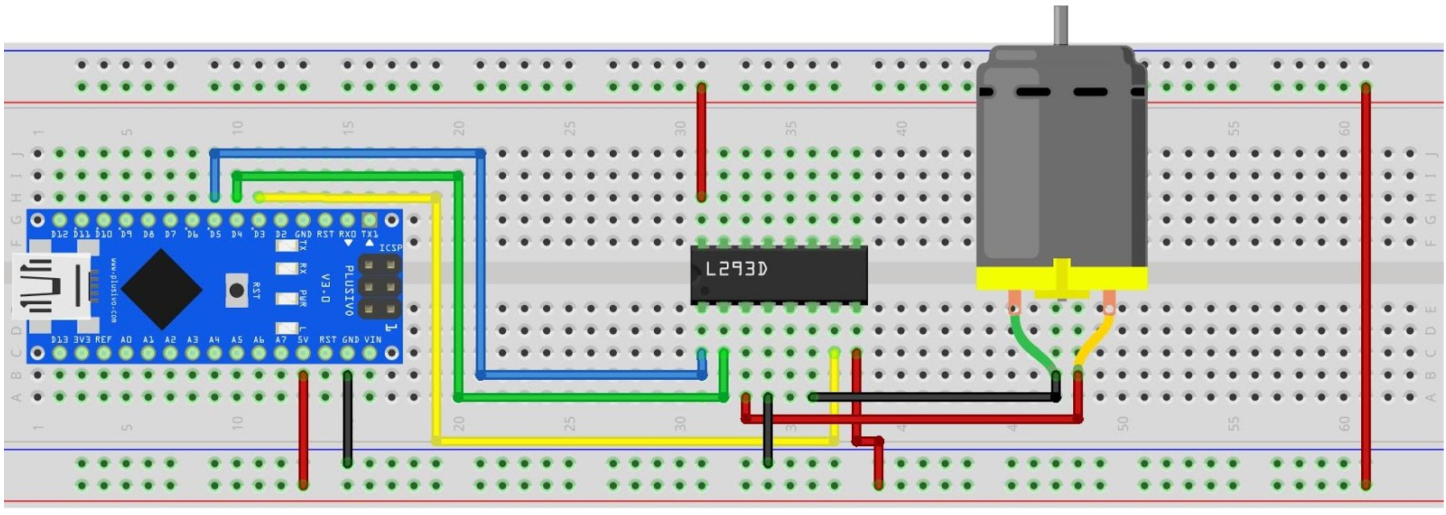


12.4 Connection

Schematic

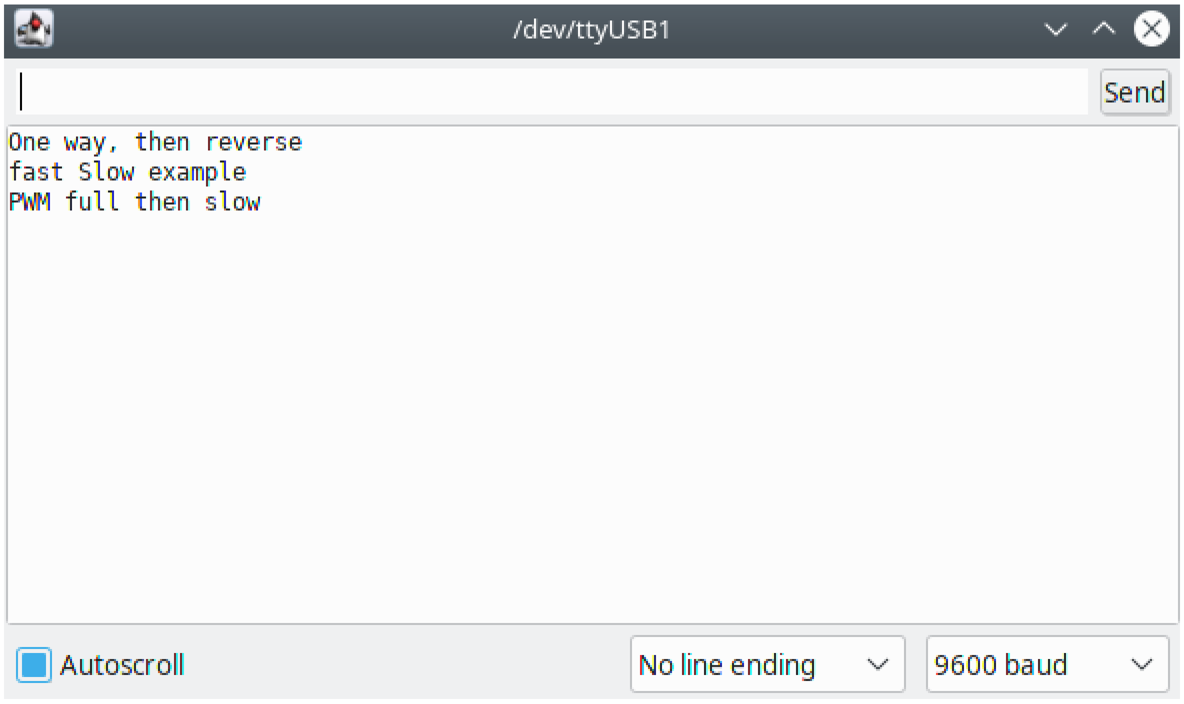


Wiring diagram



The code below does not require a separate power supply or a battery, it actually uses the 5V power from the Arduino. This can only be done because the L293D is controlling it and would be risky without it.

DO NOT connect a motor directly to the device, because if you will get an electrical feedback when you turn the motor off. If you have a small motor, this will damage your Nano, but with a large one, you can even see a flame and sparks effect.



```
One way, then reverse
fast Slow example
PWM full then slow
```

No line ending

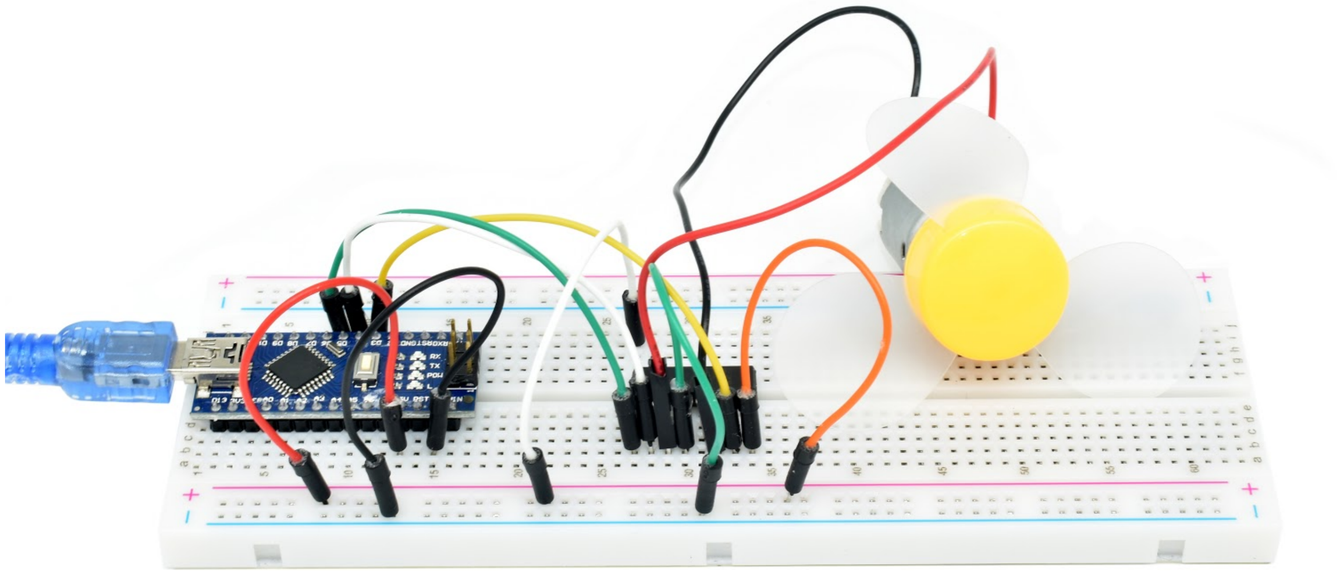
9600 baud

12.5 Code

After wiring, find and open the program located in the folder Lesson 12 DC Motors, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

Once the program loads, power ON all the power switches. There will slightly rotation of the motor clockwise and counterclockwise for 5 times. Afterwards, it will continue to dramatically rotate clockwise and counterclockwise. Then after a short pause, it will dramatically rotate counterclockwise. Then, to drive the motor, the controller board will send a PWM, so that the motor will gradually lower its maximum RPM to the minimum and increase to the maximum again. Finally, it stops for 10 seconds until the next cycle starts.

12.6 Example picture



Lesson 13: Relay

13.1 Overview

In this lesson, you will learn the basics for using a relay.

13.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x Fan blade and 3-6v motor
1 x 5v Relay
5 x M-M wires (Male to Male jumper wires)

13.3 Component Introduction

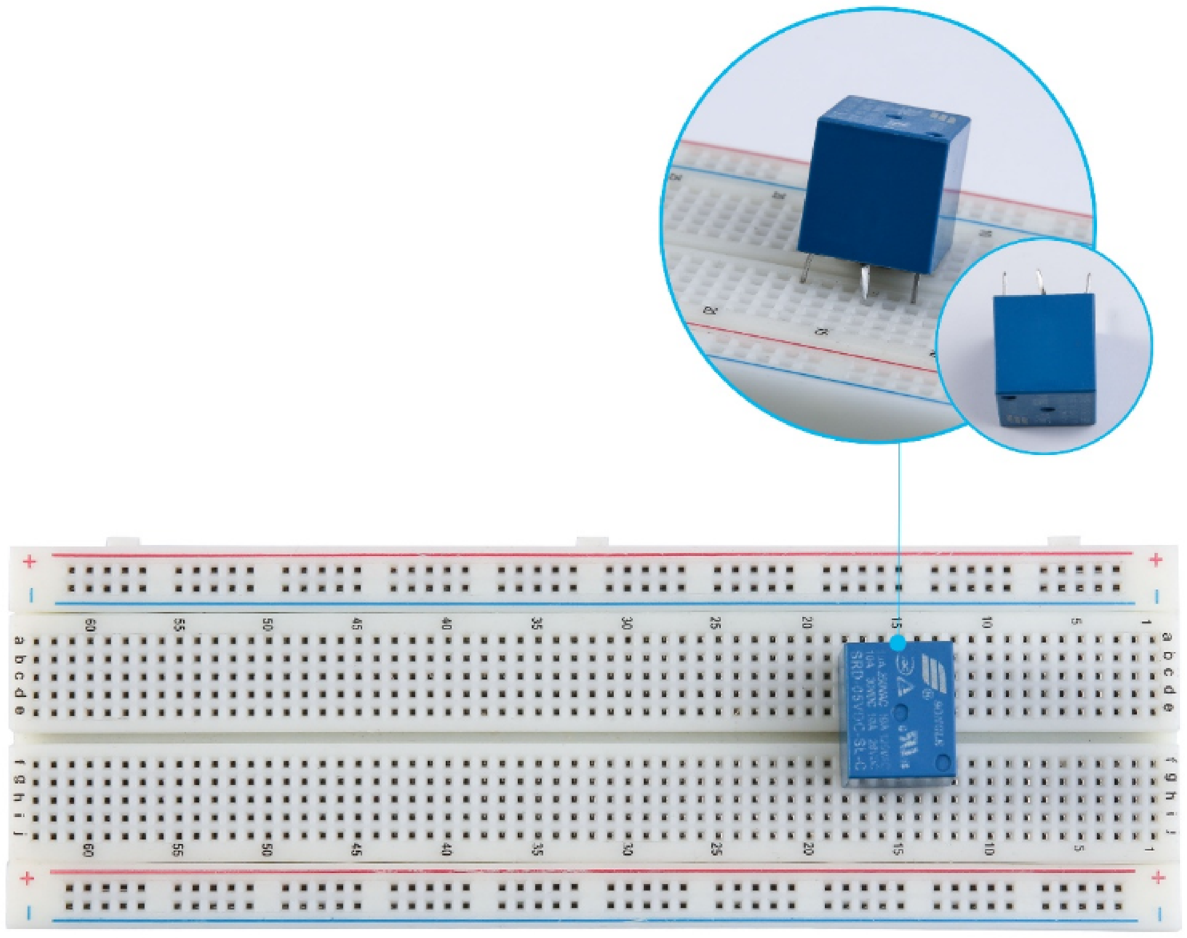
Relay

A relay is an electrically operated switch, generally using an electromagnet to mechanically operate a switch, but sometimes its operating principles are also used as in solid-state relays. Relays are used where it is necessary to control a circuit using 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 where they repeated the signal coming in from one circuit and re-transmitted it on another circuit. As early computers, what they do is to perform logical operations further down the line. The way they are working is that they repeated the signal coming in from one circuit and re-transmitted it on another circuit.

A contactor is a type of relay that can handle the high power needed to directly control an electric motor or other loads. Solid-state relays control power circuits using a semiconductor device to perform the switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are utilized to safeguard electrical circuits from overload or faults. In modern electric power systems, these functions are performed by digital instruments called "protective relays".

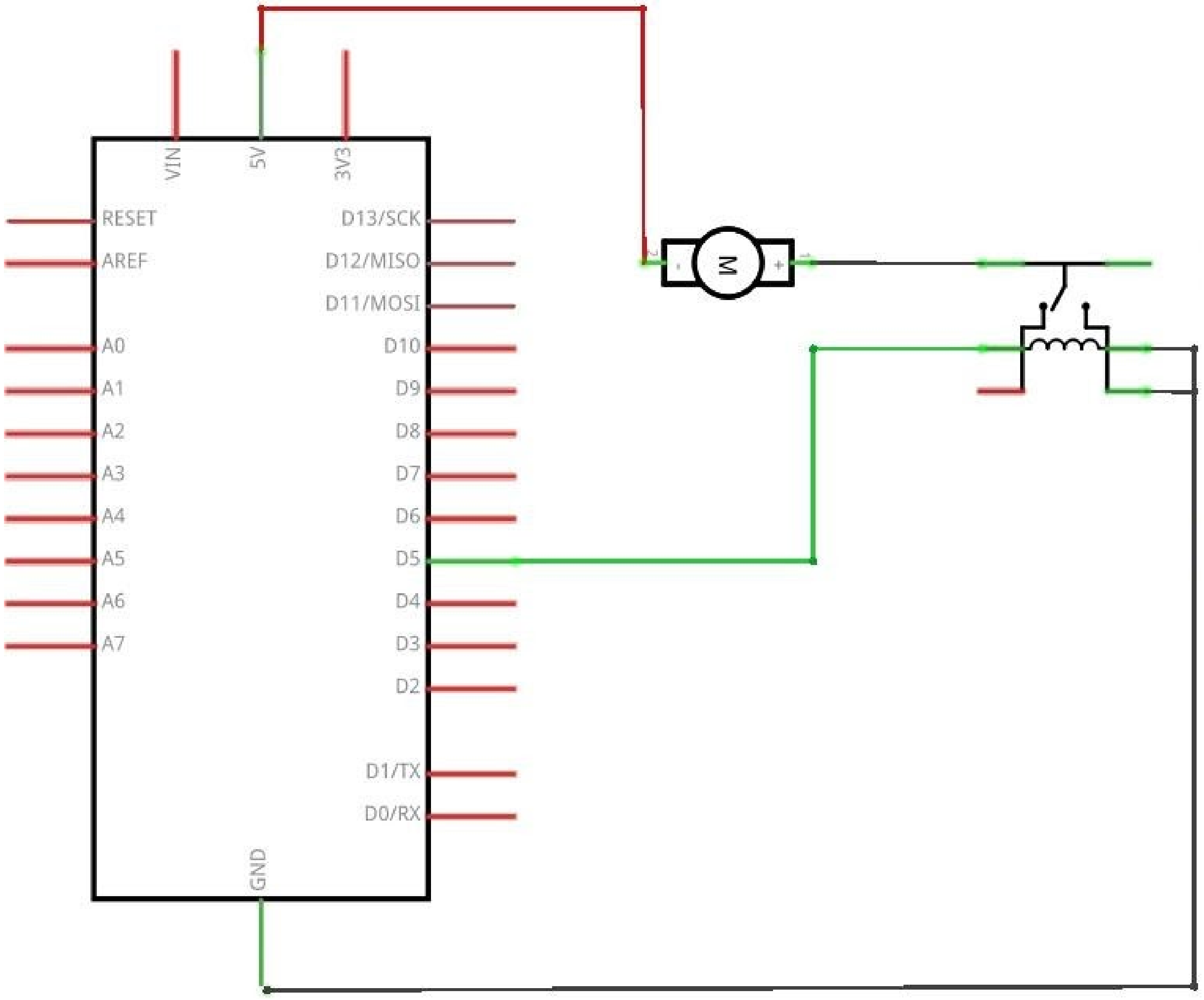
You can find the schematic showing how to drive a relay with our device below.

If you are unsure about how to insert the relay into the breadboard. As you can see in the photo below, you will need to slightly bend one of the pins of the relay so you can insert it into the breadboard.

