Preface

Our Company

SDLSmartLabs is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing.

No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

This is an entry-level learning kit for Arduino. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

Parts List

UNO R3

USB Cable

RFID Module

4X4 Keypad

5V Step Motor

Moto Drivers

74HC595

SW-520D

Flame

LM35

IR Receiver

CDS

LED

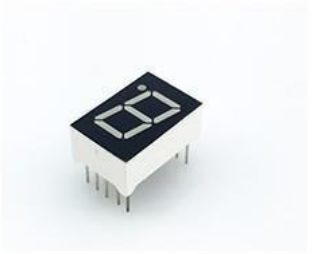
5V Relay

Big Sound

Key Switch*4

9V battery holder

830 breadboard



7segment

4 7segment

8*8 matrix

Jumper Wire

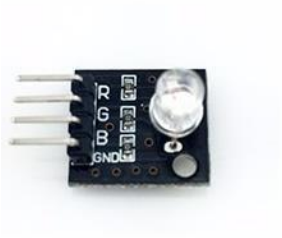
F-MDupont wire

Water

B10K Variable

2.54mm pin

Clock



RGB

DHT11

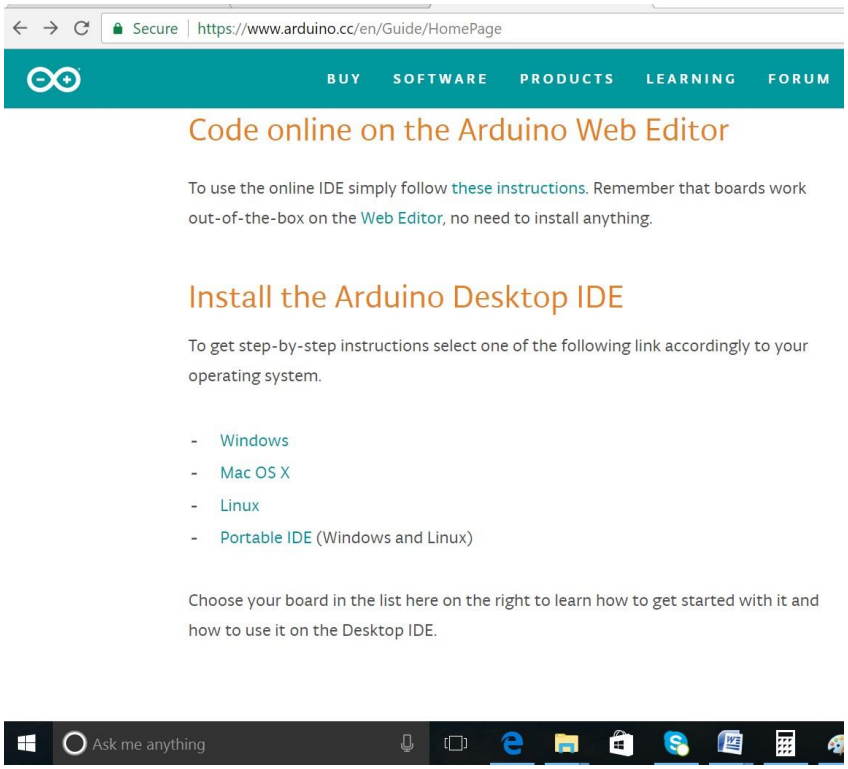
Table of Contents

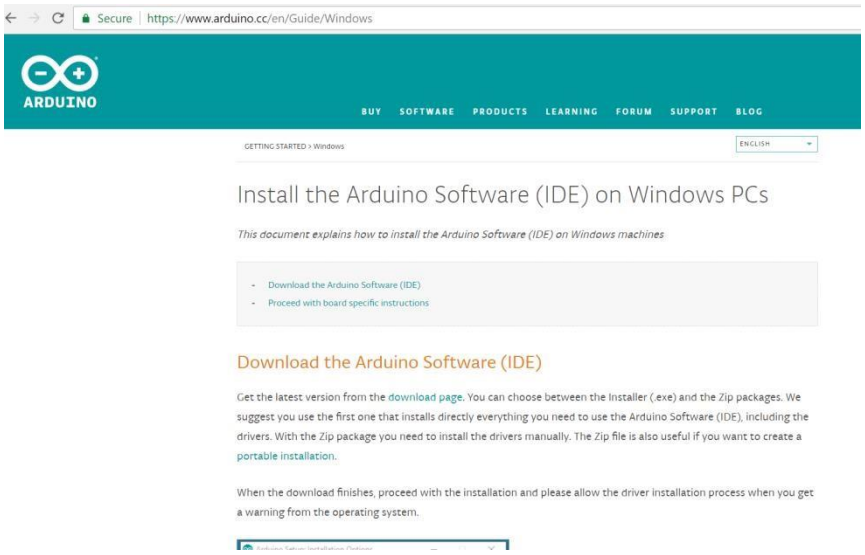
Introduction: Installing IDE

STEP-1: Download the Arduino IDE (Integrated Development Environment)

Access the Internet:
In order to get your Arduino up and running, you'll need to download some software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the Arduino to do exactly what you want. It's like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

www.arduino.cc/en/Main/Software

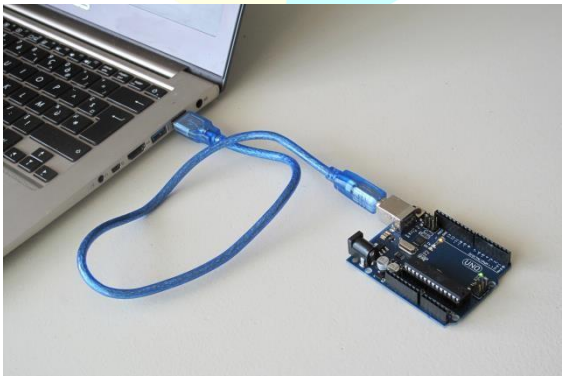




For different operating system platforms, the way of using Arduino IDE is different. Please refer to the following links: Windows User : http://www.arduino.cc/en/Guide/Windows Mac OS X User : http://www.arduino.cc/en/Guide/MacOSX Linux User : http://playground.arduino.cc/Learning/Linux For more detailed information about Arduino IDE, please refer to the following link: http://www.arduino.cc/en/Guide/HomePage

STEP-2: Connect your Arduino Uno to your Computer:

Use the USB cable provided in the kit to connect the Arduino to one of your computer's USB inputs.



STEP-3: Install Drivers

Depending on your computer's operating system, you will need to follow specific instructions. Please go to the URLs below for specific instructions on how to install the drivers onto your Arduino Uno.

Windows Installation Process:

Go to the web address below to access the instructions for installations on a Windows-based computer.

http://arduino.cc/en/Guide/Windows

Macintosh OS X Installation Process:
Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

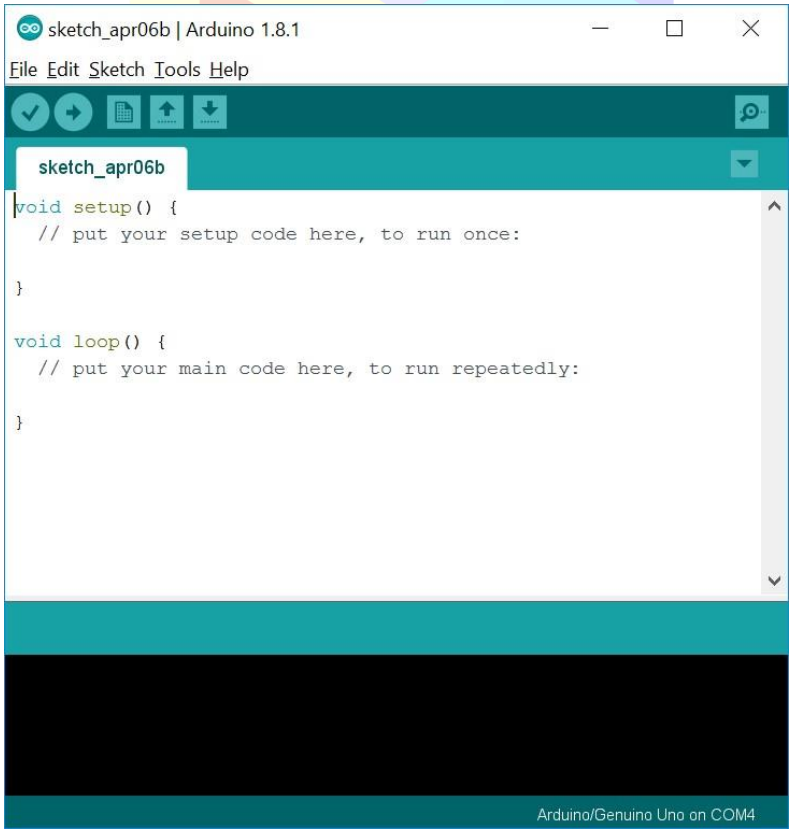
http://arduino.cc/en/Guide/MacOSX

Linux:
32 bit / 64 bit, Installation Process Go to the web address below to access the instructions for installations on a Linux-based computer.

http://www.arduino.cc/playground/Learning/Linux

STEP-4: Open the Arduino IDE

Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away, this is just an introduction. The step is to set your IDE to identify your Arduino Uno.



GUI (Graphical User Interface)

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details. Note: If you are using an external programmer with your board, you can hold down the "shift" ke on your computer when using this icon. The text will change to "Upload using Programmer" *New*

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the curre window overwriting its content.

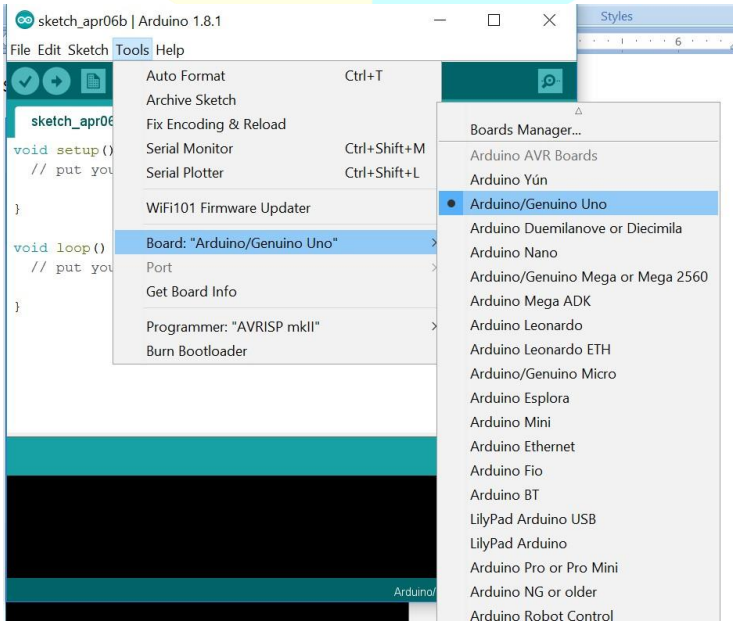
Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbookmenu instead. *Save*

Saves your sketch.

Serial Monitor

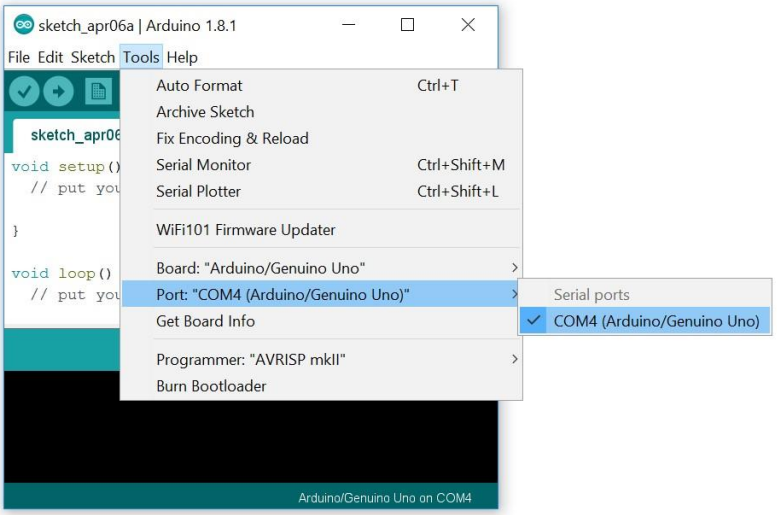
Opens the serial monitor.

STEP-5: Select your board: Arduino Uno



STEP-6: Select your Serial Device

Windows: Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be com3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



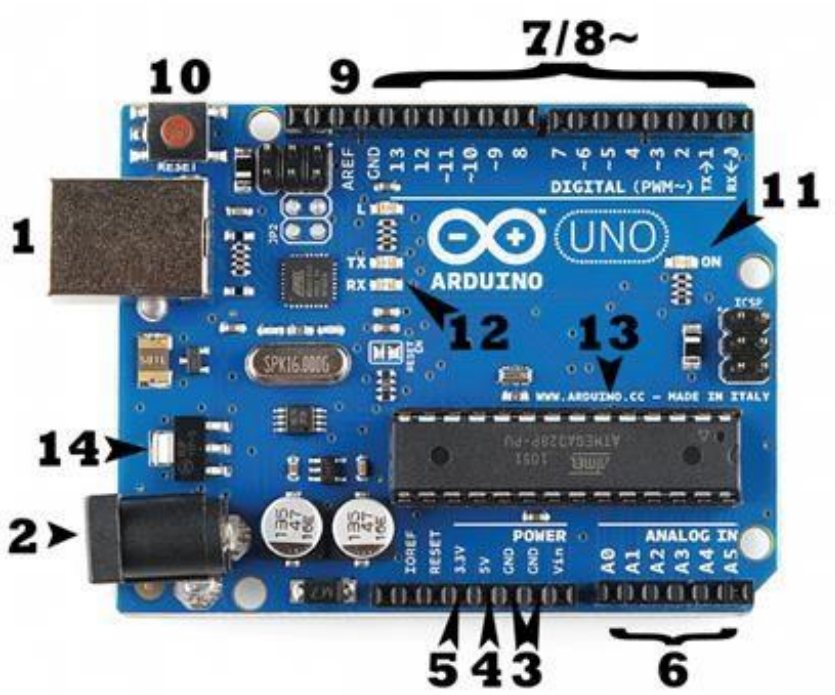
Mac OS: Select the serial device of the Arduino board from the Tools > Serial Port menu. On the Mac, this should be something with /dev/tty.usbmodem (for the Uno or Mega 2560) or /dev/tty.usbserial (for older boards) in it.

Linux: http://playground.arduino.cc/Learning/Linux

About Arduino Uno R3 board

What's on the board?

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common:



Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

.

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjuction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3):** Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7):** Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8):** You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

- **AREF (9):** Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10).** Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' **(11).** This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12).** These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit **(13).** Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

Lesson 1 Add Libraries

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

What are Libraries?

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

How to Install a Library

Using the Library Manager

To install a new library into your Arduino IDE, you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click Sketch > Include > Library > Manage Libraries.

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

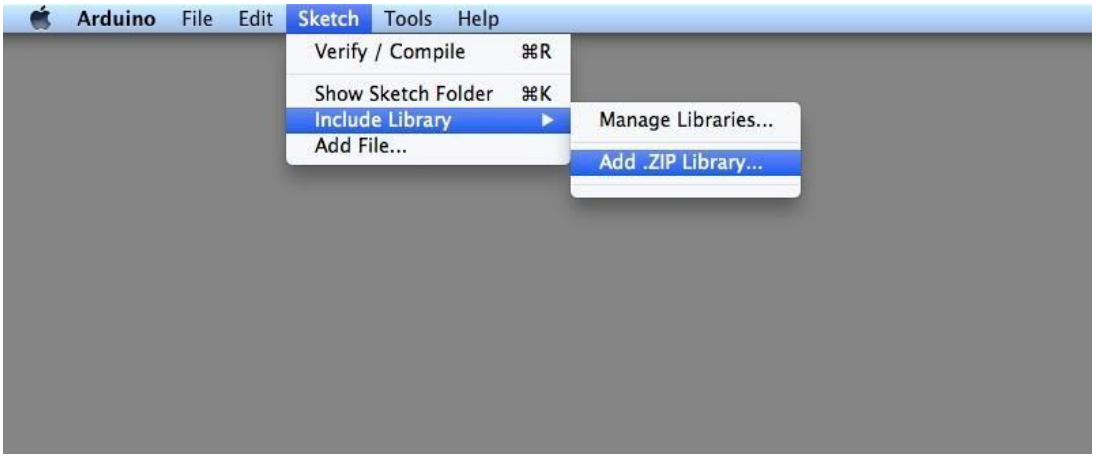
Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager.

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

Importing a .zip Library

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

In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library.



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

Return to the Sketch > Import Library menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

NB: The Library will be available to use in sketches, but examples for the library will not be shown in the File > Examples until after the IDE has restarted.

Manual installation

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

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

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

My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.cpp

My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.h

My Documents\Arduino\libraries\ArduinoParty\examples

....

or like this (on Mac and Linux):

Documents/Arduino/libraries/ArduinoParty/ArduinoParty.cpp

Documents/Arduino/libraries/ArduinoParty/ArduinoParty.h

Documents/Arduino/libraries/ArduinoParty/examples

....

There may be more files than just the .cpp and .h files so make sure they're all there. (The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder. For example: Documents\Arduino\libraries\ArduinoParty.cpp and Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp won't work.)

Restart the Arduino application. Make sure the new library appears in the Sketch > Import Library menu. That's it. You've installed a library!

Summary

In this lesson, we will install all the libraries that we will use in the tutorial. Open the libraries folder and install the ZIP files one by one so we won't need to do this step in the following lessons. We just connect the component as the schematic and upload the code provided. Then the kit will work.

Lesson 2 Blink

Overview

In this lesson, you will learn how program your UNO R3 controller board to make the Arduino's built-in LED blink.

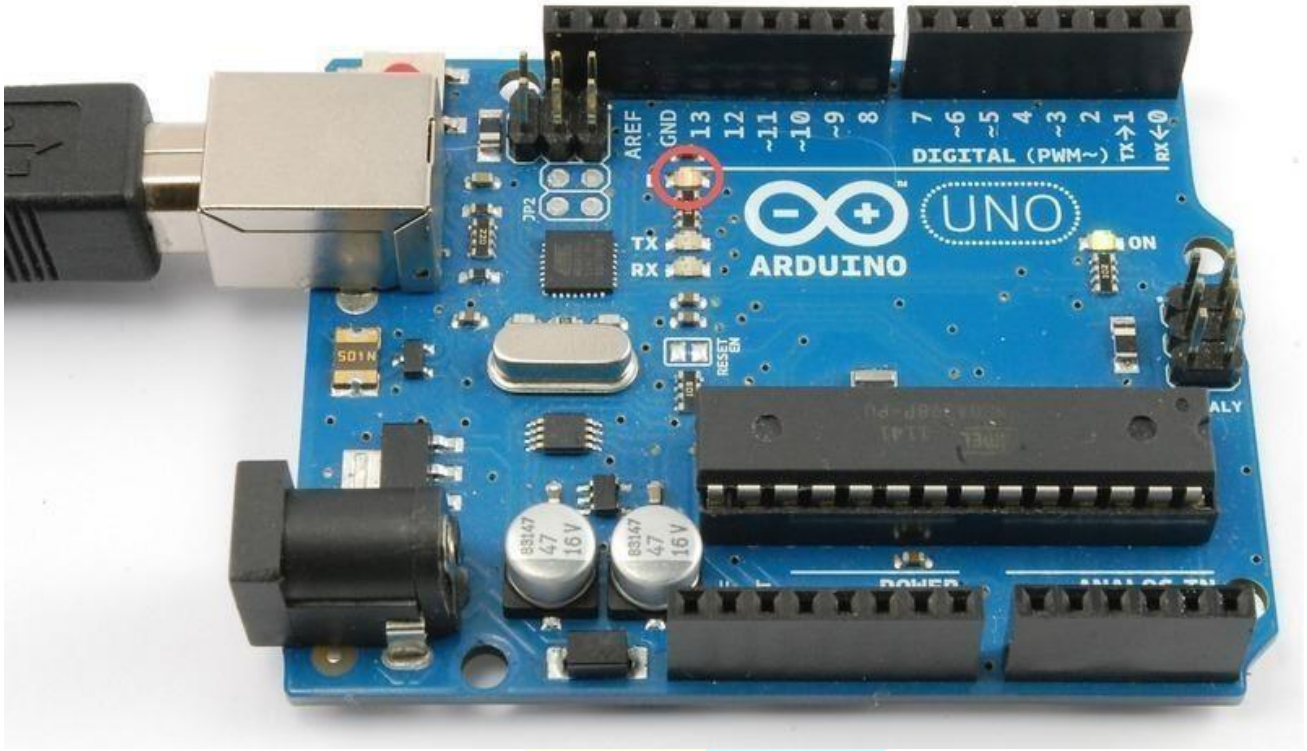
Component Required:

(1) x Quaduino Uno R3

Principle

The UNO R3 board has rows of connectors along both sides that are used to connect to several electronic devices and plug-in 'shields' that extends its capability.

It also has a single LED that you can control from your sketches. This LED is built onto the UNO R3 board and is often referred to as the 'L' LED as this is how it is labeled on the board.



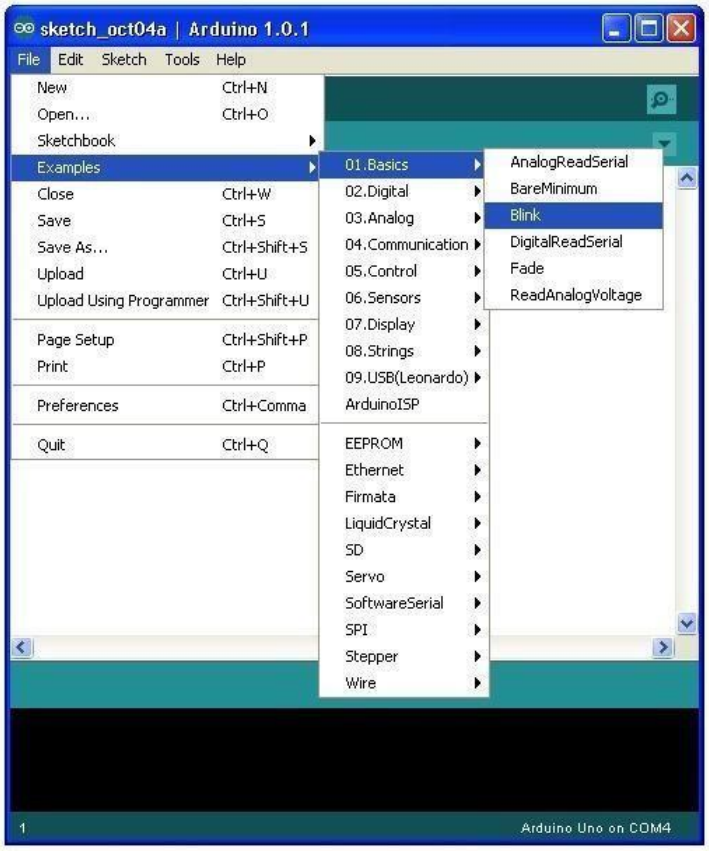
You may find that your UNO R3 board 's 'L' LED already blinks when you connect it to a USB plug. This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

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

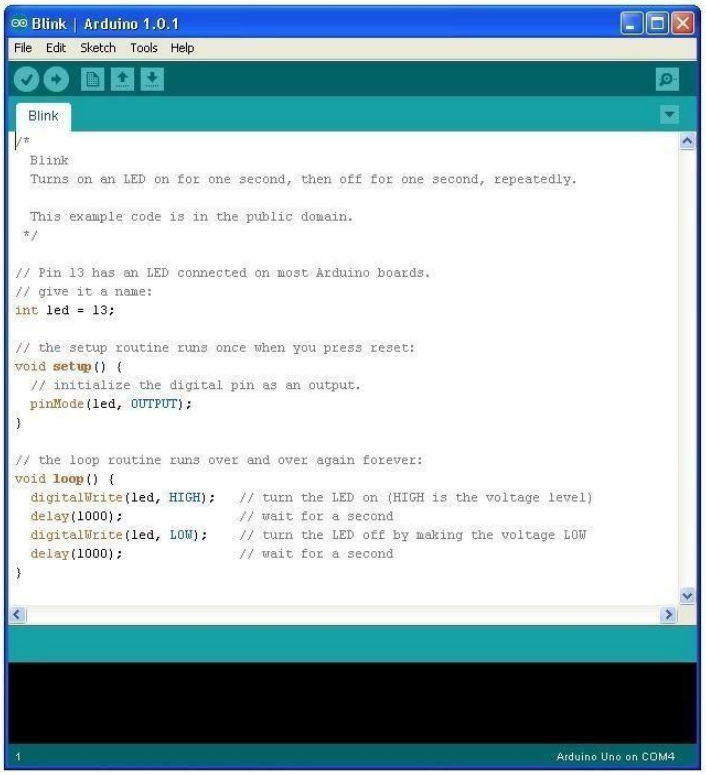
In Lesson 0, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

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

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



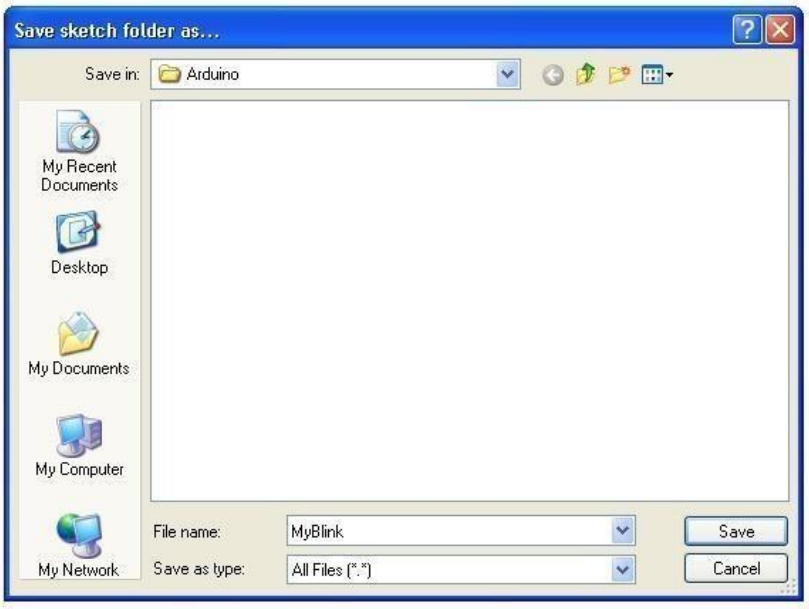
When the sketch window opens, enlarge it so that you can see the entire sketch in the window.



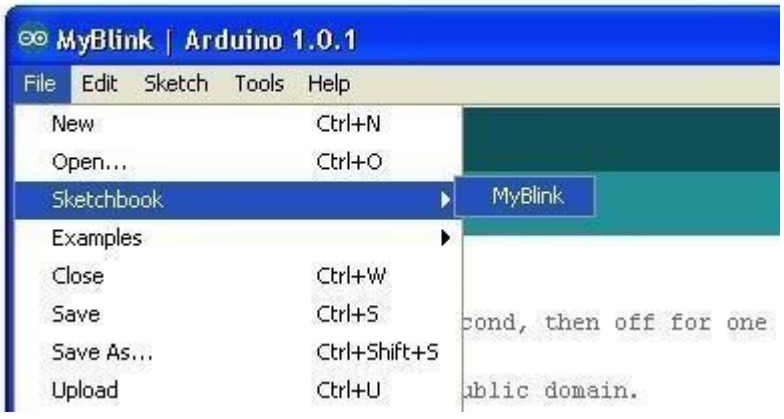
The example sketches included with the Arduino IDE are 'read-only'. That is, you can upload them to an UNO R3 board, but if you change them, you cannot save them as the same file.

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

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

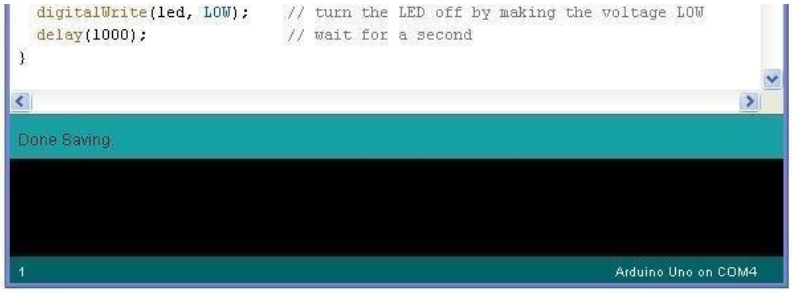


You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File > Sketchbook menu option.



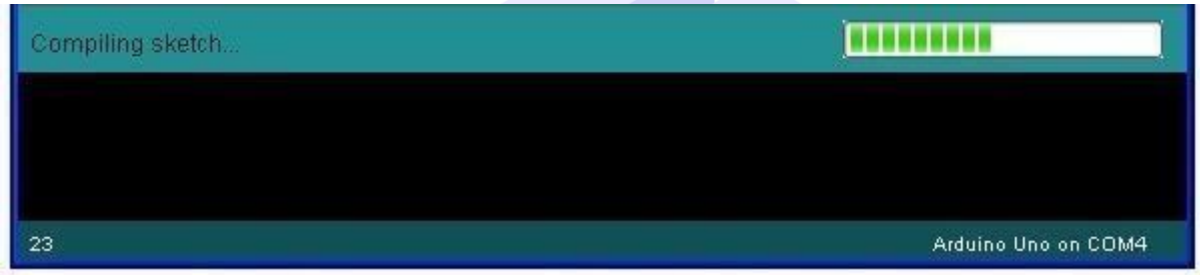
Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly. You may need to refer back to Lesson 0.

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

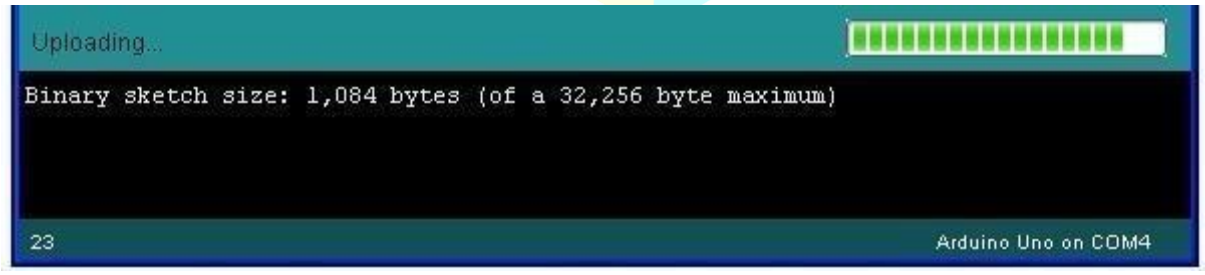


Click on the 'Upload' button. The second button from the left on the toolbar.

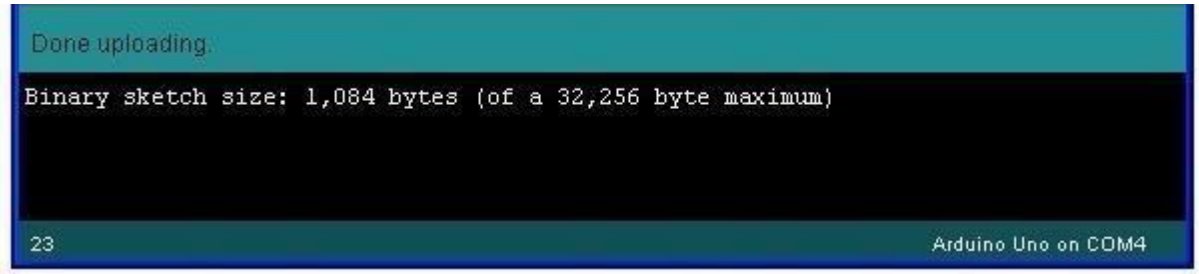
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first, it will say 'Compiling Sketch...'. This converts the sketch into a format suitable for uploading to the board.



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



Finally, the staus will change to 'Done'.



The other message tells us that the sketch is using 1,084 bytes of the 32,256 bytes available.After the 'Compiling Sketch..' stage you could get the following error message:

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

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

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

Open the code

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

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

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

The first line of code is:

```
int led = 13;
```

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

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

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

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

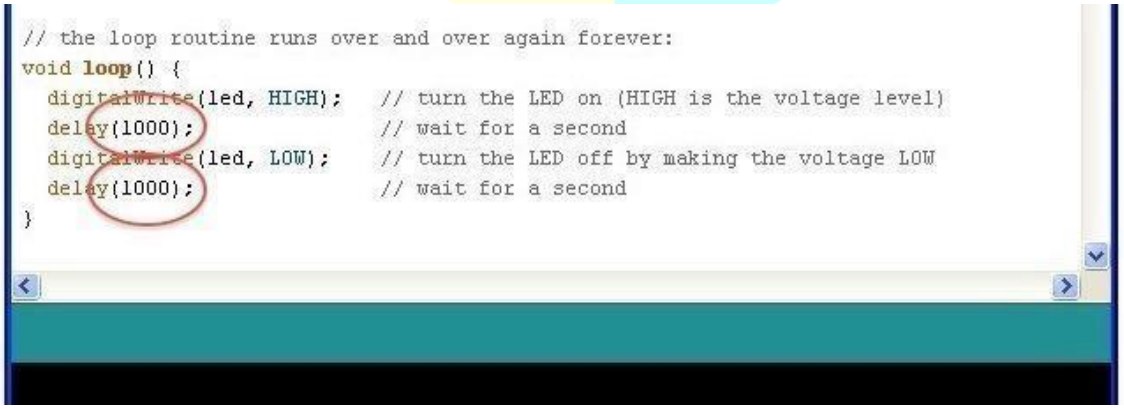
In this case, there is just one command there, which, as the comment states tells the Arduino board that we are going to use the LED pin as an output.