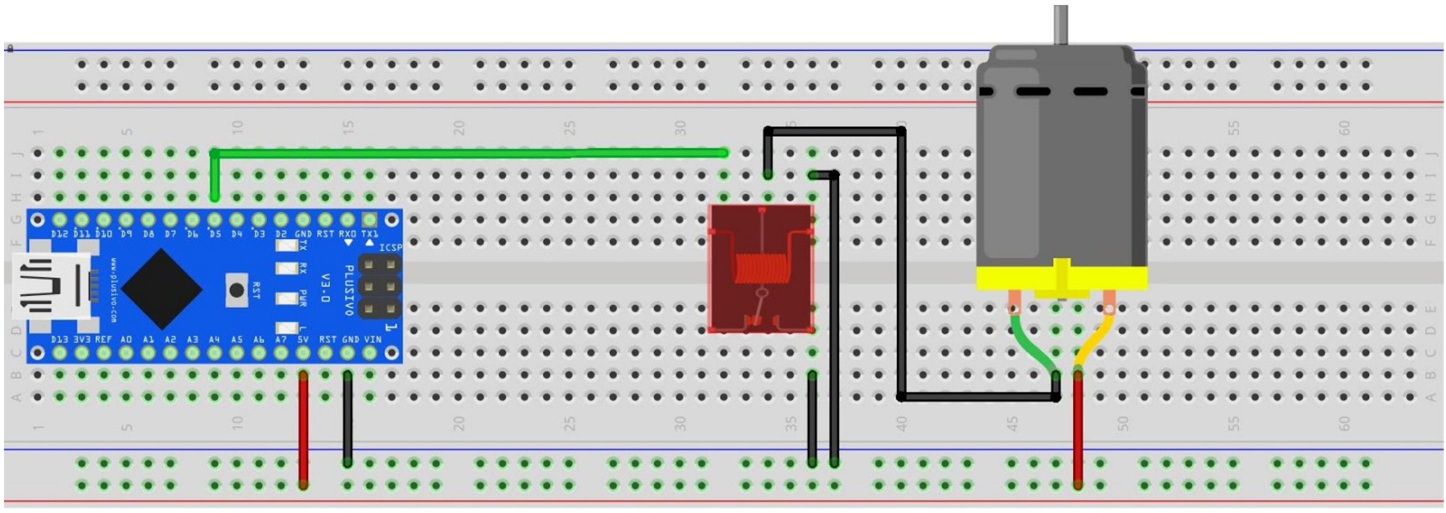


13.4 Connection

Schematic



Wiring diagram

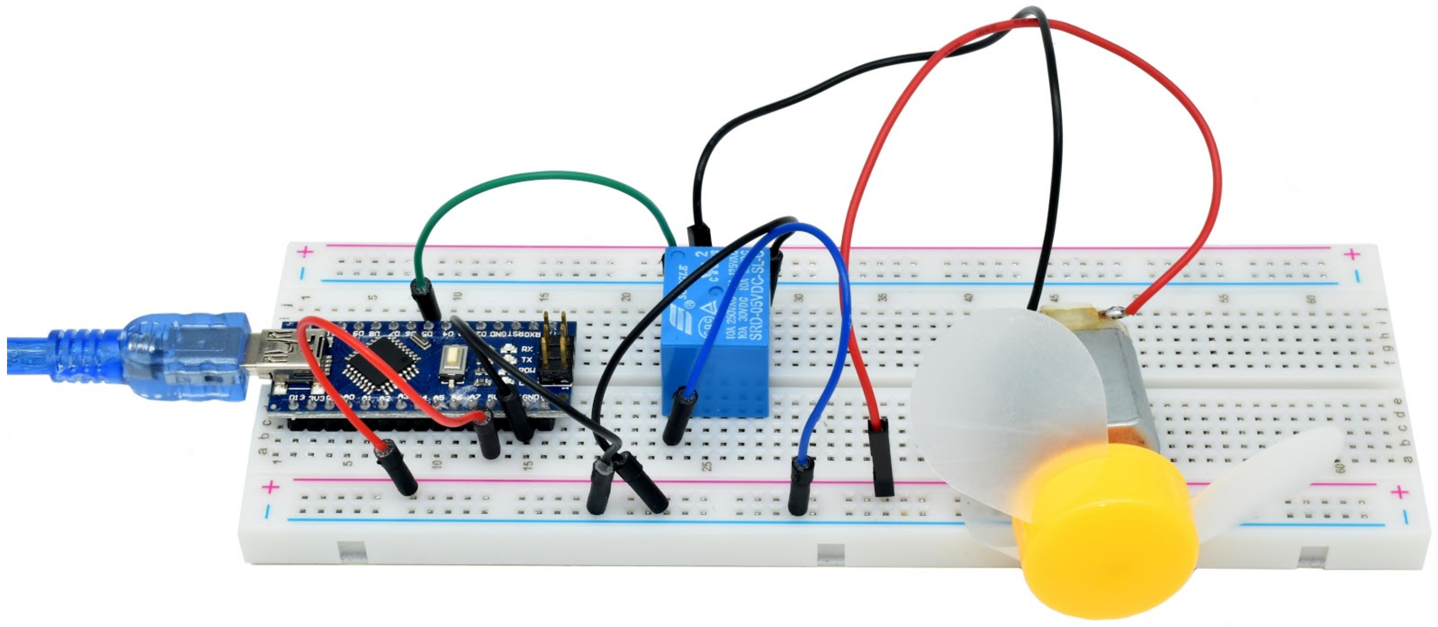


13.5 Code

After wiring, find and open the program located in the folder - Lesson 13 Relay, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

After the program loads. The relay will pick up with a ringing sound. Then, the motor will rotate and eventually, the relay will be released, and the motor stops.

13.6 Example picture



Lesson 14: Controlling DC motor with the remote

14.1 Overview

In previous lesson we controlled LEDs with remote, in this lesson we will learn how to control DC motor with remote, using 0 and 1 buttons.

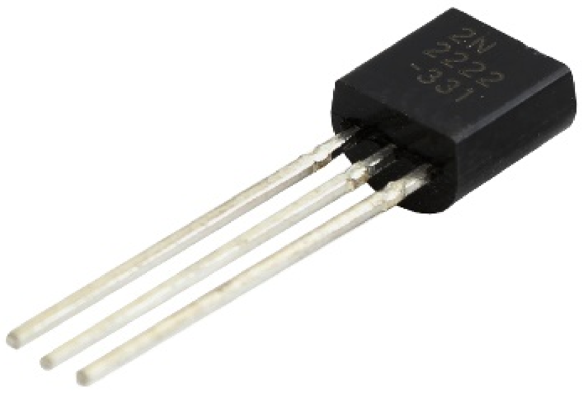
14.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x Transistor 2N2222
1 x Fan blade and 3-6v motor
1 x 2 k ohm resistor
1 x IR receiver module
1 x IR remote
6 x M-M wires (Male to Male jumper wires)

14.3 Component Introduction

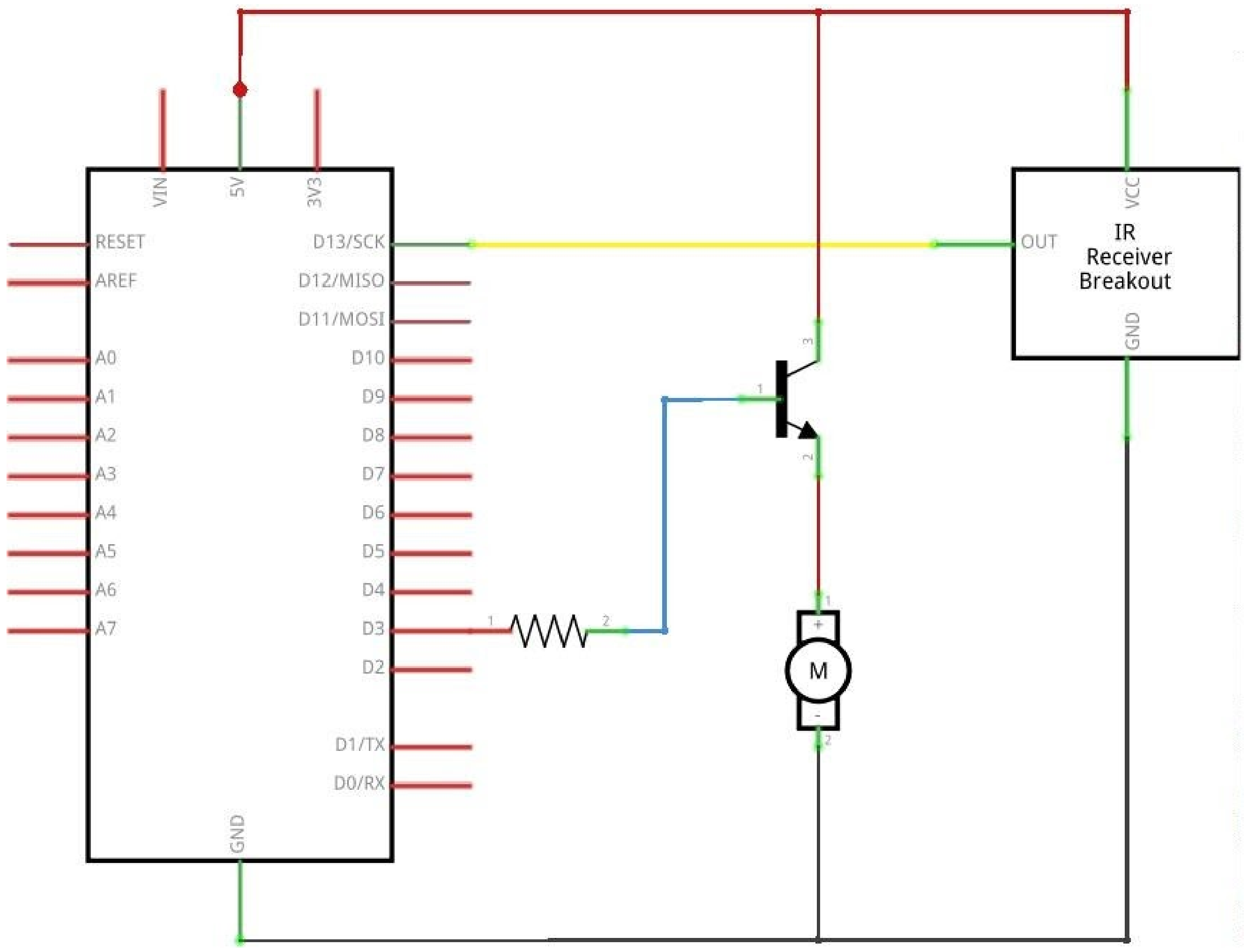
Transistor

The transistor is a circuit element that can be used as an electrically operated switch. It is made of semiconductor materials, has three terminals and can interrupt or close the circuit between two terminals depending on the command applied on the control pin.

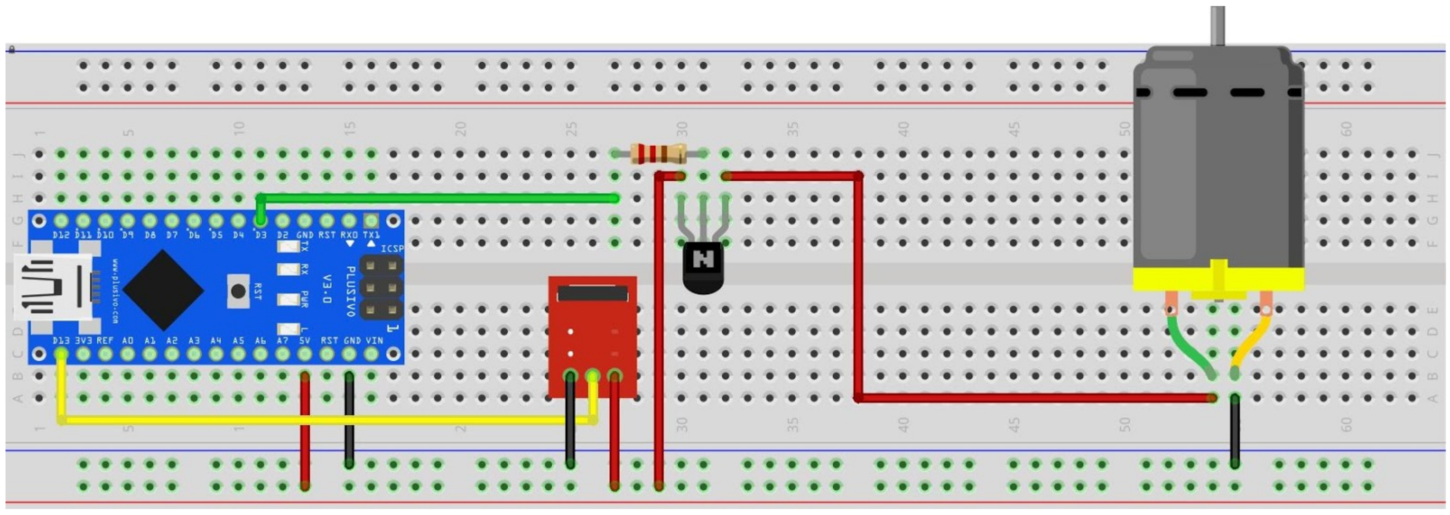


14.4 Connection

Schematic



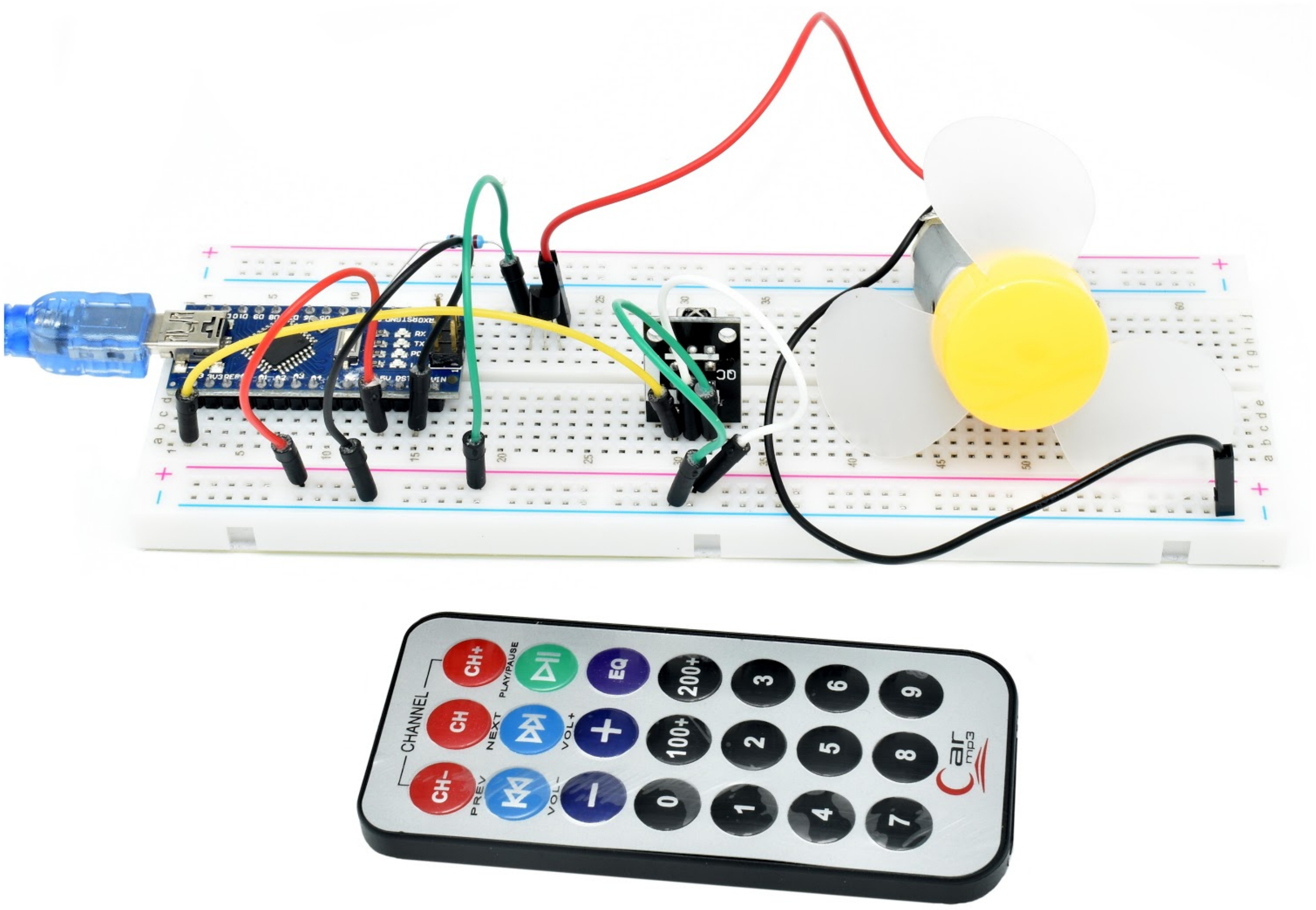
Wiring diagram



14.5 Code

After wiring, find and open the program located in the folder Lesson 12 DC Motors, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

14.6 Example picture



Lesson 15: Eight LED with 74HC595

15.1 Overview

This lesson will show you how to use eight LEDs with an Arduino without using up 8 output pins! Even though you could wire up eight LEDs with a resistor for each of them to a pin, you would quickly run out of pins for other components, as we usually have an array of devices such as buttons, sensors, servos, etc. for most projects. Instead, we are using a chip called the 74HC595 Serial to Parallel Converter, which has eight outputs and three inputs that you could use to give data to.

This chip makes the board a little slower in use (you can change the LEDs about 500,000 times a second compared to 8,000,000 a second) but it's still much faster than the human eye can distinguish, so it's not an issue.

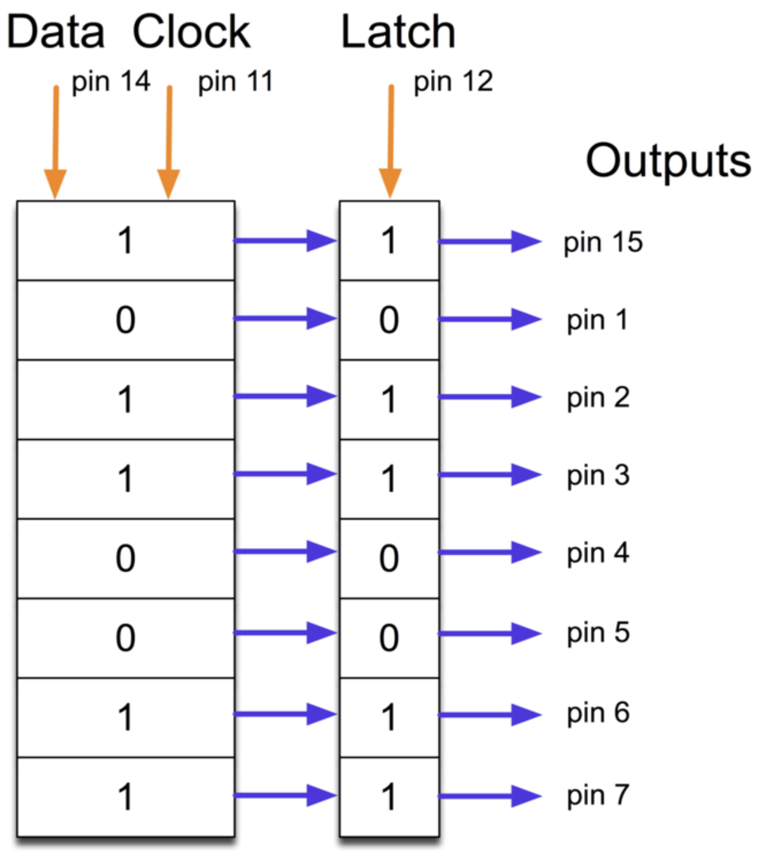
15.2 Components Required

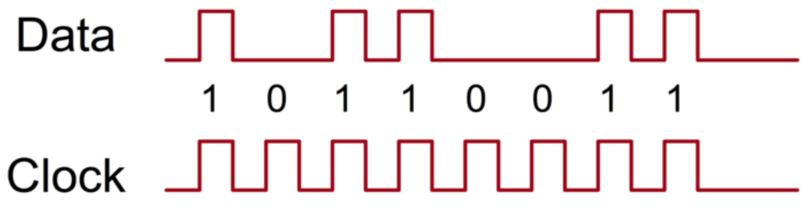
1 x Nano R3
1 x Breadboard 830p
8 x LEDs
8 x 220 ohm resistors
1 x 74hc595 IC
17 x M-M wires (Male to Male jumper wires)

15.3 Component Introduction

74HC595 Shift Register

The shift register is a kind of chip containing eight memory locations, with the values 1 or 0. We input the data using the 'Data' and 'Clock' pins of the chip to set these values on or off.



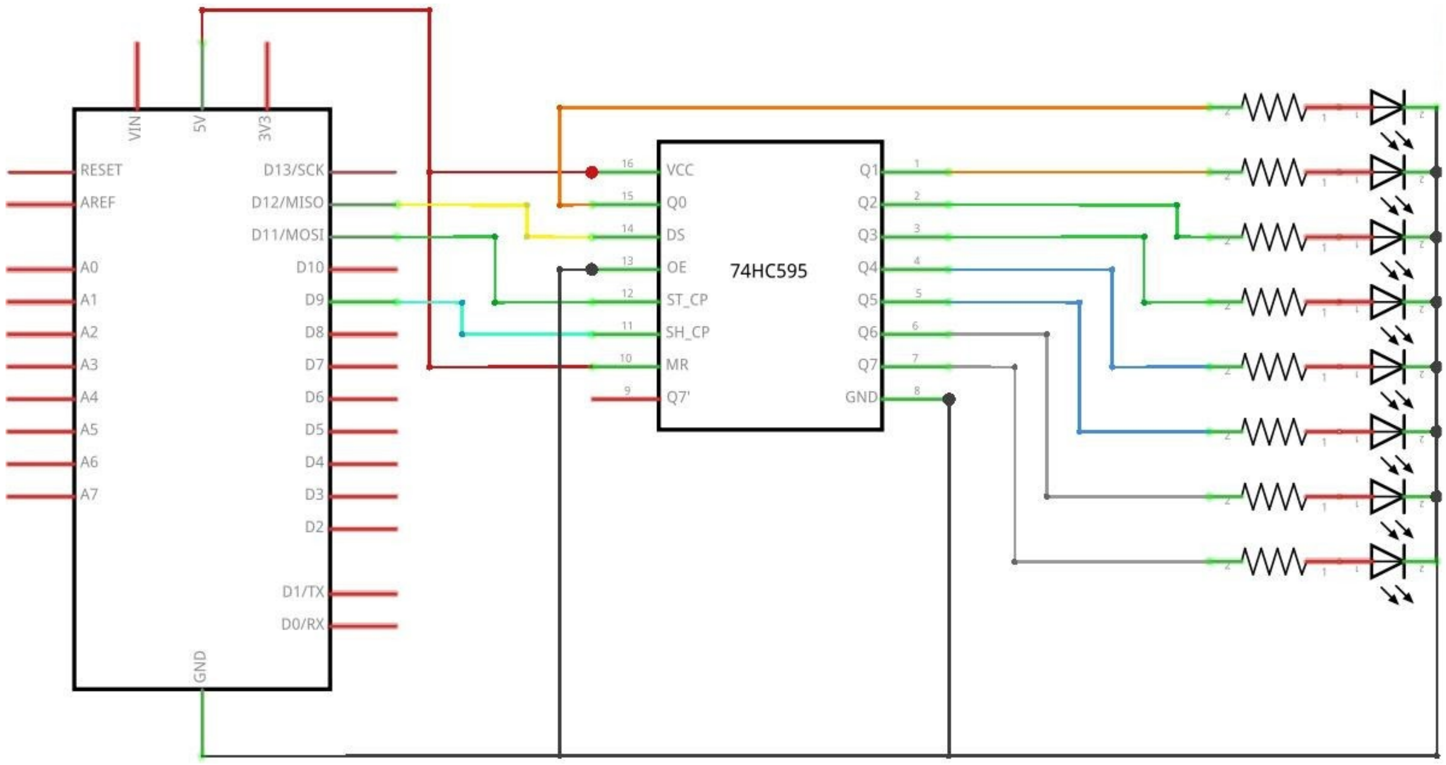


The clock pin should receive eight pulses. For each of these, if the data pin is either high or low, then a 1 or a 0 goes into the shift register respectively. After all the pulses are received, enabling the 'Latch' pin copies those eight values to the latch register.

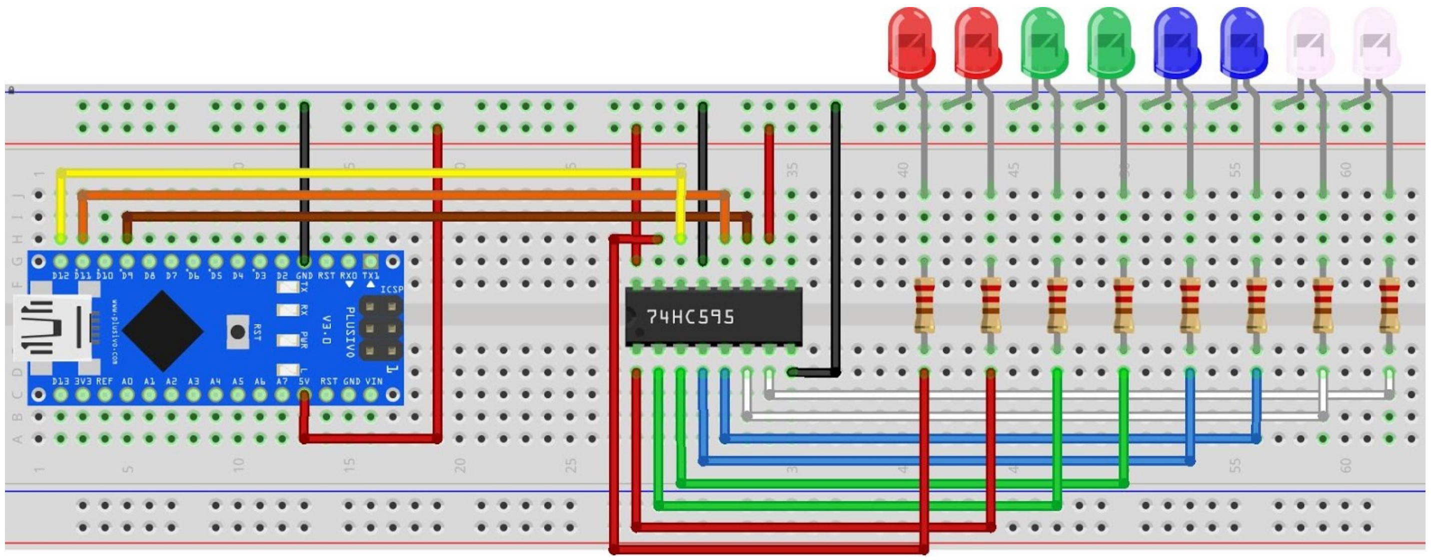
Additionally, the chip has an output enable (OE) pin, utilized to weather enable or disable the outputs collectively. This can be linked to a PWM-capable Arduino pin, allowing the use of 'analogWrite' to the light intensity of the LEDs. We connect this pin to GND since it is active low.

15.4 Connection

Schematic



Wiring diagram



We have eight LEDs and eight resistors to connect, so let us begin.

We recommend you place first the 74HC595 chip as almost everything else connects to it. Place it in such a way that towards the top of the breadboard is the U-shaped notch. Located at the left of this notch is the Pin 1 of the chip.

Digital 12 from the Nano goes to pin #14 of the shift register
Digital 11 from the Nano goes to pin #12 of the shift register
Digital 9 from the Nano goes to pin #11 of the shift register

All except one of the outputs is placed on the left side of the chip, where the LEDs can be found as well, to make the connection effortless.

Then, place the resistors on the breadboard. You should check that the leads of the resistors are not touching each other before you connect the power to your Nano. You can try to shorten the leads so that they are closer to the surface of the breadboard, if ever you are encountering any difficulties

Afterwards, put the LEDs in place. The longer positive leads must all be in the direction of the chip, irrespective of which side of the breadboard they are on.
Connect the jumper wires, but remember the one that goes from pin 8 of the IC to the GND column.

Try the sketch mentioned. Each LED should light one by one until they are all on. Afterwards, they will all turn off and the sequence repeats.

15.5 Code

After wiring, find and open the program located in the folder Lesson 15 Eight LED with 74HC595, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

We start by defining the three pins we will be utilizing. These are the 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;
```

Then, we define a variable called ‘leds’, which will hold the order of the
LEDs being turned on or off. The data 'byte' represents numbers using eight bits, each being on or off, ideal for monitoring which of the eight LEDs are powered on or off.

```
byte leds = 0;
```

The 'setup' function establishes the three pins to be used as digital outputs.

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

At first, the 'loop' function turns all of the LEDs off by setting the variable ‘leds' to 0. Afterwards, it calls the 'updateShiftRegister' function, which sends the 'leds' sequence to the shift register in order to turn all the LEDs off.

There will be a half second pause for the loop function and using the ‘for’ loop and the variable ‘i’, it counts through the number of leds (from 0 to 7). Every time, the 'bitSet' function is utilized to set the bit that controls that LED in the variable 'leds'. Then it calls the ‘updateShiftRegister' to update the LEDs. Afterwards, there is a half second delay before the incrementation of 'i' and the next LED is turned on.

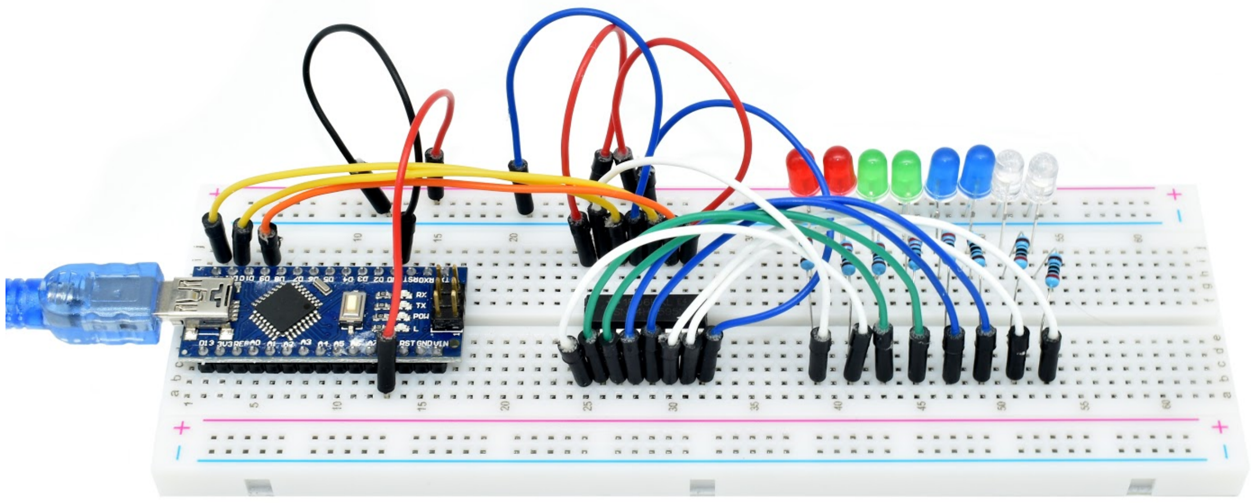
```
void loop()
{
leds = 0;
updateShiftRegister();
delay(500);
for (int i = 0; i < 8; i++)
{
bitSet(leds, i);
updateShiftRegister();
delay(500);
}
}
```

First of all, latchPin is set to low by 'updateShiftRegister' function followed with a call of the 'shiftOut' function before bringing back the ‘latchPin’ to high. We have four parameters here: the first two are for Data and Clock, the third is to identify which end of the data you wish to begin at, and the last is for the actual data to be shifted into the shift register. We are
beginning with the rightmost bit, also called the 'Least Significant Bit' (LSB).

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

In order to turn one of the LEDs off, you would call the ‘bitClear’ function with the 'leds' variable, which will have a setting of ‘0’, and then followed with calling the 'updateShiftRegister' to have the actual LEDs updated.

15.6 Example picture



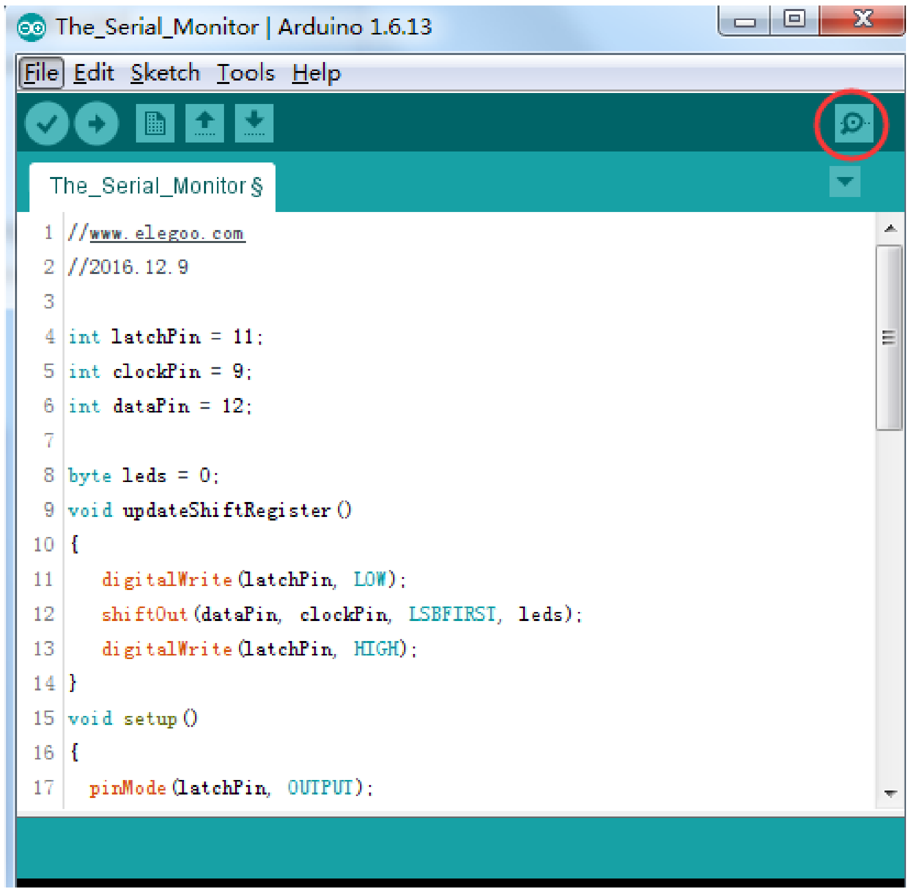
Lesson 16: The Serial Monitor

16.1 Overview

This lesson will be based on the previous lesson, with the option of managing the LEDs from your computer. We will be utilizing the Serial Monitor – the connection between the computer and the Arduino, enabling you to send and receive text messages for debugging or controlling from keyboard. After this lesson, you can send commands from your computer to turn the LEDs on, using the same exact parts and a similar breadboard layout.

16.2 Steps taken

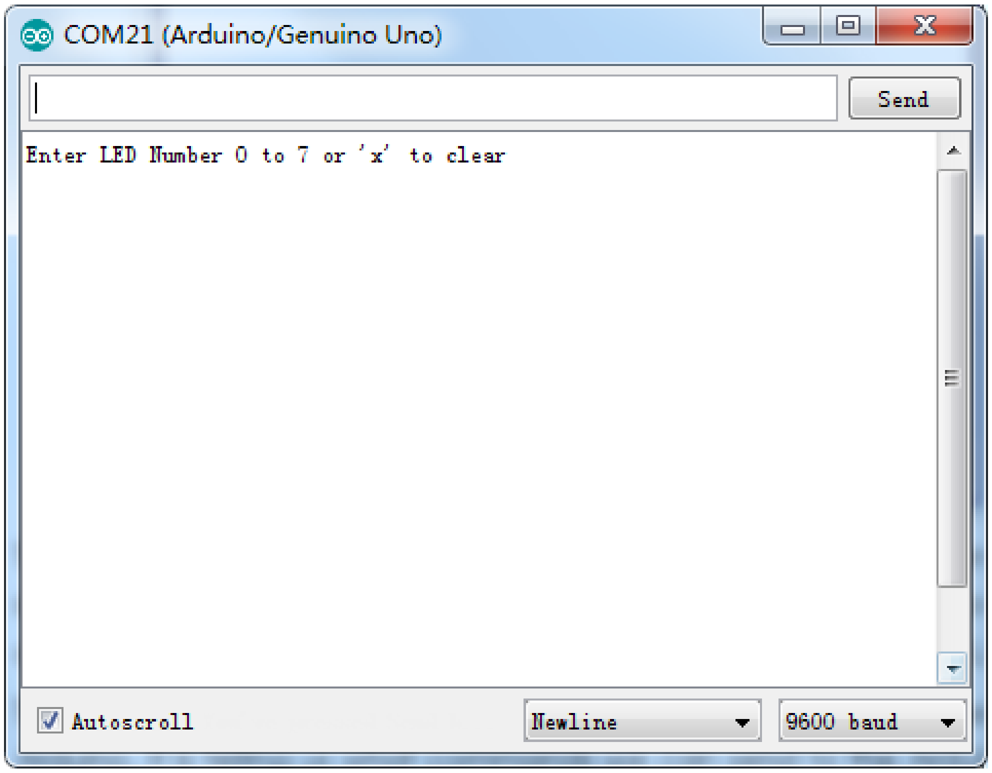
After uploading this code onto your device, click on the right-most button (as shown below) on the toolbar in the IDE.


The_Serial_Monitor | Arduino 1.6.13
File Edit Sketch Tools Help
The_Serial_Monitor §

```
//www.elegoo.com
//2016.12.9

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);
```

This window will open.

Press the Serial Monitor button to turn on the serial monitor. We discussed the serial monitor in detail in Lesson 2.

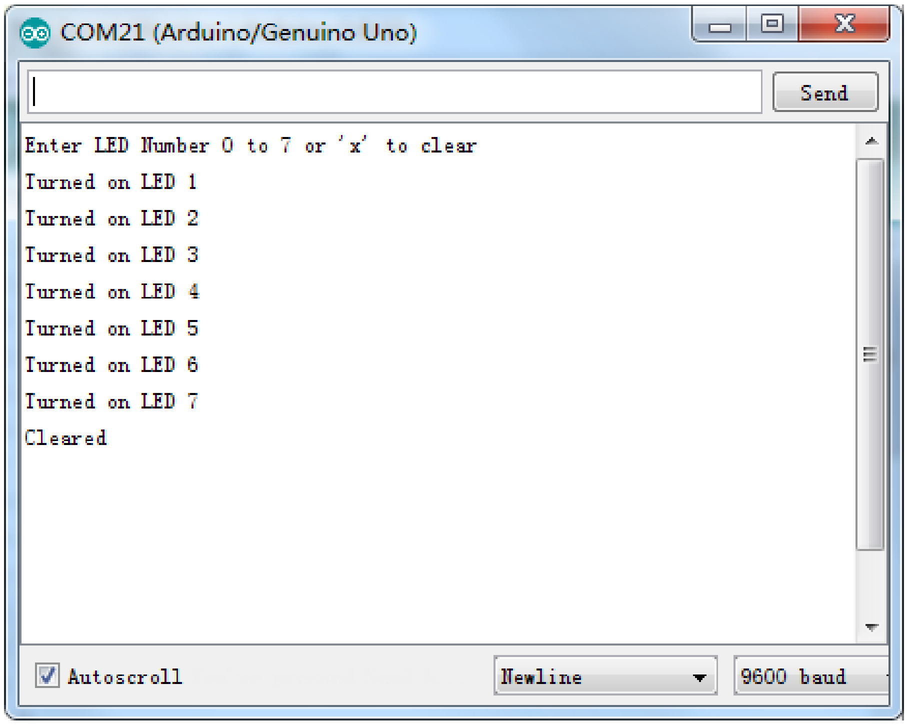


```
Enter LED Number 0 to 7 or 'x' to clear
```

This is the Serial Monitor and a part of the IDE software. Its purpose is to allow the Nano board and your computer to communicate (to send and receive the message to each other).

The message that appeared, “Enter LED Number 0 to 7 or 'x' to clear” has been sent by the device, which informs us of the commands we can give to the Arduino: 'x' to turn all the LEDs off or the number of the LED you want to turn on (where 0 to7 is from bottom LED to top LED). You can test the following commands into the top area of the Serial Monitor and press 'Send': x 0 3 5.

If the LEDs are already turned off, typing ‘x’ will not do anything, but upon entering each number, the matching LED will turn on and a confirmation message will appear, like the one 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
```

Typing ‘x’ one more time and clicking ‘Send’ will turn off all the LEDs.

16.3 Code

After wiring, find and open the program located in the folder Lesson 16 The Serial Monitor, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

As expected, this sketch is largely built on the one used in Lesson 16, so we only need to explain the new lines of code. You may deem it useful to refer to the complete sketch in your IDE. As you can see, there are three new lines at the end in the 'setup' function:

```
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' to clear");
 }
```

Initially, there is the 'Serial.begin(9600)' command which starts the communication, enabling the Nano to send out commands through the USB connection. The value 9600 is the 'baud rate' of the connection and it determines the speed of the data being sent. This can be modified to a higher value, but you will also need to change the Serial monitor to the same value. For now, leave it at 9600.

The 'while' loop assures us that there is something on the other side of the USB connection for the device to communicate with, or else the message cannot be displayed even though it is sent. This line is only needed if you are using a Leonardo board because the Arduino board routinely resets itself when you open the Serial

Monitor.