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

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

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

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



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

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

Lesson 3 LED

Overview

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

Component Required:

(1) x Quaduino Uno R3

(1) x 5mm red LED

(1) x 220 ohm resistor

(1) x 1k ohm resistor

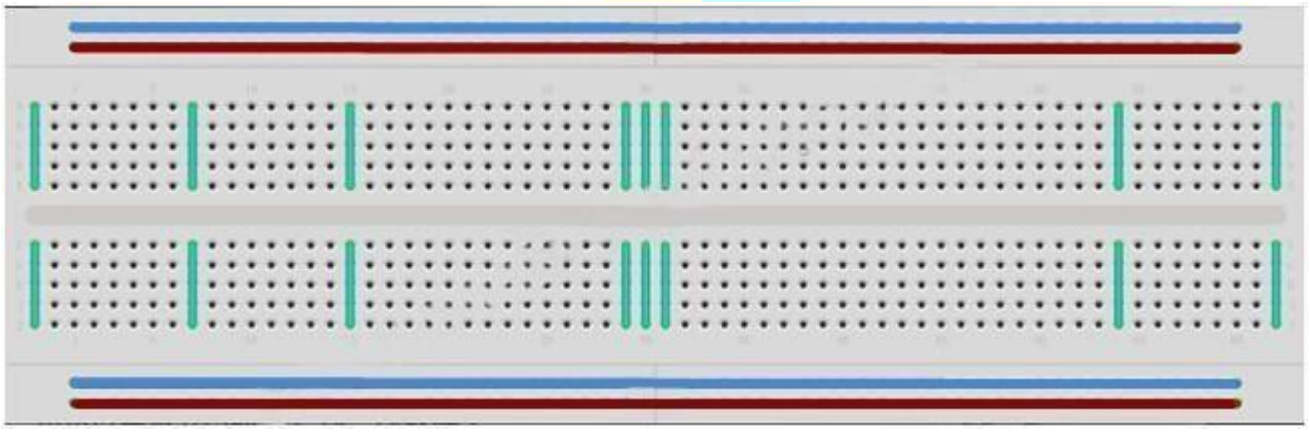
(1) x 10k ohm resistor

(2) x M-M wires

Component Introduction

BREADBOARD MB-102 :

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



Breadboards come in various sizes and configurations. The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically. A deep channel running down the middle indicates that there is a break in connections there, meaning, you can push a chip in with the legs at either side of the

channel without connecting them together. Some breadboards have two strips of holes running along the long edges of the board that are separated from the main grid. These have strips running down the length of the board inside and provide a way to connect a common voltage. They are usually in pairs for +5 volts and ground. These strips are referred to as rails and they enable you to connect power to many components or points in the board.

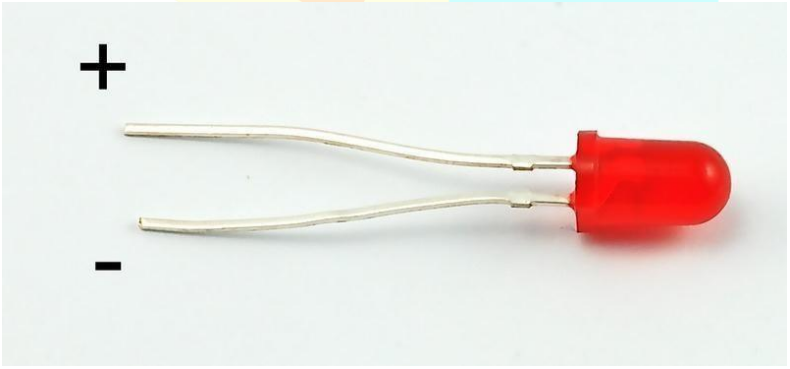
While breadboards are great for prototyping, they have some limitations. Because the connections are push-fit and temporary, they are not as reliable as soldered connections. If you are having intermittent problems with a circuit, it could be due to a poor connection on a breadboard.

LED:

LEDs make great indicator lights. They use very little electricity and they pretty much last forever.

In this lesson, you will use perhaps the most common of all LEDs: a 5mm red LED. 5mm refers to the diameter of the LED. Other common sizes are 3mm and 10mm.

You cannot directly connect an LED to a battery or voltage source because 1) the LED has a positive and a negative lead and will not light if placed the wrong way and 2) an LED must be used with a resistor to limit or 'choke' the amount of current flowing through it; otherwise, it will burn out!



I If you do not use a resistor with an LED, then it may well be destroyed almost immediately, as too much current will flow through, heating it and destroying the 'junction' where the light is produced.

There are two ways to tell which is the positive lead of the LED and which the negative.

Firstly, the positive lead is longer.

Secondly, where the negative lead enters the body of the LED, there is a flat edge to the case of the LED.

If you happen to have an LED that has a flat side next to the longer lead, you should assume that the longer lead is positive.

RESISTORS:

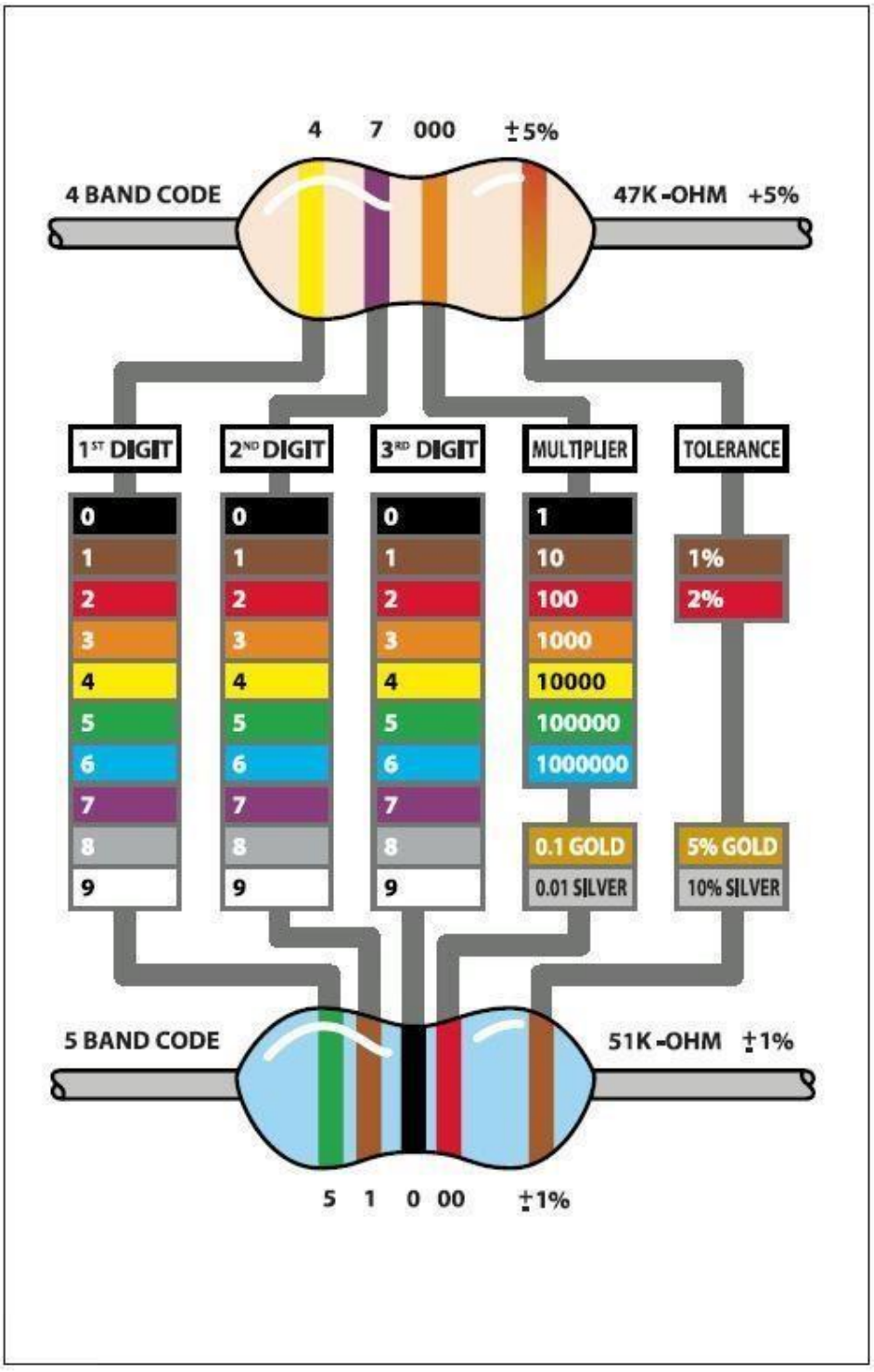
As the name suggests, resistors resist the flow of electricity. The higher the value of the resistor, the more it resists and the less electrical current will flow through it. We are going to use this to control how much electricity flows through the LED and therefore, how brightly it shines.

But first, more about resistors...

The unit of resistance is called the Ohm, which is usually shortened to Ω the Greek letter Omega. Because an Ohm is a low value of resistance (it doesn't resist much at all), we also denote the values of resistors in kΩ (1,000 Ω) and MΩ (1,000,000 Ω). These are called kilo-ohms and mega-ohms.

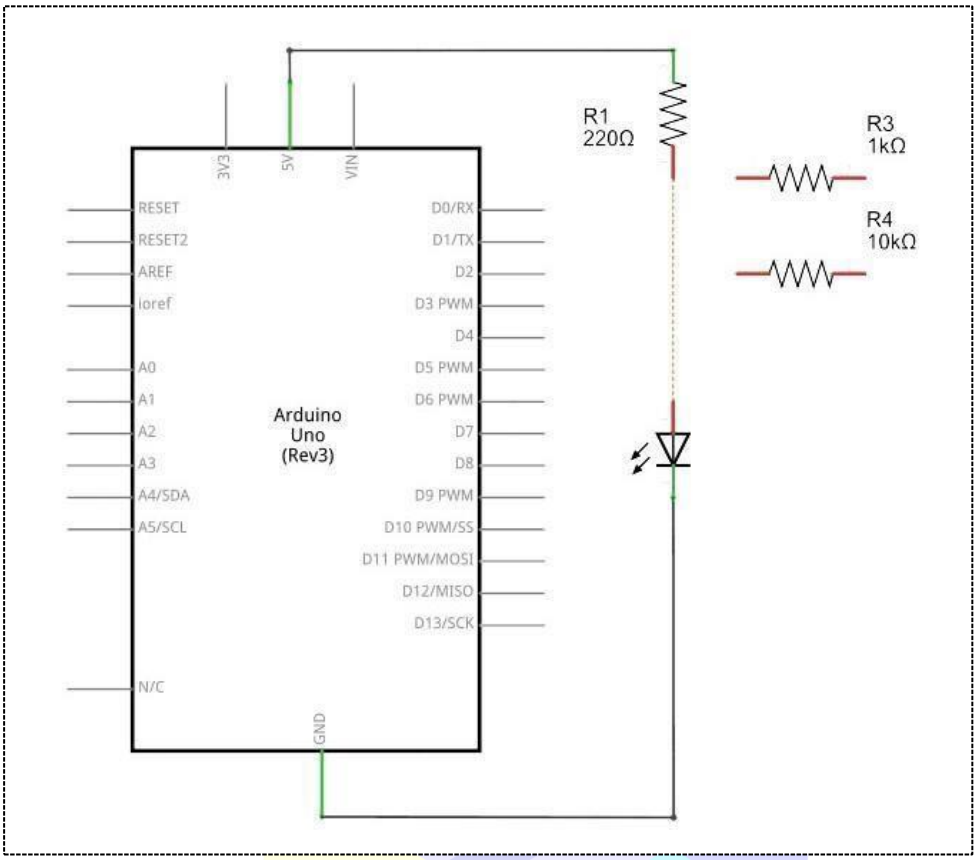
In this lesson, we are going to use three different values of resistor: 220Ω, 1kΩ and 10kΩ. These resistors all look the same, except that they have different colored stripes on them.
These stripes tell you the value of the resistor.

The resistor color code has three colored stripes and then a gold stripe at one end.

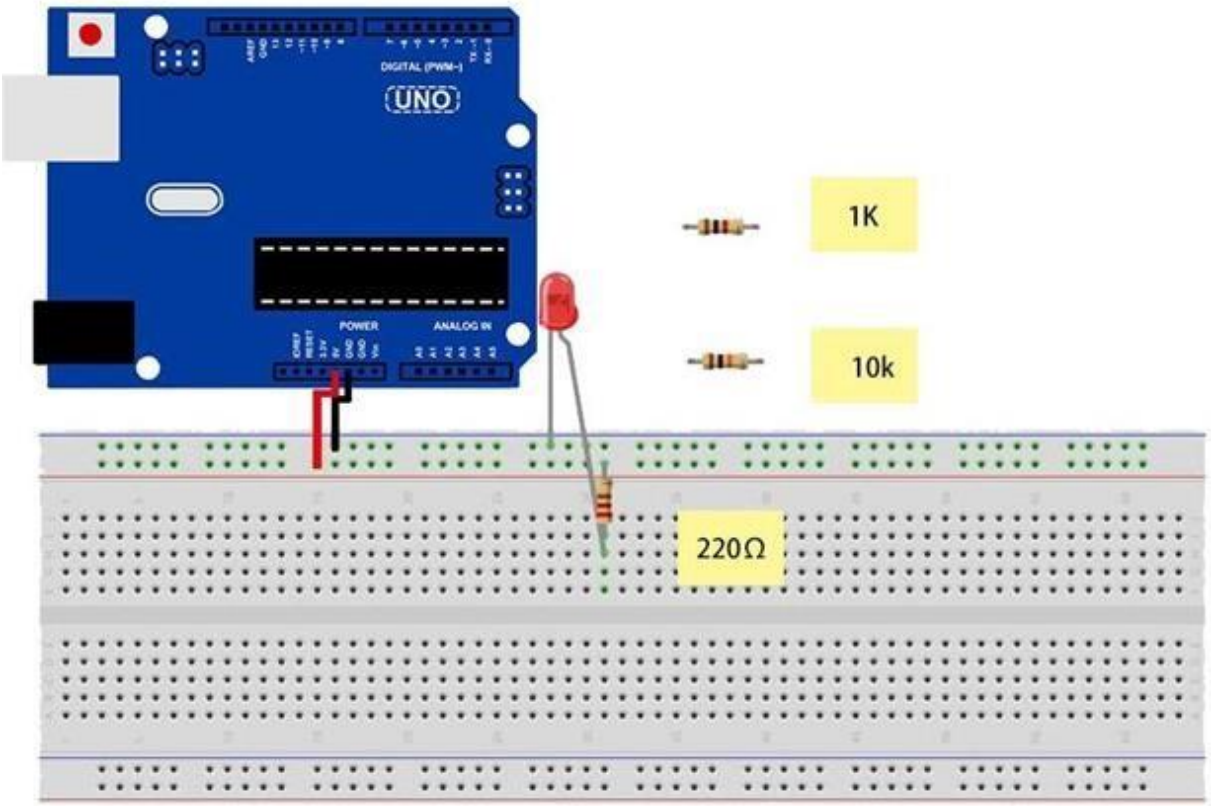


Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either way around.

Connection Schematic



wiring diagram



The UNO is a convenient source of 5 volts, which we will use to provide power to the LED and

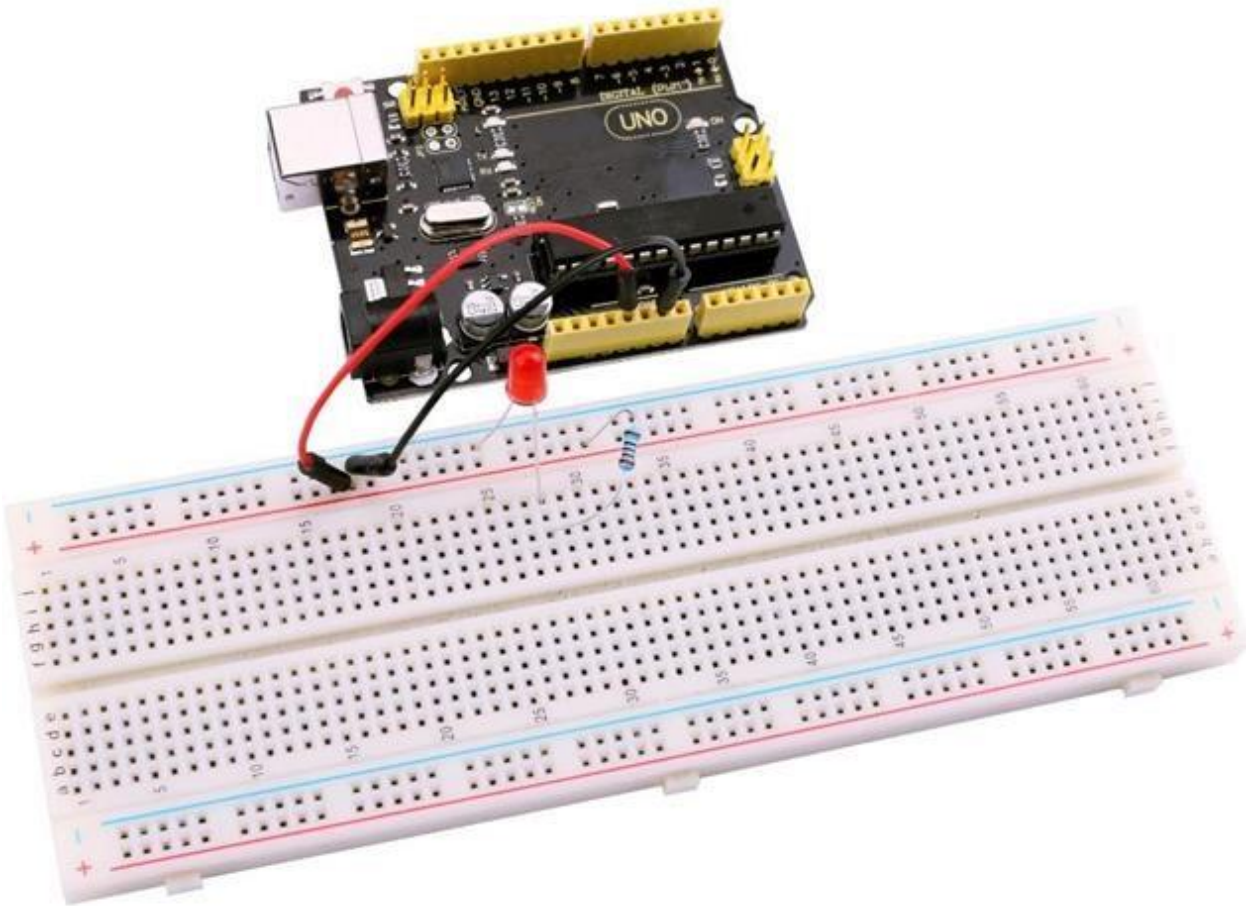
the resistor. You do not need to do anything with your UNO, except to plug it into a USB cable.

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

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

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

It does not matter which side of the LED we put the resistor, as long as it is there somewhere.6



Lesson 4 Digital Inputs

Overview

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

Pressing the button nearer the top of the breadboard will turn the LED on; pressing the other button will turn the LED off.

Component Required:

(1) x Quaduino Uno R3

(1) x Breadboard

(1) x 5mm red LED

(1) x 220 ohm resistor

(2) x push switches

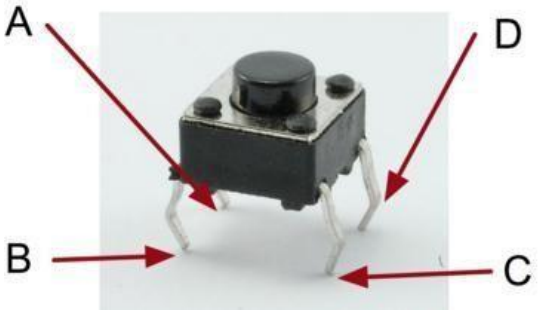
(6) x M-M wires

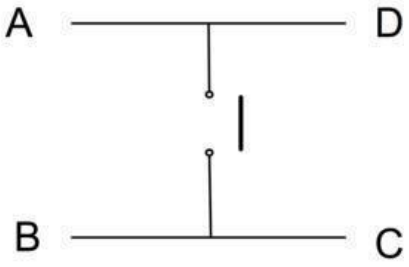
Component Introduction

PUSH SWITCHES:

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

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

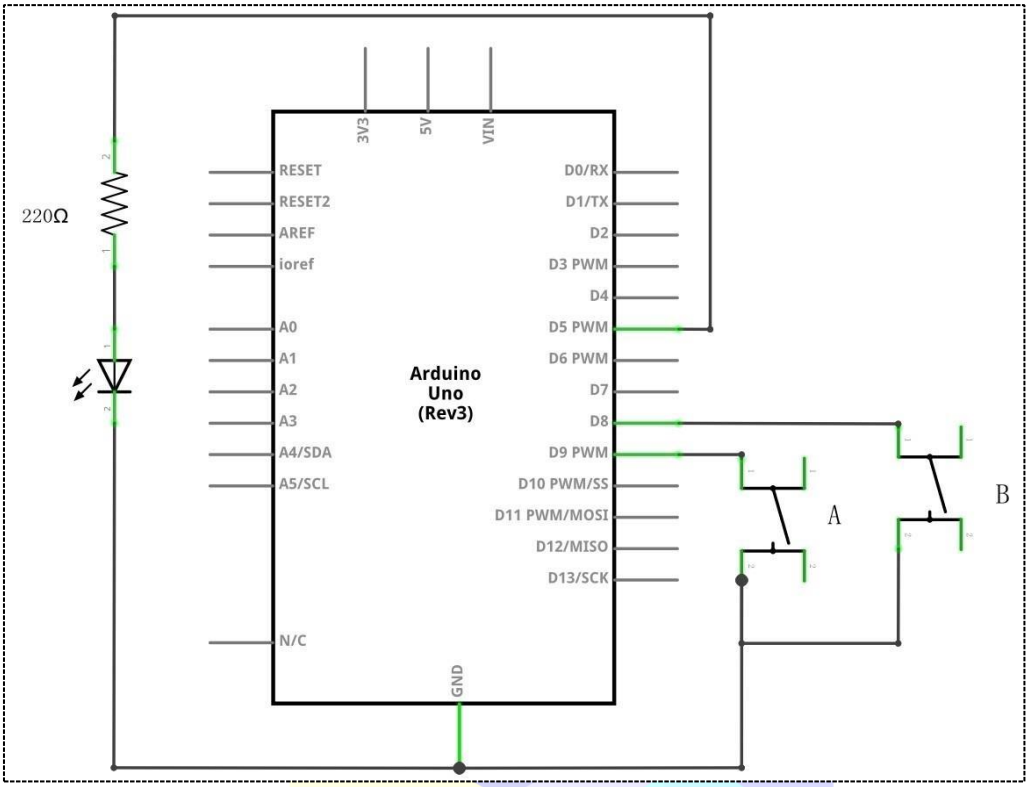




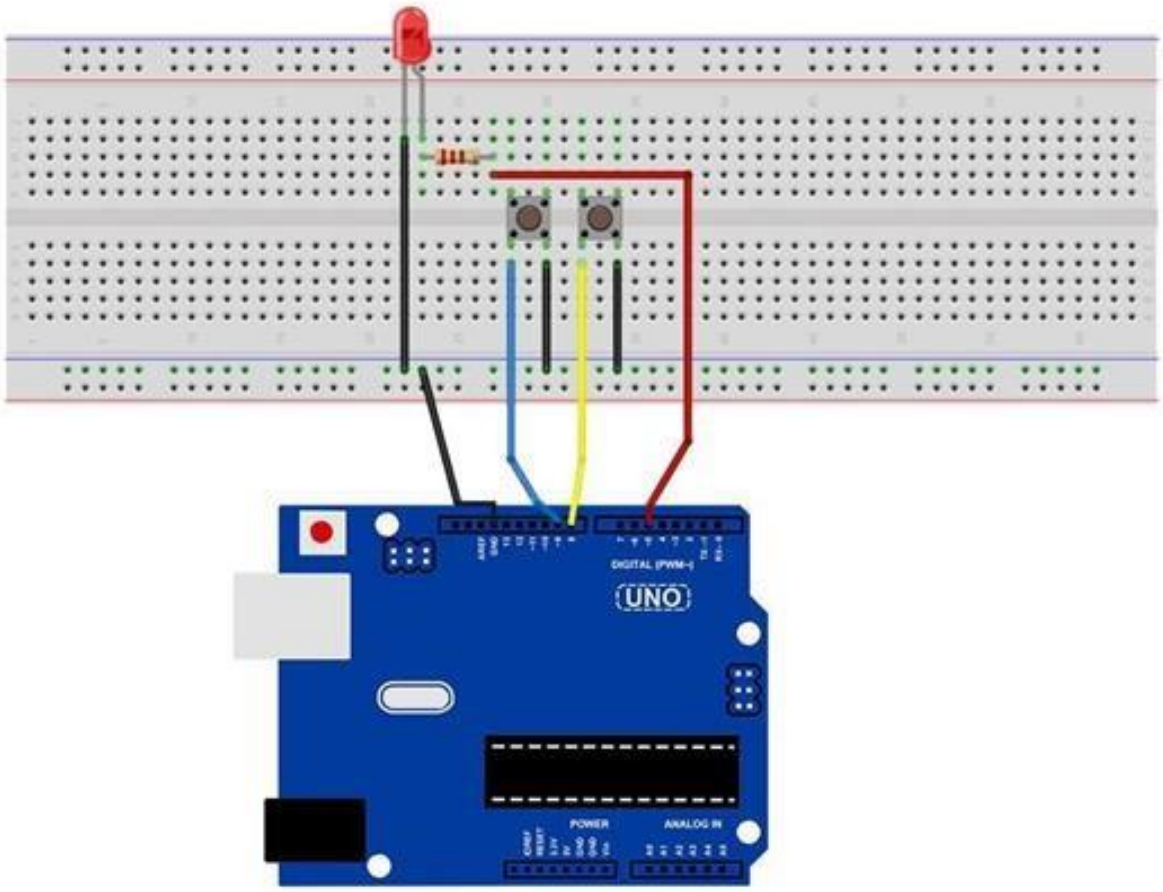
Actually, there are only really two electrical connections. Inside the switch package, pins B and C are connected together, as are A and D.

Connection

Schematic



wiring diagram



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

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

Code

Load the sketch onto your UNO board. Pressing the top button will turn the LED on while pressing the bottom button will turn it off.

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

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

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

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

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

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

```
void loop()
{

  if (digitalRead(buttonApin) == LOW)
  {

    digitalWrite(ledPin, HIGH);
  }
```

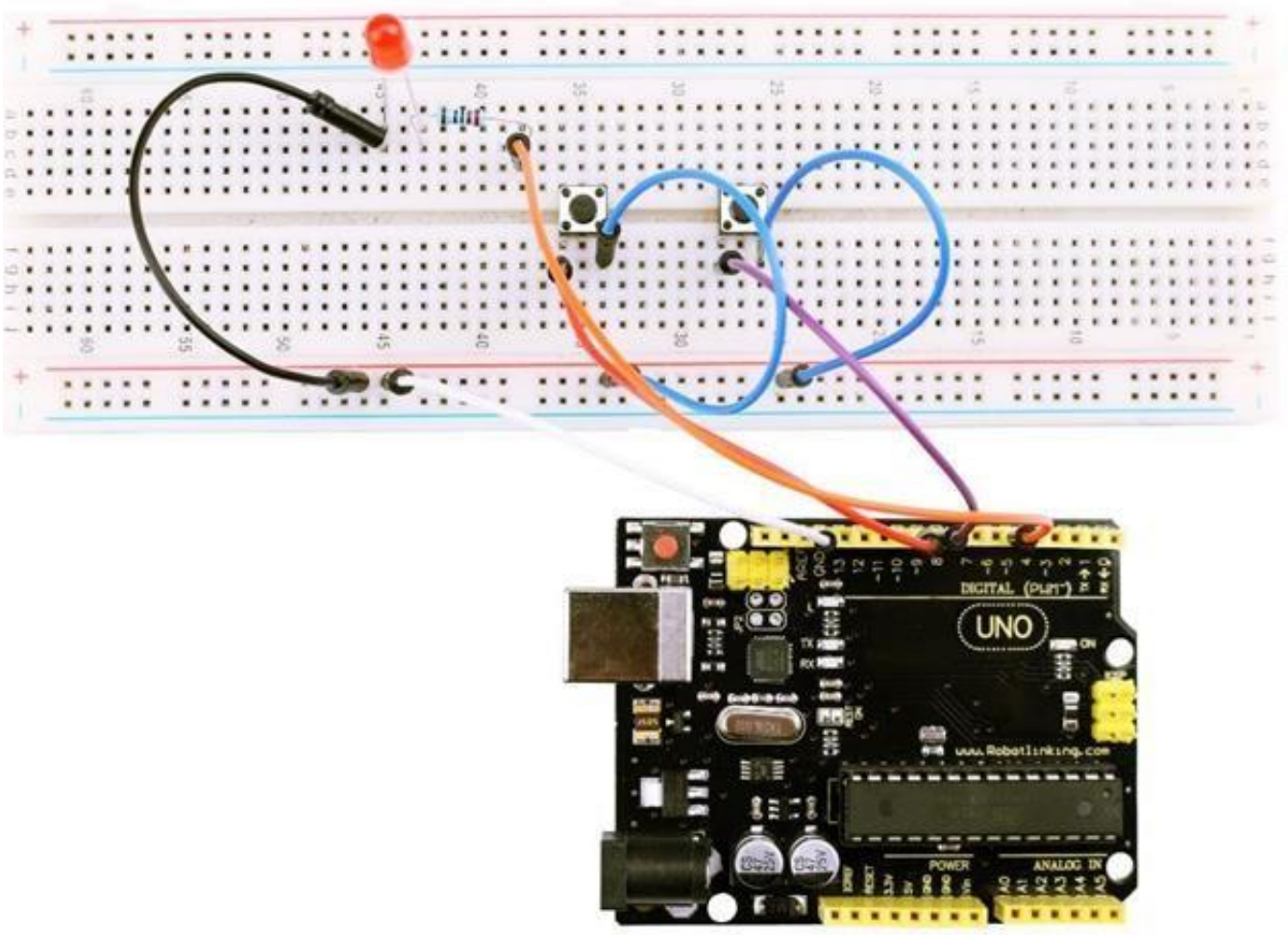
```
  if (digitalRead(buttonBpin) == LOW)

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

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

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

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



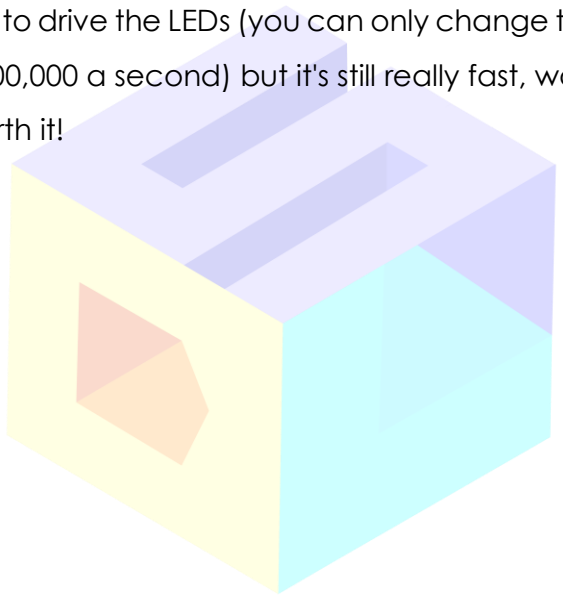
Lesson 5 Eight LED with 74HC595

Overview

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

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

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



Component Required:

(1) x Quaduino Uno R3

(1) x Breadboard

(8) x led

(8) x 220 ohm resistors

(1) x Breadboard

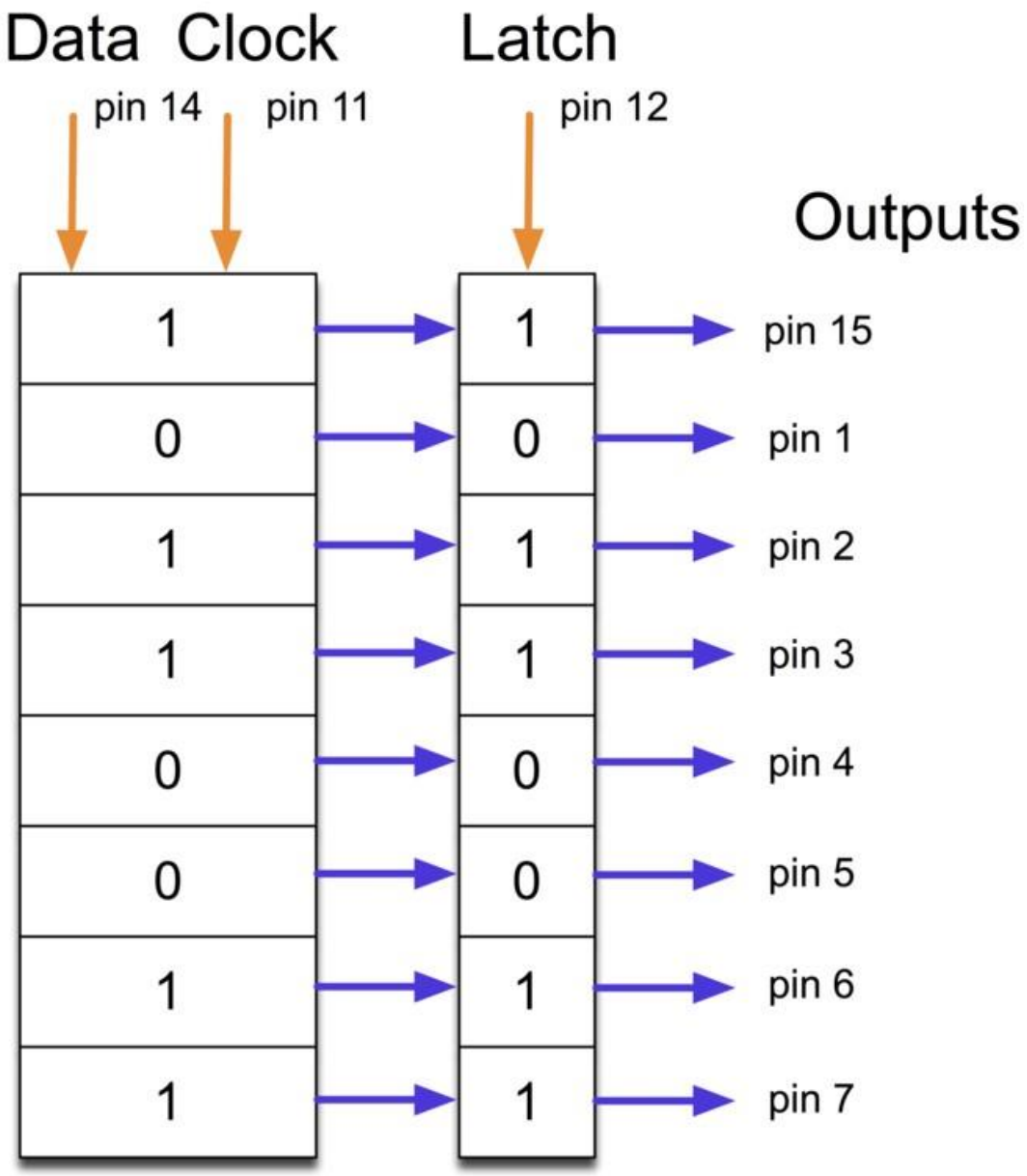
(1) x 74hc595 ic

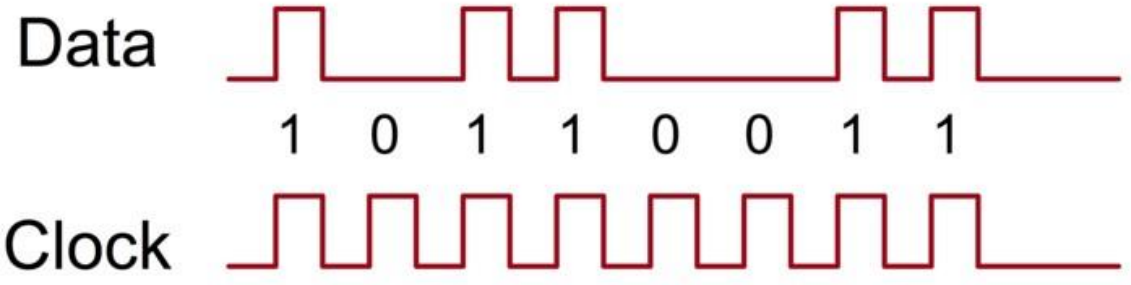
(14) x M-M wires

Component Introduction

74HC595 Shift Register:

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





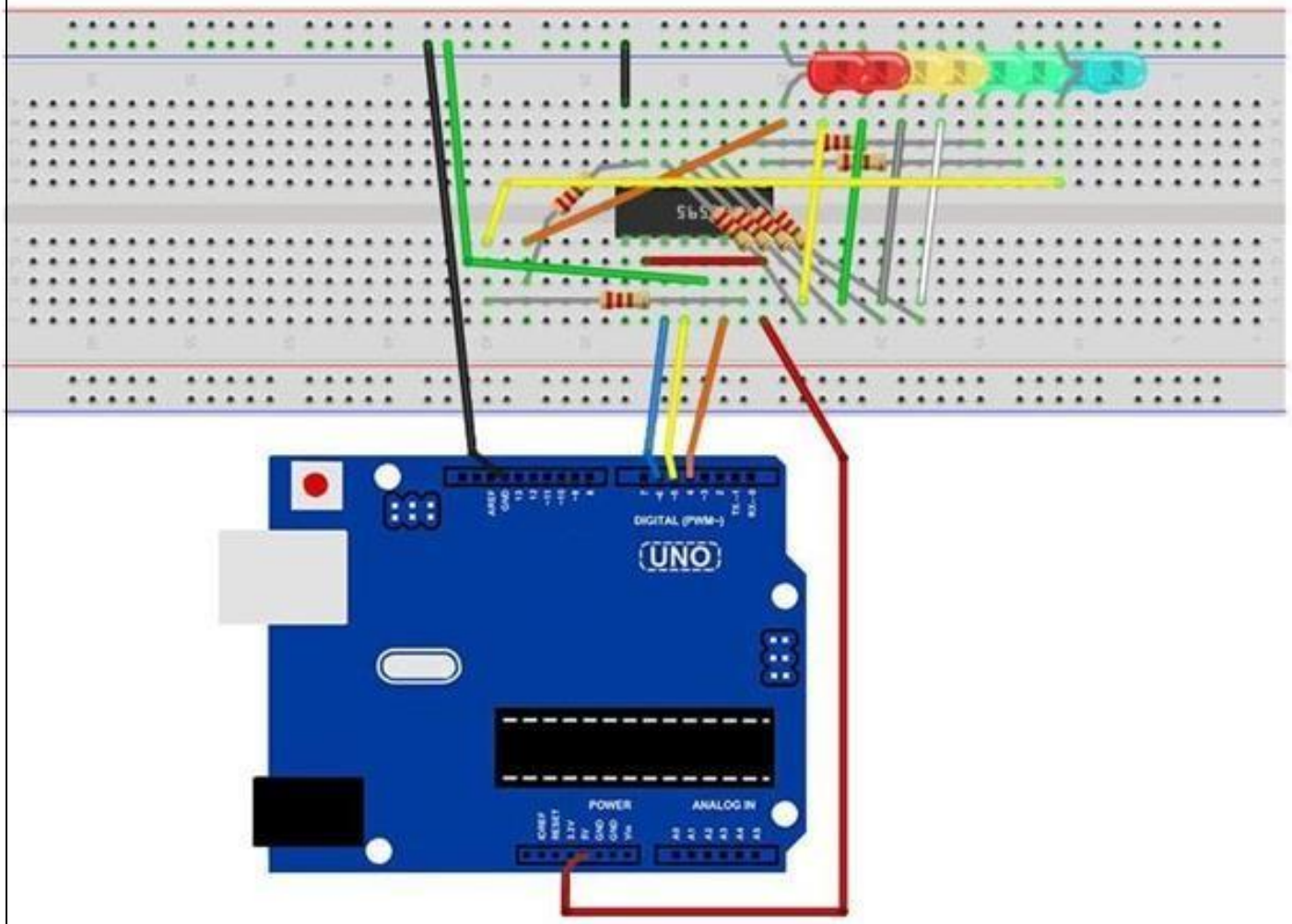
The clock pin needs to receive eight pulses. At each pulse, if the data pin is high, then a 1 gets pushed into the shift register; otherwise, a 0. When all eight pulses have been received,

enabling the 'Latch' pin copies those eight values to the latch register. This is necessary; otherwise, the wrong LEDs would flicker as the data is being loaded into the shift register.

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

Connection

wiring diagram



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

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

- Digital 4 from the UNO goes to pin #14 of the shift register
- Digital 5 from the UNO goes to pin #12 of the shift register
- Digital 6 from the UNO goes to pin #11 of the shift register

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

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

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

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

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

Code

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

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
```