The final new lines in 'setup' send out the message we see at the top of the Serial Monitor. Let’s take a look at the ‘loop’ function:

```
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");
}
}
}
```

The ‘if’ statement tells us everything that happens inside the loop and nothing else will happen if the call to the built-in function 'Serial.available()' is 'true'. If data is sent to the Nano and is ready to be processed, then Serial.available() will return 'true'. Meaning, the Serial.available() will return 'true' if the buffer, where incoming messages are held, is not empty. If a message has been accepted, then we move on to the next part of the code:

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

This extracts the next character from the buffer and assigns it to the variable 'ch'. The variable 'ch' is of type 'char', which holds a single character. This variable will either be a number between 0 and 7 or the letter 'x'.

The 'if' statement in the following line checks if it is a single digit by confirming that the value of 'ch' is between '0' and '7'. While it may seem peculiar comparing characters like this, it is perfectly reliable, as each character is illustrated by a unique number, called an ASCII value. Thus, when we compare characters using the operators <= and >=, we are actually comparing the ASCII values. In the case that the ‘if’ result is positive, we proceed on to the next line:

Next, we are subtracting the digit '0' from any digit entered. Thus, typing '0' then '0' - '0' will equal 0, whereas typing '7' then '7' - '0' will equal the number 7 as the ASCII values are actually the ones being used in the operation. Because we know the number of the LED we want to power on, we just need to store that in the variable 'leds' and have the shift register updated.

```
bitSet(leds, led);
updateShiftRegister();1
```

The following lines send a confirmation message back to the Serial Monitor.

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

You can see from the first line that it uses Serial.print rather than Serial.println, as we do not want to begin a new line after printing. We will print the message in two parts: 'Turned on LED ' and the number of the LED as an ‘int’ variable. Serial.print can take any type of variable, either a text string enclosed in double-quotes, or an 'int' or any type of variable. Ensuing the 'if' statement that handles the single digit case, there is a second 'if', one that checks whether 'ch' is the letter 'x'.

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

In that is the case, it then clears each one of the LEDs and sends a message of confirmation.

Lesson 17: Photocell

17.1 Overview

This lesson will be showing you the basics on how to measure brightness using Analog Input or how to check the intensity of light and how to use that value to manage the number of LEDs being lit using a photocell.

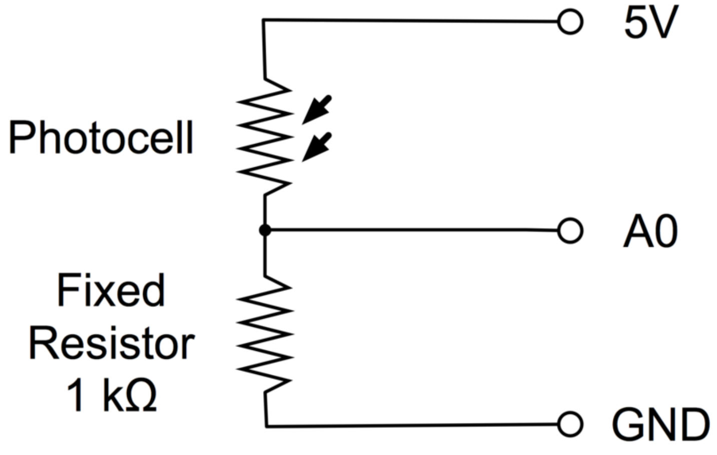
17.2 Components Required

1 x Nano R3
1 x Breadboard 830p
8 x LEDs
8 x 220 ohm resistors
1 x 1K ohm resistors
1 x 74hc595 IC
18 x M-M wires (Male to Male jumper wires)
1 x Photoresistor (Photocell)

17.3 Component Introduction

PHOTOCELL

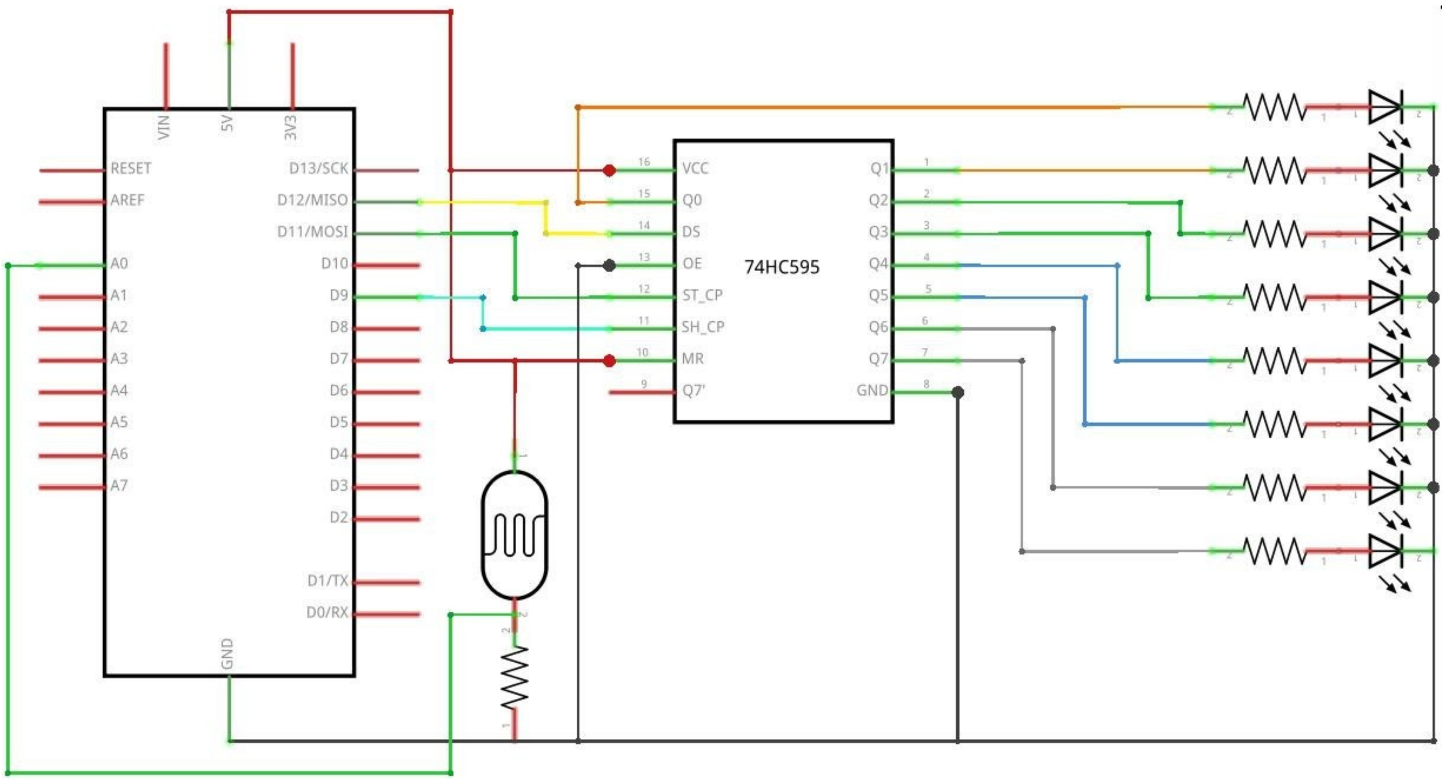
The photocell used here is a light dependent resistor (‘LDR’) type. From the name, we draw the conclusion that these components behave like a resistor, except that the resistance changes according to the level of light they are exposed to, with values between 50 kΩ in near darkness and 500 Ω in bright light. To transform this fluctuating value of resistance into something that the Arduino’s analog input can measure, it has to be converted into a voltage. The most basic way to do so is to integrate it with a fixed resistor.



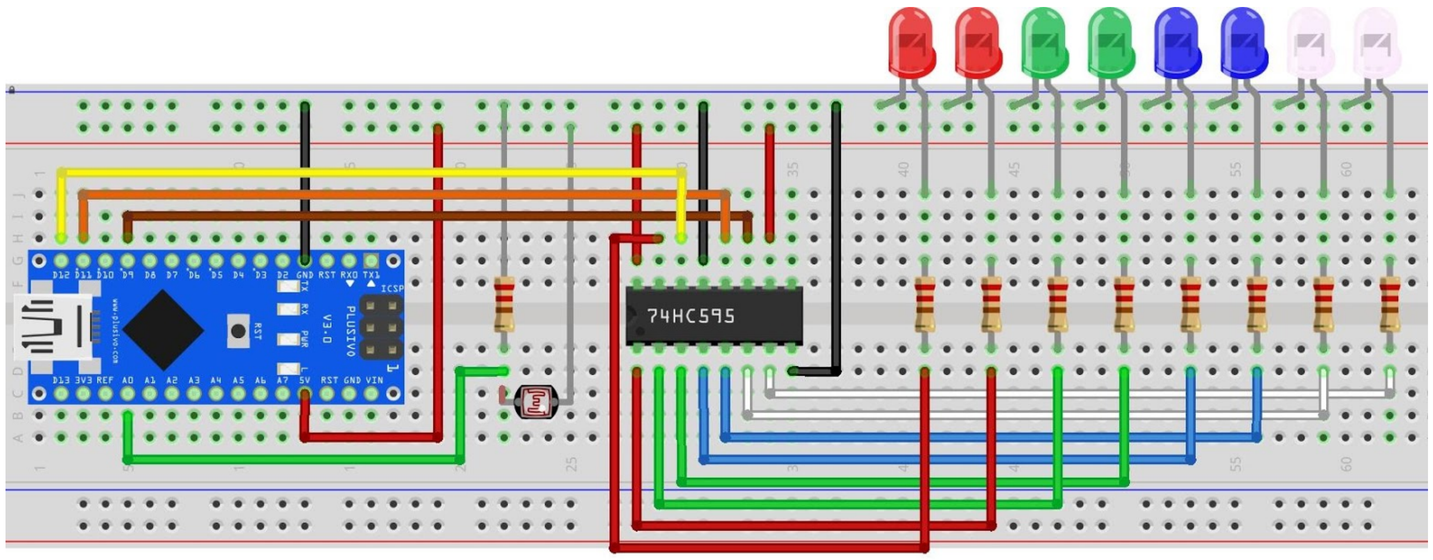
Alternatively, when the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it behaves as if the pot is being turned towards GND. Use the code found in the next section and try to cover the photocell or hold it near a light source to see how the values change.

17.4 Connection

Schematic



Wiring diagram


74HC595
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
PLUSIVO
V3.0
ICSP
RST

17.5 Code

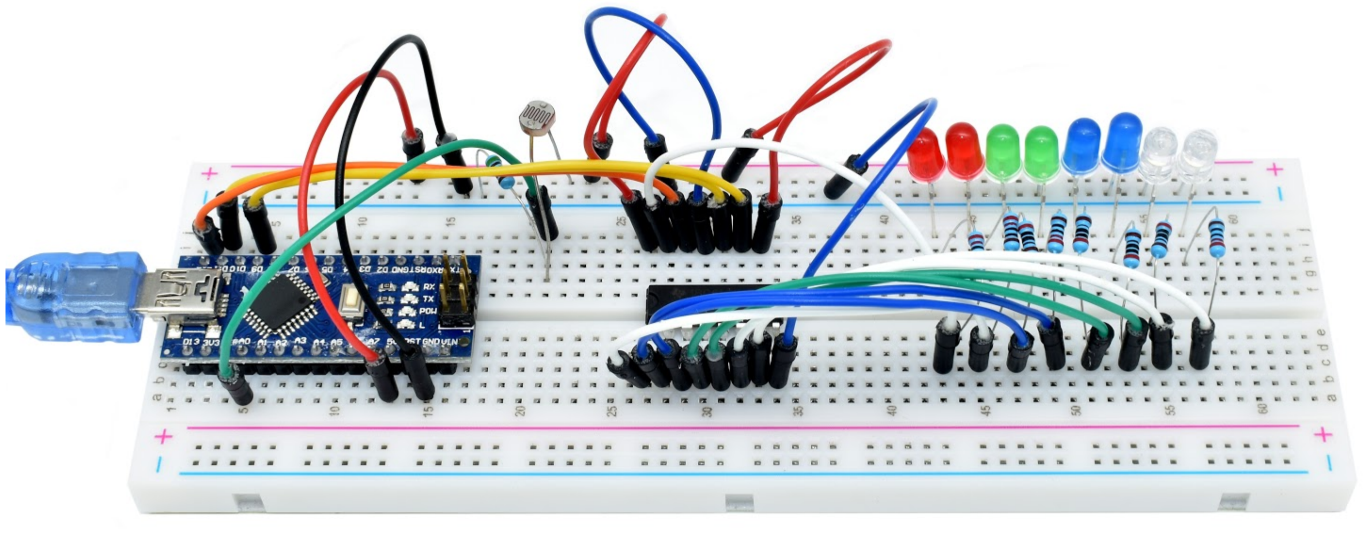
After wiring, find and open the program located in the folder Lesson 17 Photocell, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

Firstly, we see that, instead of 'potPin', we have modified the analog pin's name to be 'lightPin', as we do not have a pot connected any more. The other essential change to the code is the line where we determine the number of LEDs to light:

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

In this situation, we will divide the raw reading by 57 instead of 114, half as much as we did previously with the pot to split it into nine areas, from no LEDs lit to all eight lit. We also have to note the fixed 1 kΩ resistor. Thus, when the photocell has the same resistance as the resistor, the raw reading will be 1023 / 2 = 511, which means that all the LEDs are being lit and that 'numLEDSLit' will be 8.

17.6 Example picture



Lesson 18: Thermometer

18.1 Overview

This lesson will teach you how to show the temperature in Serial Monitor.

18.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x 10K ohm resistors
1 x Thermistor
3 x M-M wires (Male to Male jumper wires)

18.3 Component Introduction

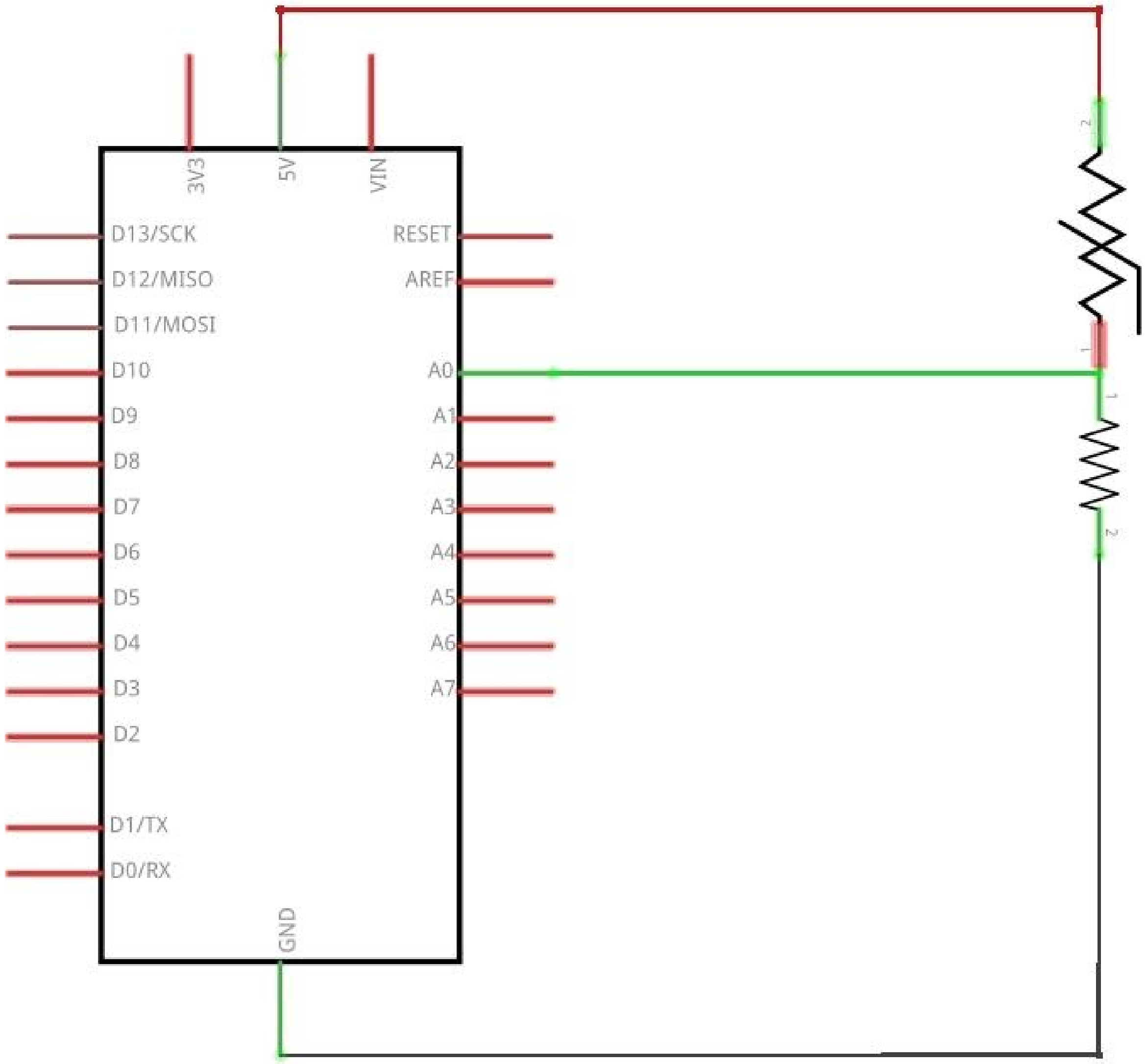
Thermistor

The thermistor is simply a thermal resistor, it changes its resistance according to the temperature. Theoretically, all resistors are thermistors, because their resistance adjusts to some extent with temperature, but it is generally too little to determine. On the other hand, with thermistors, the resistance changes intensely with temperature, say, 100 ohms or more per degree.

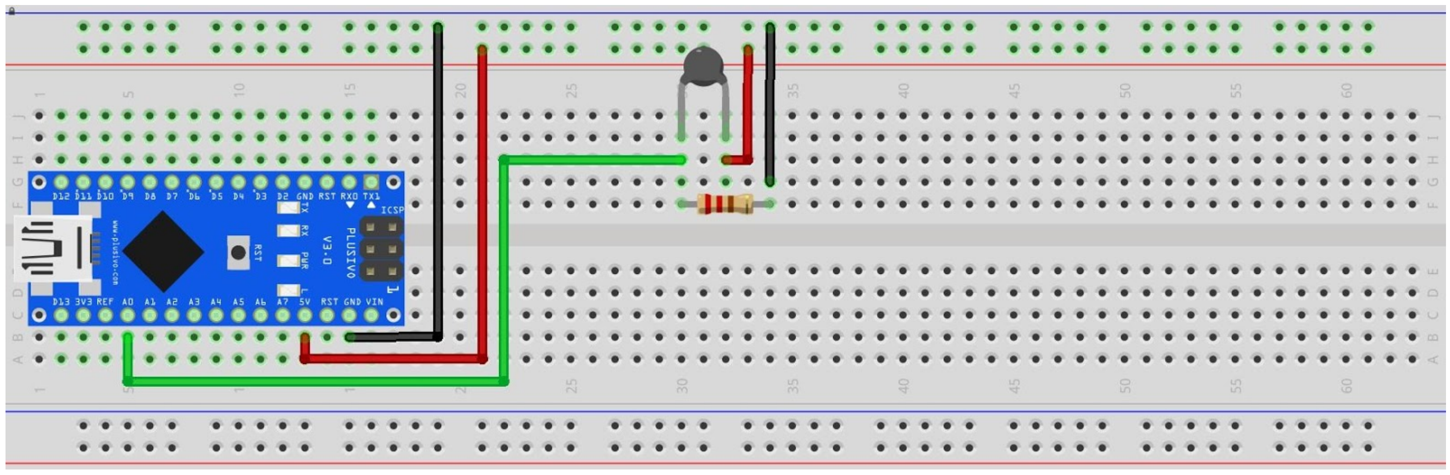
There are two kinds of thermistors, PTC (positive temperature coefficient) and NTC (negative temperature coefficient). For the most part, NTC sensors are used for temperature measurements, while PTCs are used as resettable fuses. The working mechanism is that the temperature increases the resistance, so the more current passes through them, the more they heat up and resist. For their flexibility, they prove to be very useful for protecting circuits!

18.4 Connection

Schematic



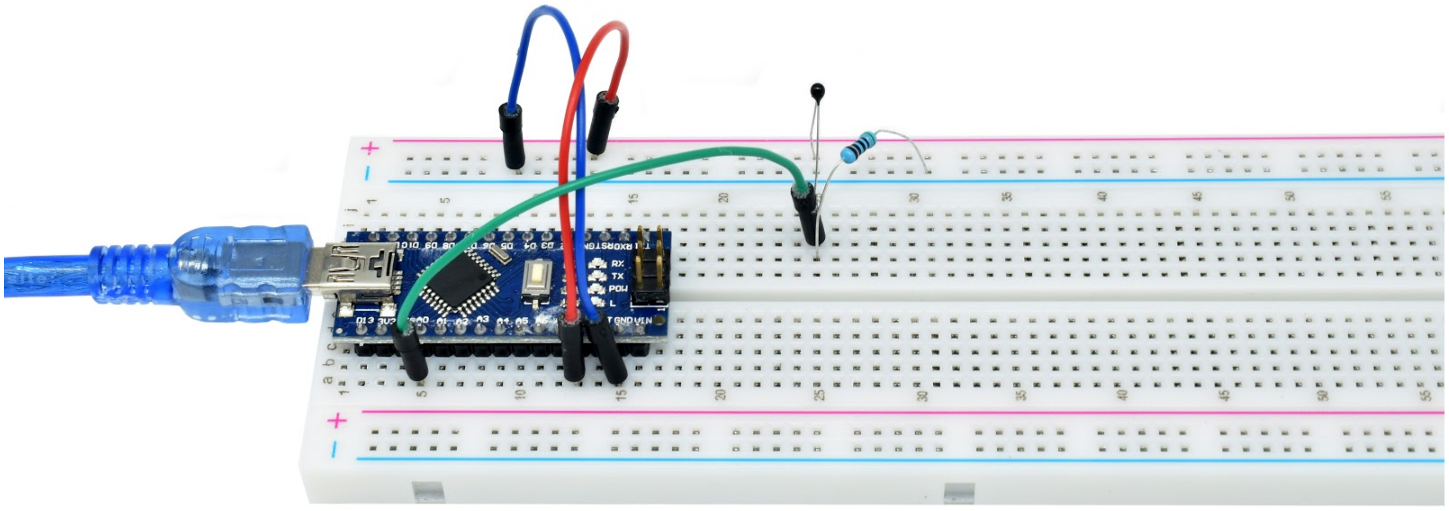
Wiring diagram



18.5 Code

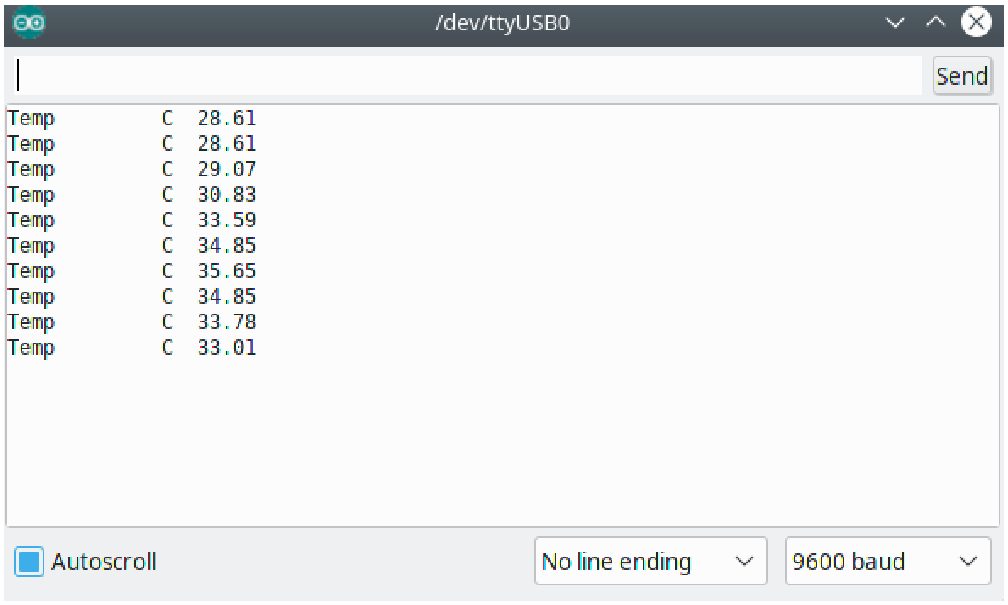
After wiring, find and open the program located in the folder Lesson 18 Thermometer, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

18.6 Example picture



Open the monitor so we can see the following values:

Press the Serial Monitor button to turn it on. The serial monitor is comprehensively introduced in Lesson 2.



```
Temp        C  28.61
Temp        C  28.61
Temp        C  29.07
Temp        C  30.83
Temp        C  33.59
Temp        C  34.85
Temp        C  35.65
Temp        C  34.85
Temp        C  33.78
Temp        C  33.01
```

Lesson 19: LED Bar with 10 Segments

19.1 Overview

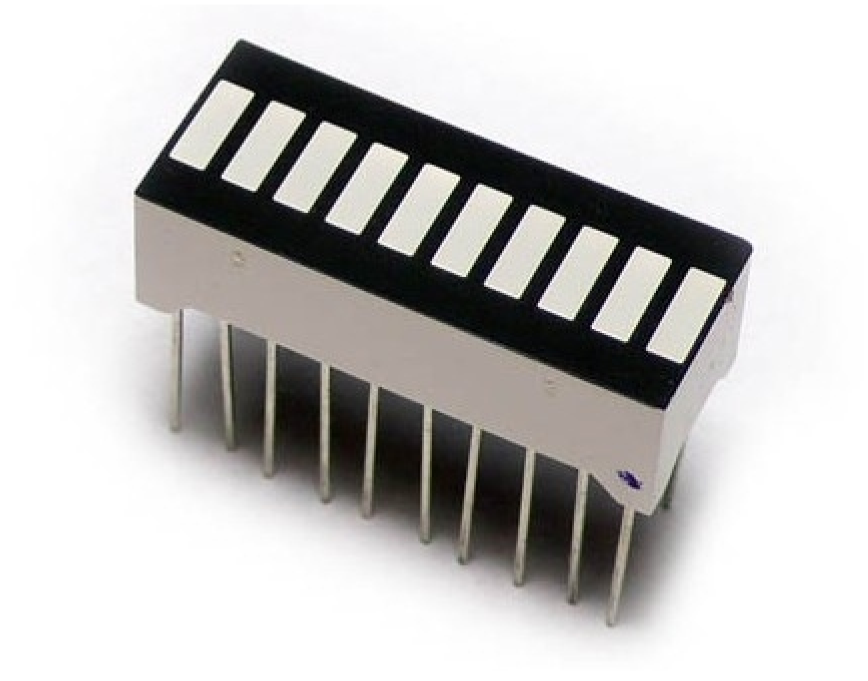
In this lesson we learn how to use LED Bar and how to increase it one cell by one.

19.2 Components Required

1 x Nano R3
1 x Breadboard 830p
10 x 220 ohm resistors
11 x M-M wires (Male to Male jumper wires)
1 x LED Bar with 10 Segments

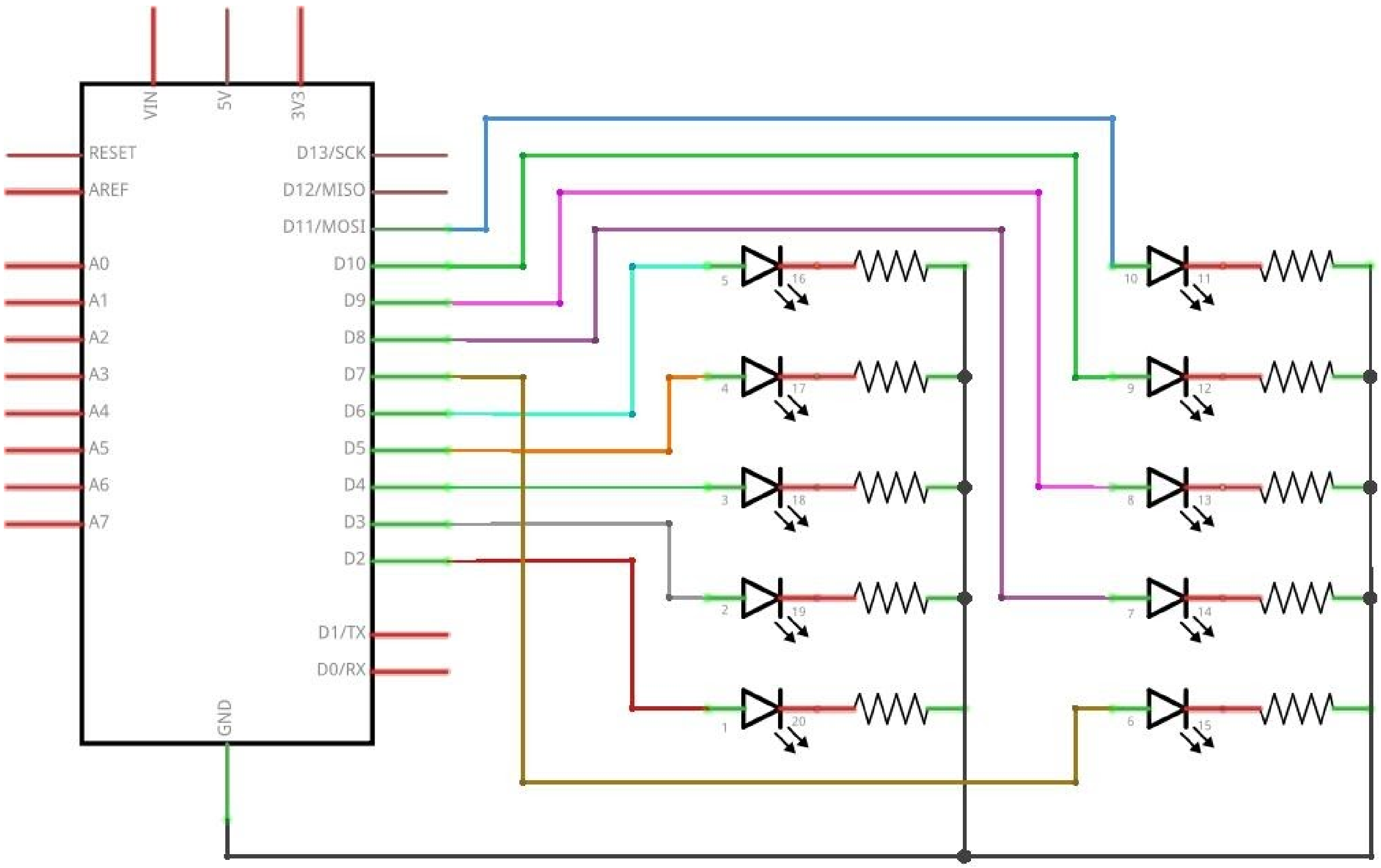
19.3 Component Introduction

It contains 10 individual LEDs, each one of them has Anode and Cathode so we have 20 pins.

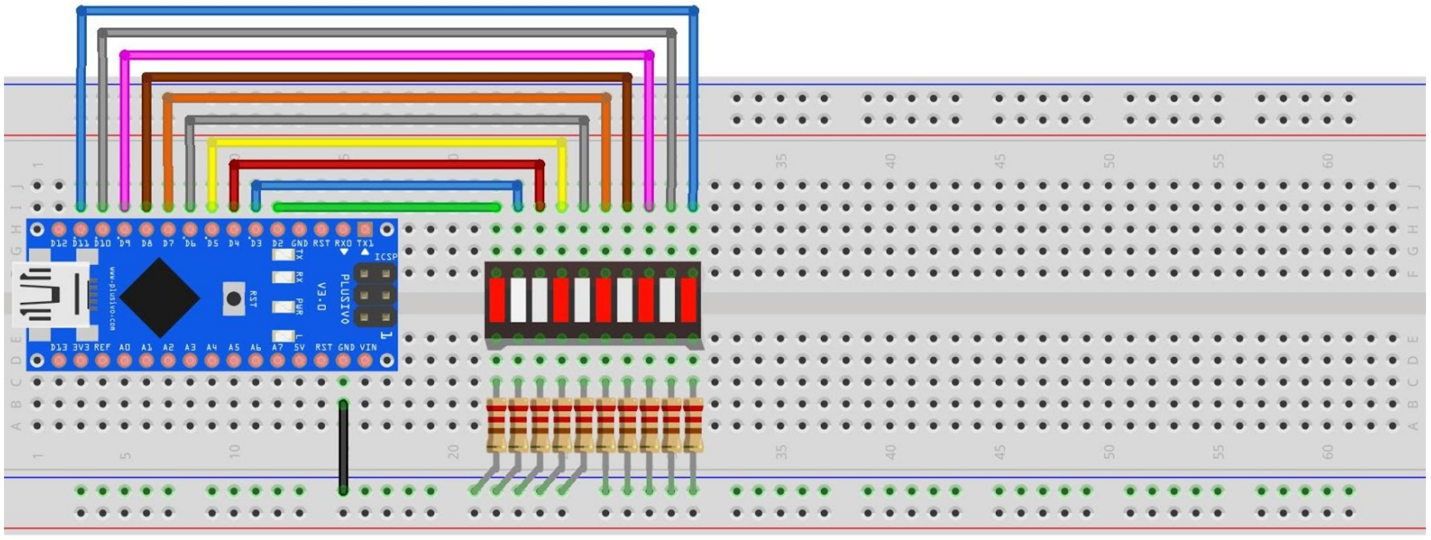


19.4 Connection

Schematic



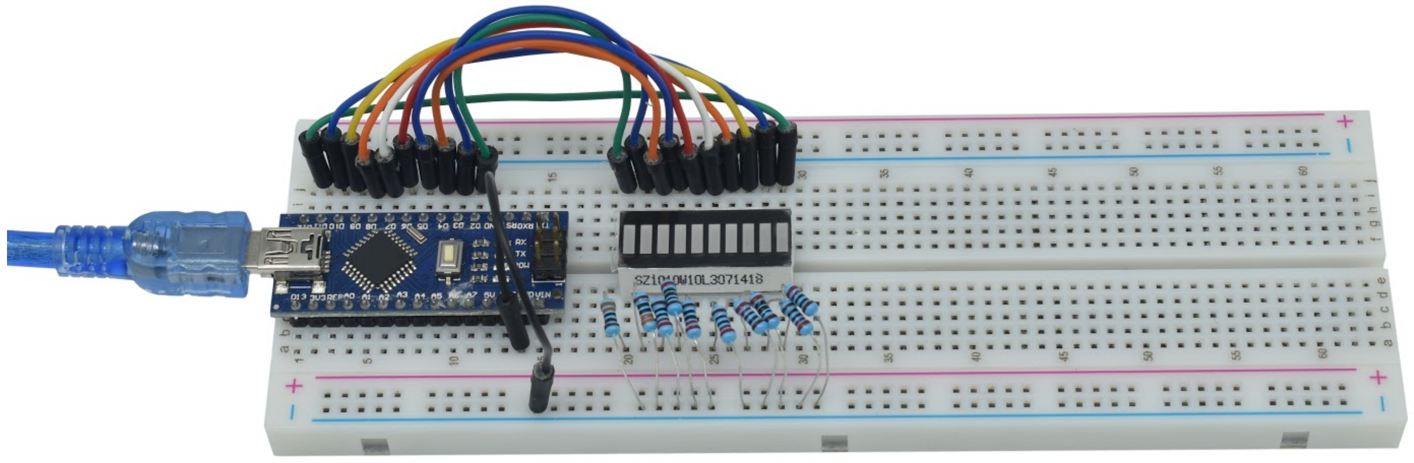
Wiring diagram



19.5 Code

After wiring, find and open the program located in the folder Lesson 19: LED Bar with 10 Segments, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

19.6 Example picture



Lesson 20: LED Bar with 74hc595

20.1 Overview

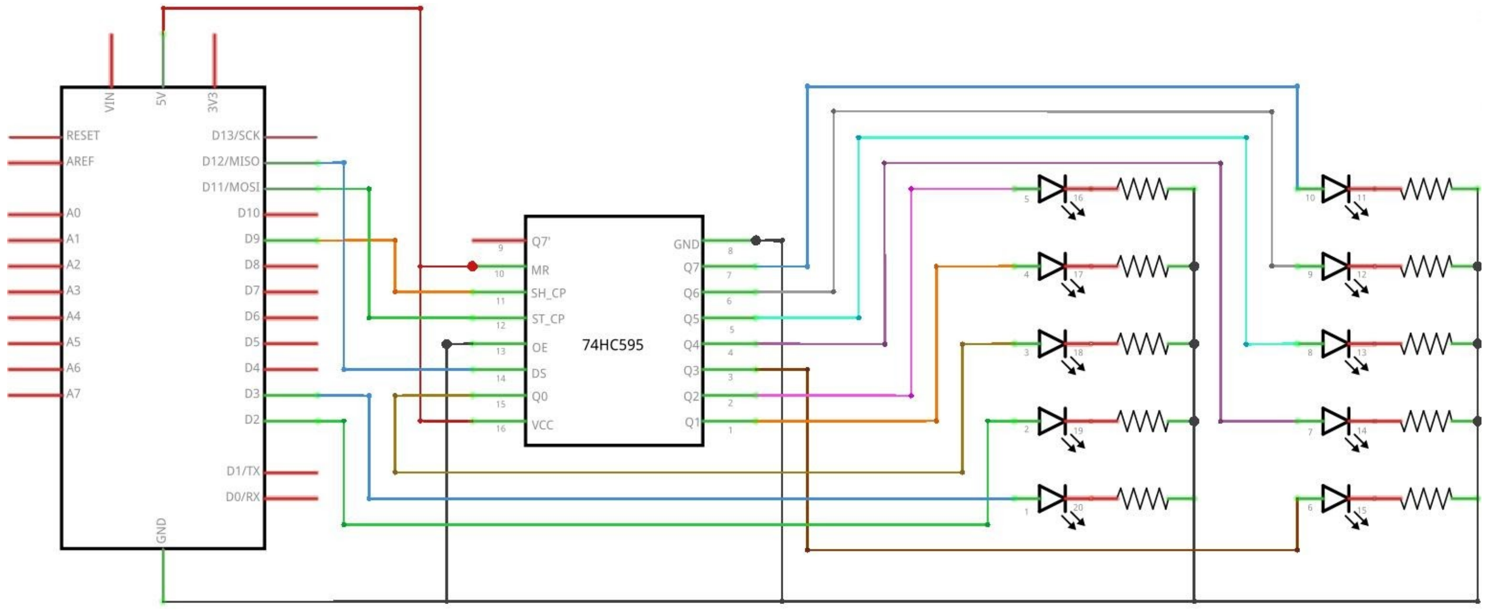
This lesson is the same as the previous one, but we are going to add 74hc595 IC so we will use only 5 Outputs from Nano.