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

```
byte leds = 0;
```

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

```
setup()
{
```

```
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT); pinMode(clockPin,
  OUTPUT);
}
```

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

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

There is then a half second delay before 'i' is incremented and the next LED is lit.

```
void
loop()
{

  leds = 0; updateShiftRegister();
  delay(500);
  for (int i = 0; i < 8; i++)
  {

    bitSet(leds, i);
    updateShiftRegister();
    delay(500);
  }
}
```

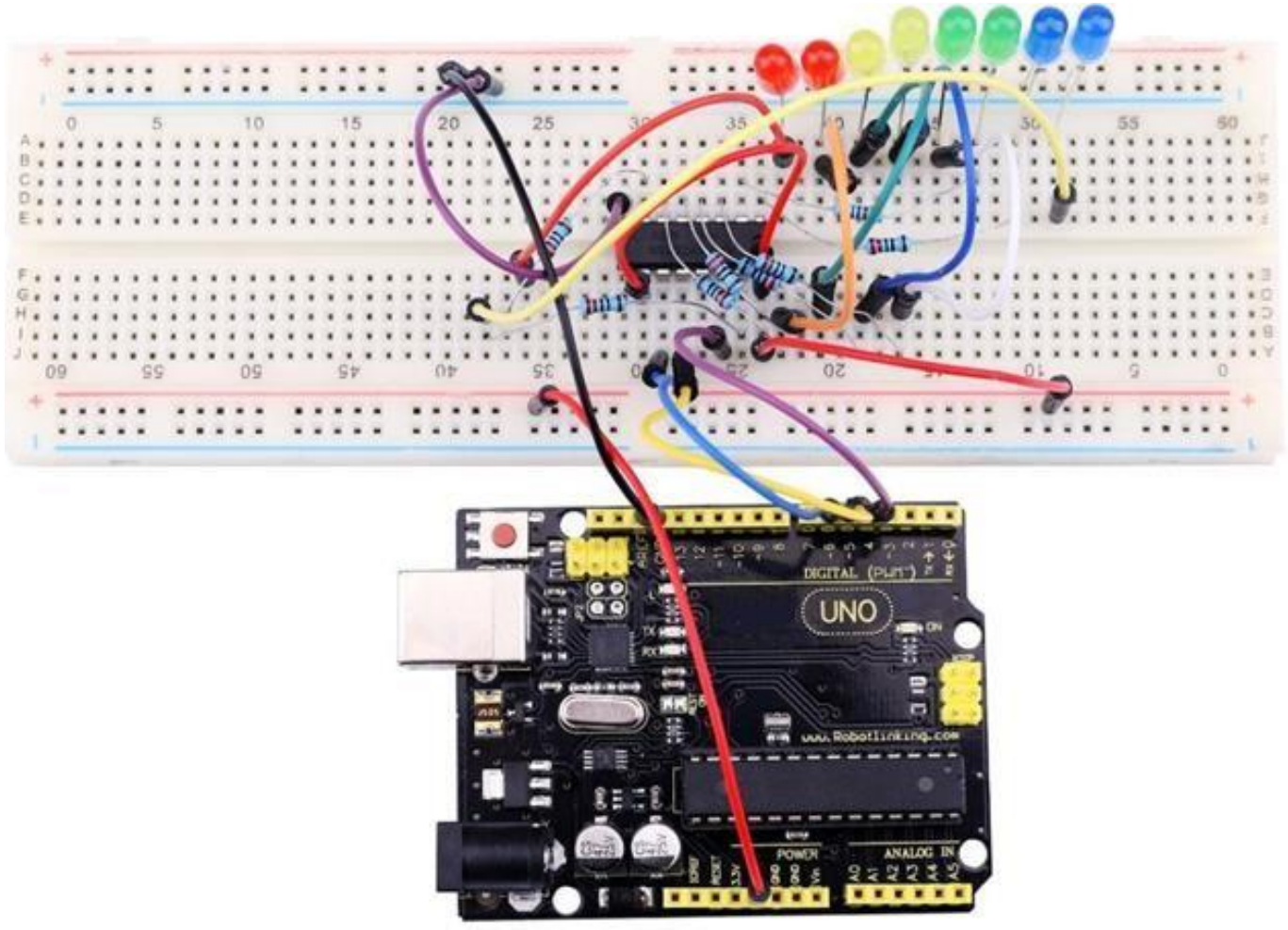
The function 'updateShiftRegister', first of all sets the latchPin to low, then calls the UNO function 'shiftOut' before putting the 'latchPin' high again. This takes four parameters, the first two are the pins to use for Data and Clock respectively.

The third parameter specifies which end of the data you want to start at. We are going to start with the right most bit, which is referred to as the 'Least Significant Bit' (LSB).

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

```
void updateShiftRegister()
{

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



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

Lesson 6 The Serial Monitor

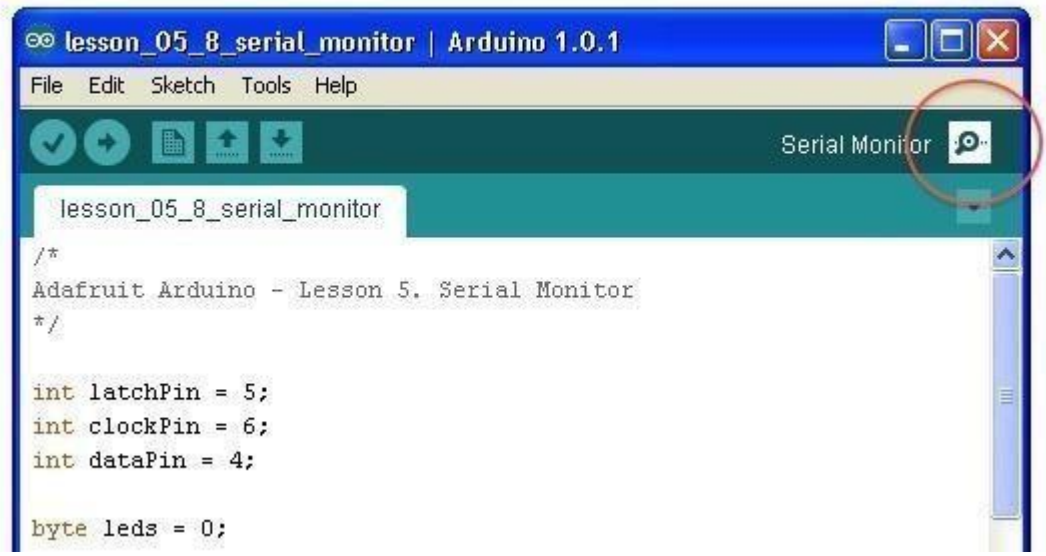
Overview

In this lesson, you will build on Lesson 6, adding the facility to control the LEDs from your computer using the Arduino Serial Monitor. The serial monitor is the 'tether' between the computer and your UNO. It lets you send and receive text messages, handy for debugging and also controlling the UNO from a keyboard! For example, you will be able to send commands from your computer to turn on LEDs.

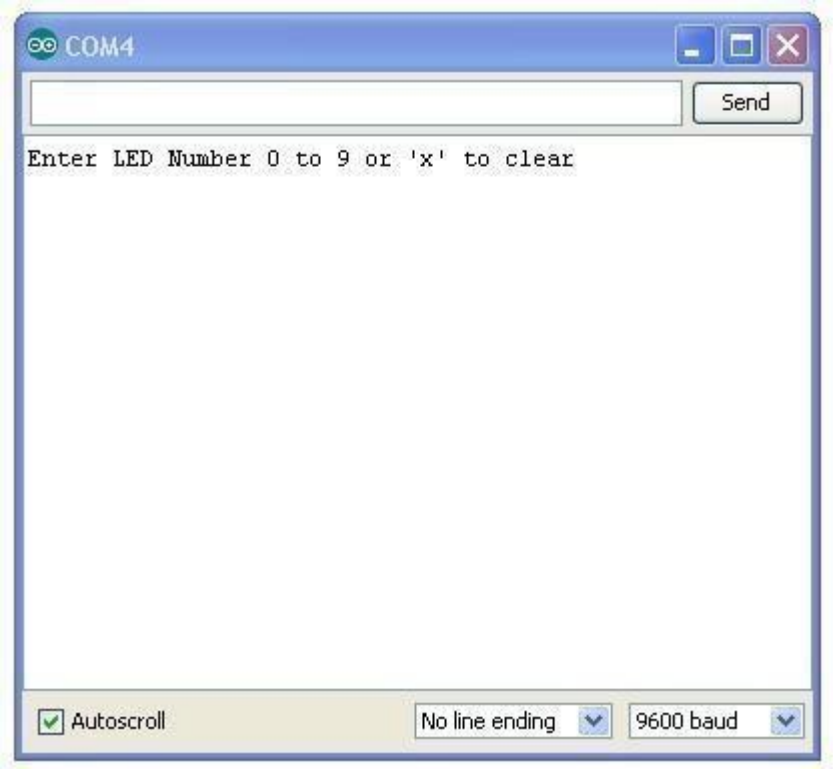
In this lesson, you will use exactly the same parts and a similar breadboard layout as Lesson 6. So, if you have not already done so, follow Lesson 6 now.

Steps taken

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



The following window will open.

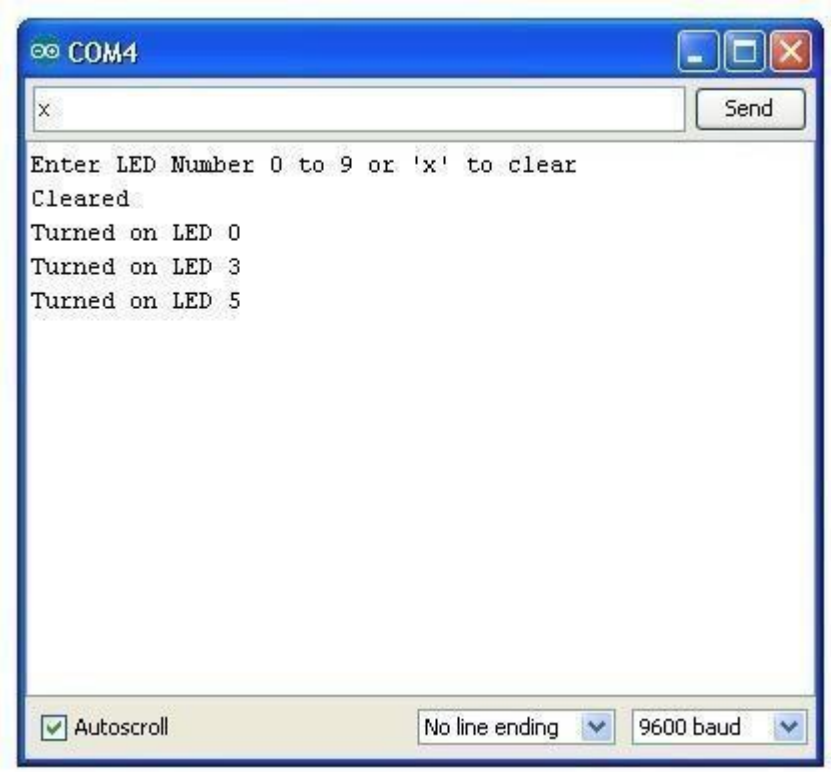


This window is called the Serial Monitor and it is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an UNO board (over USB) and also to receive messages from the UNO.

The message "Enter LED Number 0 to 9 or 'x' to clear" has been sent by the Arduino. It is telling us what commands we can send to the Arduino: either send the 'x' (to turn all the LEDs off) or the number of the LED you want to turn on (where 0 is the bottom LED, 1 is the next one up, all the way to 7 for the top LED).

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

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



TYpe x again and press „Send" to turn off all LEDs.

Code

As you might expect, the sketch is based on the sketch used in Lesson 6. So, we will just cover the new bits here. You will find it useful to refer to the full sketch in your Arduino IDE.

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

```
void
setup()
{

  pinMode(latchPin, OUTPUT); pinMode(dataPin,
  OUTPUT); pinMode(clockPin, OUTPUT);
  updateShiftRegister(); Serial.begin(9600); while (!
  Serial); // Wait until Serial is ready - Leonardo
  Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}
```

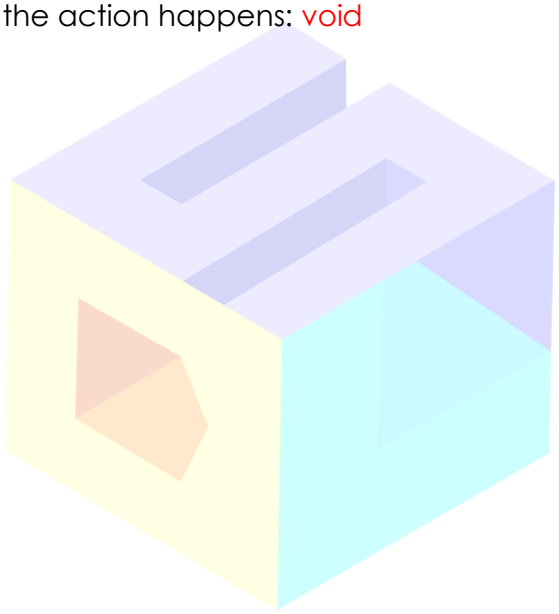
Firstly, we have the command 'Serial.begin(9600)'. This starts serial communication, so that the UNO can send out commands through the USB connection. The value 9600 is called the

'baud rate' of the connection. This is how fast the data is to be sent. You can change this to a higher value, but you will also have to change the Arduino Serial monitor to the same value. We will discuss this later; for now, leave it at 9600.

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

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

The 'loop' function is where all the action happens: void loop()



```
{

  if (Serial.available())
  {

    char ch = Serial.read(); if
    (ch >= '0' && ch <= '7')
    {

      int led = ch - '0'; bitSet(leds,
      led); updateShiftRegister();
      Serial.print("Turned on LED ");
      Serial.println(led);
    }

    if (ch == 'x')
    {
```

```
        leds = 0;
        updateShiftRegister();
        Serial.println("Cleared");
      }
    }
}
```

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

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

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

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

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

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

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

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

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

```
led = ch – '0';
```

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

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

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

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

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

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

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

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