20.2 Components Required

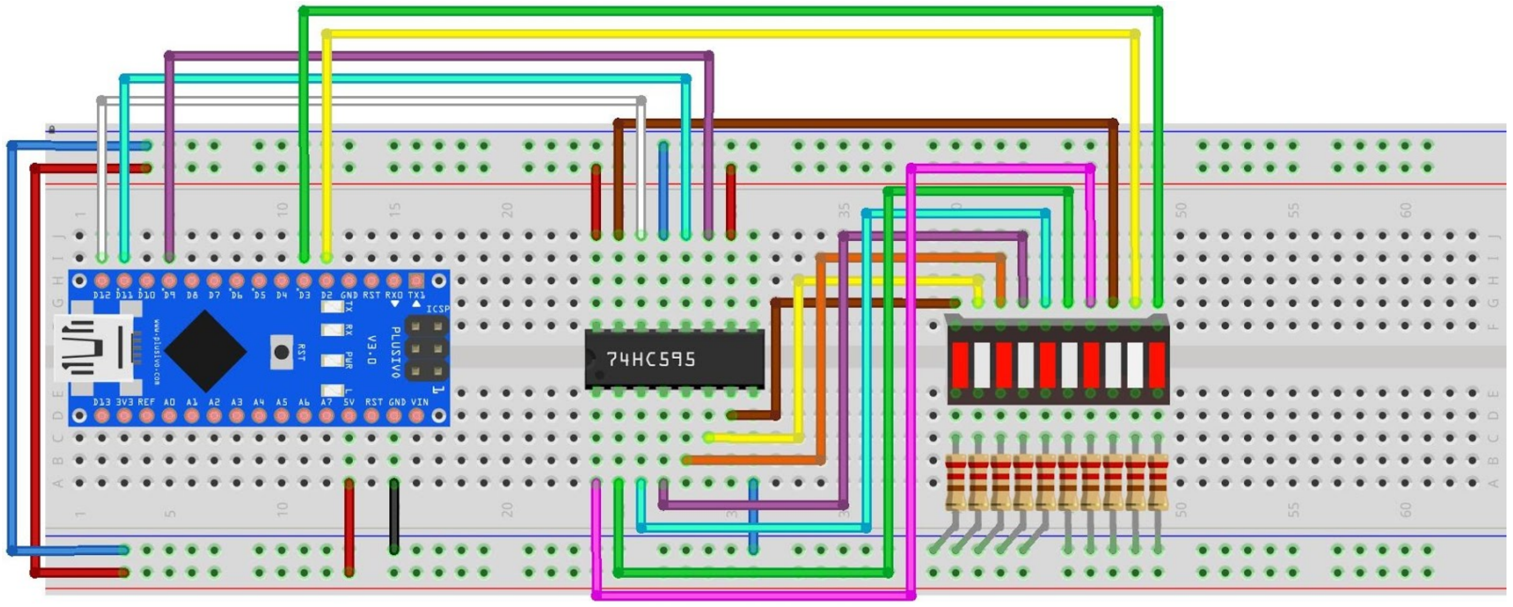
1 x Nano R3
1 x Breadboard 830p
10 x 220 ohm resistors
19 x M-M wires (Male to Male jumper wires)
1 x LED Bar with 10 Segments
1 x 74hc595 IC

20.3 Connection

Schematic



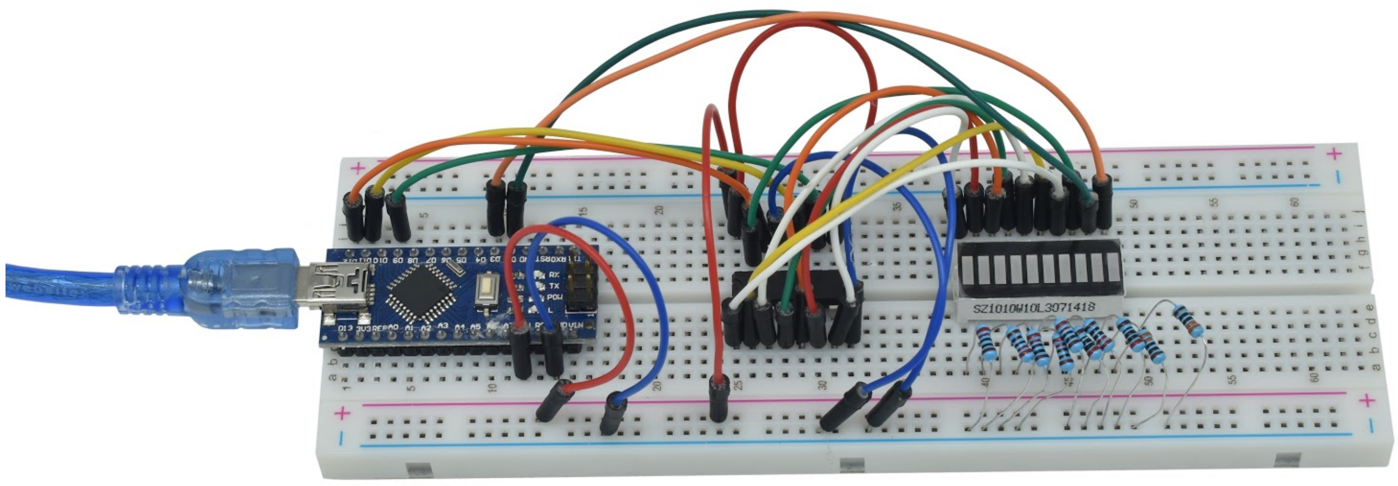
Wiring diagram



20.4 Code

After wiring, find and open the program located in the folder Lesson 20: LED Bar with 74hc595, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

20.5 Example picture



Lesson 21: LED Bar with Potentiometer

21.1 Overview

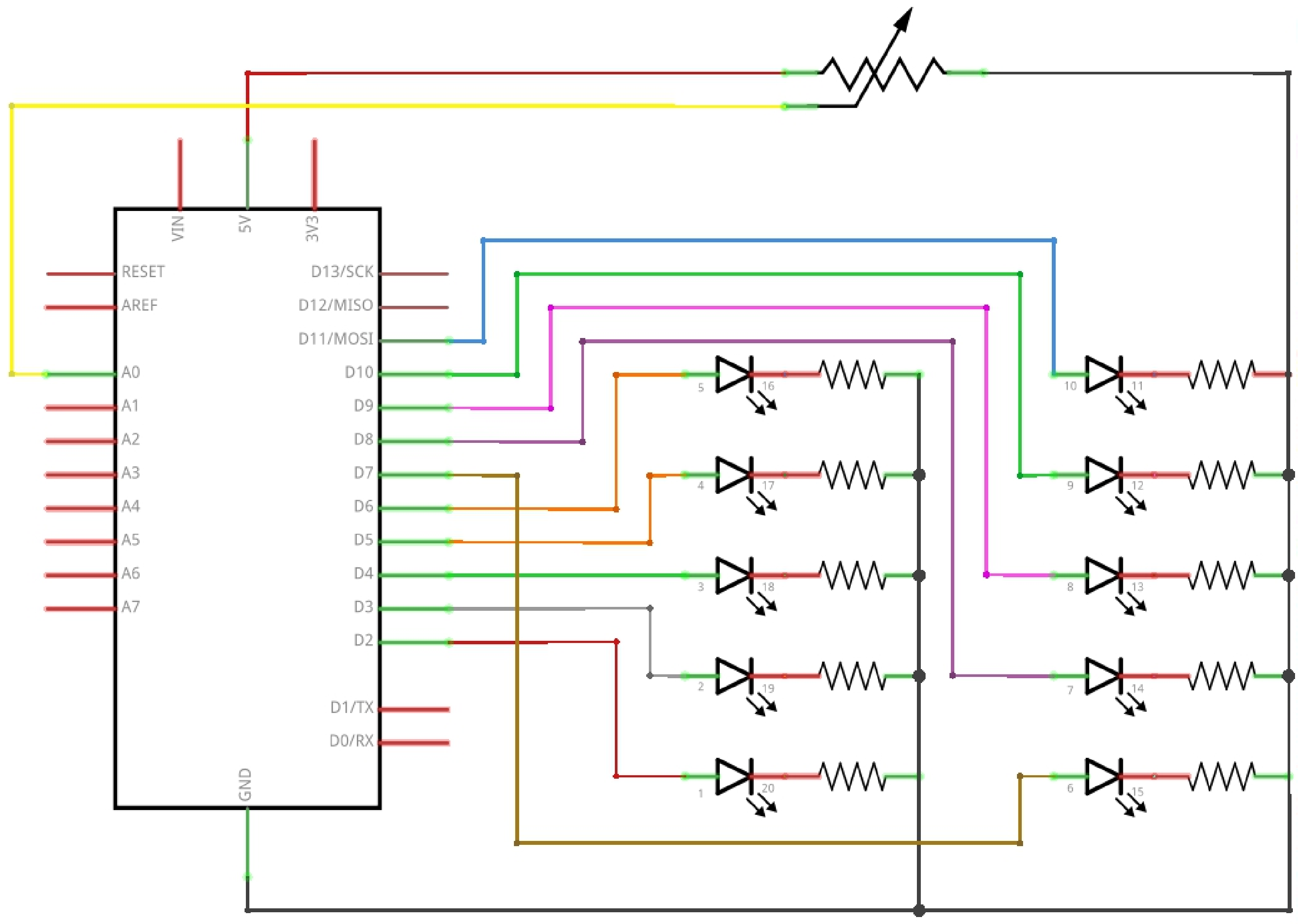
In this lesson we are going to use 10K Potentiometer to light up the LEDs.

21.2 Components Required

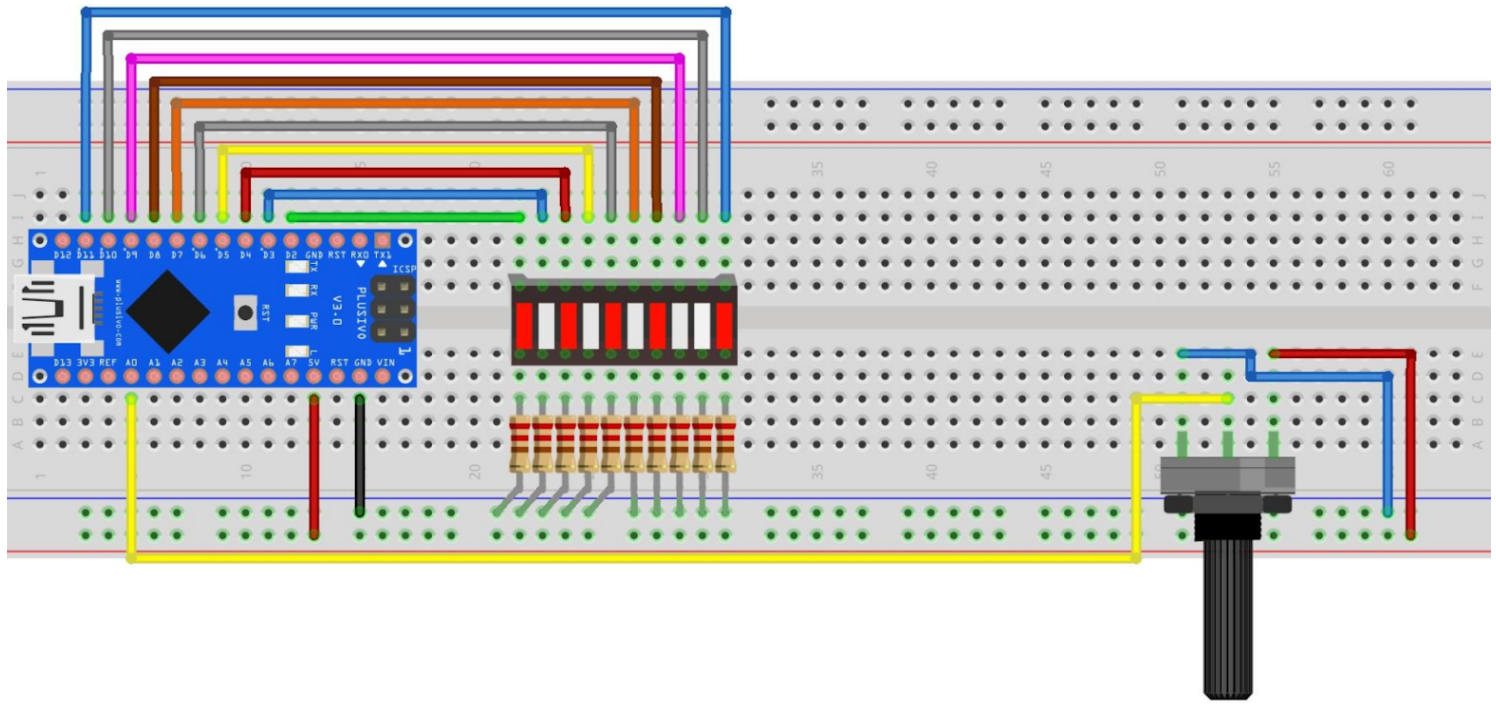
1 x Nano R3
1 x Breadboard 830p
10 x 220 ohm resistors
15 x M-M wires (Male to Male jumper wires)
1 x LED Bar with 10 Segments
1 x Potentiometer 10kΩ

21.3 Connection

Schematic



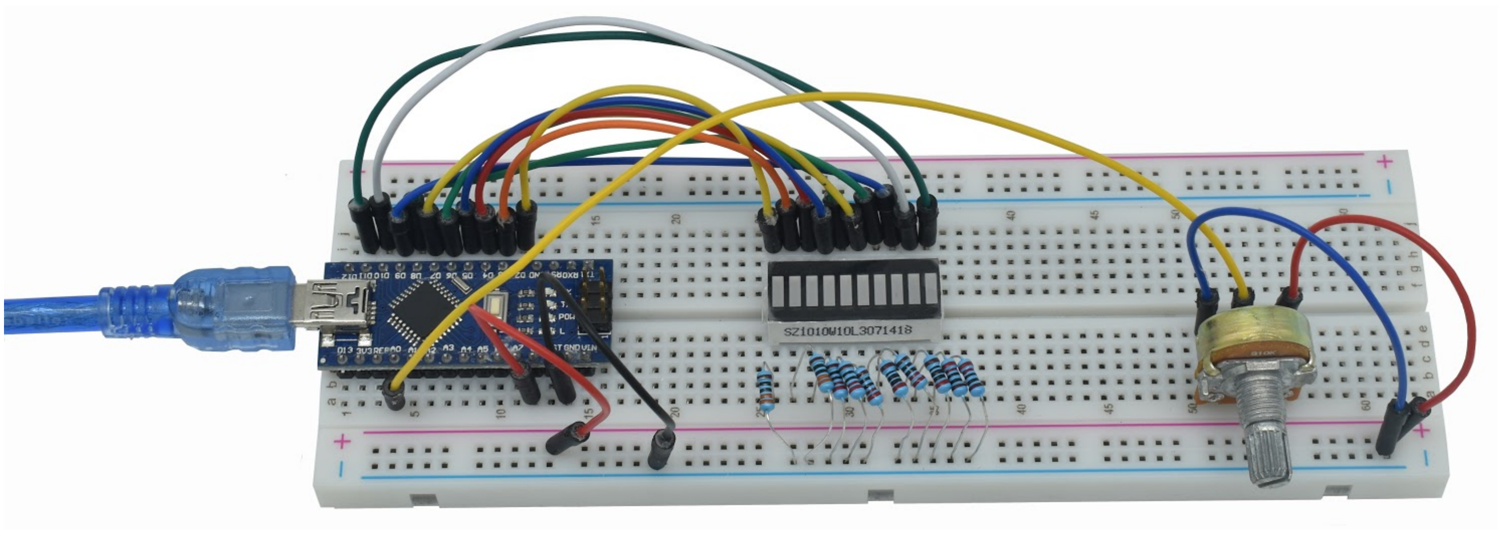
Wiring diagram



21.4 Code

After wiring, find and open the program located in the folder Lesson 21: LED Bar with Potentiometer, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

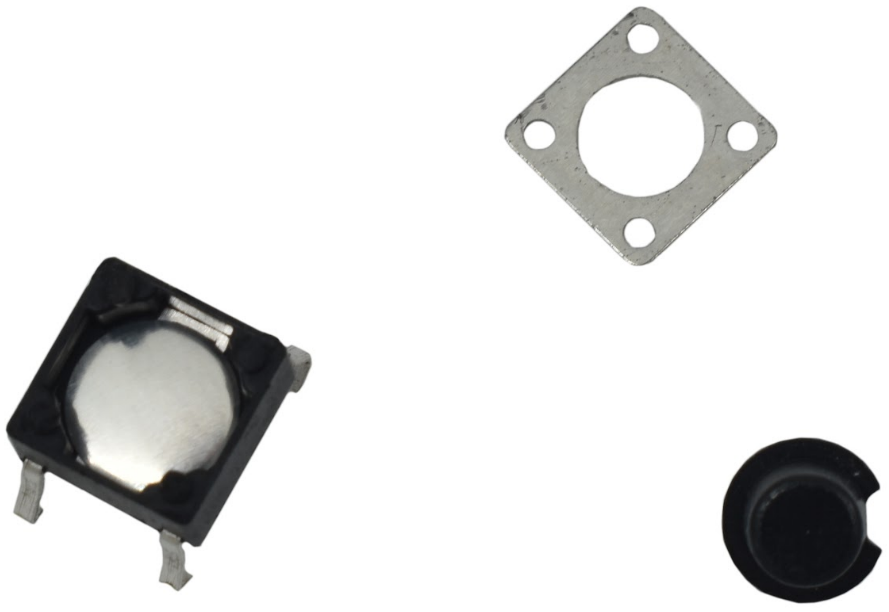
21.5 Example picture



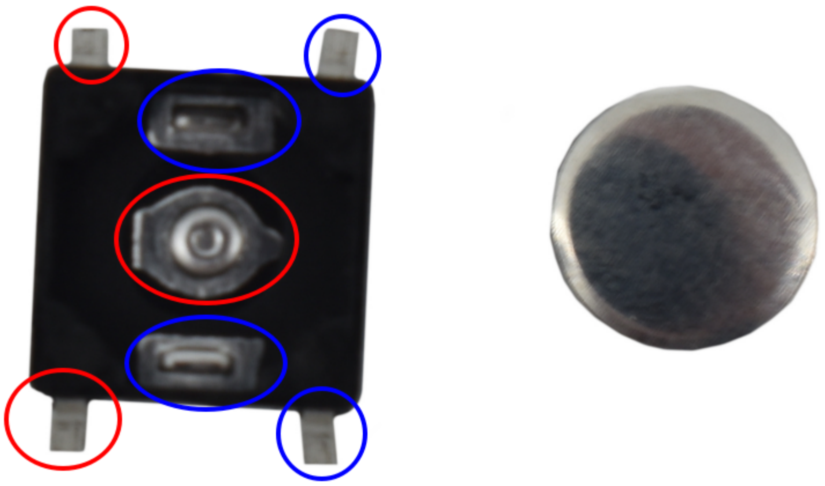
Lesson 22: Debouncing

22.1 Overview

To understand this lesson, let us take a push button off and take a look inside of it.



There is a metal domed disk inside of it so when you press on it, the disk makes a contact between the legs. As you can see below, the first contact is red, and the second one is blue.



The disk is boncy, so when you press on it, it will not make contact directly, indeed it will make the contact on and off many times before setting down.