```
if (ch == 'x')
{

  leds = 0;
  updateShiftRegister();
  Serial.println("Cleared");
}
```

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

Lesson 7 Photocell

Overview

In this lesson, you will learn how to measure light intensity using an Analog Input. You will build on lesson 8 and use the level of light to control the number of LEDs to be lit.

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

Component Required:

(1) x Quaduino Uno R3

(1) x Breadboard

(8) x led

(8) x 220 ohm resistors

(1) x 1k ohm resistors

(1) x Breadboard

(1) x 74hc595 ic

(1) x Photocell

(14) x M-M wires

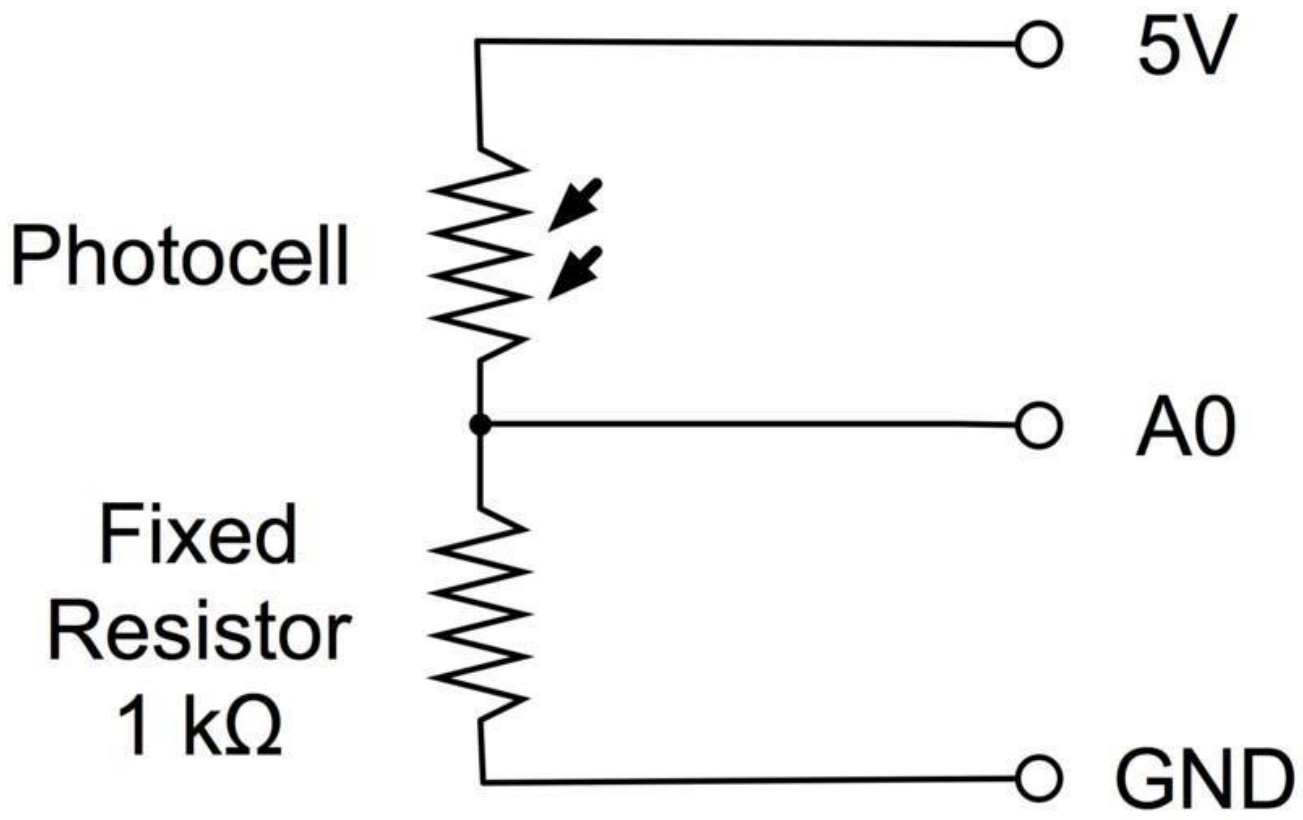
Component Introduction

PHOTOCELL:

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

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

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



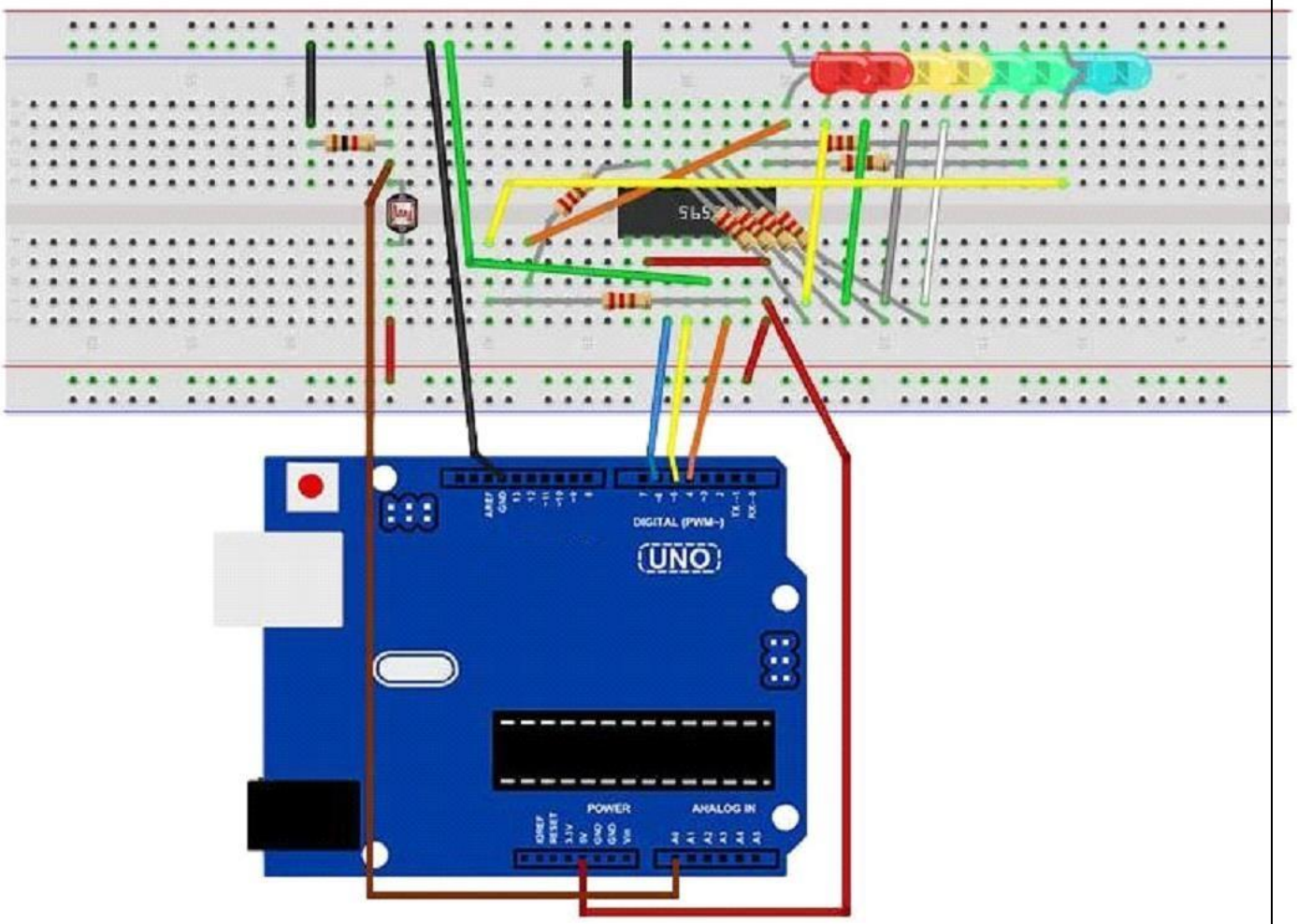
The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.

When the photocell is in dull light, the resistance becomes greater than the fixed 1 kΩ resistor and it is as if the pot were being turned towards GND.

Load up the sketch given in the next section and try covering the photocell with your finger, and then holding it near a light source.

Connection

wiring diagram

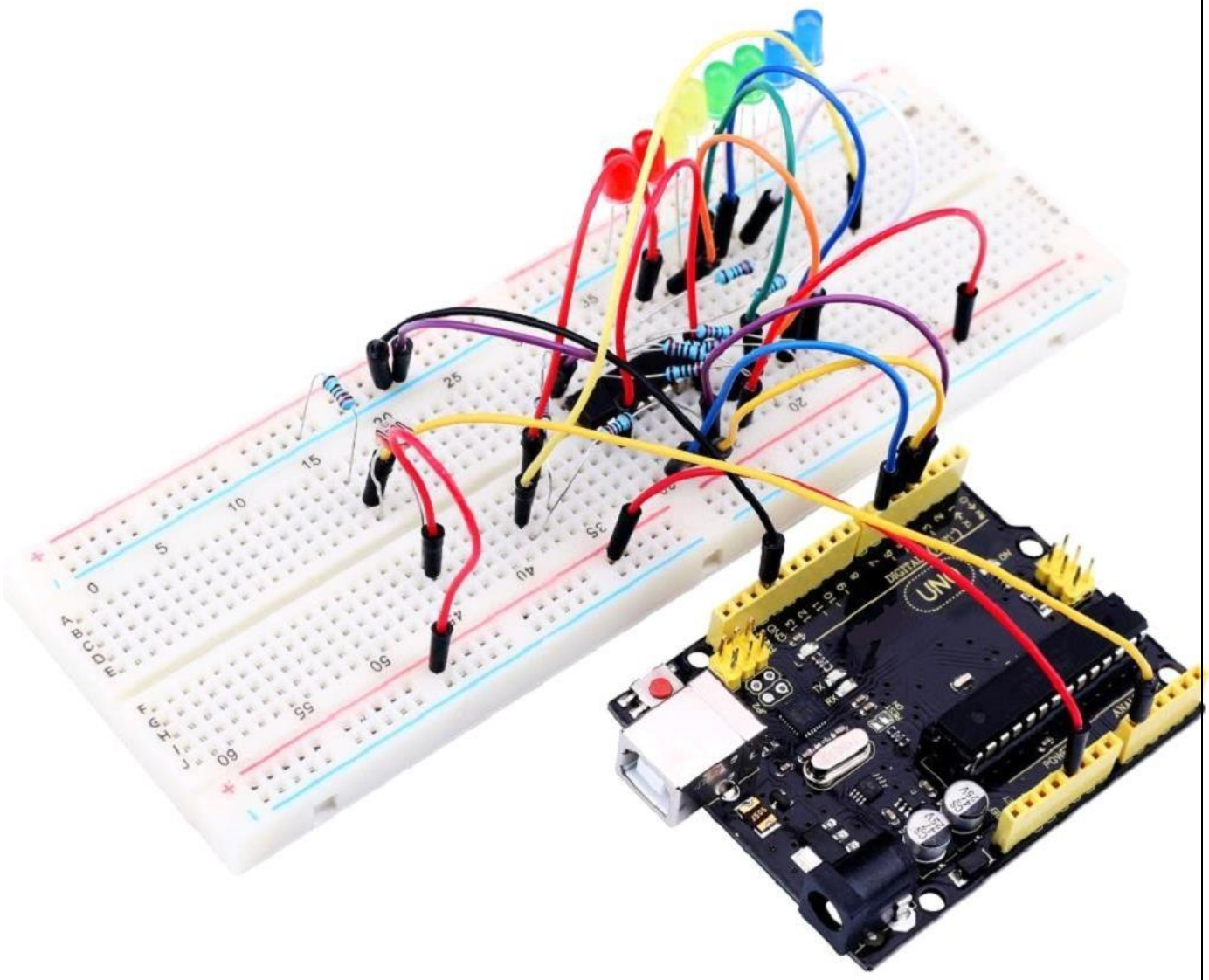


Code

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

The only other substantial change to the sketch is the line that calculate how many of the LEDs to light: int numLEDSLit = reading / 57; // all LEDs lit at 1k

This time, we divide the raw reading by 57 rather than 114. In other words, we divide it by half as much as we did with the pot to split it into nine zones, from no LEDs lit to all eight lit. This extra factor is to account for the fixed 1 kΩ resistor. This means that when the photocell has a resistance of 1 kΩ (the same as the fixed resistor), the raw reading will be 1023 / 2 = 511. This will equate to all the LEDs being lit and then a bit (numLEDSLit) will be 9.



Lesson 8 Making Sounds

Overview

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

Component Required:

(1) x Quaduino Uno R3

(1) x Active buzzer

(2) x F-M wires

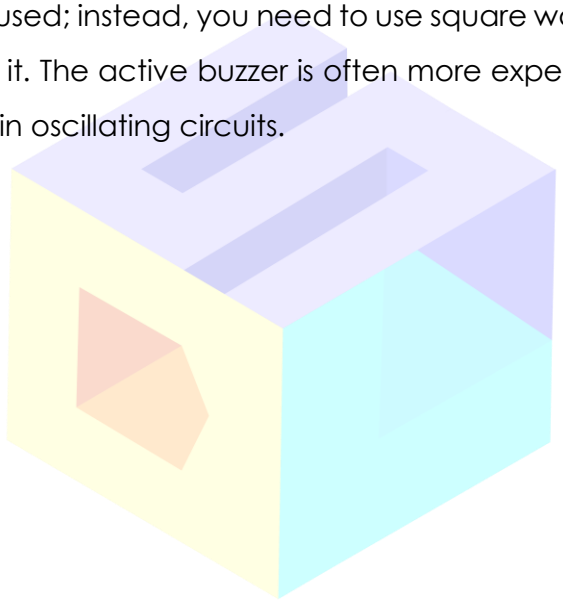
Component Introduction

BUZZER:

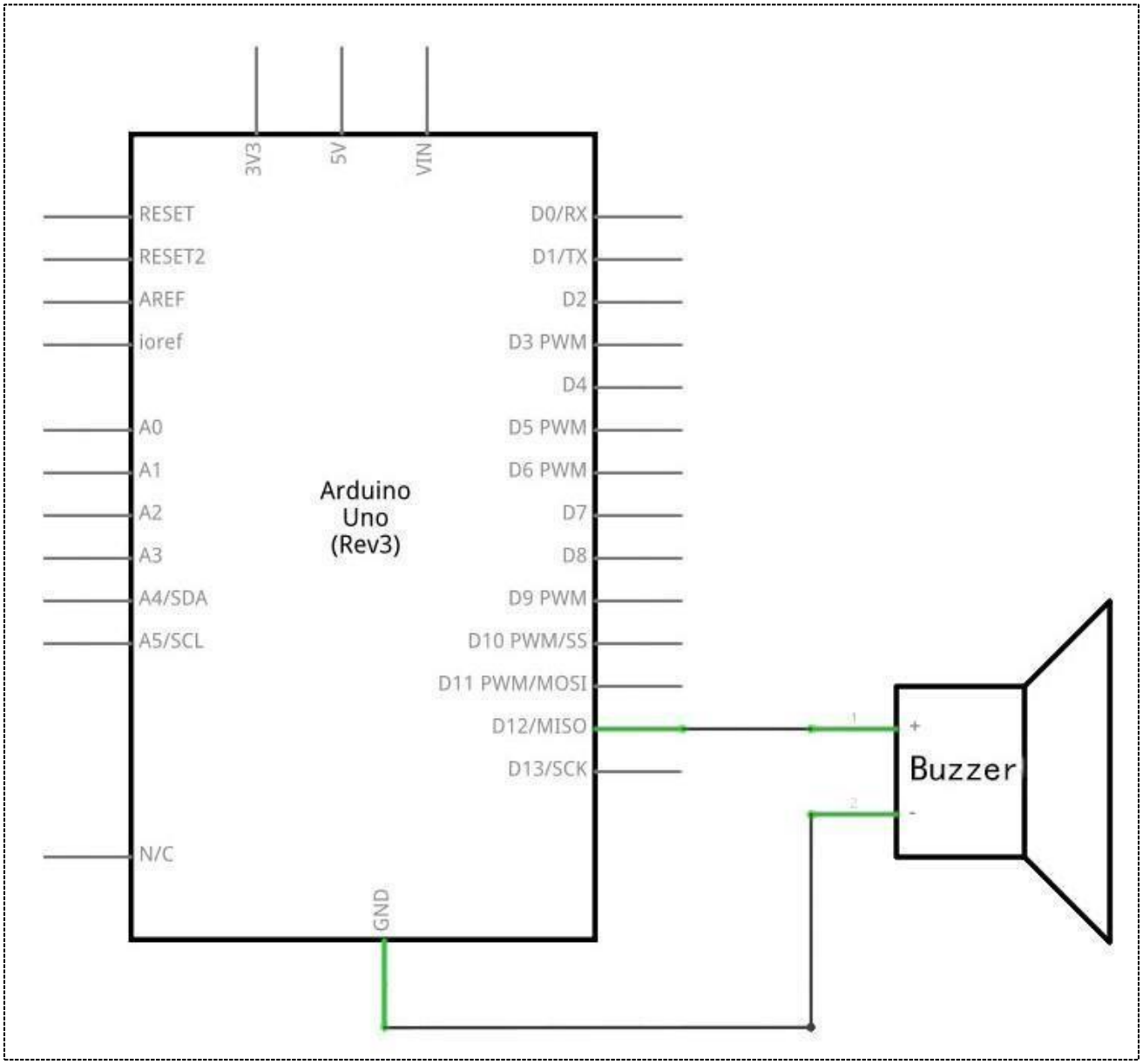
Electronic buzzers are DC-powered and equipped with an integrated circuit. They are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products for voice devices.

Buzzers can be categorized as active and passive ones. Turn the pins of two buzzers face up. The one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.

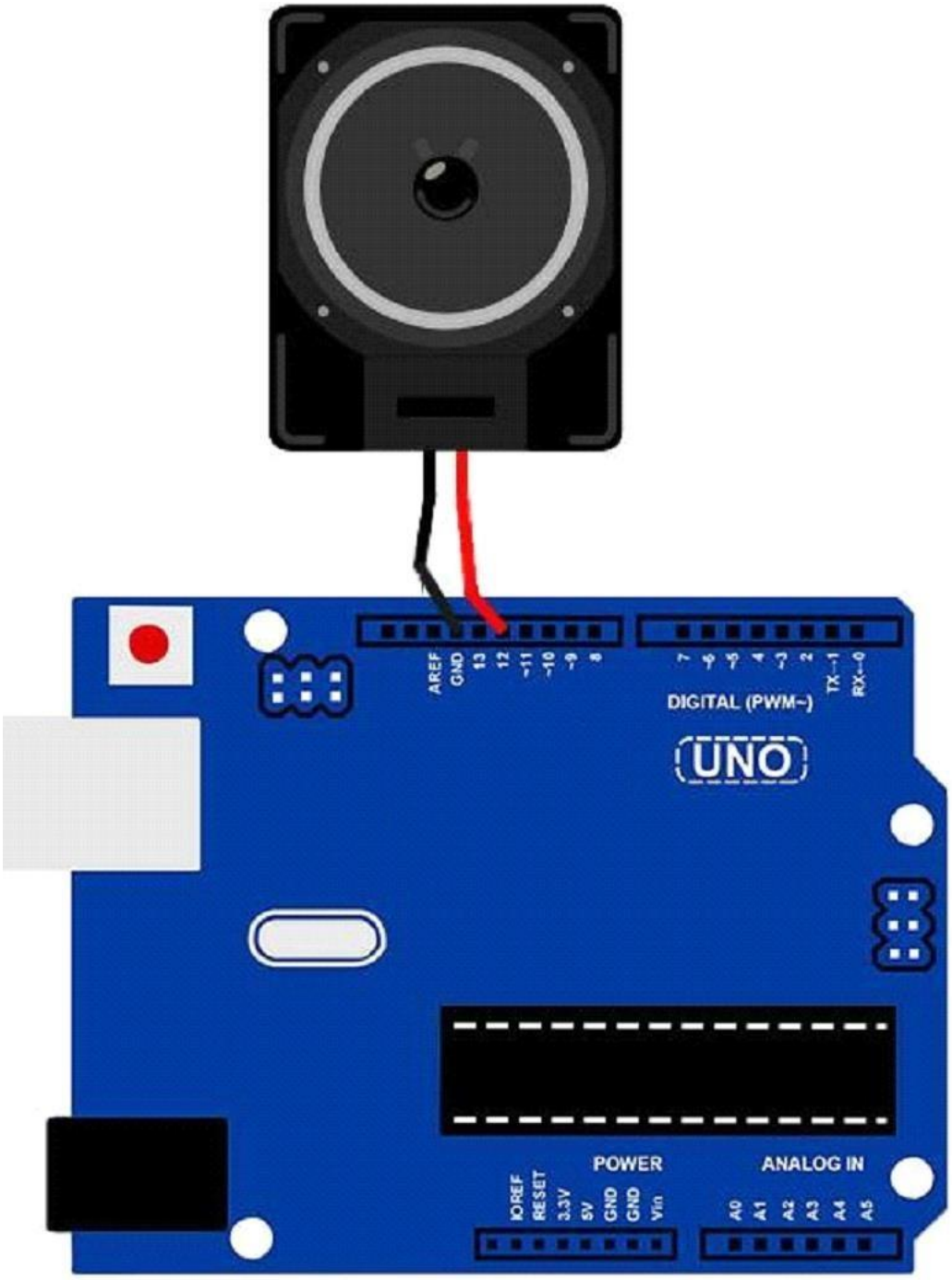
The difference between the two is that an active buzzer has a built-in oscillating source, so it will generate a sound when electrified. A passive buzzer does not have such a source so it will not tweet if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K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



Connection Schematic

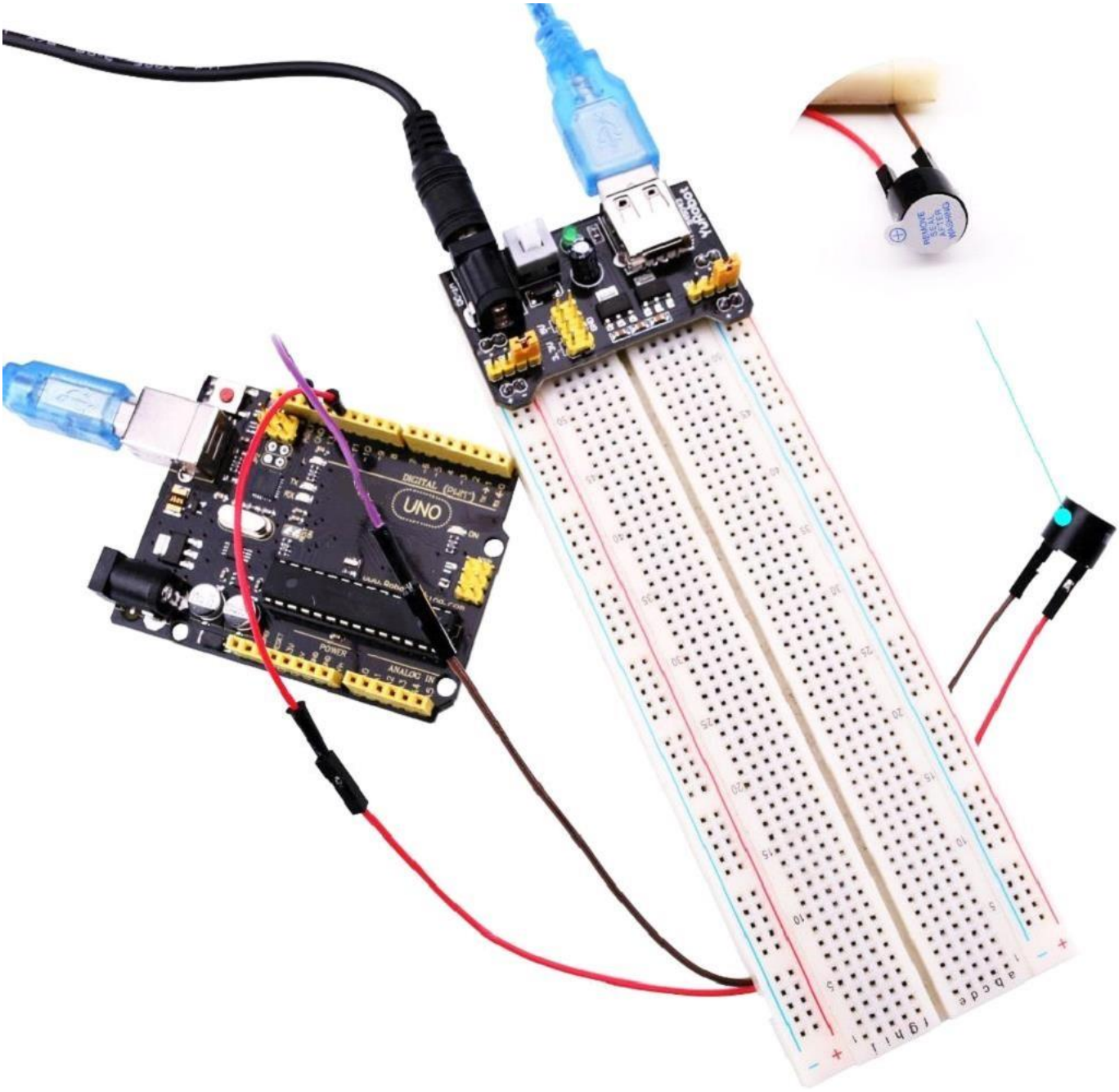


wiring diagram



Code

See the code file.



Lesson 9 Passive Buzzer

Overview

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

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

Component Required:

(1) x Quaduino Uno R3

(1) x Passive buzzer

(2) x F-M wires

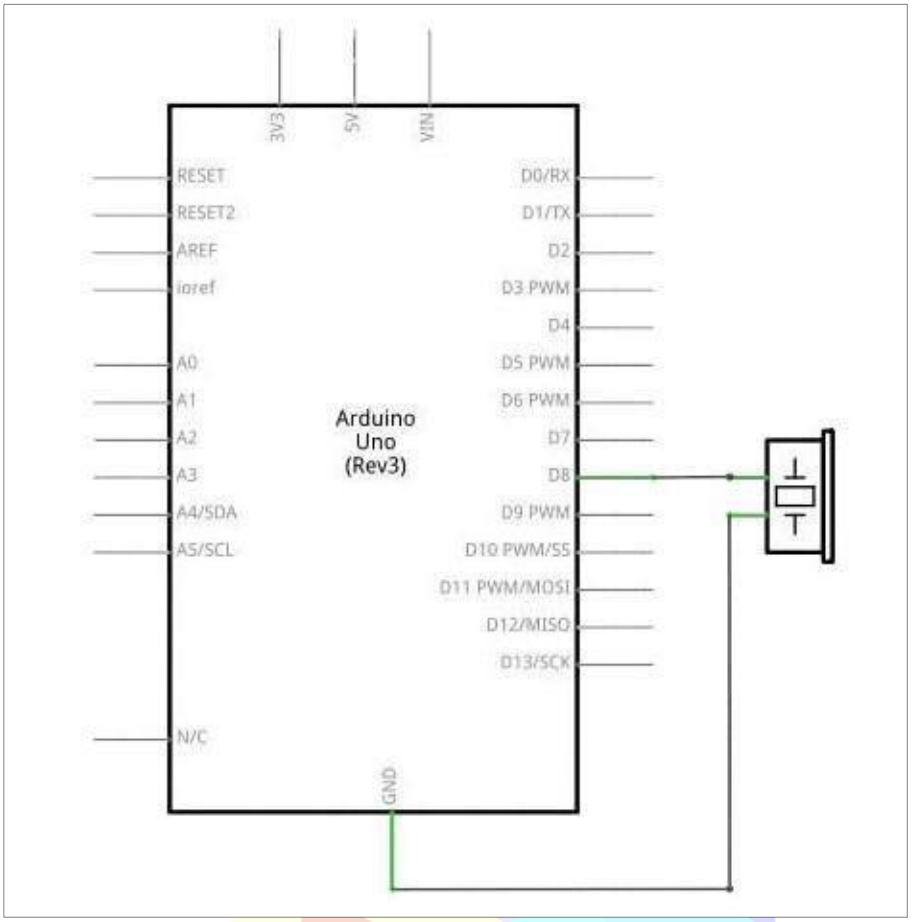
Component Introduction

Passive Buzzer:

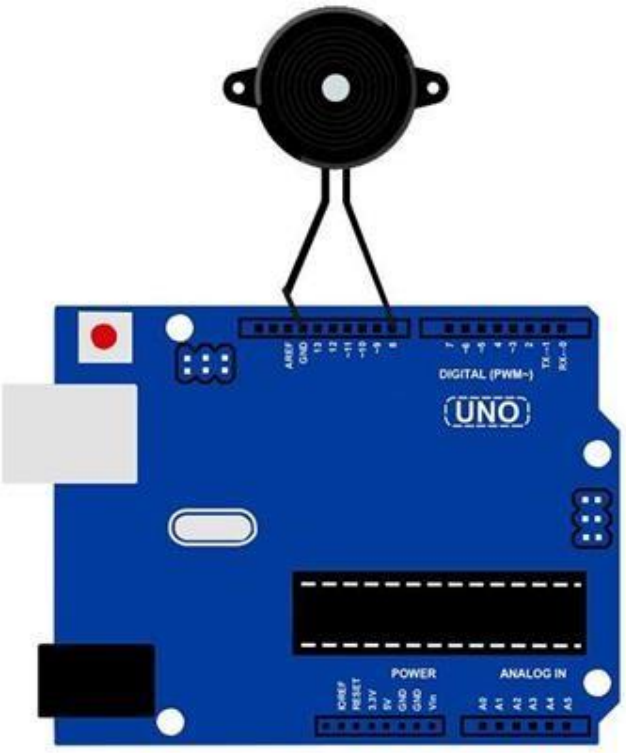
The working principle of passive buzzer it to use PWM generating audio to make the air to vibrate. Appropriately changed as long as the vibration frequency, it can generate different sounds. For example, sending a pulse of 523Hz, it can generate Alto Do, pulse of 587Hz, it can generate midrange Re, pulse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

We should be careful not to use the UNO R3 board analog Write () function to generate a pulse to the buzzer, because the pulse output of analog Write () is fixed (500Hz).

Connection Schematic



wiring diagram



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

Code

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

Lesson 10 Ball Switch

Overview

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

Component Required:

(1) x Quaduino Uno R3

(1) x Ball switch

(2) x F-M wires

Component Introduction

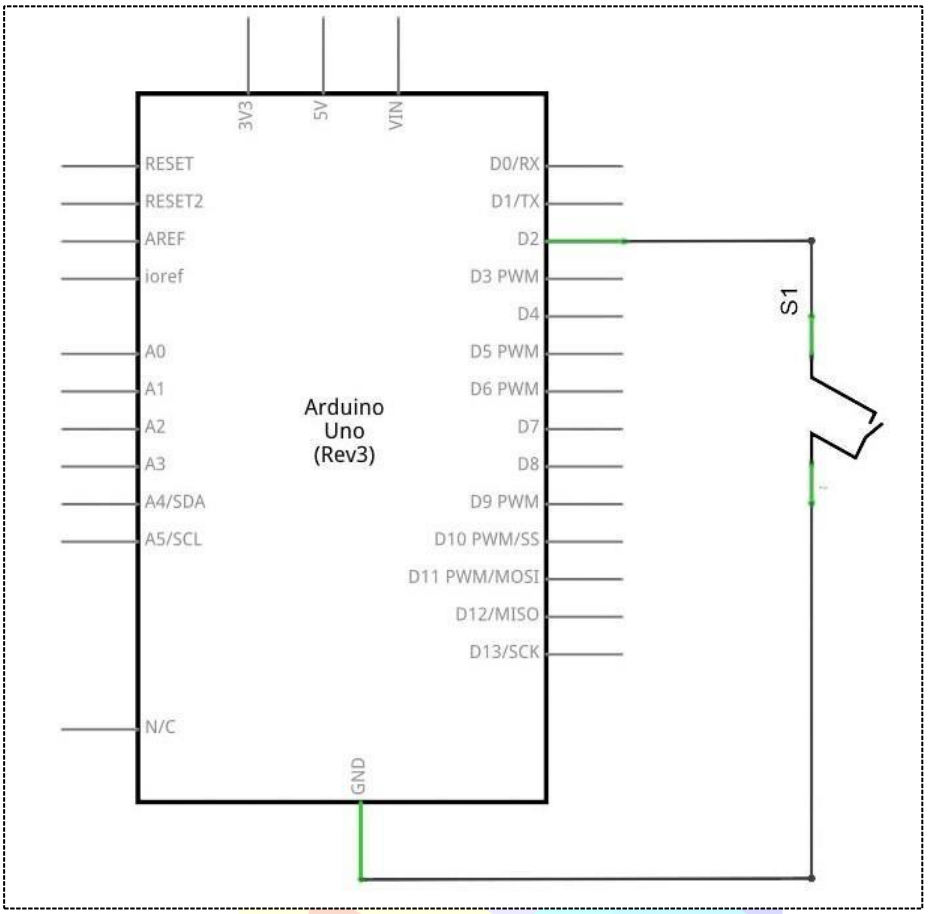
Tilt sensor:

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

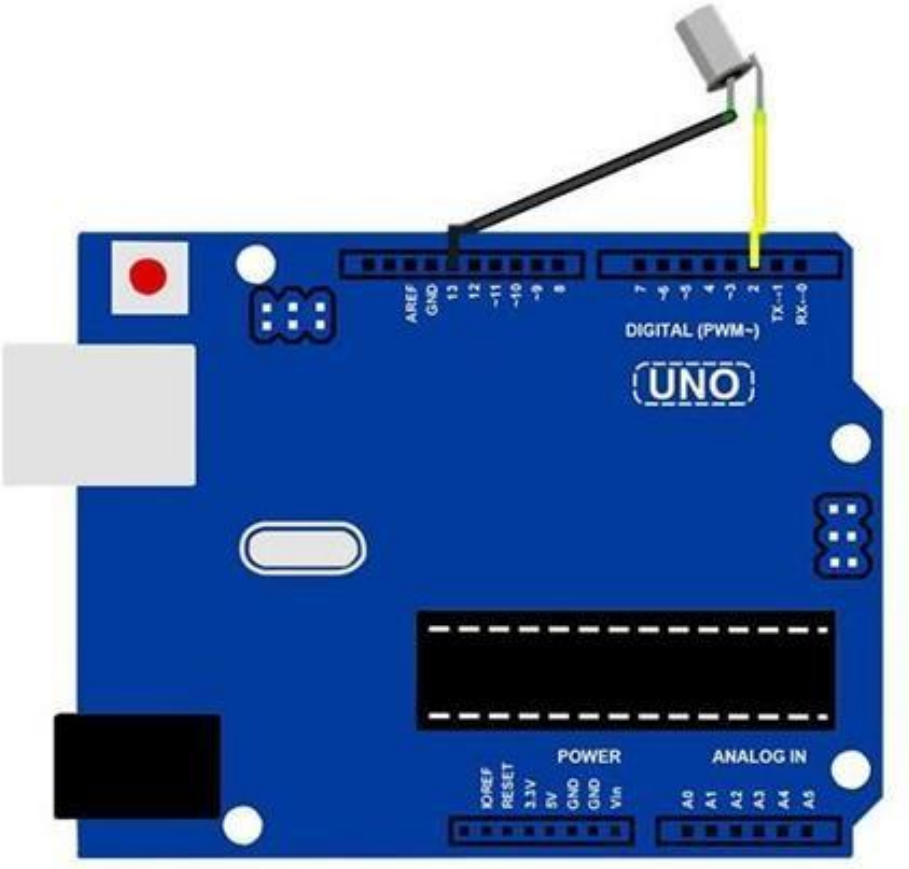
They are usually made up of a cavity of some sort (cylindrical is popular, although not always) with a conductive free mass inside, such as a blob of mercury or rolling ball. One end of the cavity has two conductive elements (poles). When the sensor is oriented so that that end is downwards, the mass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

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

Connection Schematic

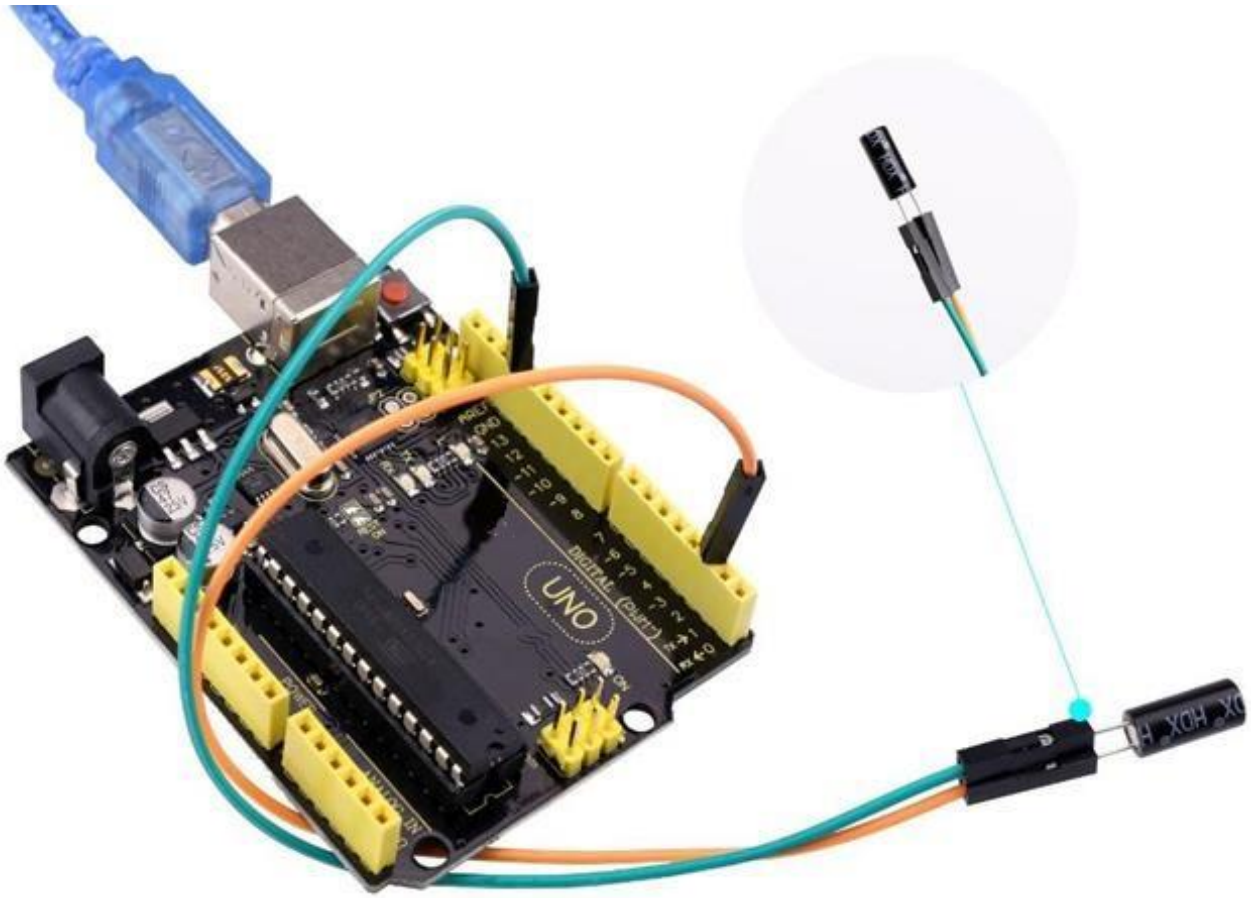


wiring diagram



Code

See the code file.



Lesson 11 Relay Module

Overview

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

Component Required:

(1) x Quaduino Uno R3

(1) x Relay module

(3) x F-M wires

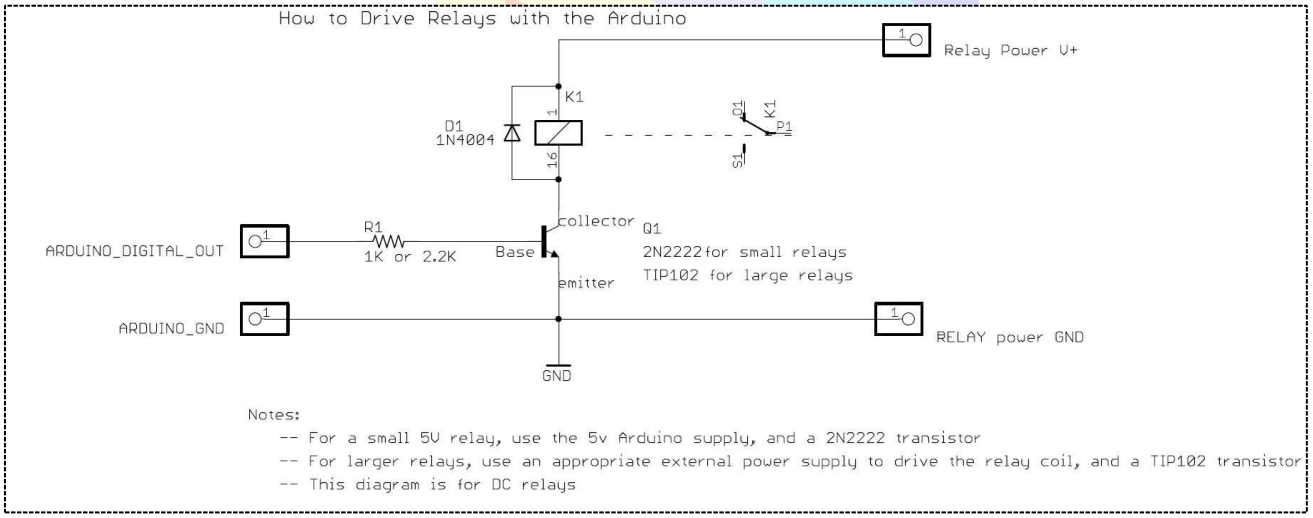
Component Introduction

Relay:

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

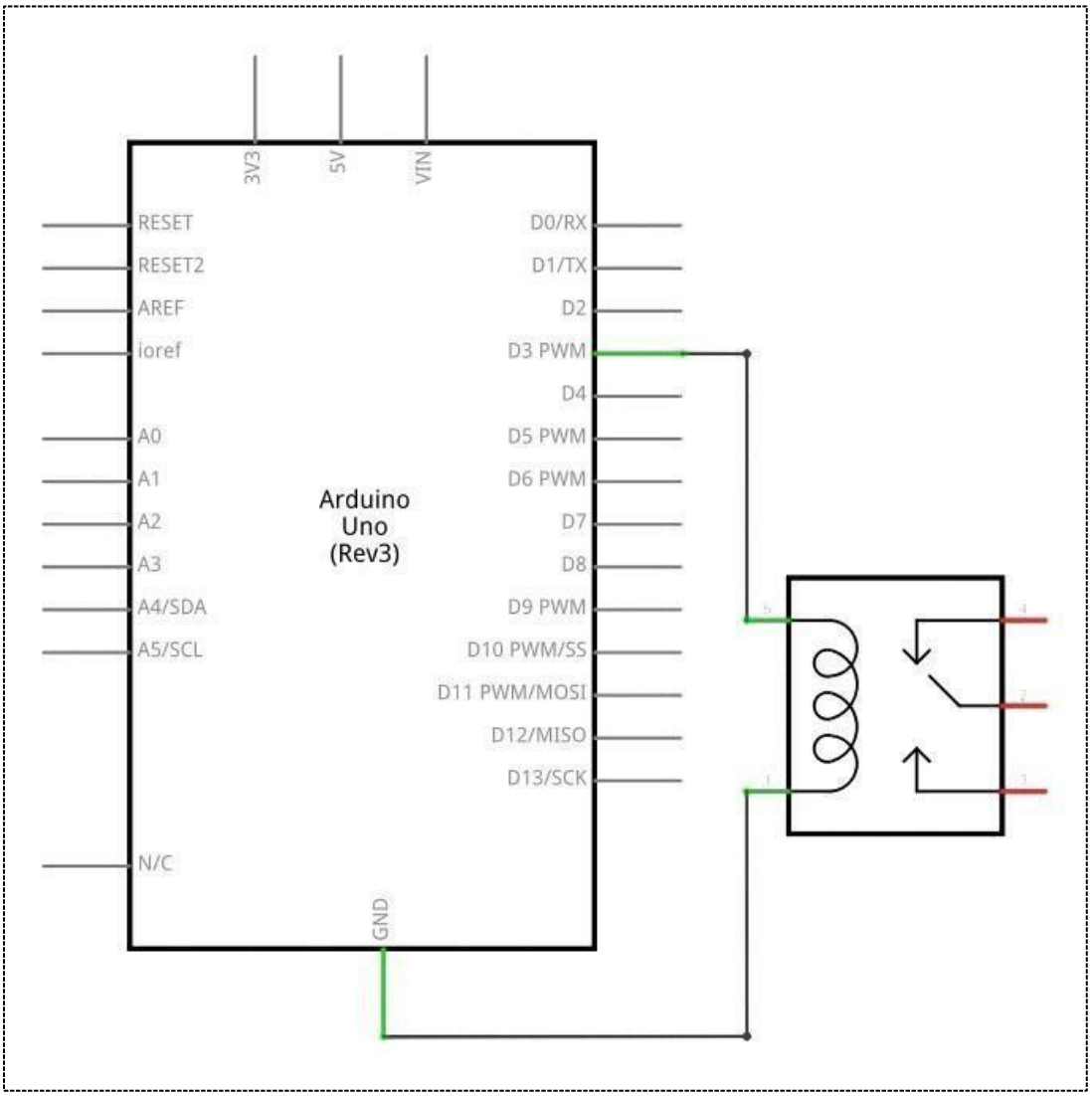
A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform the switching. Relays with calibrated operating characteristics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

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

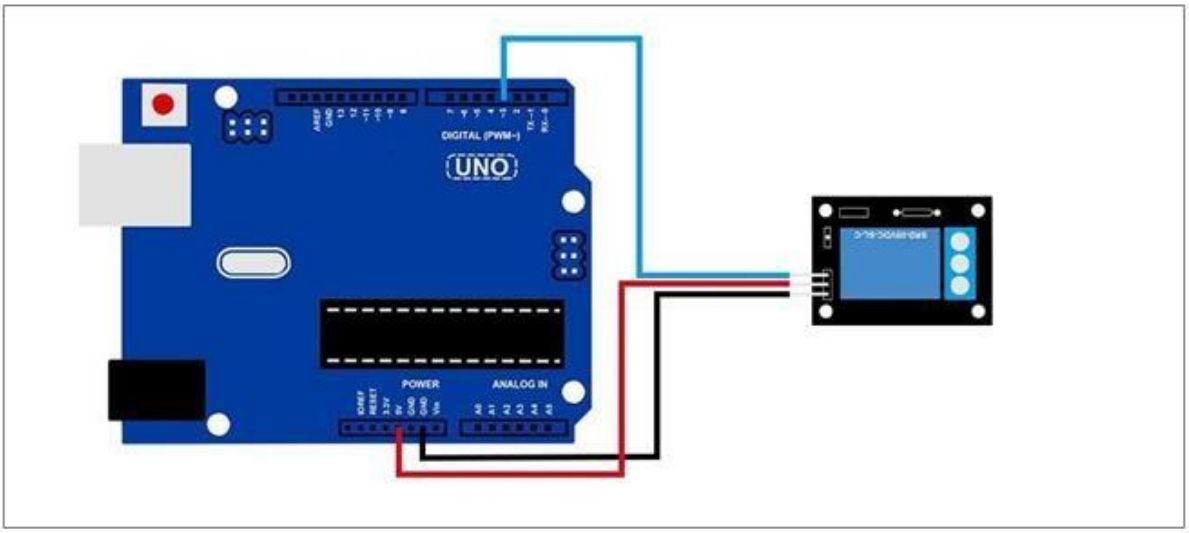


Connection

Schematic

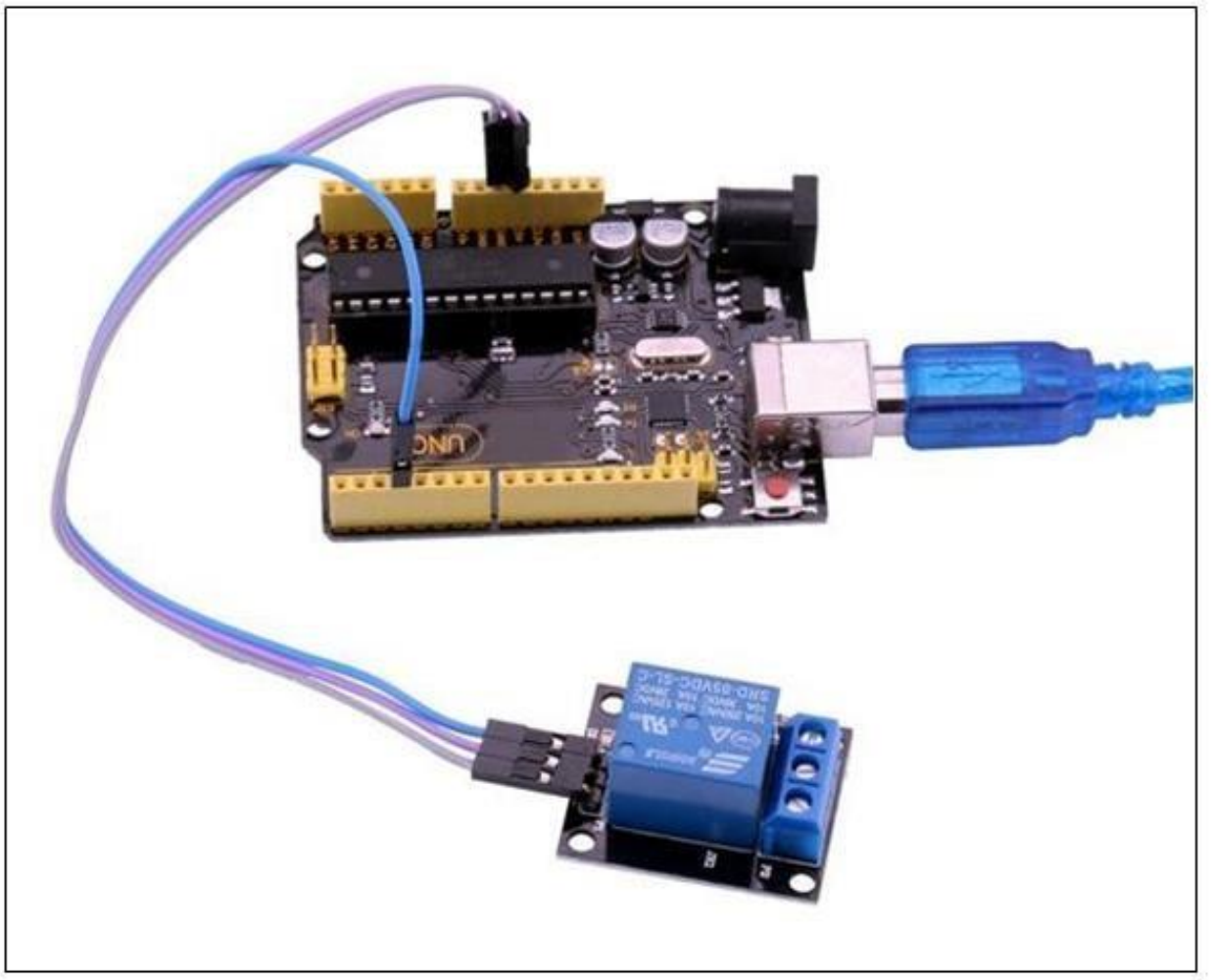


wiring diagram



The code

See the code file.



Lesson 12 74HC595 And Segment Display

Overview

After learning Lesson 6 and Lesson 7, we will use the 74HC595 shift register to control the segment display. The segment display will show number from 9-0.

Component Required:

(1) x Quaduino Uno R3

(2) x Breadboard

(1) x 74HC595

(1) x Segment Display

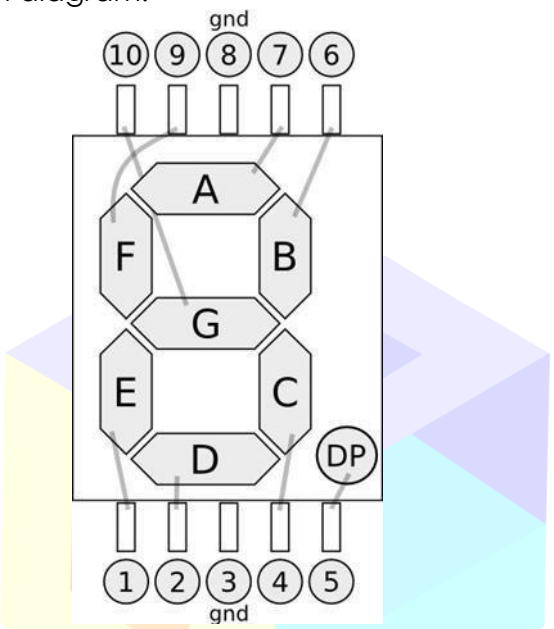
(8) x 220 ohm resistor

(20) x M-M wires

Component Introduction

Seven segment display

Below is the seven-segment pin diagram.

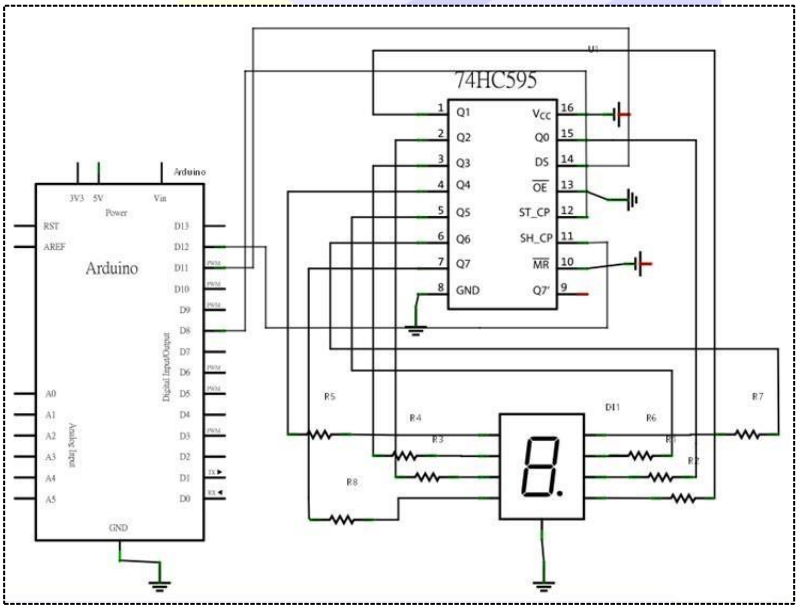


0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device, if you are using a common anode, the table should be replaced every 1 0 0 should all replaced by 1):

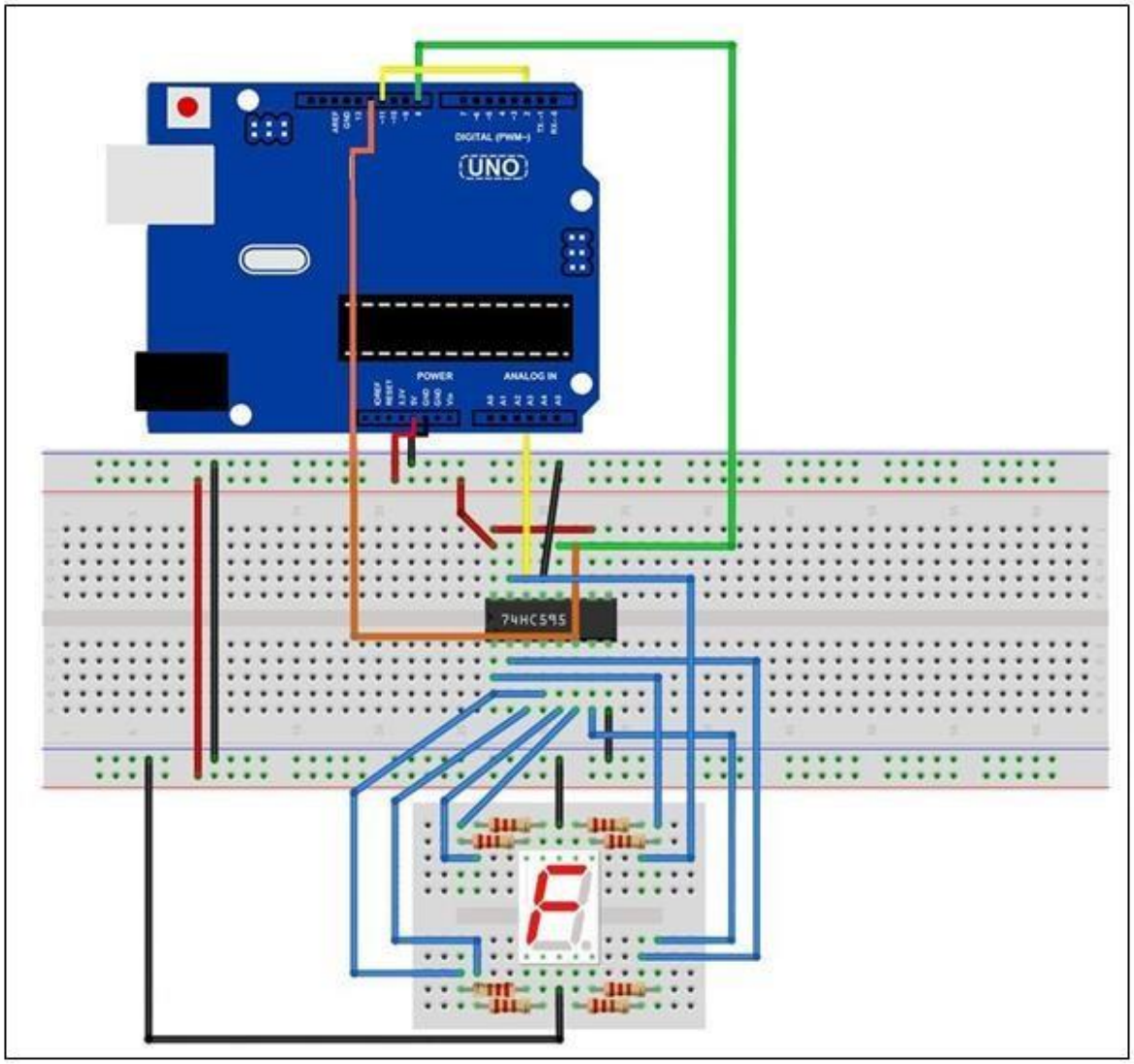
Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1

4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

Connection Schematic



wiring diagram



The following table shows the seven-segment display 74HC595 pin correspondence table:

74HC595 pin	Seven shows remarkable control pin (stroke)
Q0	7 (A)
Q1	6 (B)
Q2	4 (C)
Q3	2 (D)

Q4	1 (E)
Q5	9 (F)

Q6	10 (G)
Q7	5 (DP)

Step one: Connect 74HC595

First, the wiring is connected to power and ground:

- Vcc (pin 16) and MR (pin 10) connected to 5V
- GND (pin 8) and OE (pin 13) to ground

Connection DS, ST_CP and SH_CP pin:

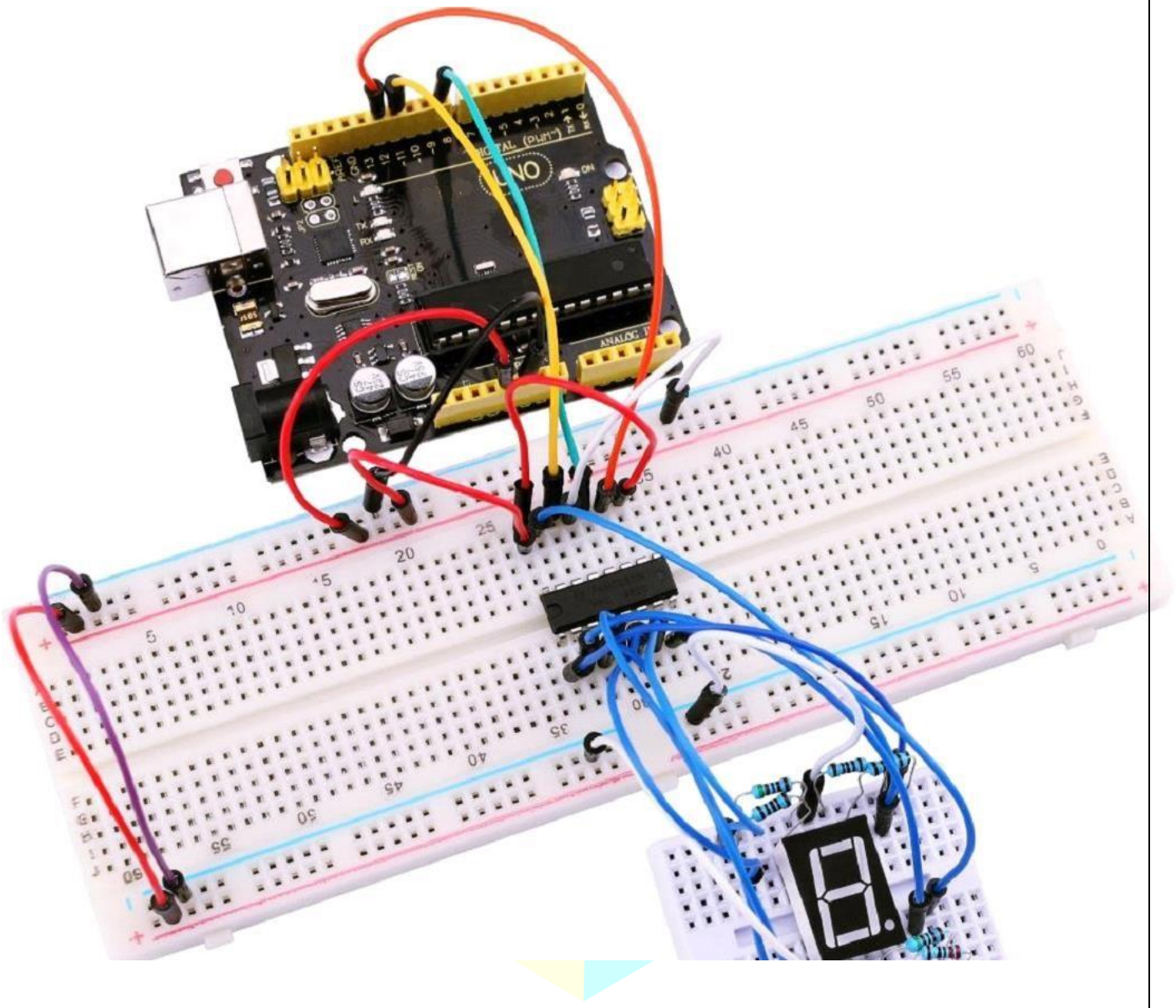
- DS (pin 14) connected to UNO R3 board pin 11 (the figure below the blue line)
- ST_CP (pin 12, latch pin) connected to UNO R3 board pin 8 (FIG green line below)
- SH_CP (pin 11, clock pin) connected to UNO R3 board pin 12 (the figure below the yellow line)

Step two: Connect the seven segment display

- The seven-segment display 3, 8 pin to UNO R3 board GND (This example uses the common cathode, if you use the common anode, please connect the 3, 8 pin to UNO R3 board + 5V)
- According to the table above, connect the 74HC595 Q0 ~ Q7 to seven-segment display corresponding pin (A ~ G and DP), and then each foot in a 220 ohm resistor in series.

Code

See the code file.



Lesson 13 Four Digital Seven Segment Display

Overview

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

When using 1-digit 7-segment display and it is common anode, the common anode pin connects to the power source; if it is common cathode, the common cathode pin connects to the GND.

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

Component Required:

(1) x Quaduino Uno R3

(1) x Breadboard

(1) x 74HC595

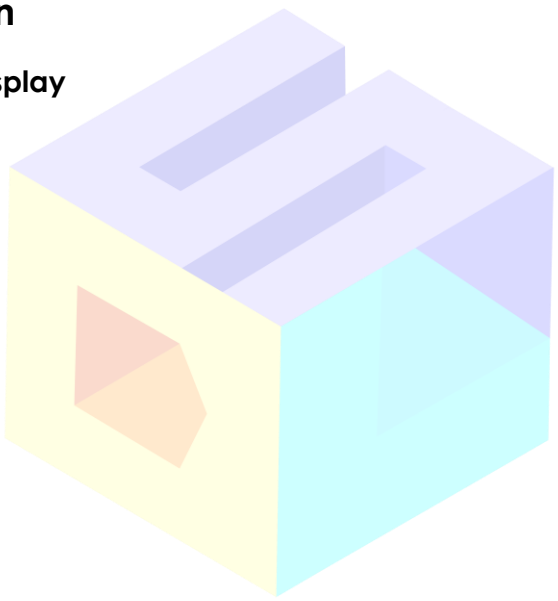
(1) x Four Digital Tube Segment Display

(4) x 220 ohm resistor

(20) x M-M wires

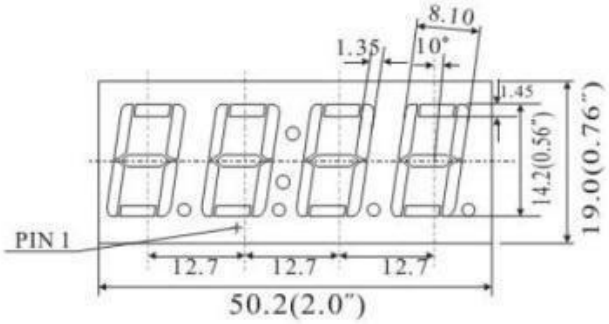
Component Introduction

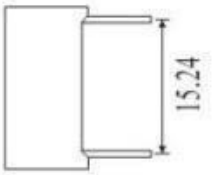
Four Digital Seven segment display

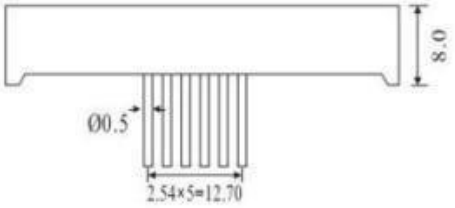


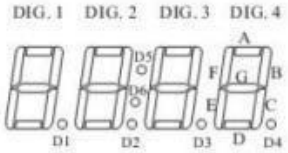
Package Dimensions

CPS05643AB



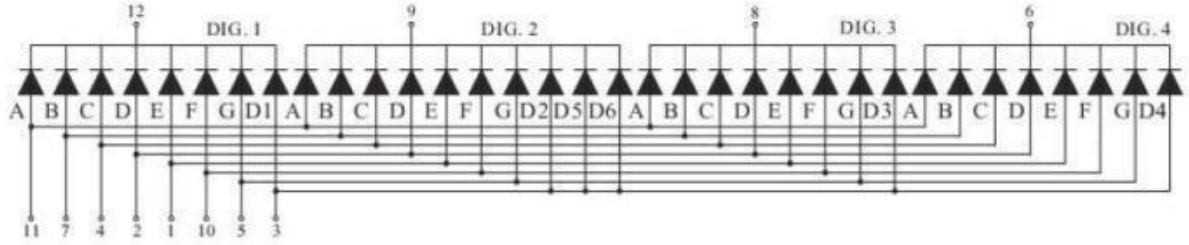




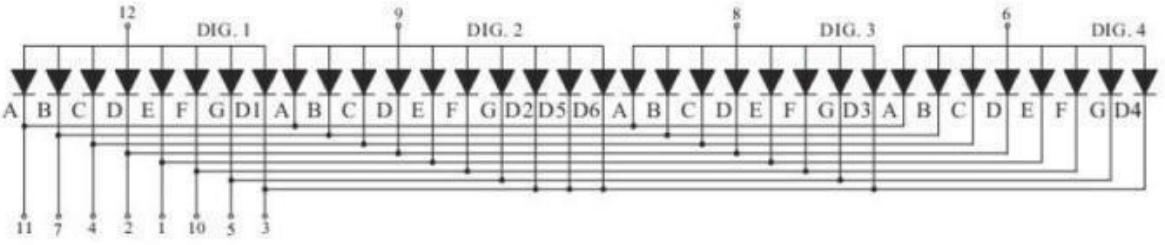


UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

Internal Circuit Diagram



5643A

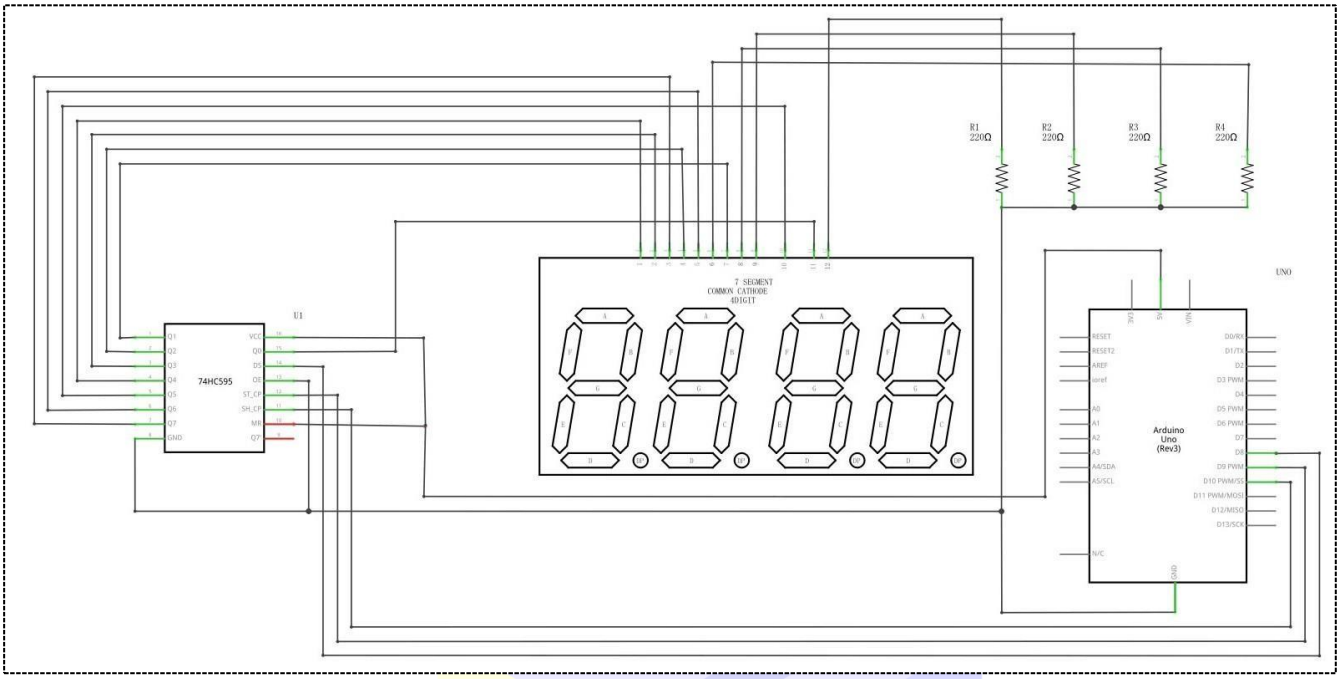


5643B

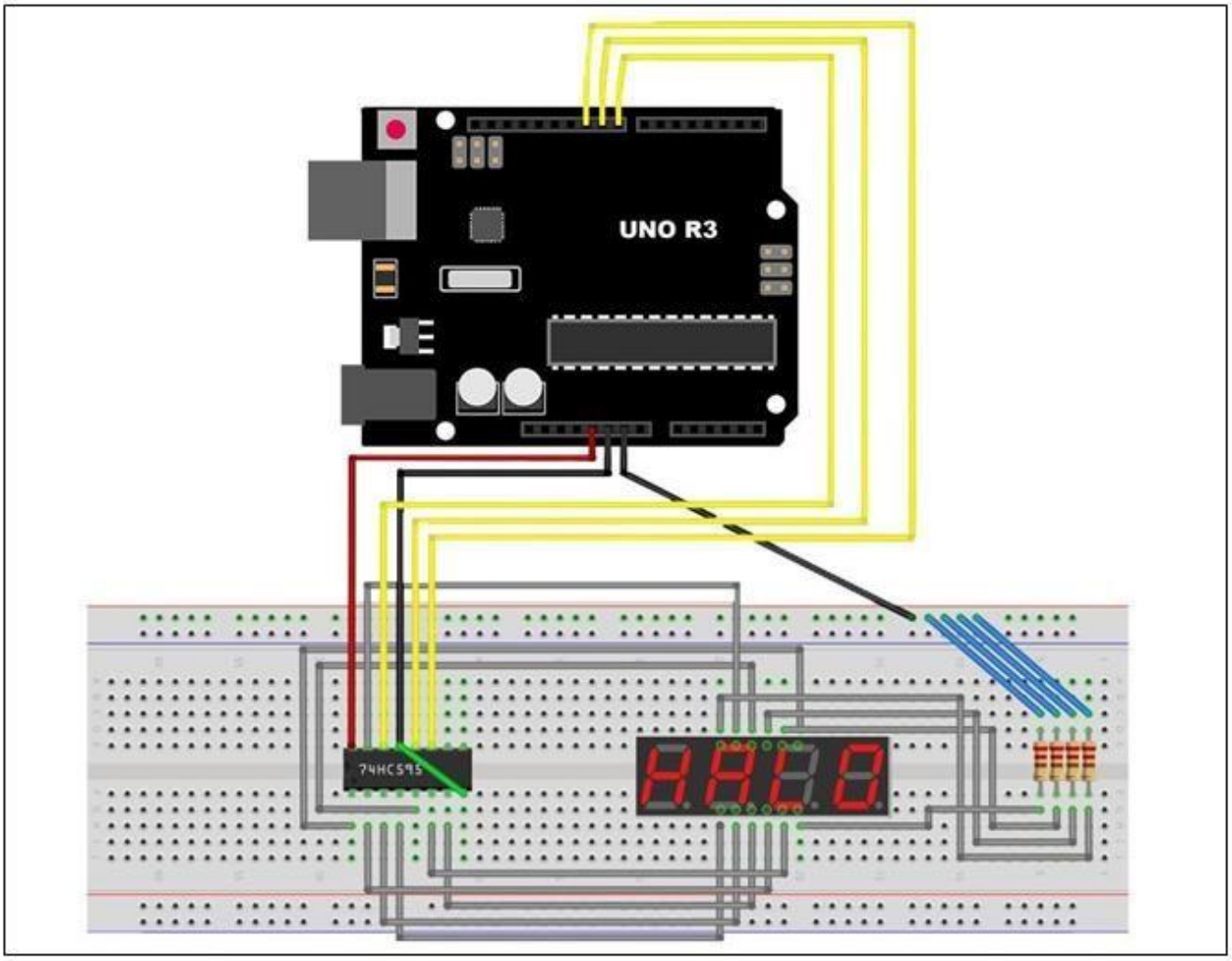
Four Digits Displays Series

Connection

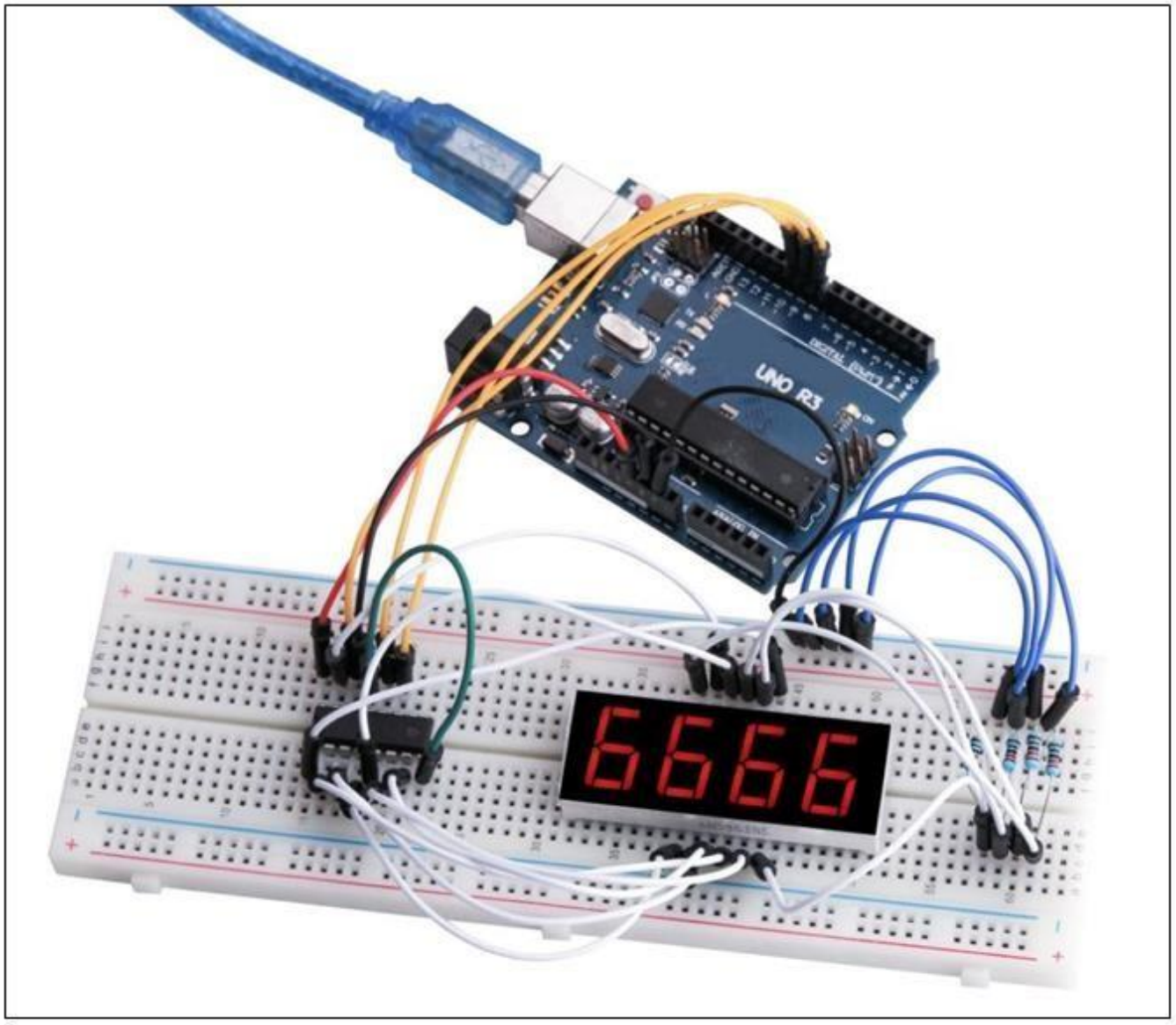
Schematic



wiring diagram



Code



See the code file.

Lesson 14 LED Dot Matrix

Overview

In this lesson, we will use the MAX7219 to drive a matrix digital tube.

Since we use the MAX7219 LED driver chip, we will be able to turn on and off the LEDs by using only 3 pins on our UNO.

Component Required:

(1) x Quaduino Uno R3

(1) x Breadboard

(1) x Max7219

(1) x 1k ohm resistor

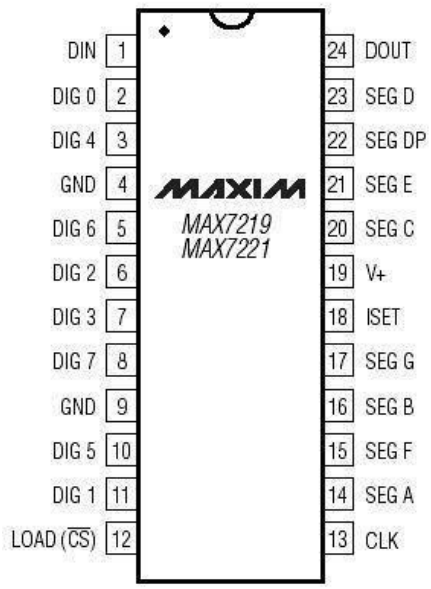
(1) x 10uf capacitor

(1) x 100nf ceramic capacitor

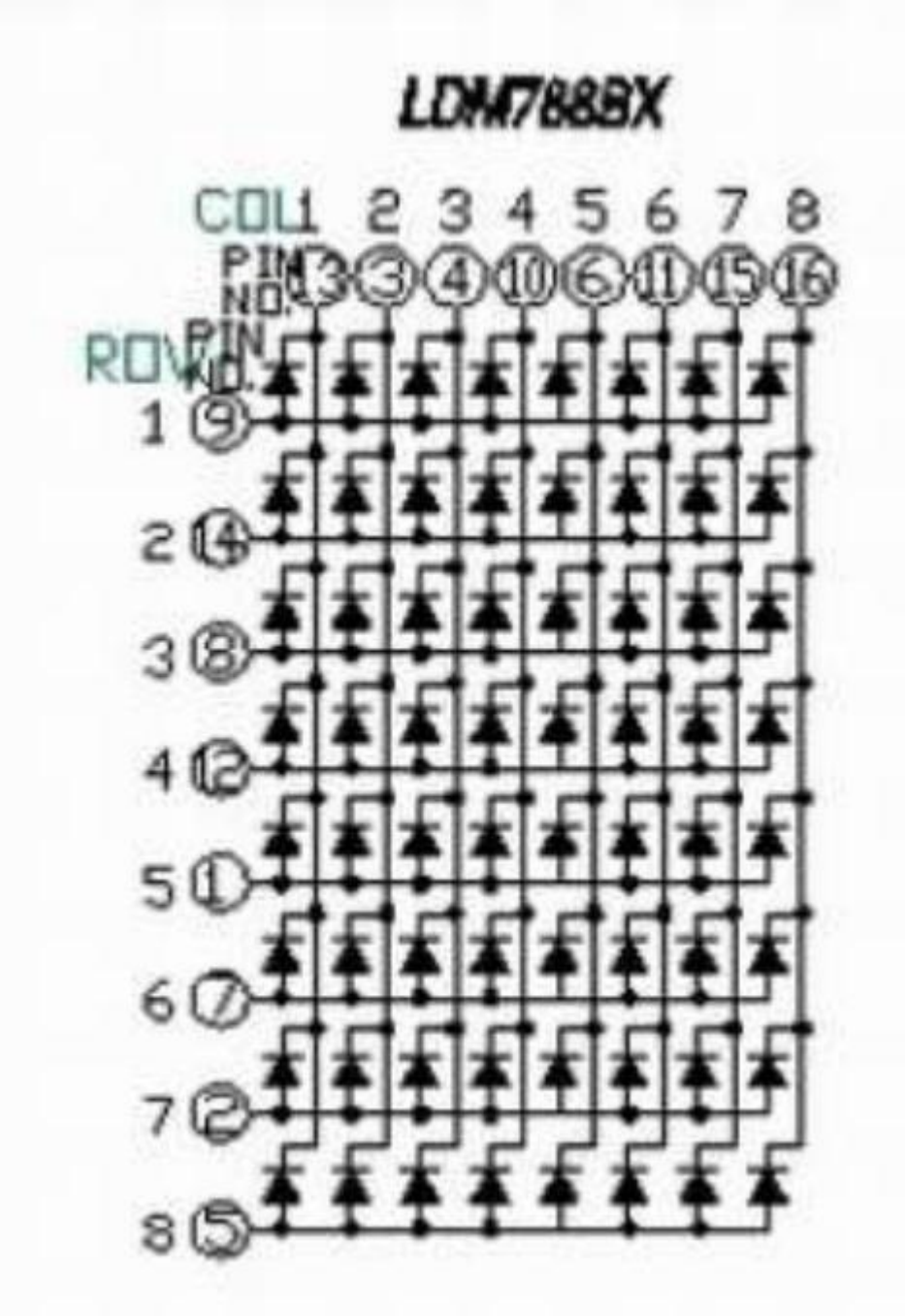
(1) x LED dot matrix

Component Introduction

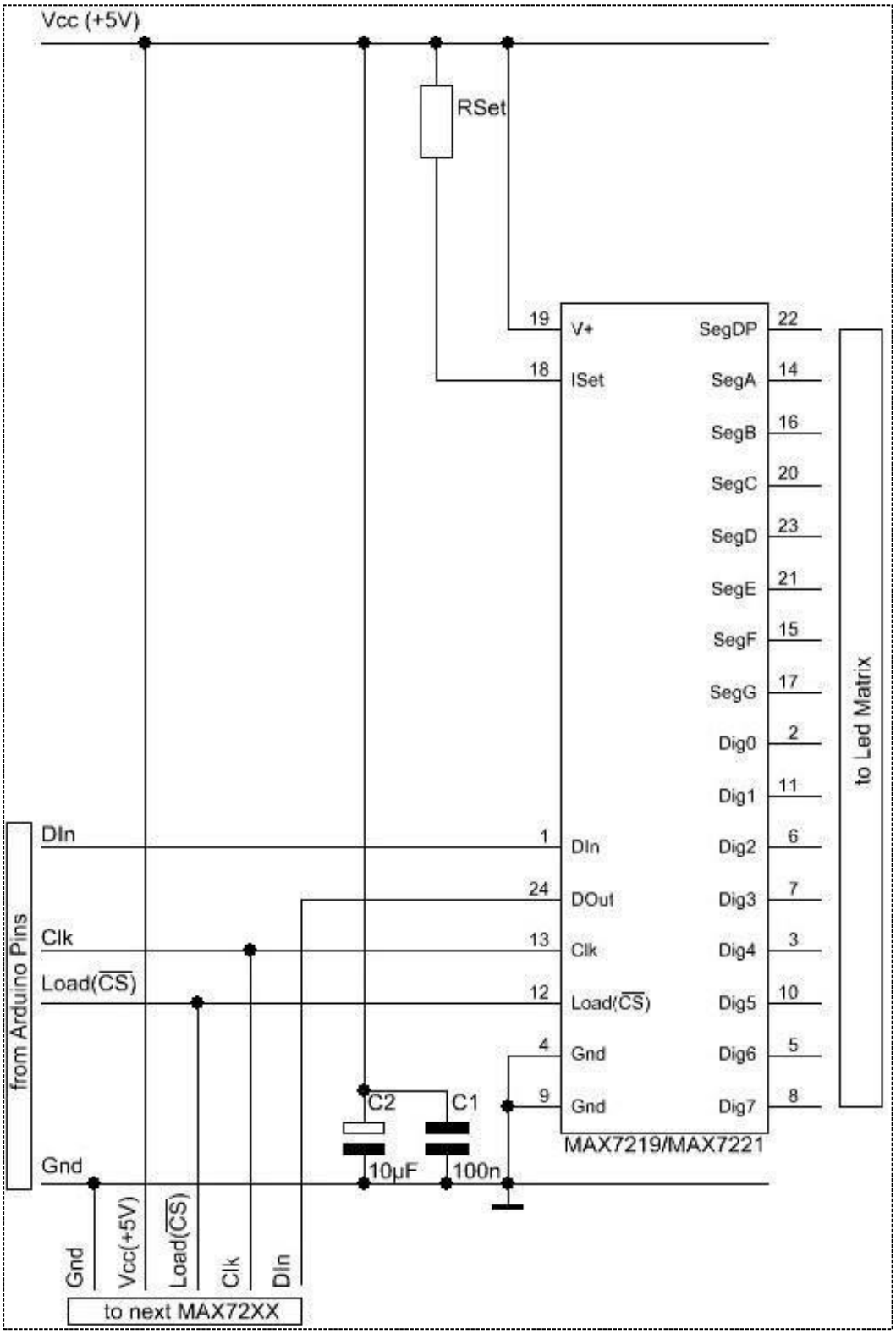
MAX7219:



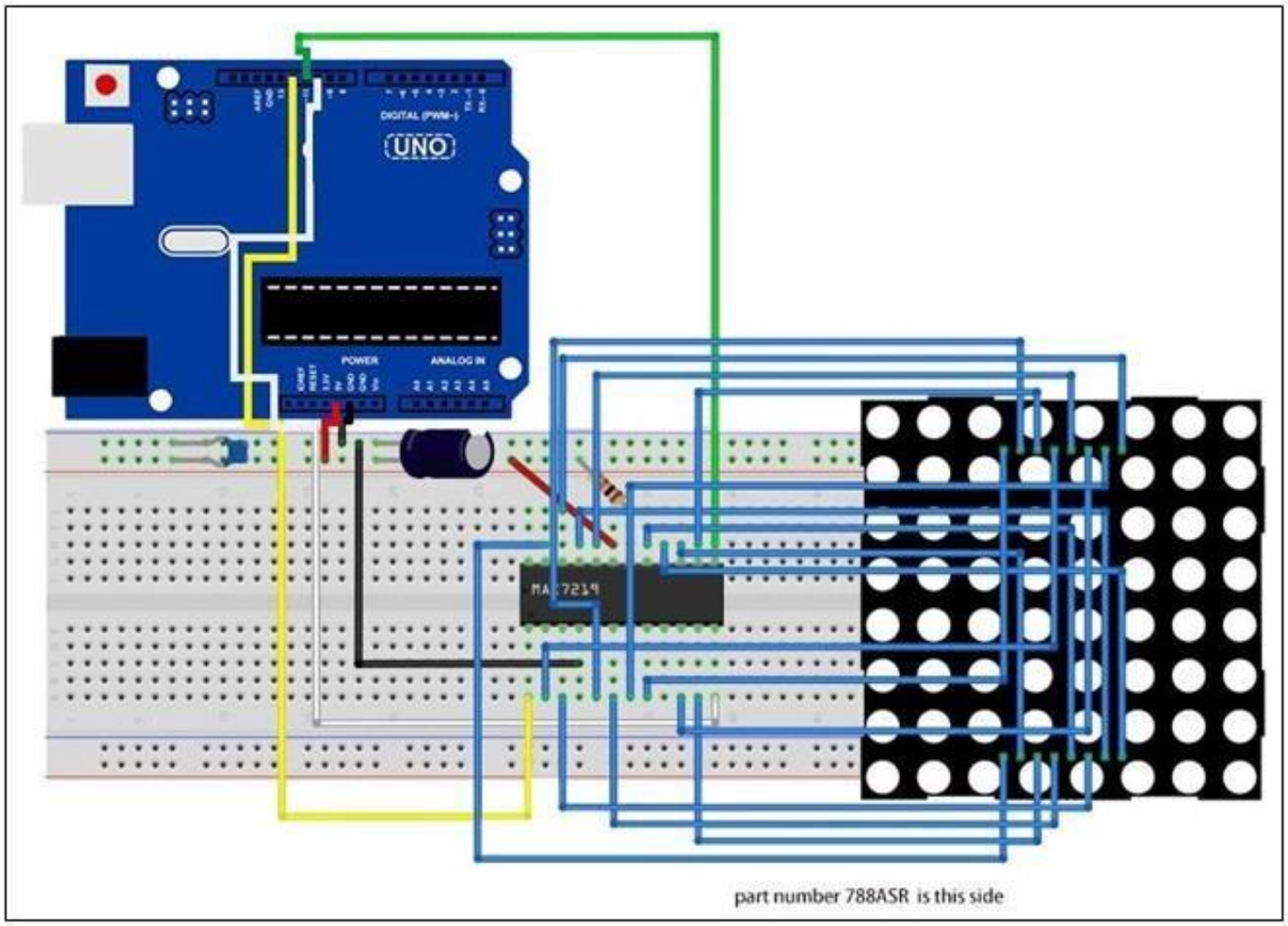
MAX7219:



Connection schematic


Vcc (+5V)
RSet
19
V+
18
ISet
SegDP
22
SegA
14
SegB
16
SegC
20
SegD
23
SegE
21
SegF
15
SegG
17
Dig0
2
Dig1
11
Dig2
6
Dig3
7
Dig4
3
Dig5
10
Dig6
5
Dig7
8
to Led Matrix
DIn
1
DIn
24
DOut
Clk
13
Clk
Load(CS)
12
Load(CS)
4
Gnd
9
Gnd
C2
C1
10µF
100n
MAX7219/MAX7221
from Arduino Pins
Gnd
Gnd
Vcc(+5V)
Load(CS)
Clk
DIn
to next MAX72XX

wiring diagram



The Code

Our Sketch will make use of the "LedControl" Library to communicate with the MAX7219 modules.

Now that we have the physical setup, all we need now is the code.

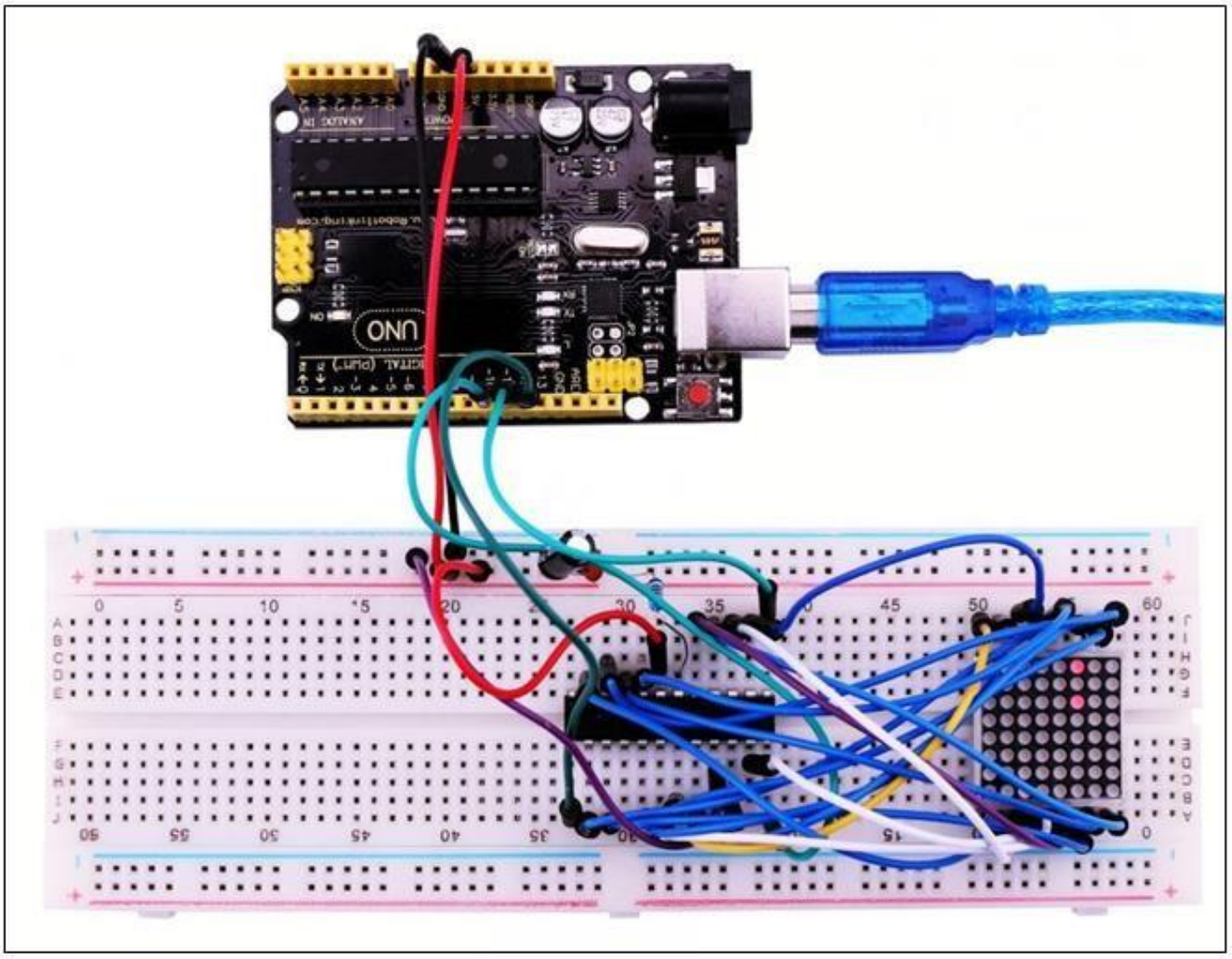
Before you can run this, you have to make sure that you have install the < LedControl> library. Or you need to install again. If you do not do this, your code won't work.

Since we are defining all the characters possible (0-9 a-z etc..), instead of using the SRAM (2048 bytes), we will put that information in Flash memory on our UNO since that information will not change and also there"s more Flash memory available (32k).

We do this by using the "#include <avr/pgmspace.h>" and then "PROGMEM prog_uchar CH[] =" to put our array information in Flash memory.