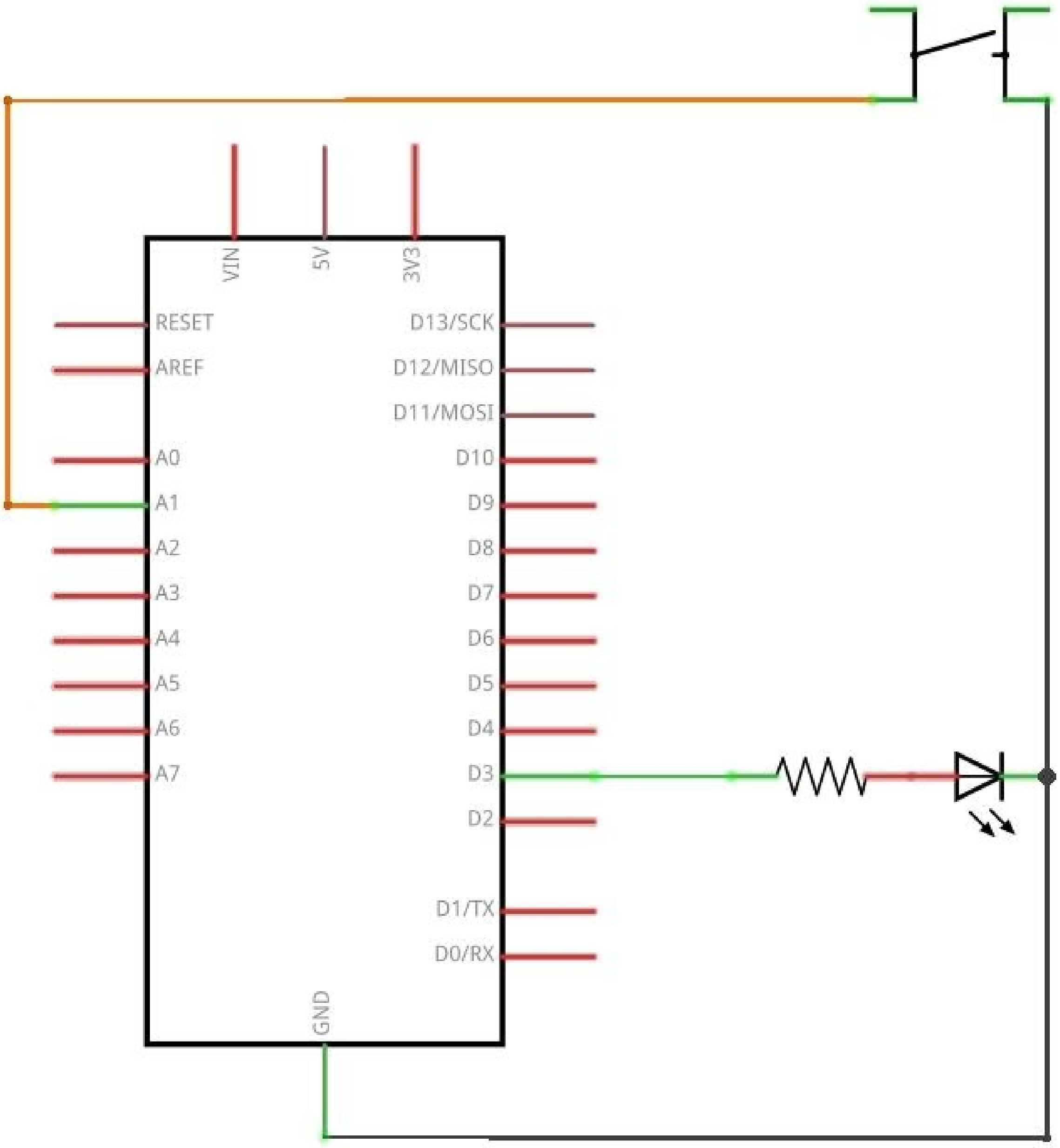
In this lesson we are going to use code to debounce a push button.

22.2 Components Required

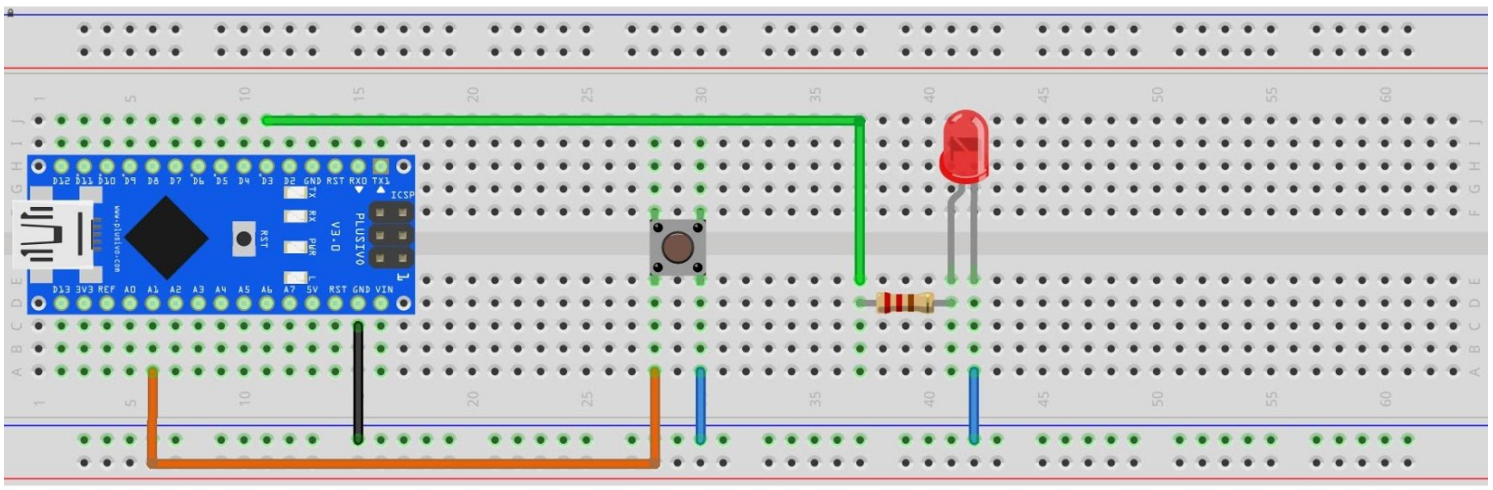
1 x Nano R3
1 x Breadboard 830p
1 x 5mm red LED
1 x 220 ohm resistor
1 x push Button
5 x M-M wires (Male to Male jumper wires)

22.3 Connection

Schematic



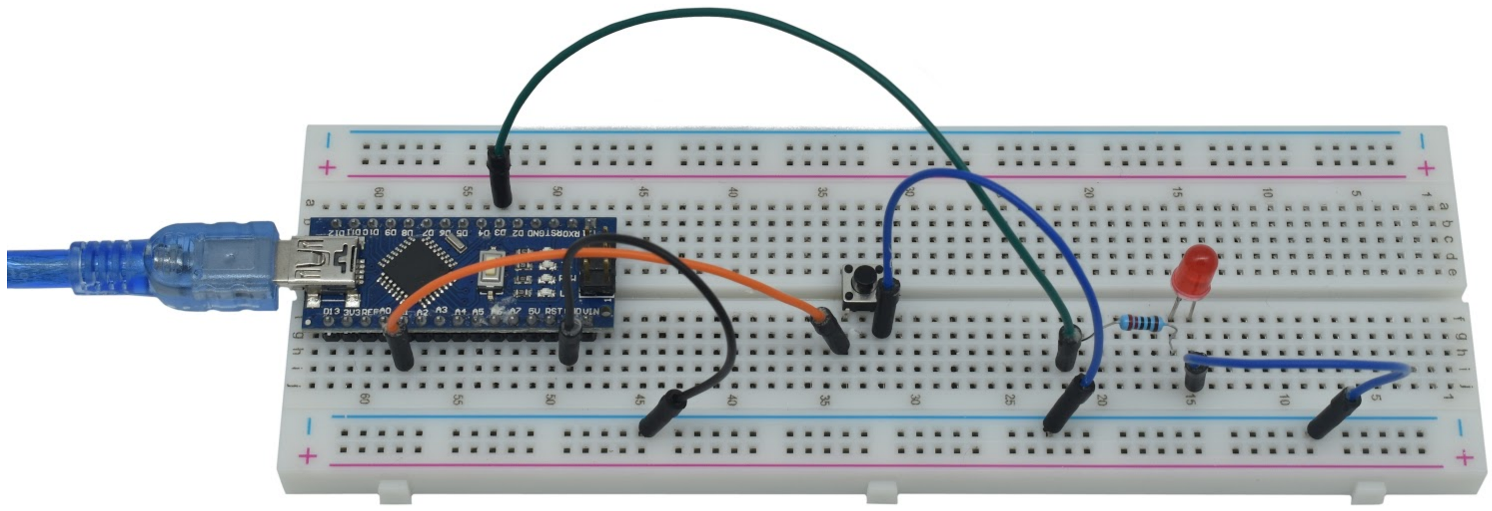
Wiring diagram



22.4 Code

After wiring, find and open the program located in the folder Lesson 22: Debouncing, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

22.5 Example picture



Lesson 23: LED Bar with 2 Push buttons

23.1 Overview

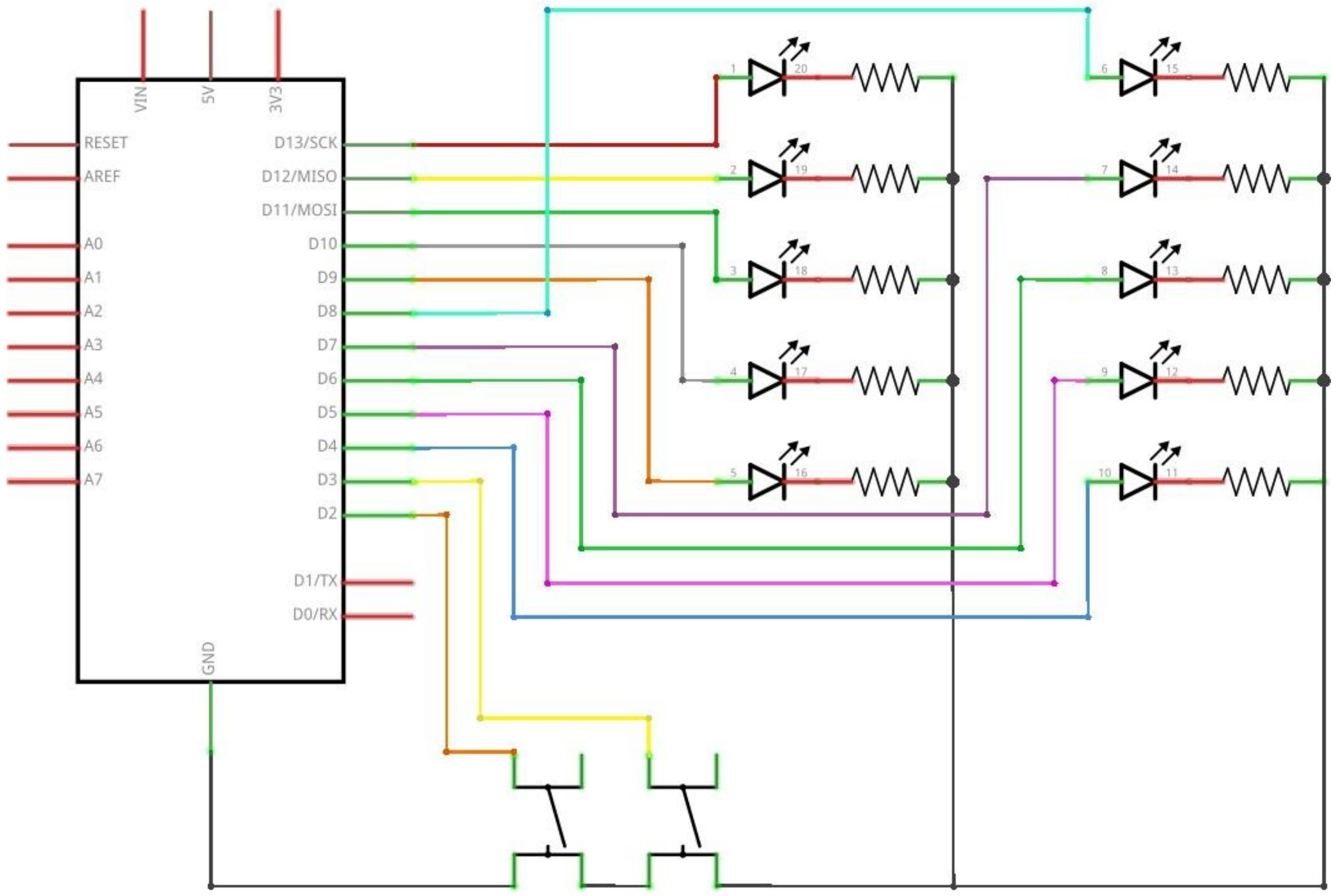
In this lesson we are going to use 2 push buttons to make a counter with the LED Bar.

23.2 Components Required

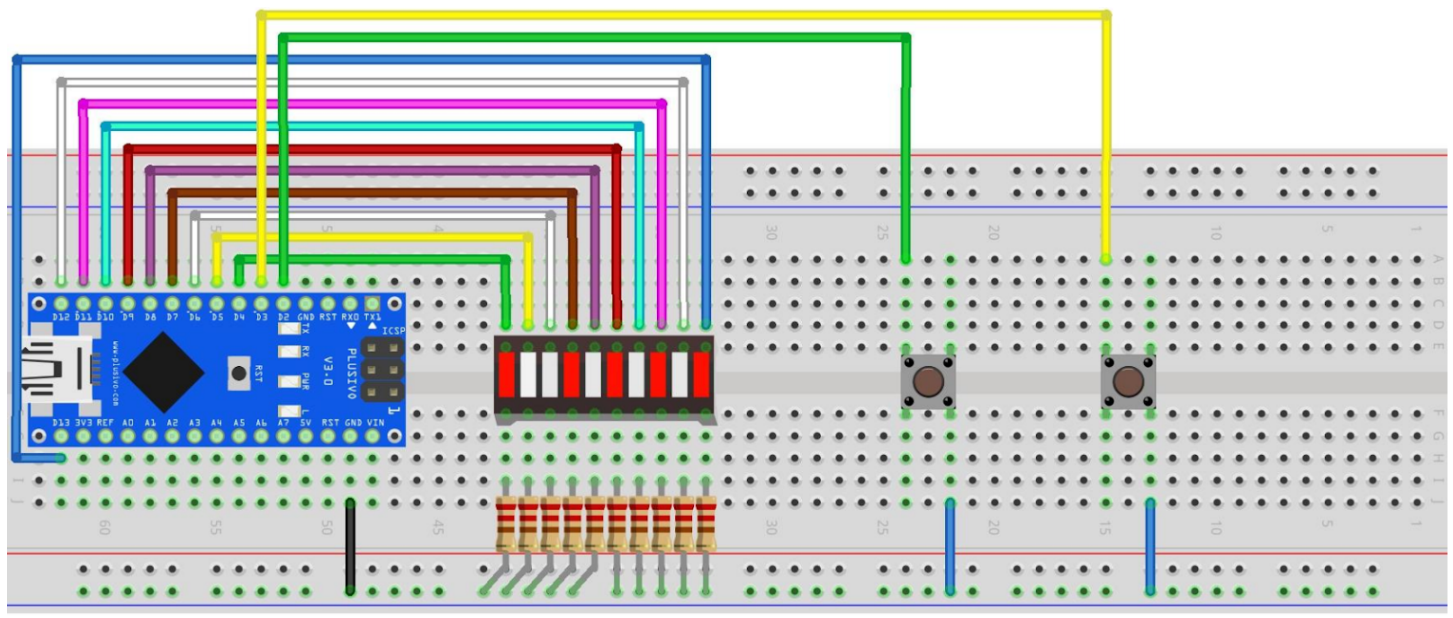
1 x Nano R3
1 x Breadboard 830p
10 x 220 ohm resistors
15 x M-M wires (Male to Male jumper wires)
1 x LED Bar with 10 Segments
2 x push Button

23.3 Connection

Schematic



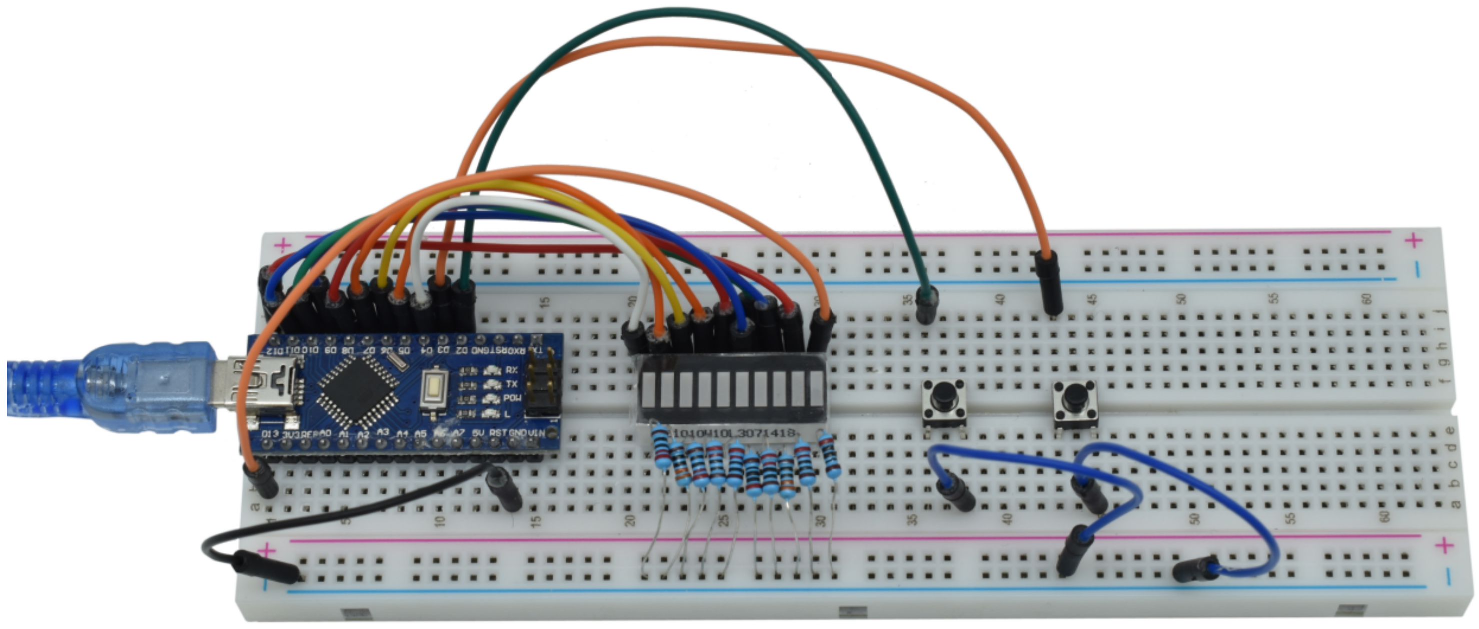
Wiring diagram



23.4 Code

After wiring, find and open the program located in the folder Lesson 23: LED Bar with 2 Push buttons, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

23.5 Example picture



Lesson 24: Four Digits Seven Segments Display

24.1 Overview

This lesson will be teaching you the basics on using a 4-digit 7-segment display. For 1-digit 7-segment display, please note that common cathode pin connects to ground while common anode pin connects to the power source.

For this display, the one that controls which digit to show is the common anode or common cathode pin. Although just one digit is working at a time, you are capable of seeing all the numbers displayed because the flashing is so quick that you barely see the interruptions, according to the principle of Persistence of Vision.