Note: Flash (PROGMEM) memory can only be populated when we upload the code to our UNO. You can"t change the values in the flash after the program has started running, unlike

SRAM where our sketch is run.



Lesson 15 Servo

Overview

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your UNO R3 board. These pulses tell the servo what position it should move to. A servo has three wires, the brown wire is GND, the red one is VCC, and the orange one is signal line.

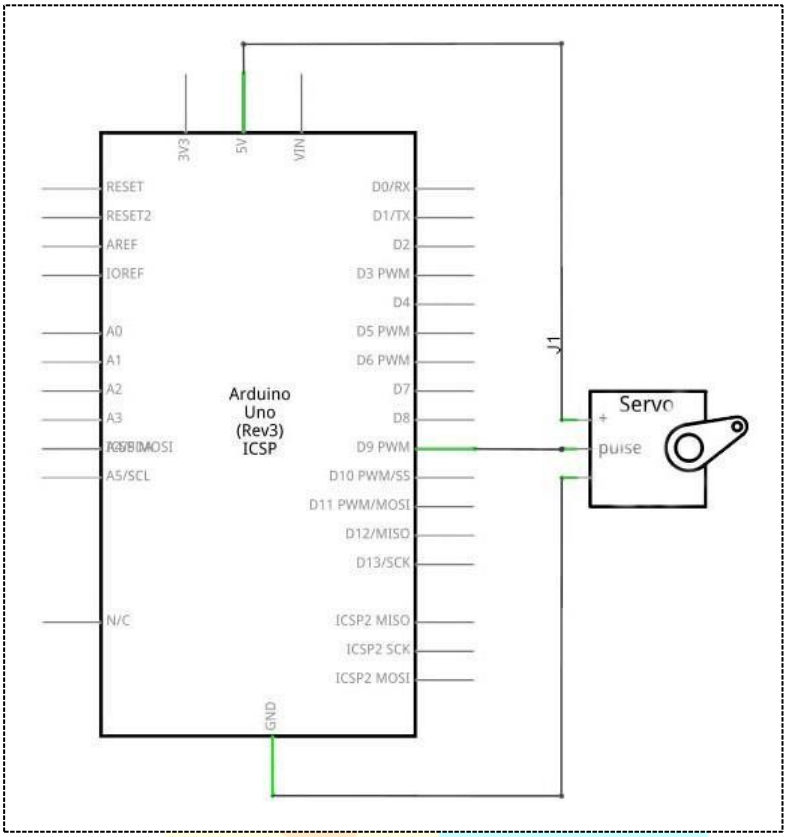
Component Required:

(1) x Quaduino Uno R3

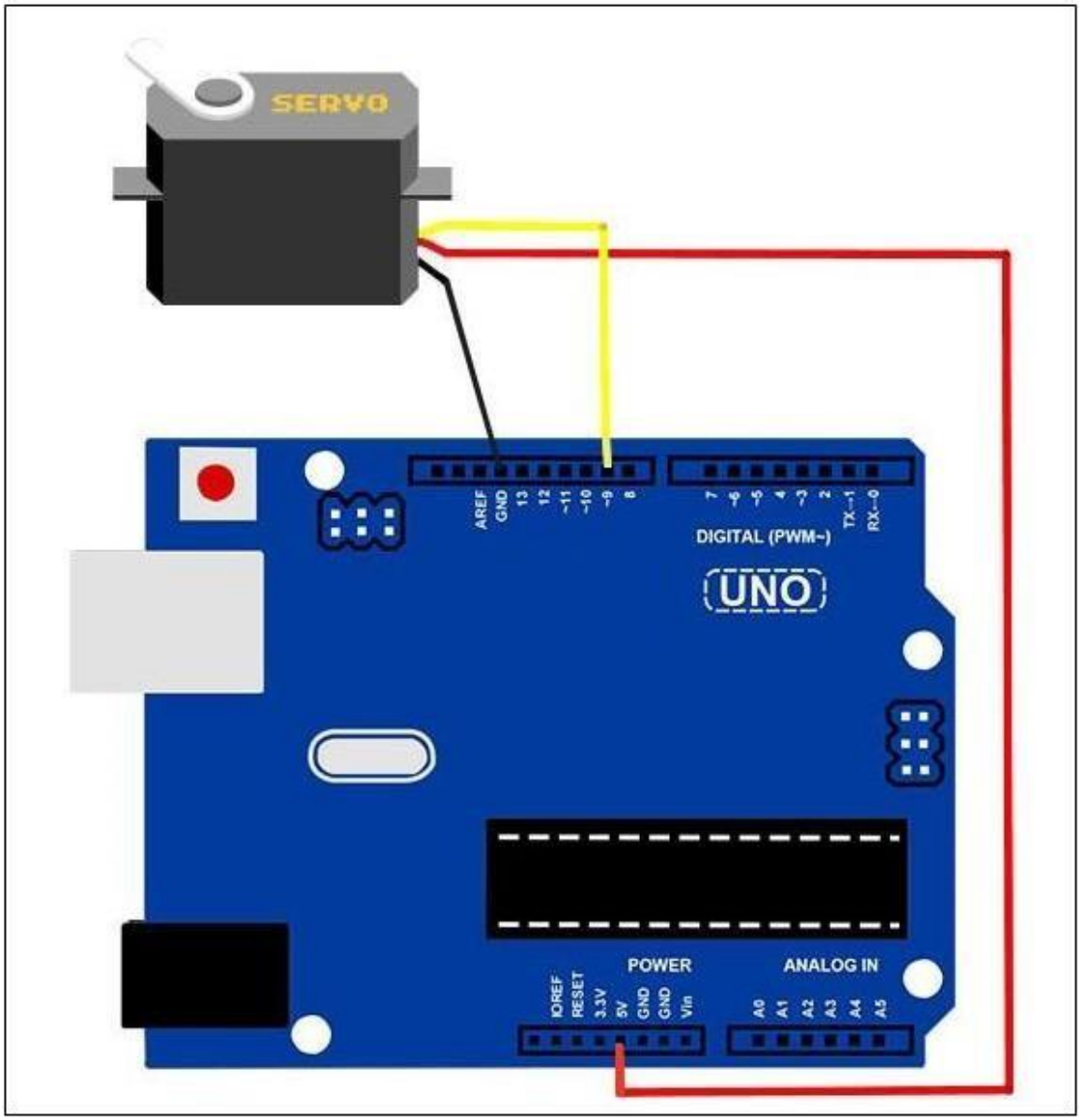
(1) x Servo

(2) x F-M wires

Connection Schematic

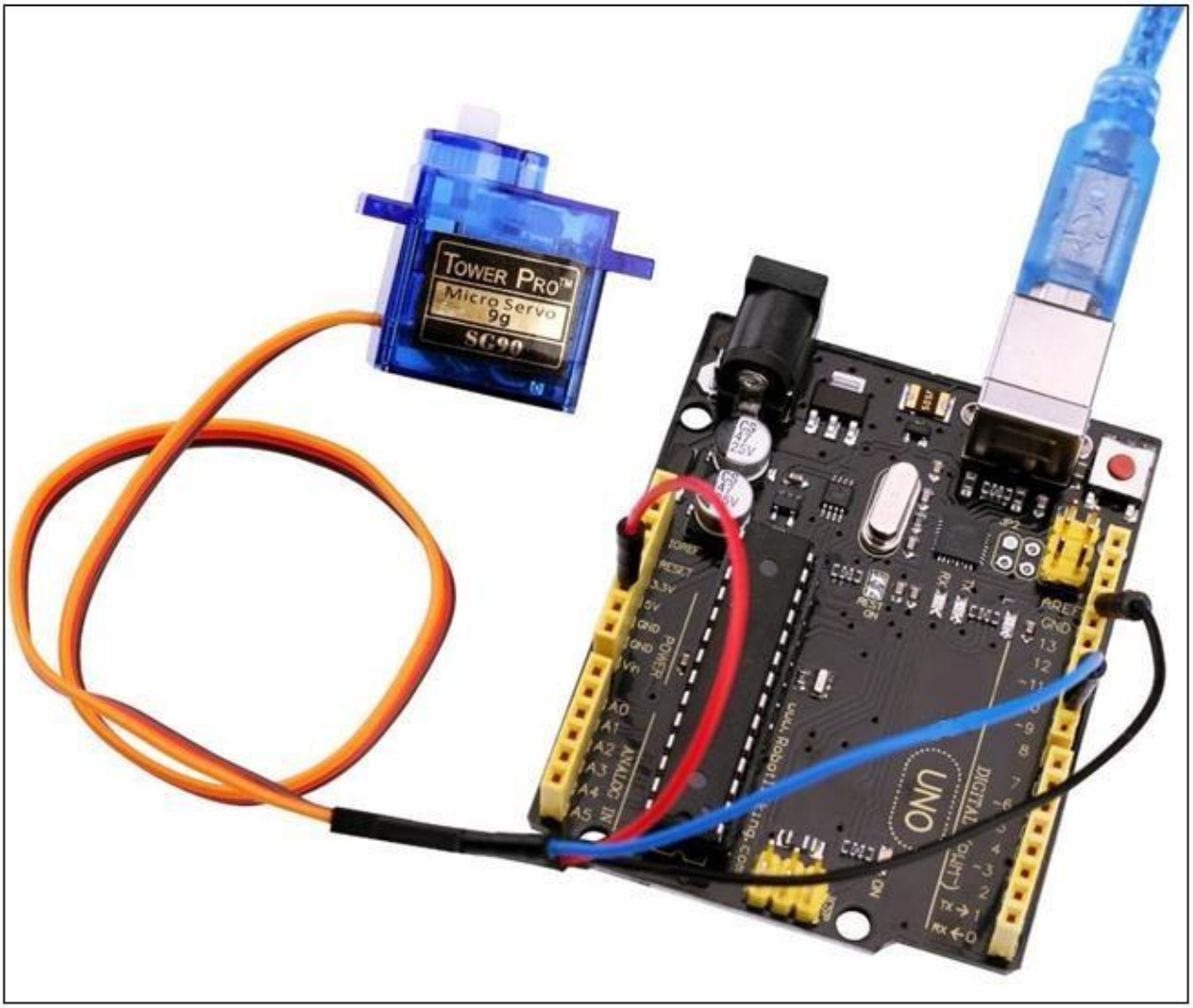


wiring diagram



Code

See the code file.



Lesson 16 Stepper Motor

Overview

In this lesson, you will learn a fun and easy way to drive a stepper motor.

The stepper we are using comes with its own driver board making it easy to connect to our UNO.

Component Required:

(1) x Quaduino Uno R3

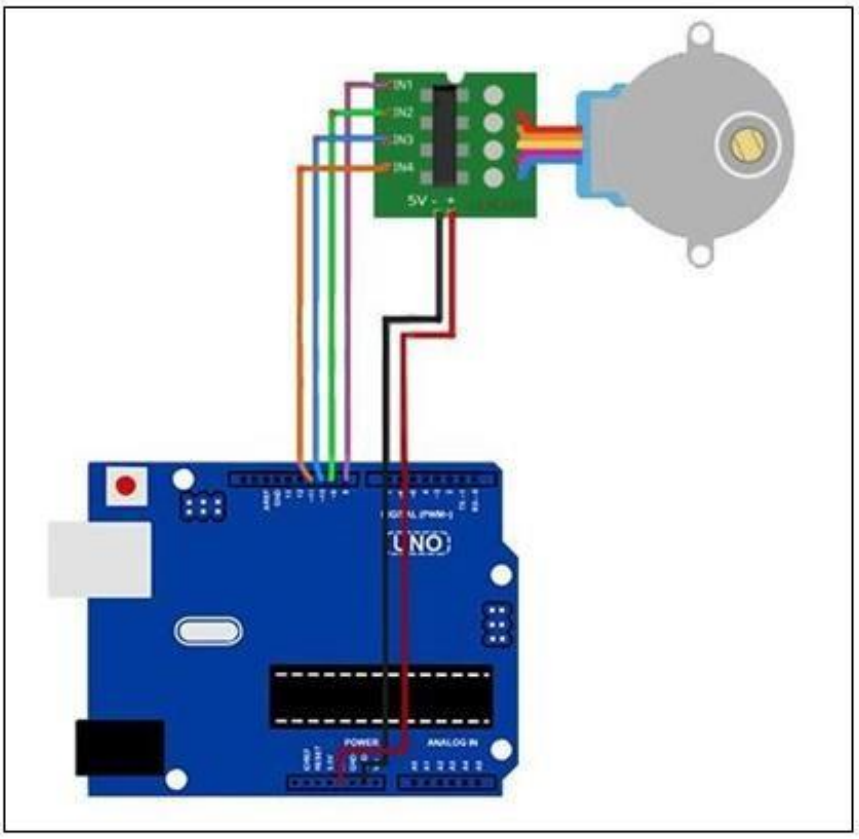
(1) x Uln2003 stepper motor driver module

(1) x Stepper motor

(9) x F-M wires

Connection

wiring diagram



We are using 4 pins to control the Stepper.

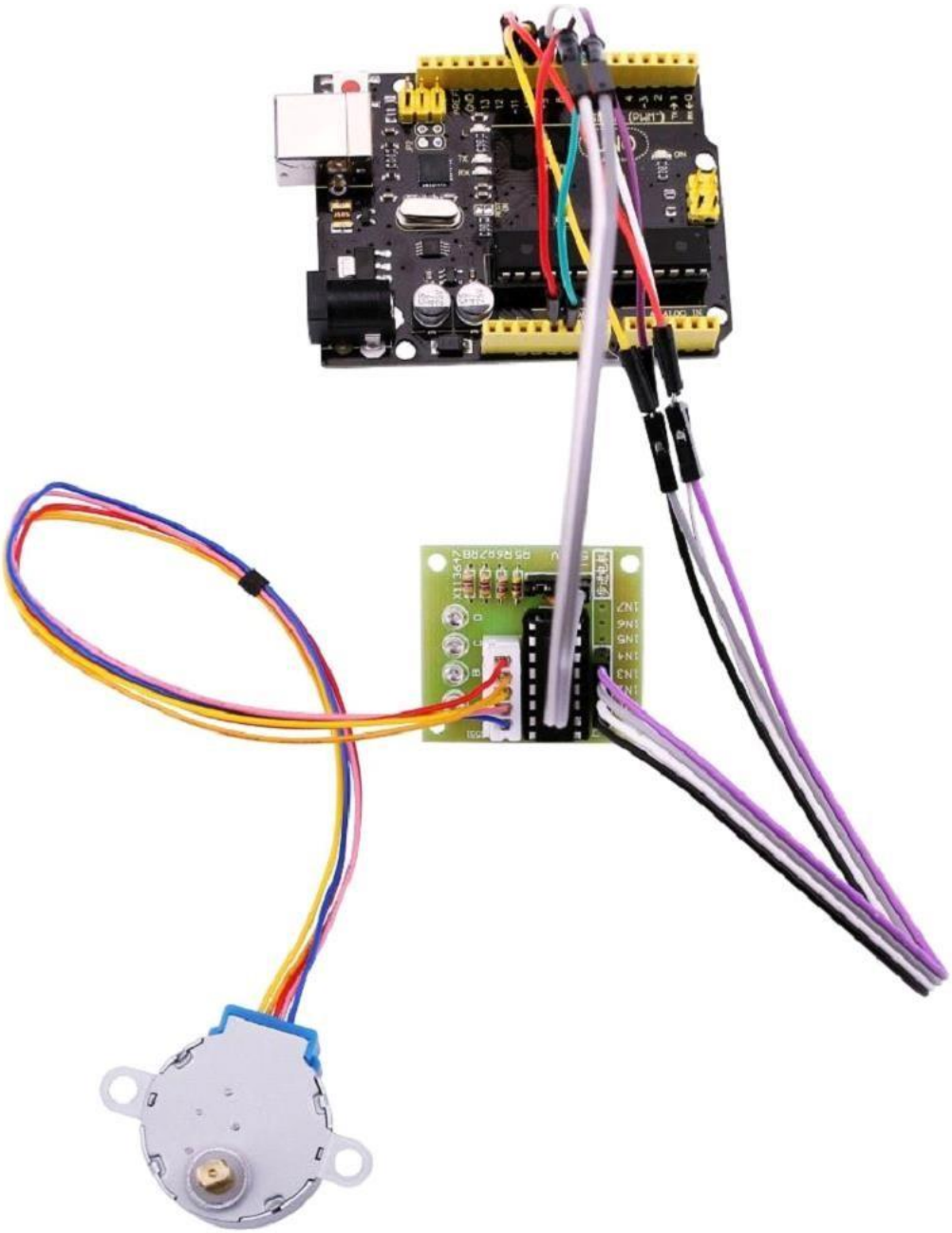
Pin 8-11 are controlling the Stepper motor.

We connect the 5V and Ground from to UNO to the Stepper motor.

Code

Now that we have the physical setup, all we need now is the code.

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



Lesson 17 LCD IIC MODULE

Overview

In this lesson, you will learn how to wire up and use an alphanumeric LCD display with IIC converter.

The display has an LED backlight and can display two rows with up to 16 characters on each row. You can see the rectangles for each character on the display and the pixels that make up each character. The display is just white on blue and is intended for showing text.

In this lesson, we will run the Arduino example program for the LCD1602 IIC module.

Component Required:

(1) x Quaduino Uno R3

(1) x LCD1602 IIC module

(4) x F-M wires

Component Introduction

LCD1602

introduction to the pins of LCD1602:

VSS: A pin that connects to ground

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

VO: A pin that adjust the contrast of LCD1602

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

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

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

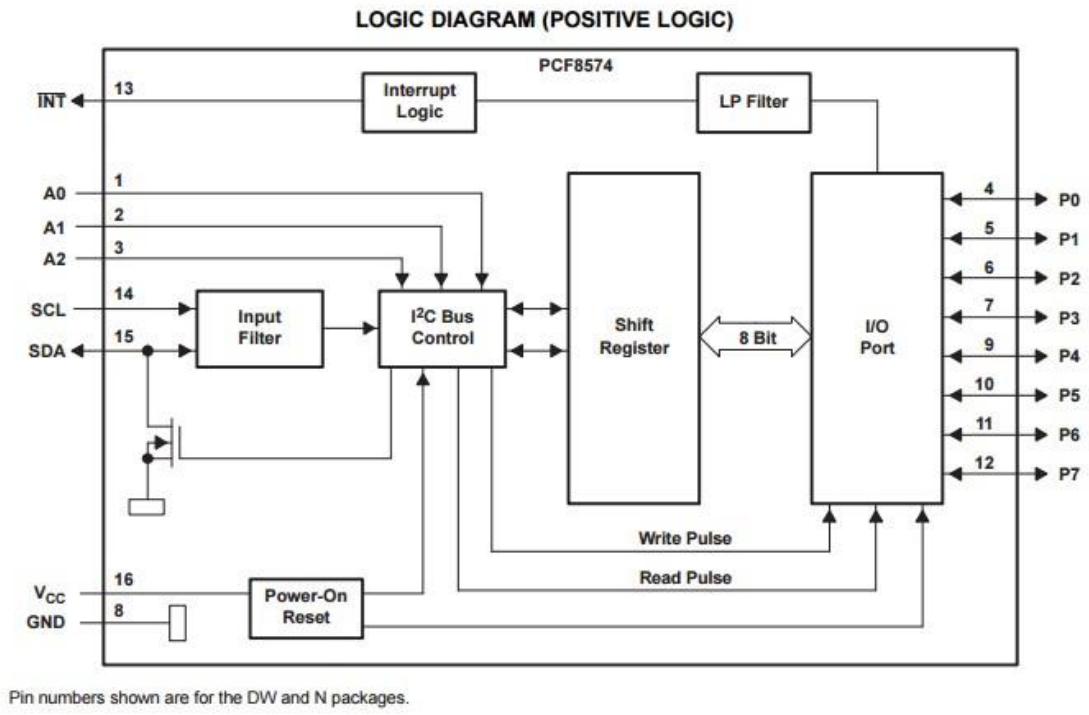
D0-D7 : Pins that read and write data

A and K: Pins that control the LED backlight

PCF8574

This 8-bit input/output (I/O) expander for the two-line bidirectional bus (I 2 C) is designed for 2.5-V to 6-V VCC operation. The PCF8574 provides general-purpose remote I/O expansion for most microcontroller families via the I 2 C interface [serial clock (SCL), serial data (SDA)]. The device features an 8-bit quasi-bidirectional I/O port (P0–P7), including latched outputs with high-current drive capability for directly driving LEDs. Each quasi-bidirectional I/O can be used as an input or output without the use of a data-direction control signal. At power on, the I/Os are high. In this mode, only a current source to VCC is active. An additional strong pull up to VCC allows fast rising edges into heavily loaded outputs. This device turns on when an output

is written high and is switched off by the negative edge of SCL. The I/Os should be high before being used as inputs.



mode, only a current source to VCC is active. An additional strong pull up to VCC allows fast rising edges into heavily loaded outputs. This device turns on when an output is written high and is switched off by the negative edge of SCL. The I/Os should be high before being used as inputs.

I2C Interface

I2C communication with this device is initiated by a master sending a start condition, a high-to-low transition on the SDA I/O while the SCL input is high. After the start condition, the device address byte is sent, most-significant bit (MSB) first, including the data direction bit (R/W). This device does not respond to the general call address. After receiving the valid address byte, this device responds with an acknowledge, a low on the SDA I/O during the high of the acknowledge-related clock pulse. The address inputs (A0–A2) of the slave device must not be changed between the start and the stop conditions. The data byte follows the address acknowledge. If the R/W bit is high, the data from this device are the values read from the P port. If the R/W bit is low, the data are from the master, to be output to the P port. The data byte is followed by an acknowledge sent from this device. If other data bytes are sent from the master, following the acknowledge, they are ignored by this device. Data are output only if complete bytes are received and acknowledged. The

output data will be valid at time, t pv, after the low-to-high transition of SCL and during the clock cycle for the acknowledge. A stop condition, which is a low-to-high transition on the SDA I/O while the SCL input is high, is sent by the master.

Top to bottom:

GND - GND

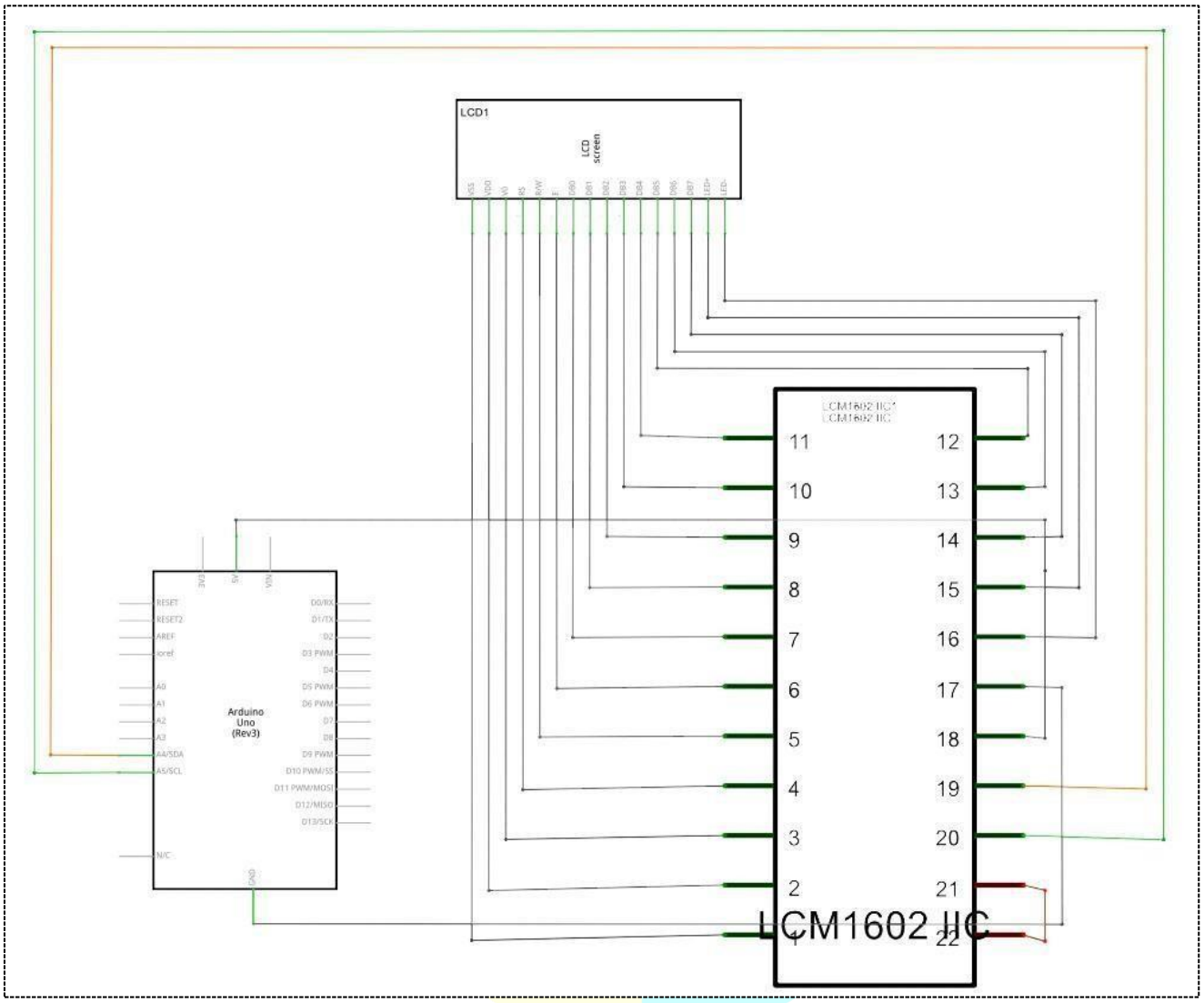
VCC - 5V

SDA - ANALOG Pin 4

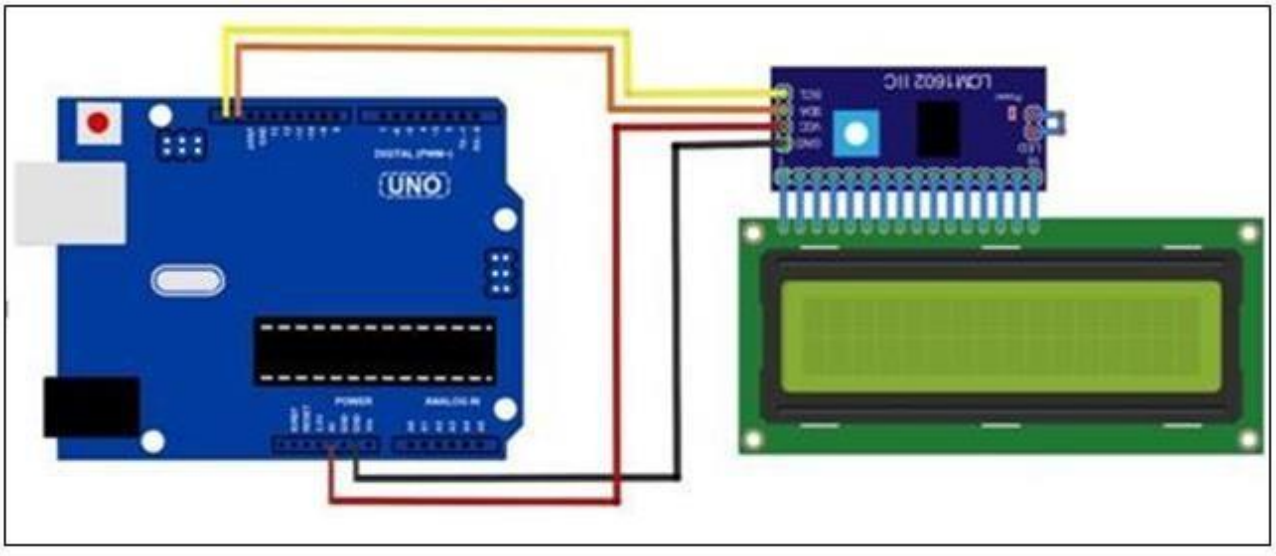
SCL - ANALOG pin 5