24.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x 74HC595 IC
1 x 4 Digit 7-Segment Display
8 x 220 ohm resistors
1 x Potentiometer 10kΩ

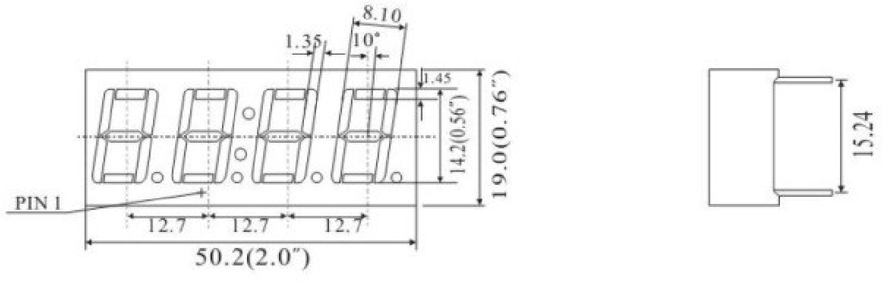
32 x M-M wires (Male to Male jumper wires)

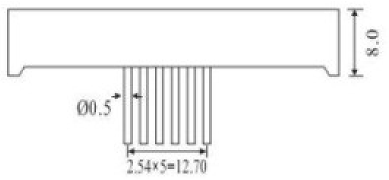
24.3 Component Introduction

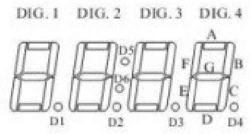
Four Digits Seven segments display

Package Dimensions

CPS05643AB

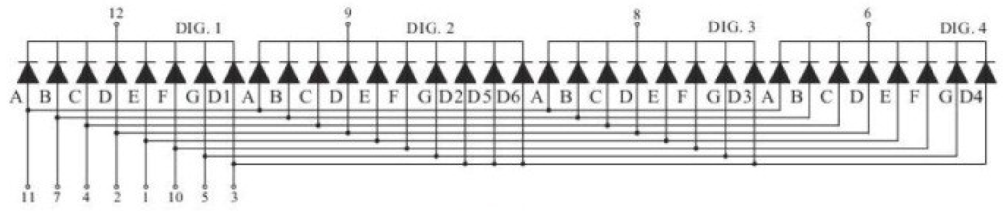






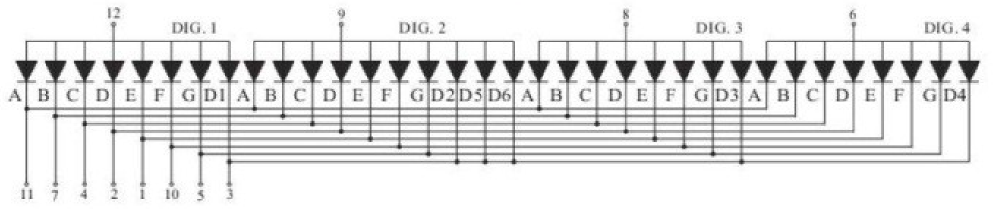
UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

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



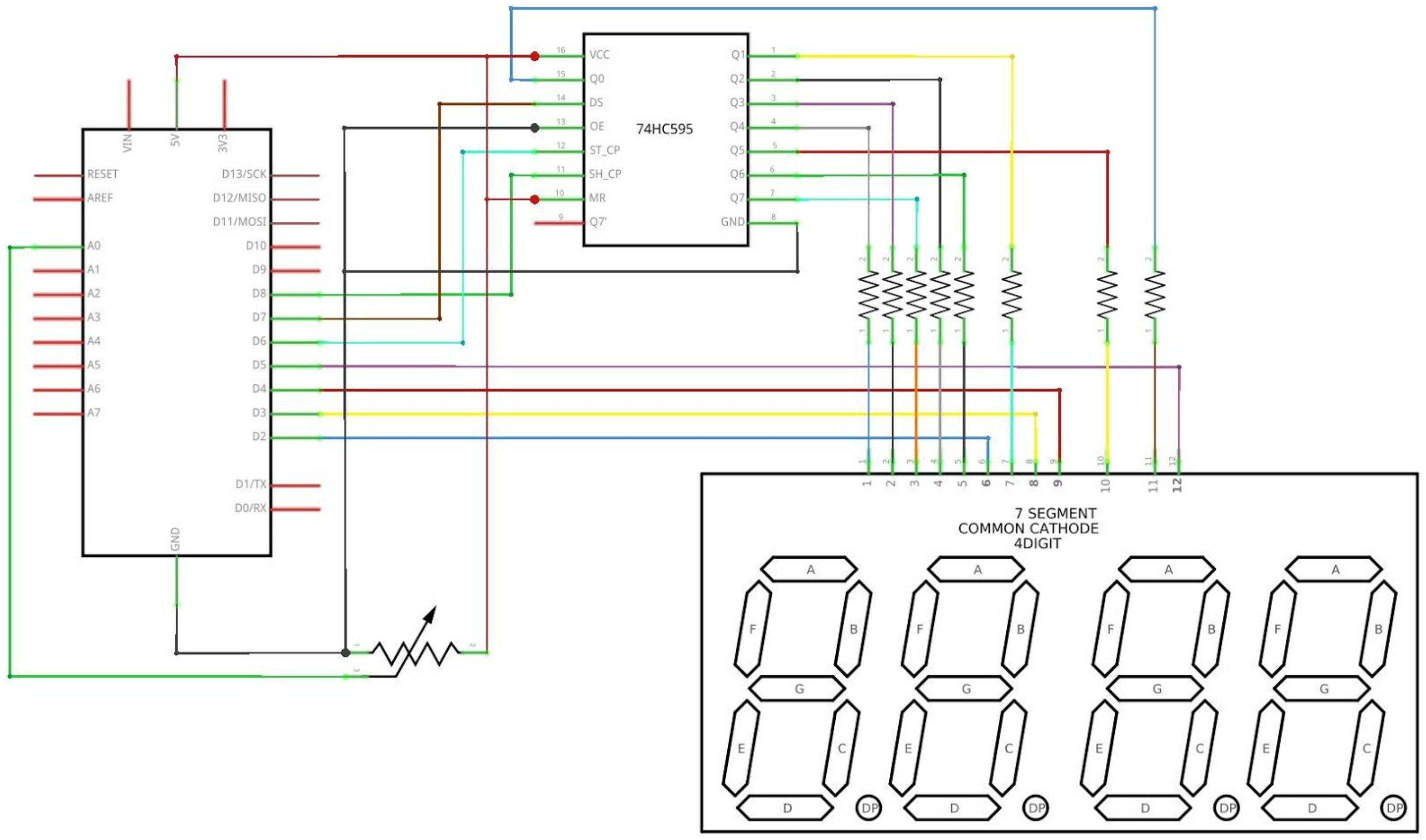
12
DIG. 1
9
DIG. 2
8
DIG. 3
6
DIG. 4
A B C D E F G D1 A B C D E F G D2 D5 D6 A B C D E F G D3 A B C D E F G D4
11 7 4 2 1 10 5 3

5643B

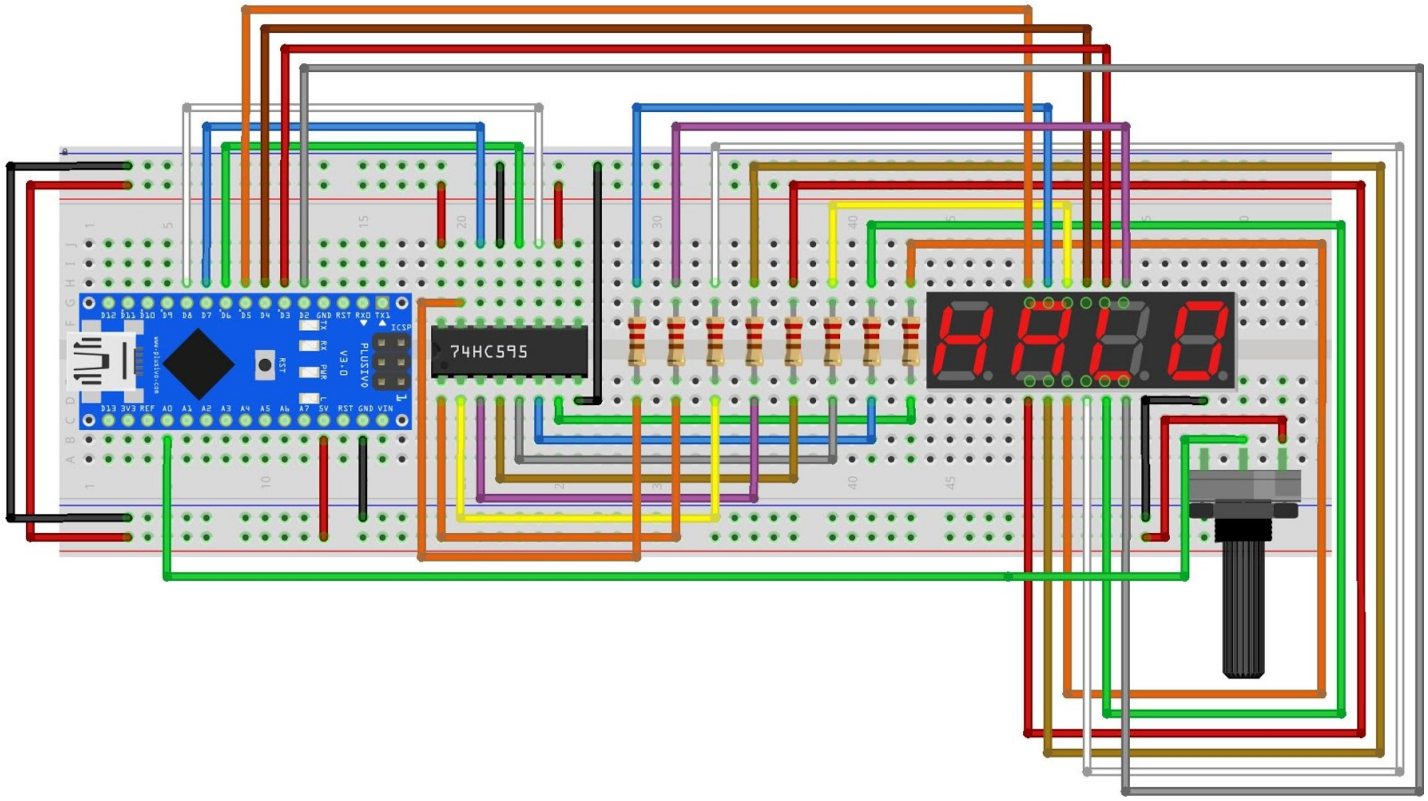
Four Digits Displays Series

24.4 Connection

Schematic



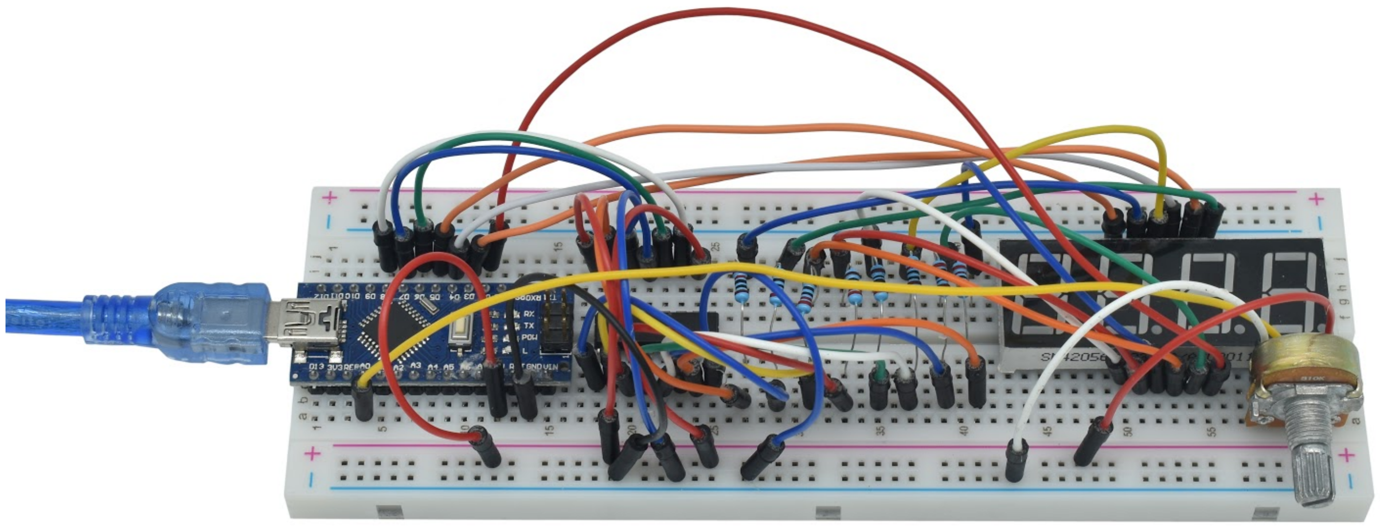
Wiring diagram



24.5 Code

After wiring, find and open the program located in the folder Lesson 24 Four Digits Seven Segments Display, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

24.6 Example picture



Lesson 25: Thermometer Display on 4 Digit Seven Segment

25.1 Overview

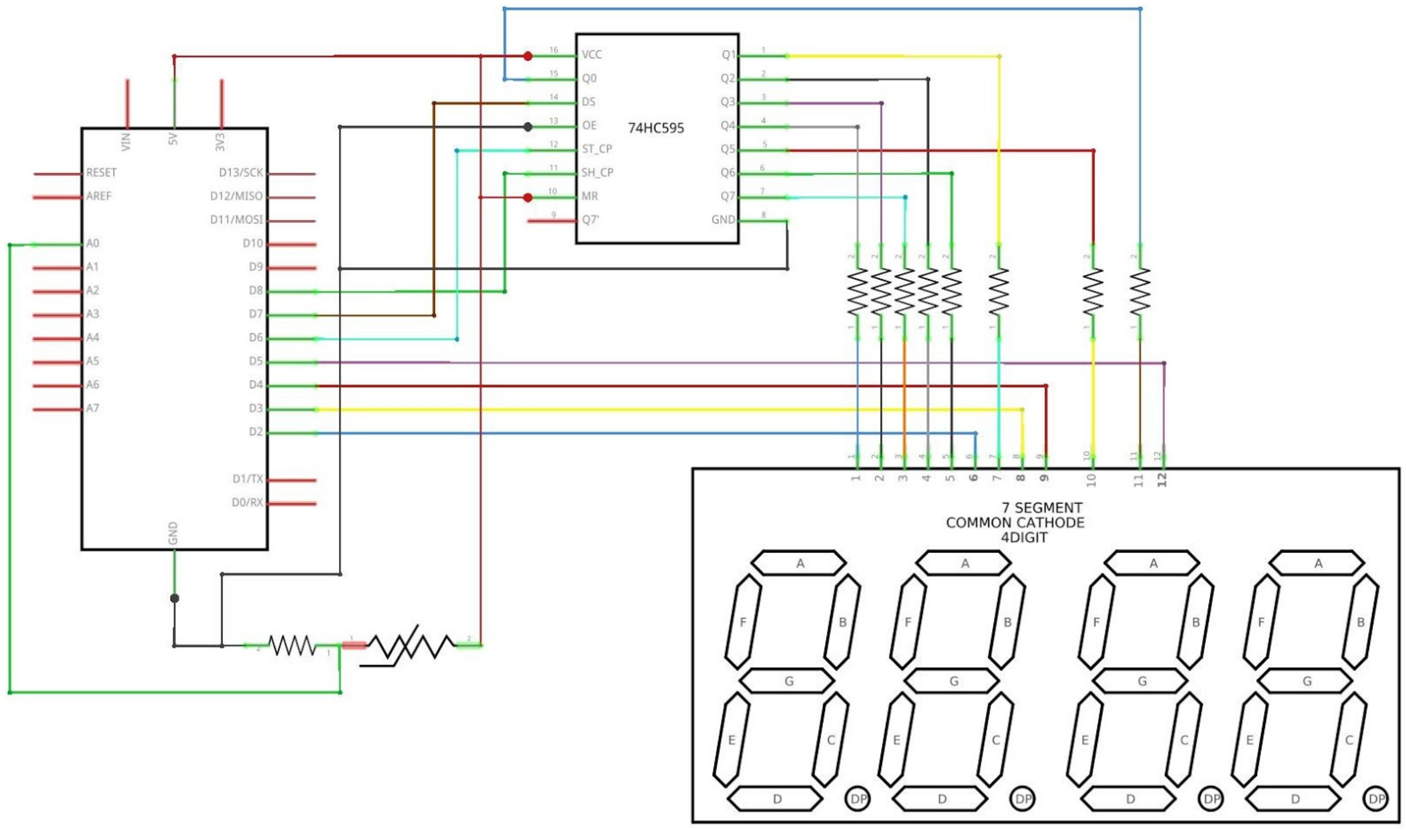
This lesson based on the previous lesson, instead of showing the voltage on 4 digit seven segment, we will show the temperature using thermistor.

25.2 Components Required

1 x Nano R3
1 x Breadboard 830p
1 x 74HC595 IC
1 x 4 Digit 7-Segment Display
8 x 220 ohm resistors
1 x 10k ohm resistors
1 x Thermistor
30 x M-M wires (Male to Male jumper wires)

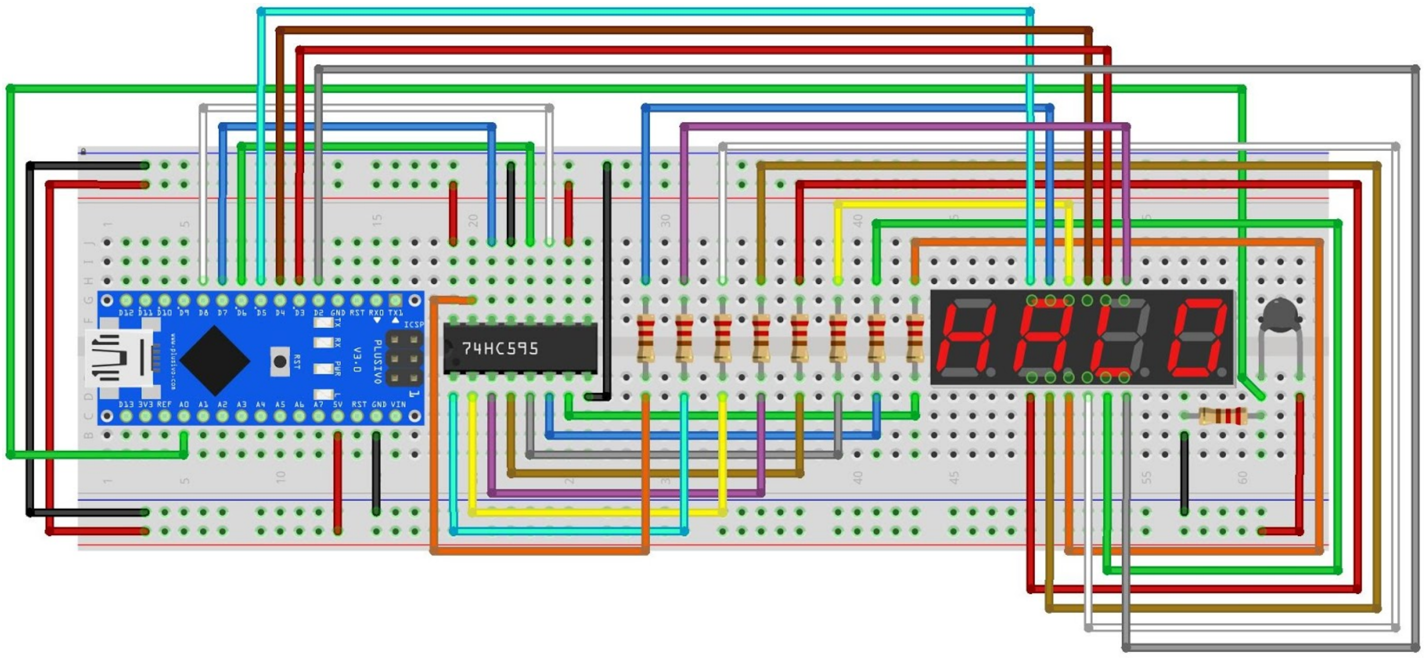
25.3 Connection

Schematic



g

Wiring diagram



25.4 Code

After wiring, find and open the program located in the folder Lesson 25 Thermometer Display on 4 Digit Seven Segment, and UPLOAD the code. If there are any errors, see Lesson 3 for details about program uploading.

25.5 Example picture