On most Arduino boards, SDA (data line) is on analog input pin 4, and SCL (clock line) is on analog input pin 5. On the Arduino Mega, SDA is digital pin 20 and SCL is 21. On the newer Arduino UNO (The "V3" pinout), the SCL and SDA pins are also on two new leftmost top pins.

Connection Schematic



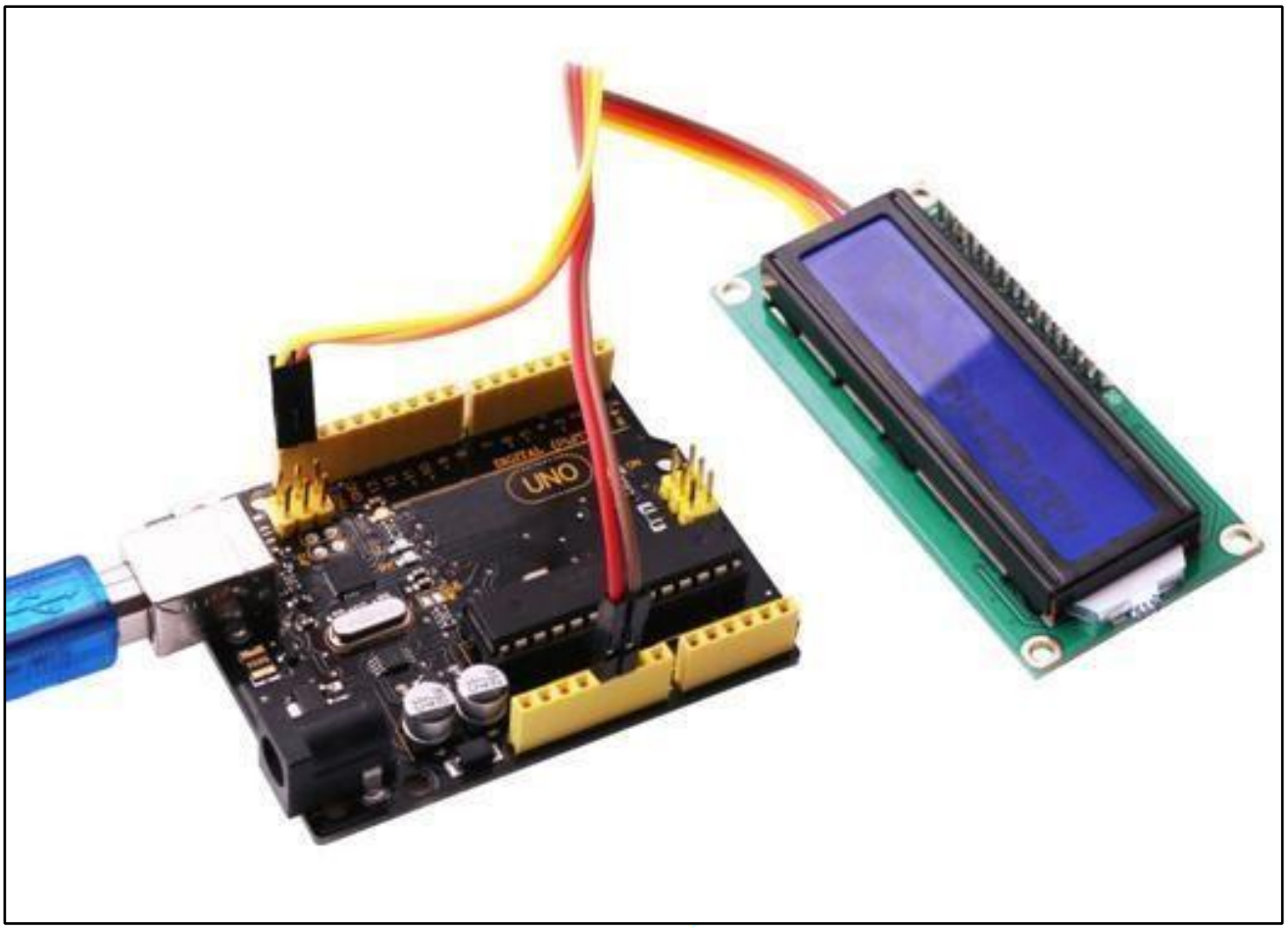
wiring diagram



The Code

Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the < LiquidCrystal_I2C> library. Or you need to install again. If you do not do this, your code won't work.



Lesson 18 SHOW RESISTORS OF PETENTIONMETER WITH LCD

Overview

In this lesson, you will use a LCD display to show the resistors of the potentiometer.

Component Required:

(1) x Quaduino Uno R3

(1) x LCD1602 IIC

(1) x 10k ohm resistor

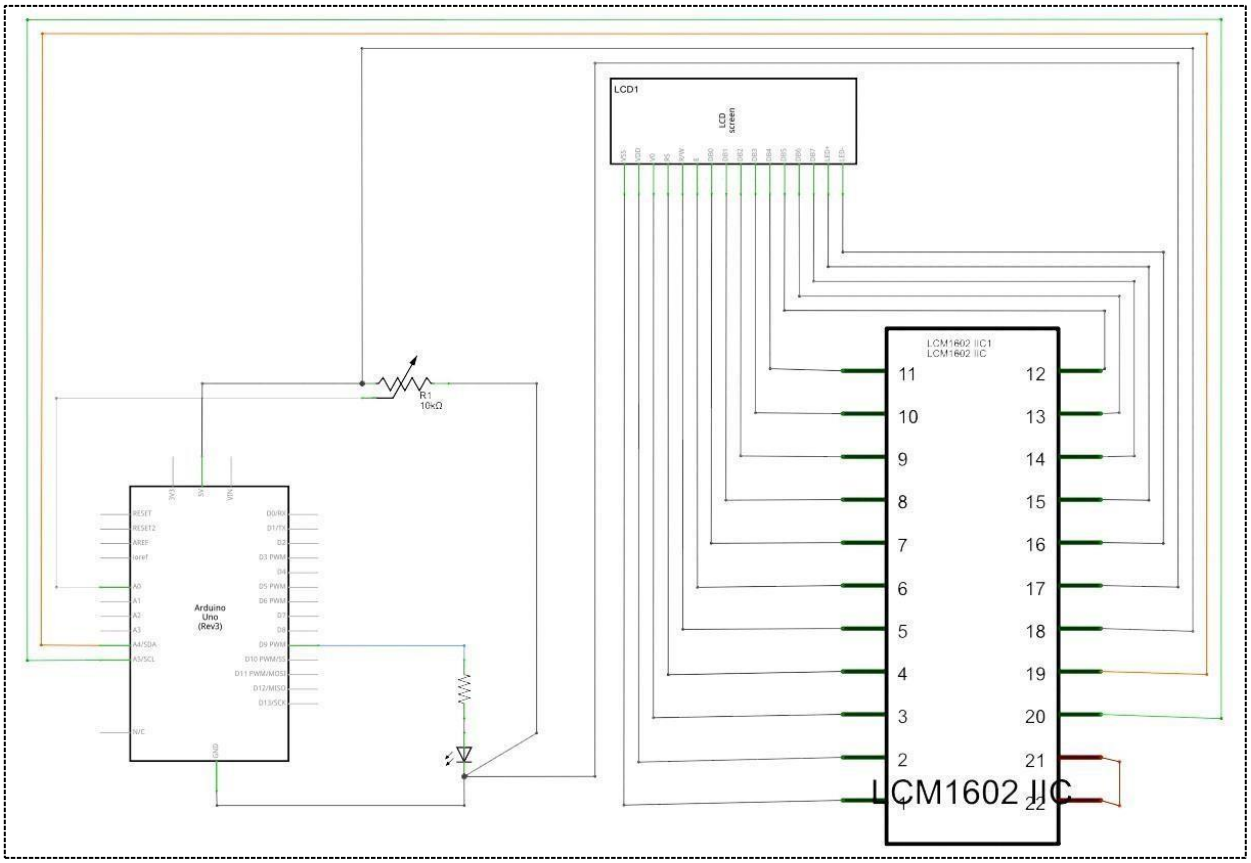
(1) x Potentiometer

(1) x Breadboard

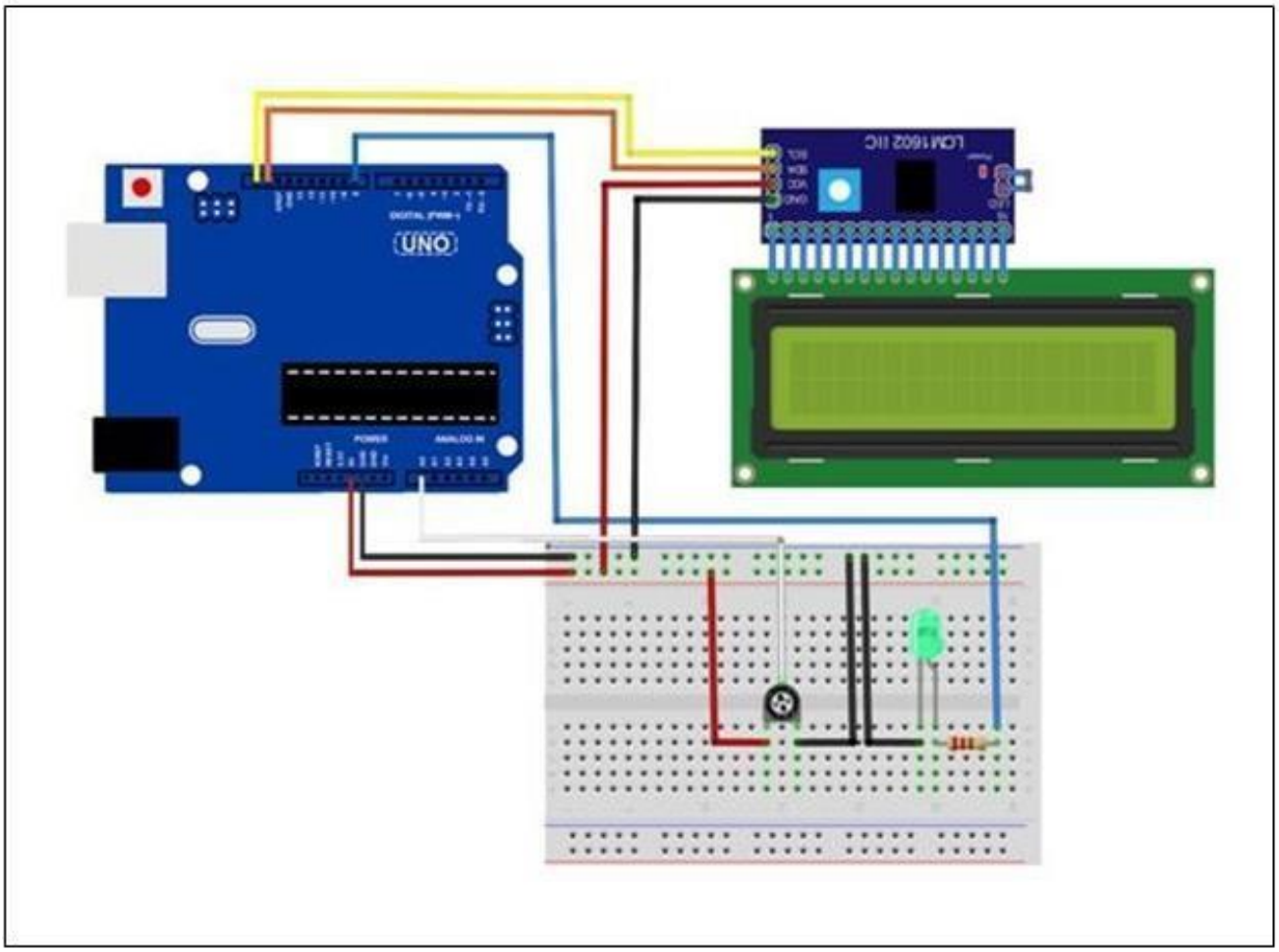
(4) x F-M wires

(8) x M-M wires

Connection Schematic



wiring diagram



The Code

Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the < LiquidCrystal_I2C> library. Or you need to install again. If you do not do this, your code won't work.

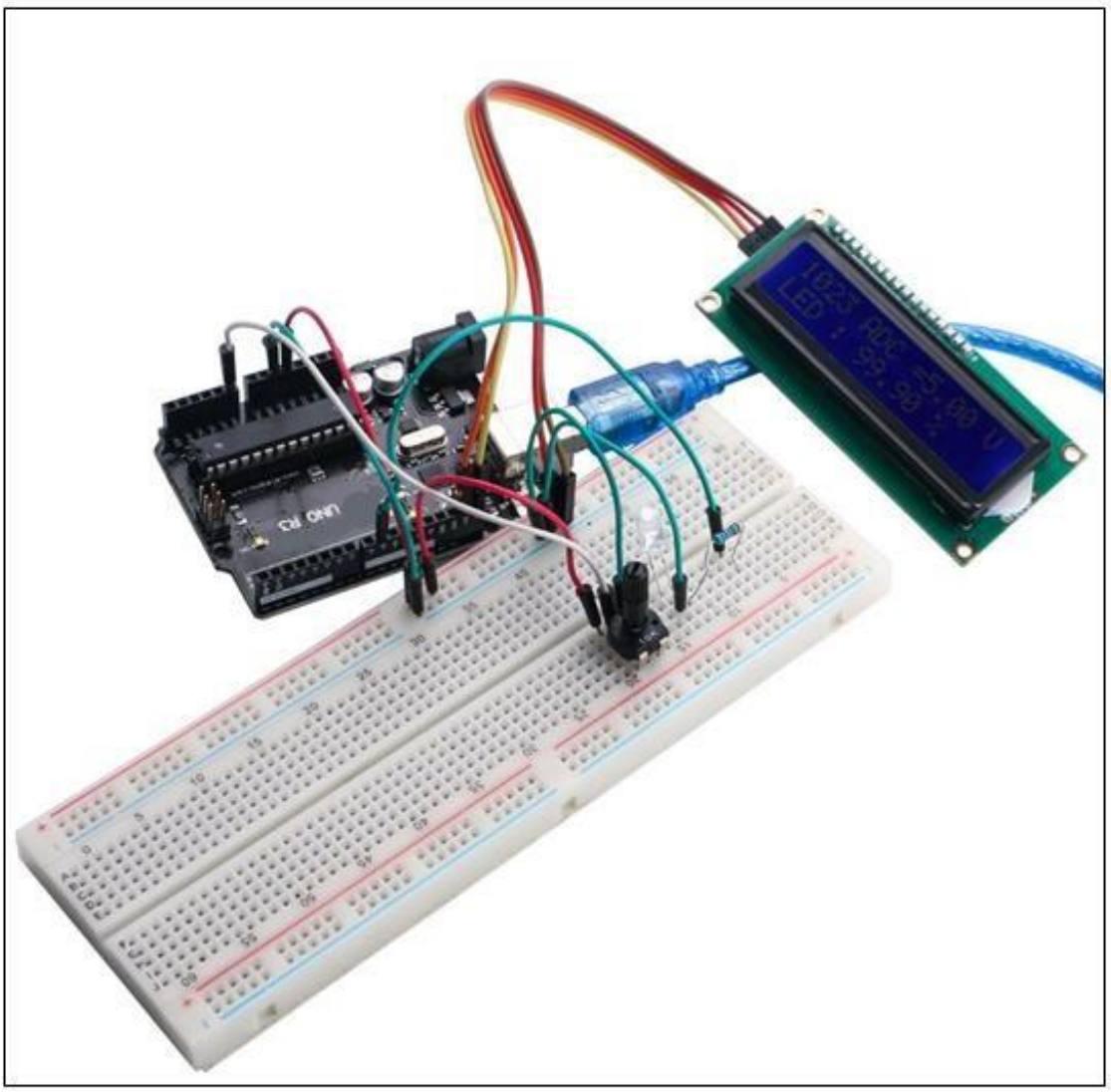
First I wanted to display the raw analog values 0 to 1023 which came from 10k pot. on A0.

That proved to be a bit complicated, when displaying the sensor value, it was written left to right.

As the number became < 1000 the last digit didn't refresh. To correct this I've searched for a function that place leading zeros (or any other character of your choosing)

After that I wanted to scale the raw analog values from 0 to 5 V.

At last I used the analog input to dim a LED and placed the percentage of the power on the LCD.

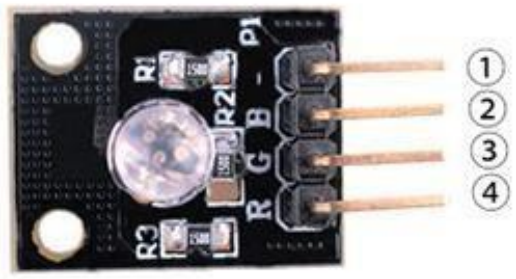


Lesson 19 RGB MODULE

Overview

In this experiment, we will learn how to use RGB module.

RGB module are made from a patch of full-color LED. By adjusting the voltage input of R, G, B pins, we can adjust the strength of the three primary colors (red/blue/green) so as to implementation result of full color effect.



1.GND:ground
2.BLUE
3.GREED
4.RED

Component Required:

(1) x Quaduino Uno R3

(1) x USB cable

(1) x RGB module

(x) x F-M wires

Component Introduction

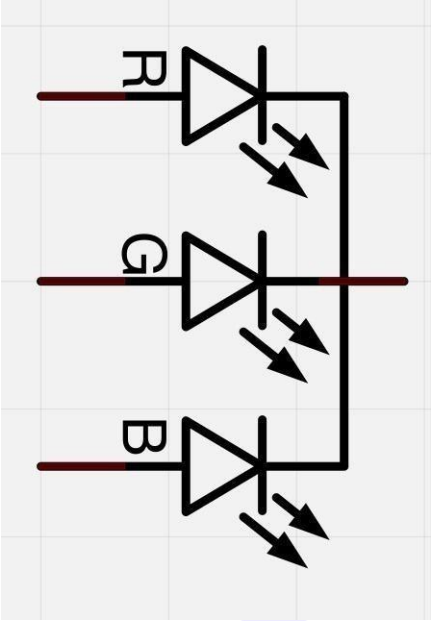
RGB:

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

We mix colors just like you would mix audio with a 'mixing board' or paint on a palette - by

adjusting the brightness of each of the three LEDs. The hard way to do this would be to use different value resistors (or variable resistors) as we played with in lesson 2. That's a lot of work! Fortunately for us, the Arduino has an analogWrite function that you can use with pins marked with a ~ to output a variable amount of power to the appropriate LEDs.

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



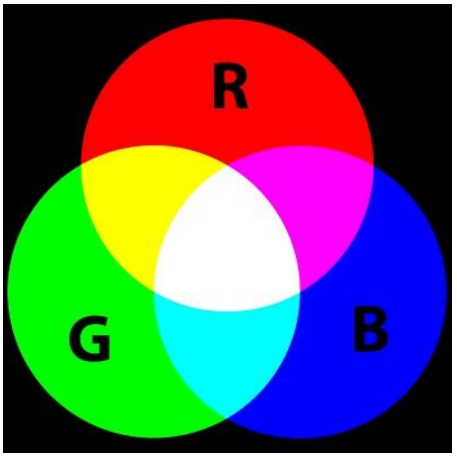
The common negative connection of the LED package is the second pin from the flat side of the LED package. It is also the longest of the four leads. This lead will be connected to ground.

Each LED inside the package requires its own 270Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to UNO output pins using these resistors.

COLOR:

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

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



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

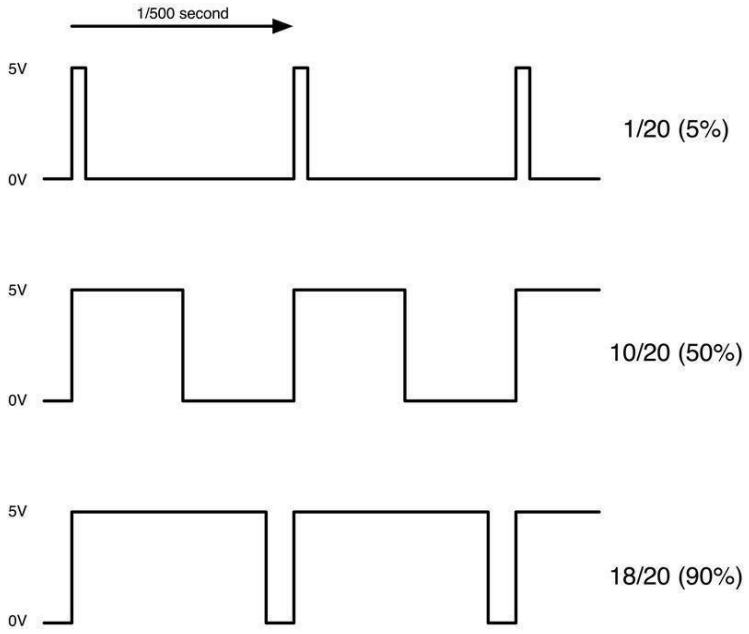
We can control the brightness of each of the red, green and blue parts of the LED separately, making it possible to mix any color we like.

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

Theory(PWM)

Pulse Width Modulation (or PWM) is a technique for controlling power. We also use it here to control the brightness of each of the LEDs.

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



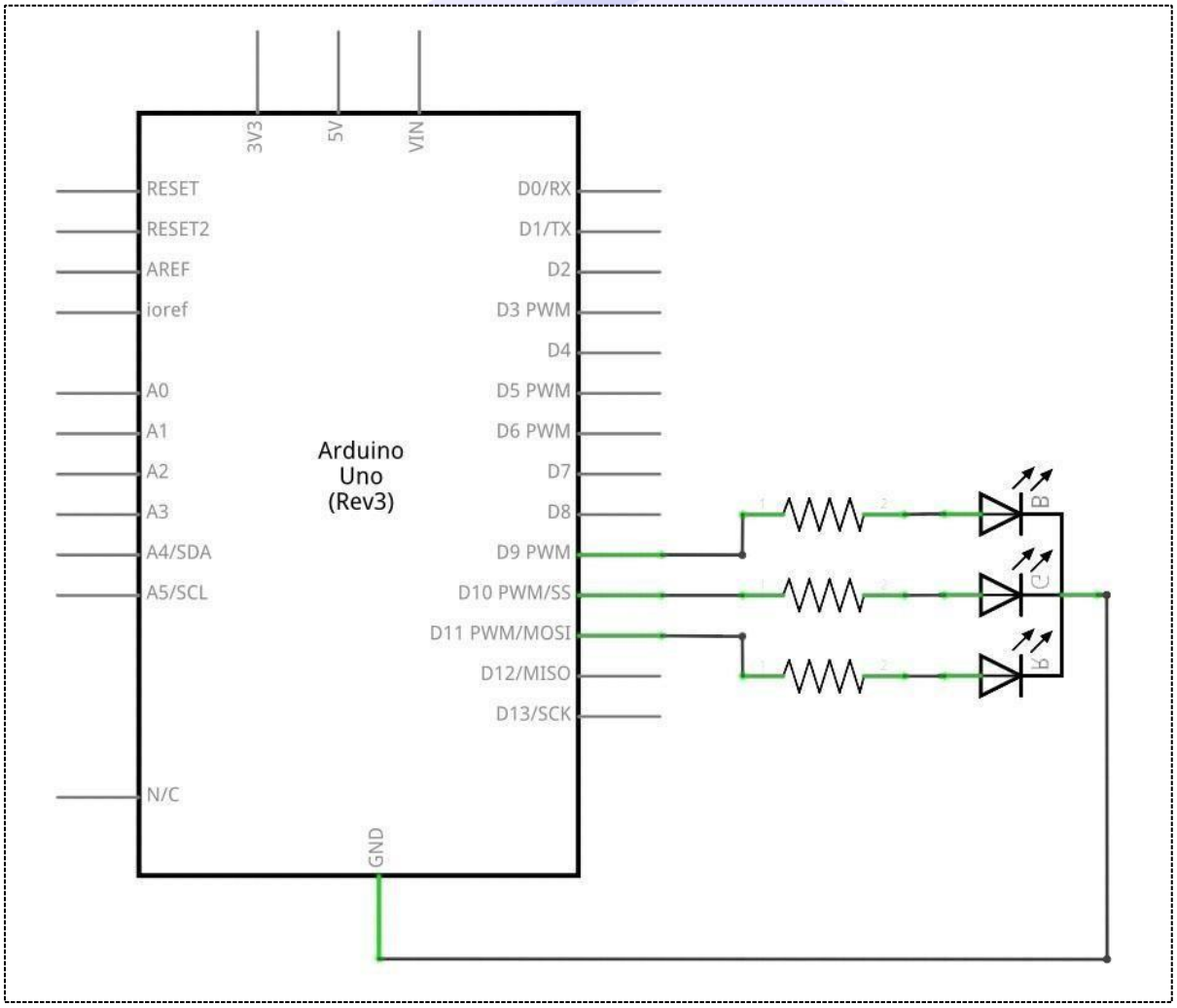
Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analogWrite' function. So 'analogWrite(0)' will not produce any pulse at all and 'analogWrite(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the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

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

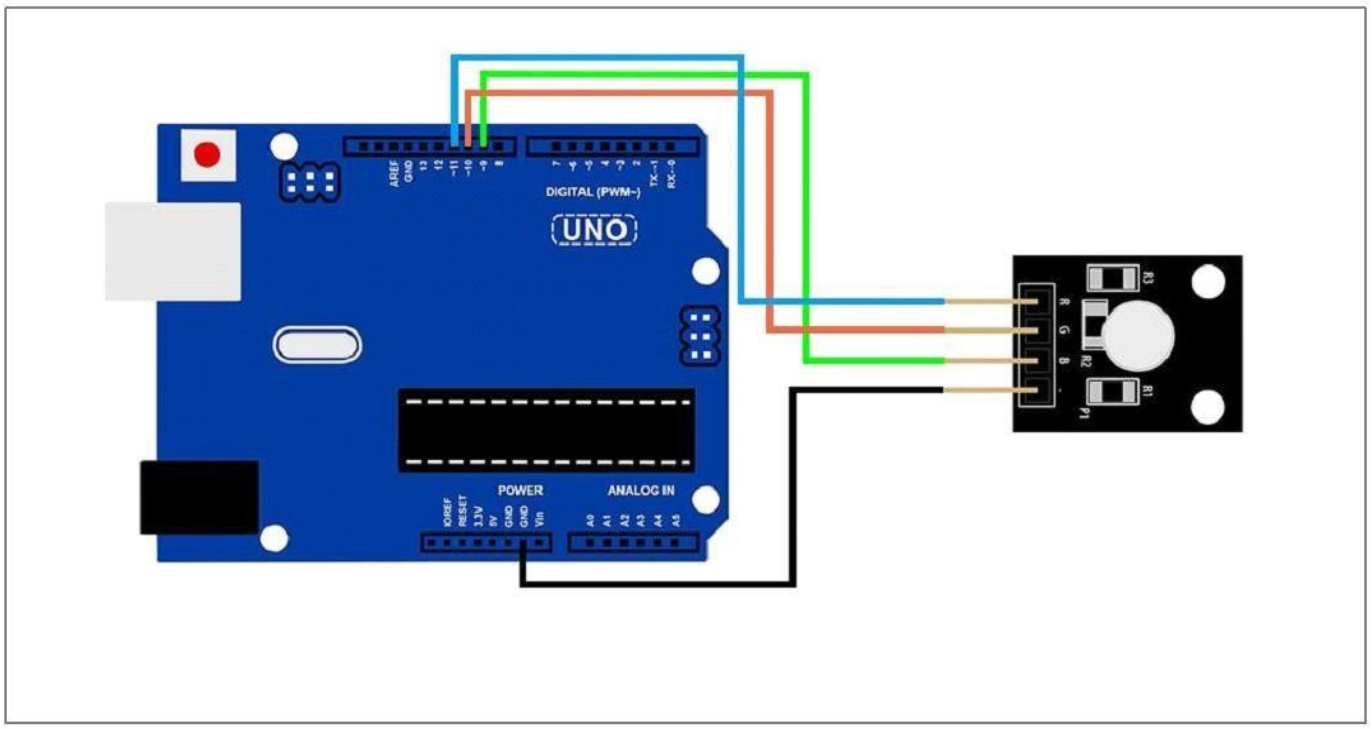
If however the output is at 5V for 90% of the time then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is changing.

Connection

Schematic

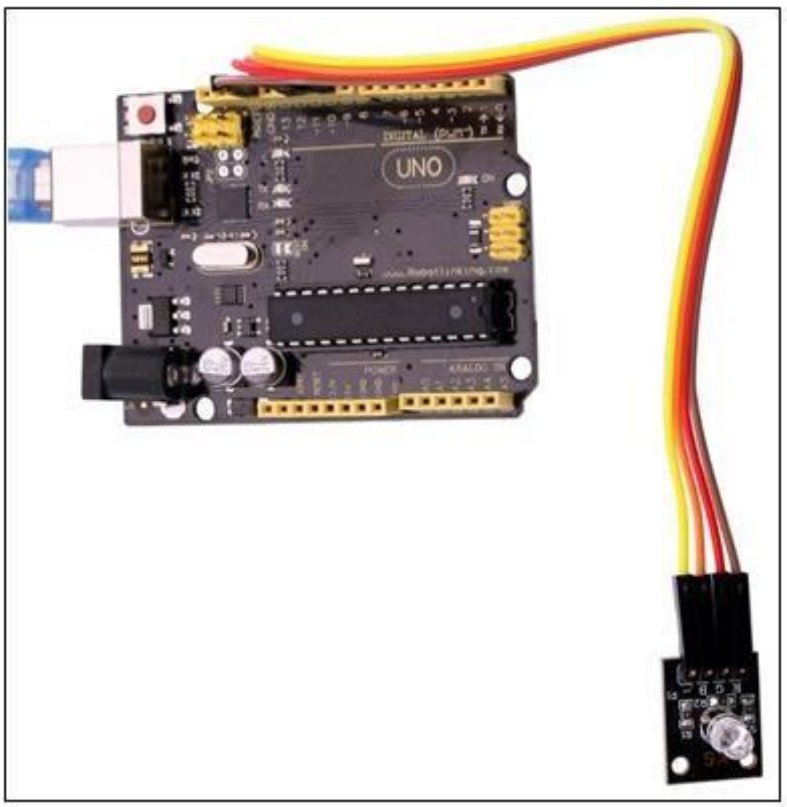


wiring diagram



Result

After we connect the circuit as the picture, we upload the program of each module. We can see the module changing their color as the code set. If you want to make it change the color in different way, you can revise the code.



Lesson 20 Ultrasonic Sensor Module

Overview

Ultrasonic sensor are great for all kind of projects that need distance measurements, avoiding obstacles as examples.

The HC-SR04 are inexpensive and easy to use since we will be using a Library specifically designed for these sensor.

Component Required:

(1) x Quaduino Uno R3

(1) x Ultrasonic sensor module

(4) x F-M wires

Component Introduction

Ultrasonic sensor

Ultrasonic ranging module HC - SR04 provides 2cm - 400cmnon-contactmeasurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,

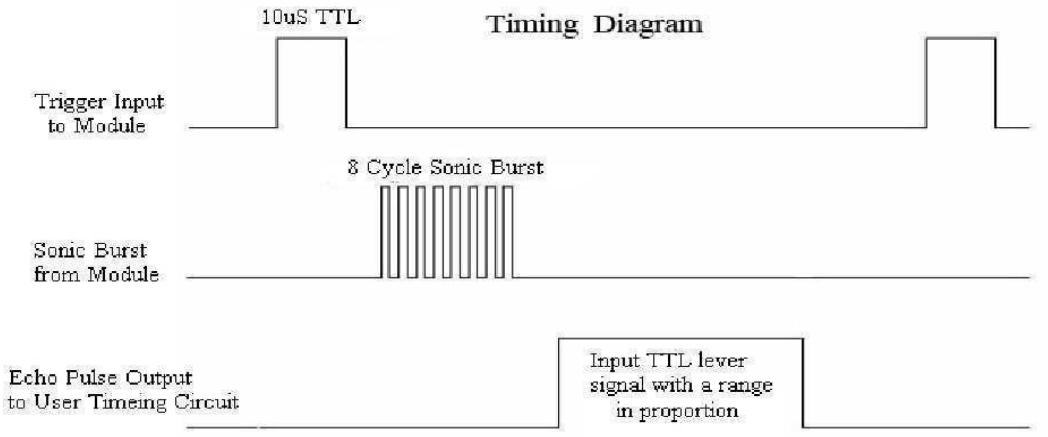
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

Test distance = (high level time × velocity of sound (340M/S) /2

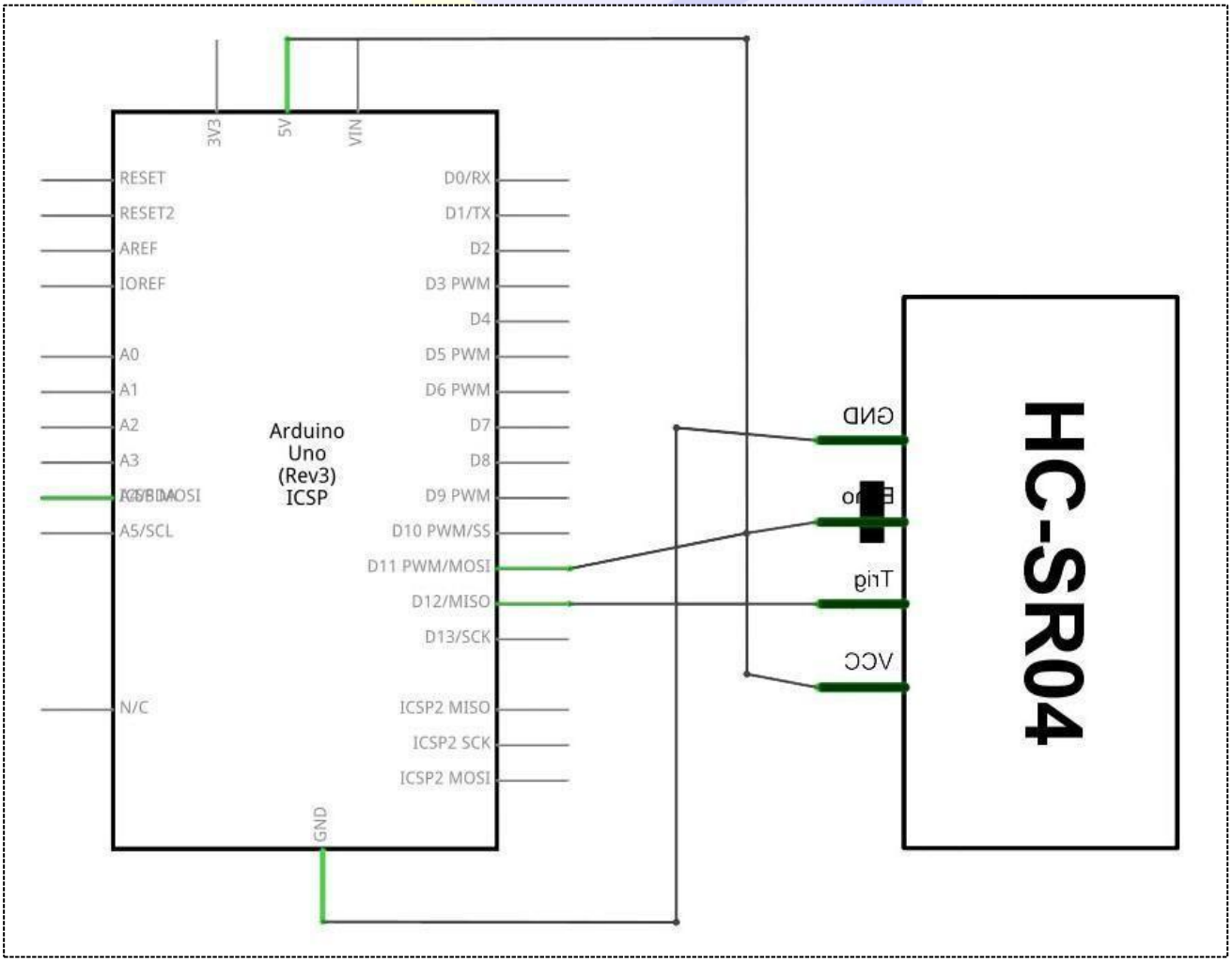
The Timing diagram is shown below. You only need to supply a short10uSpulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or:

the range = high level time * velocity (340M/S) / 2; we suggest to use over60msmeasurement cycle, in order to prevent trigger signal to the echo signal.

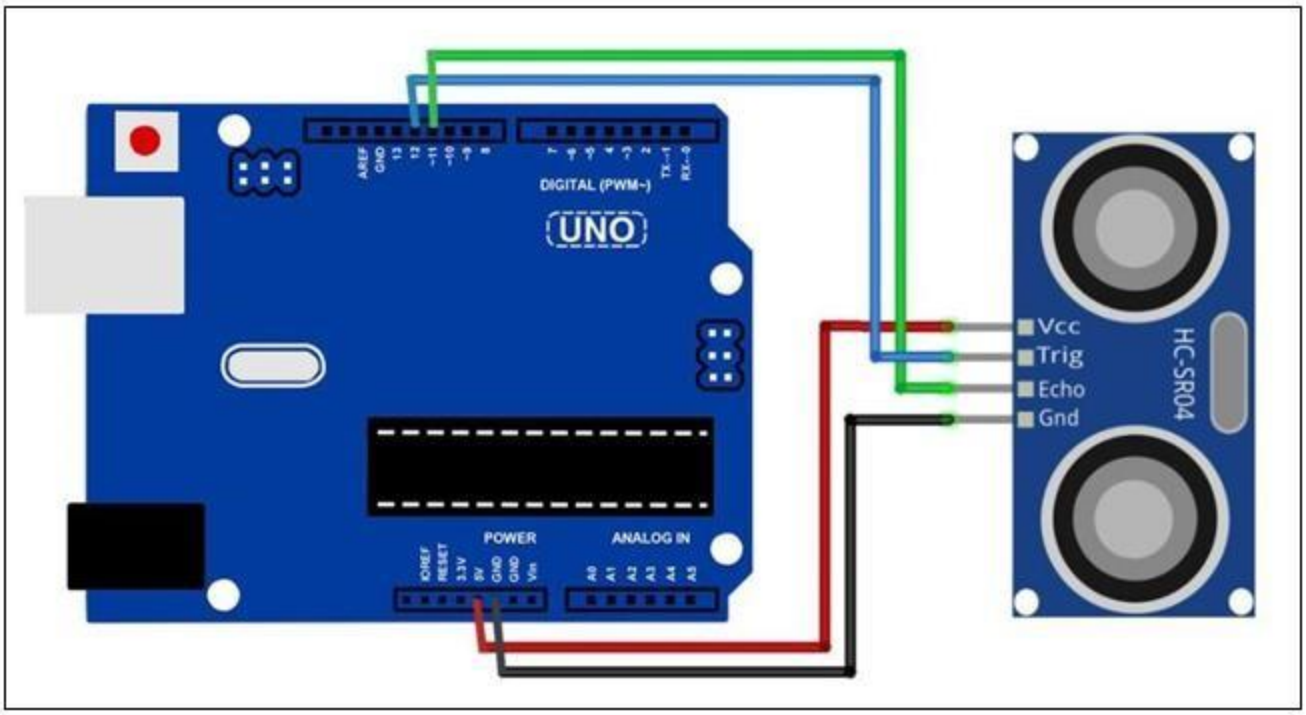


Connection

Schematic



wiring diagram



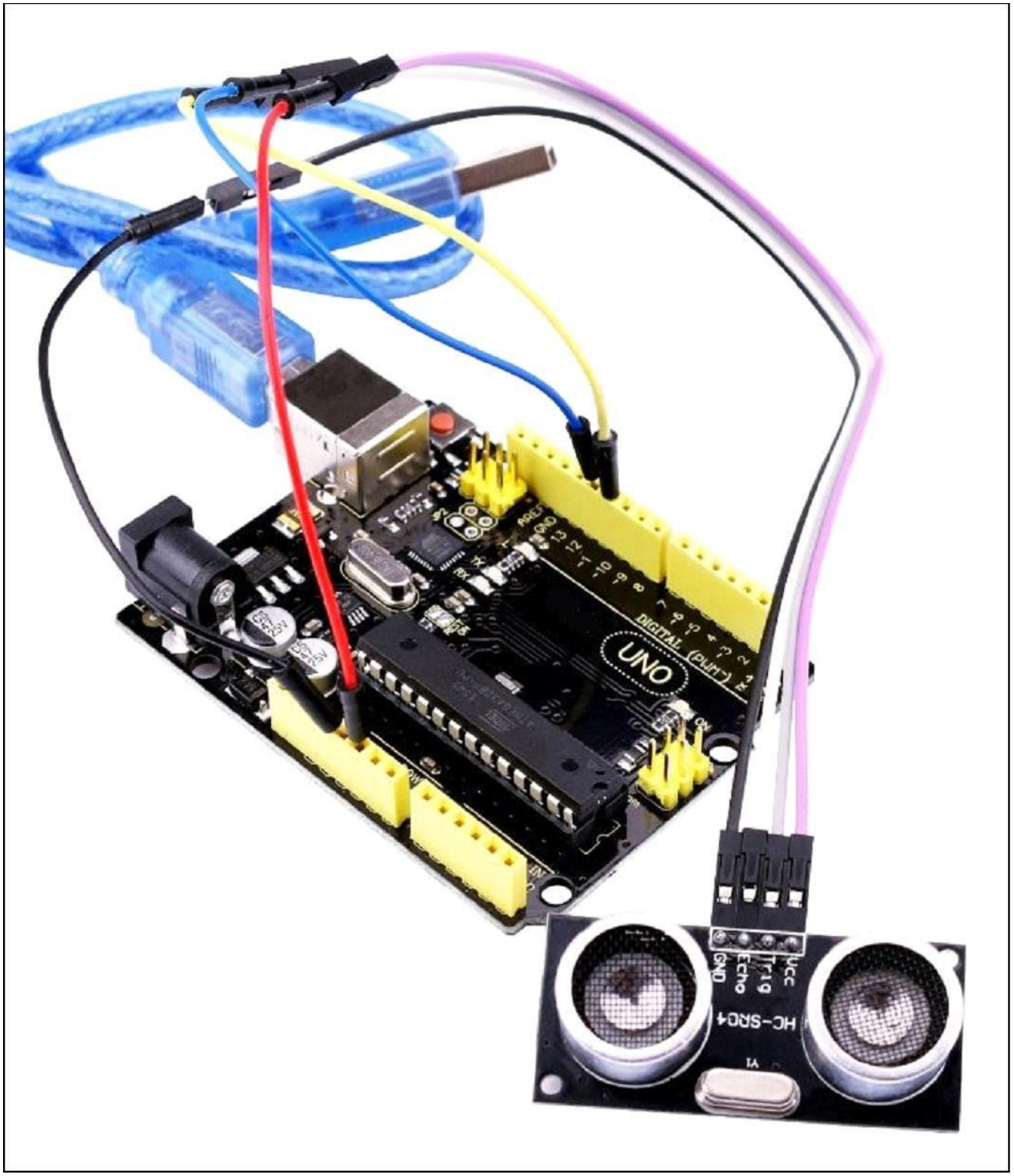
Code

Using a Library designed for these sensors will make our code short and simple.

We include the library at the beginning of our code, and then by using simple commands we can control the behavior of the sensor.

Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the < HC-SR04_Library> library. Or you need to install again. If you do not do this, your code won't work.



Lesson 21 Keypad Module

Overview

In this project, we will go over how to integrate a keyboard with an UNO R3 board so that the UNO R3 can read the keys being pressed by a user.

Keypads are used in all types of devices, including cell phones, fax machines, microwaves, ovens, door locks, etc. They're practically everywhere. Tons of electronic devices use them for user input.

So knowing how to connect a keypad to a microcontroller such as an UNO R3 board is very valuable for building many different types of commercial products.

At the end when all is connected properly and programmed, when a key is pressed, it show up at the Serial Monitor on your computer. Whenever you press a key, it shows up on the Serial Monitor. Later, in another project, we will connect the keypad circuit, so that it will get displayed on an LCD. But for now, for simplicity purposes, we start at simply showing the key pressed on the computer.

For this project, the type of keypad we will use is a matrix keypad. This is a keypad that follows an encoding scheme that allows it to have much less output pins than there are keys. For example, the matrix keypad we are using has 16 keys (0-9, A-D, *, #), yet only 8 output pins. With a linear keypad, there would have to be 17 output pins (one for each key and a ground pin) in order to work. The matrix encoding scheme allows for less output pins and thus much less connections that have to made for the keypad to work. In this way, they are more efficient than linear keypads, being that they have less wiring.

Component Required:

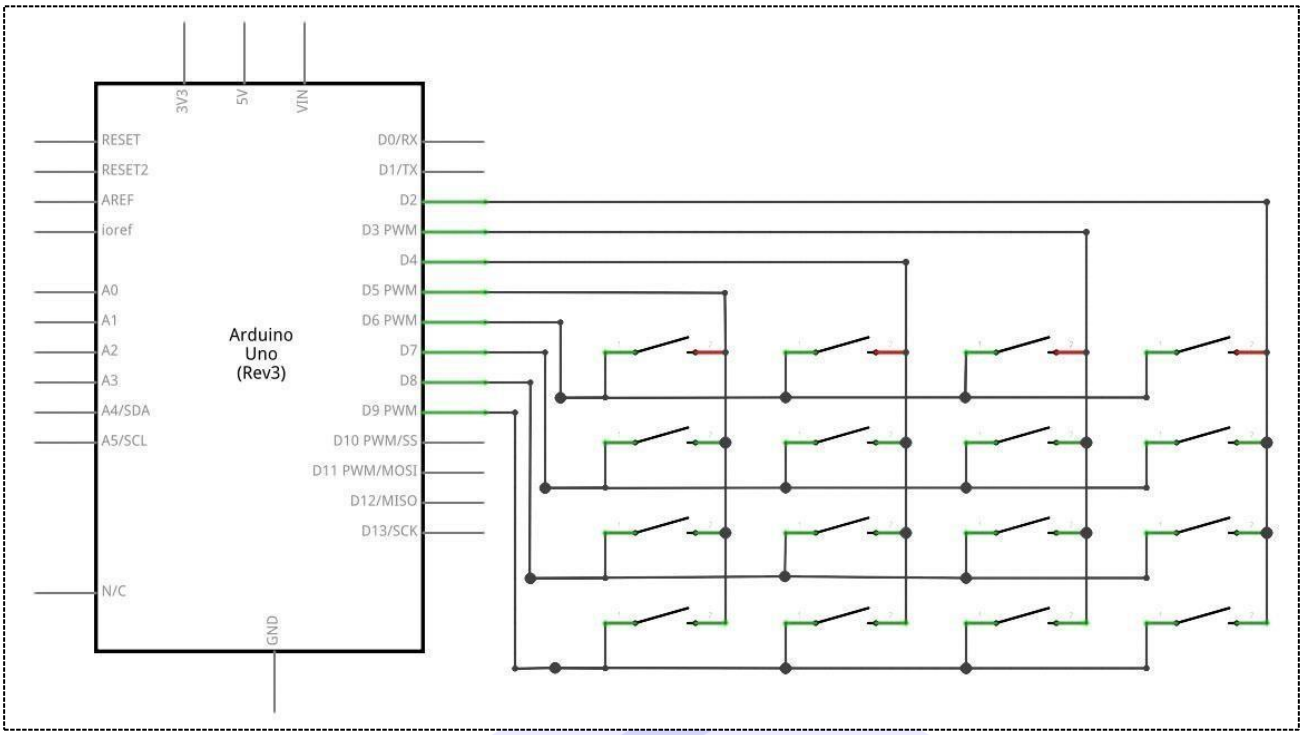
(1) x Quaduino Uno R3

(1) x Membrane switch module

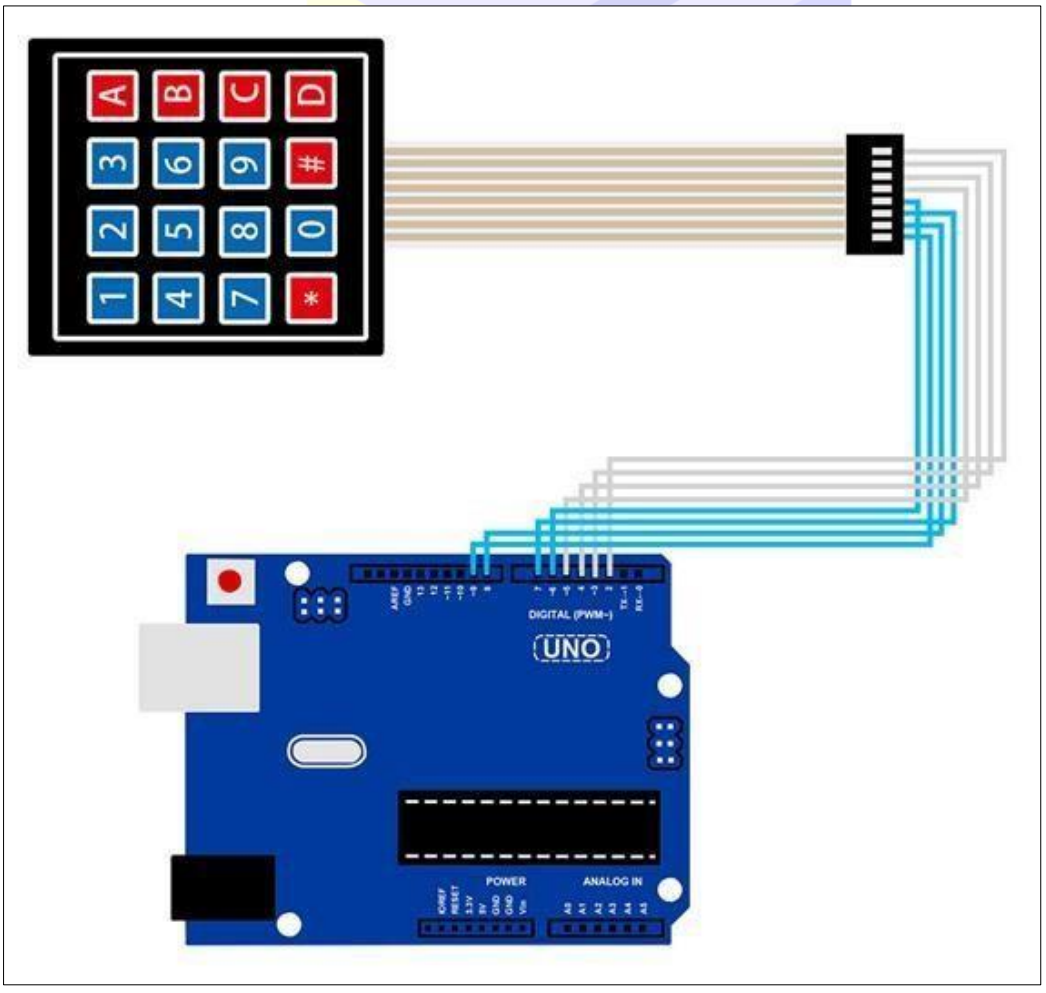
(8) x M-M wires

Connection

Schematic



wiring diagram



When connecting the pins to the UNO R3 board, we connect them to the digital output pins, D9-D2. We connect the first pin of the keypad to D9, the second pin to D8, the third pin to D7, the fourth pin to D6, the fifth pin to D5, the sixth pin to D4, the seventh pin to D3, and the eighth pin to D2.

These are the connections in a table:

Keypad Pin	Connects to Arduino Pin...
1	D9
2	D8
3	D7
4	D6
5	D5
6	D4
7	D3
8	D2

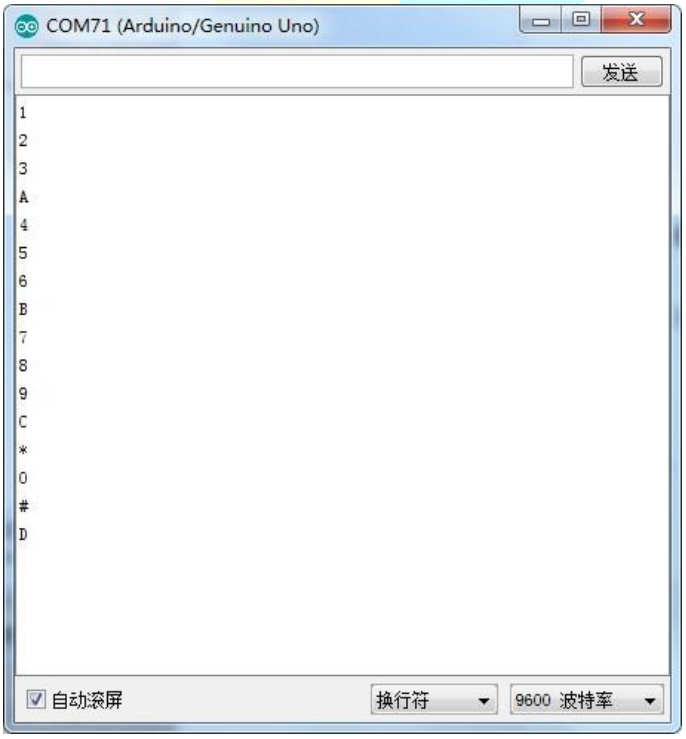
Code

Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the <Keypad> library.

Or you need to install again. If you do not do this, your code won't work.

With this code, once we press a key on the keypad, it should show up on the serial monitor of the arduino software once the code is compiled and uploaded to the UNO R3 board.



Lesson 22 DHT11 Temperature and Humidity Sensor

Overview

In this tutorial we will learn how to use a DHT11 Temperature and Humidity Sensor.

It"s accurate enough for most projects that need to keep track of humidity and temperature readings.

Again we will be using a Library specifically designed for these sensors that will make our code short and easy to write.

Component Required:

(1) x Quaduino Uno R3

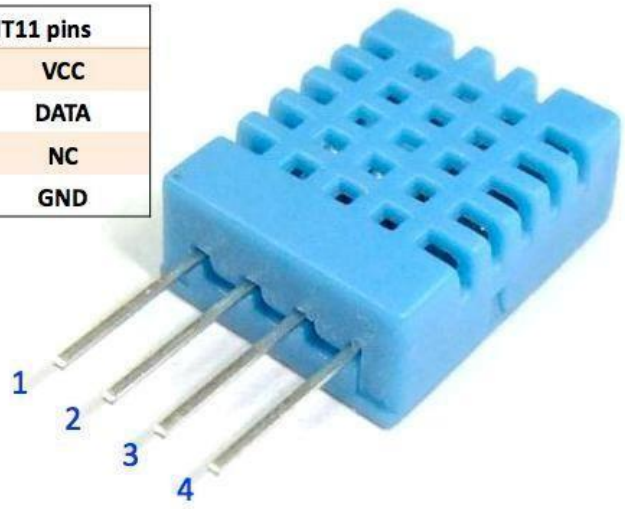
(1) x DHT11 module

(3) x F-M wires

Component Introduction

Temp and humidity sensor:

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



DHT11 digital temperature and humidity sensor is a composite Sensor contains a calibrated digital signal output of the temperature and humidity. Application of a dedicated digital modules collection technology and the temperature and humidity sensing technology, to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet components and an NTC temperature measurement devices, and connected with a high-performance 8-bit microcontroller.

Applications: HVAC, dehumidifier, testing and inspection equipment, consumer goods, automotive, automatic control, data loggers, weather stations, home appliances, humidity regulator, medical and other humidity measurement and control.

Product parameters

Relative humidity:

Resolution: 16Bit

Repeatability: ±1% RH

Accuracy: At 25℃ ±5% RH

Interchangeability: fully interchangeable

Response time: 1 / e (63%) of 25℃ 6s

1m / s air 6s

Hysteresis: <± 0.3% RH

Long-term stability: <± 0.5% RH / yr in

Temperature:

Resolution: 16Bit

Repeatability: ±0.2℃ Range:

At 25℃ ±2℃ Response time:

1 / e (63%) 10S

Electrical Characteristics

Power supply: DC 3.5～5.5V

Supply Current: measurement 0.3mA standby 60μA

Sampling period: more than 2 seconds

Pin Description:

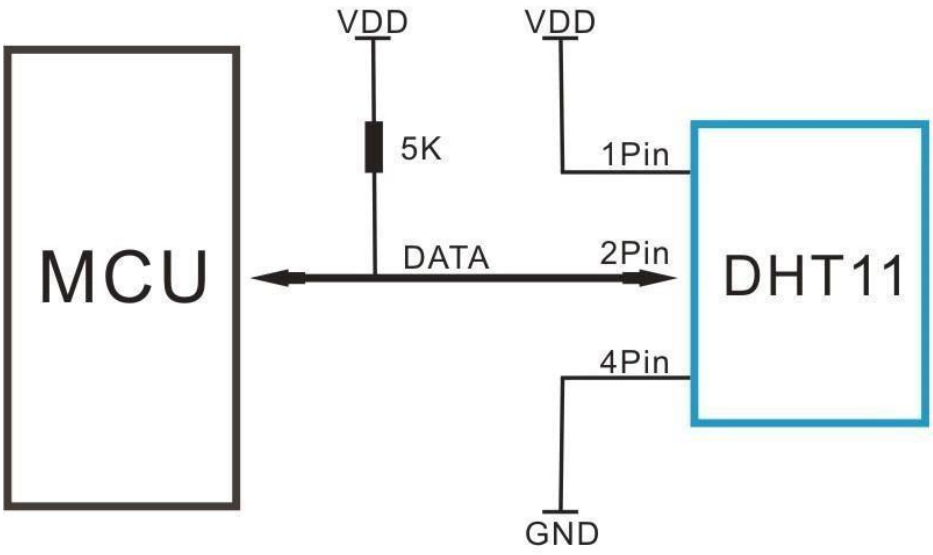
1, the VDD power supply 3.5～5.5V DC

2 DATA serial data, a single bus

3, NC, empty pin

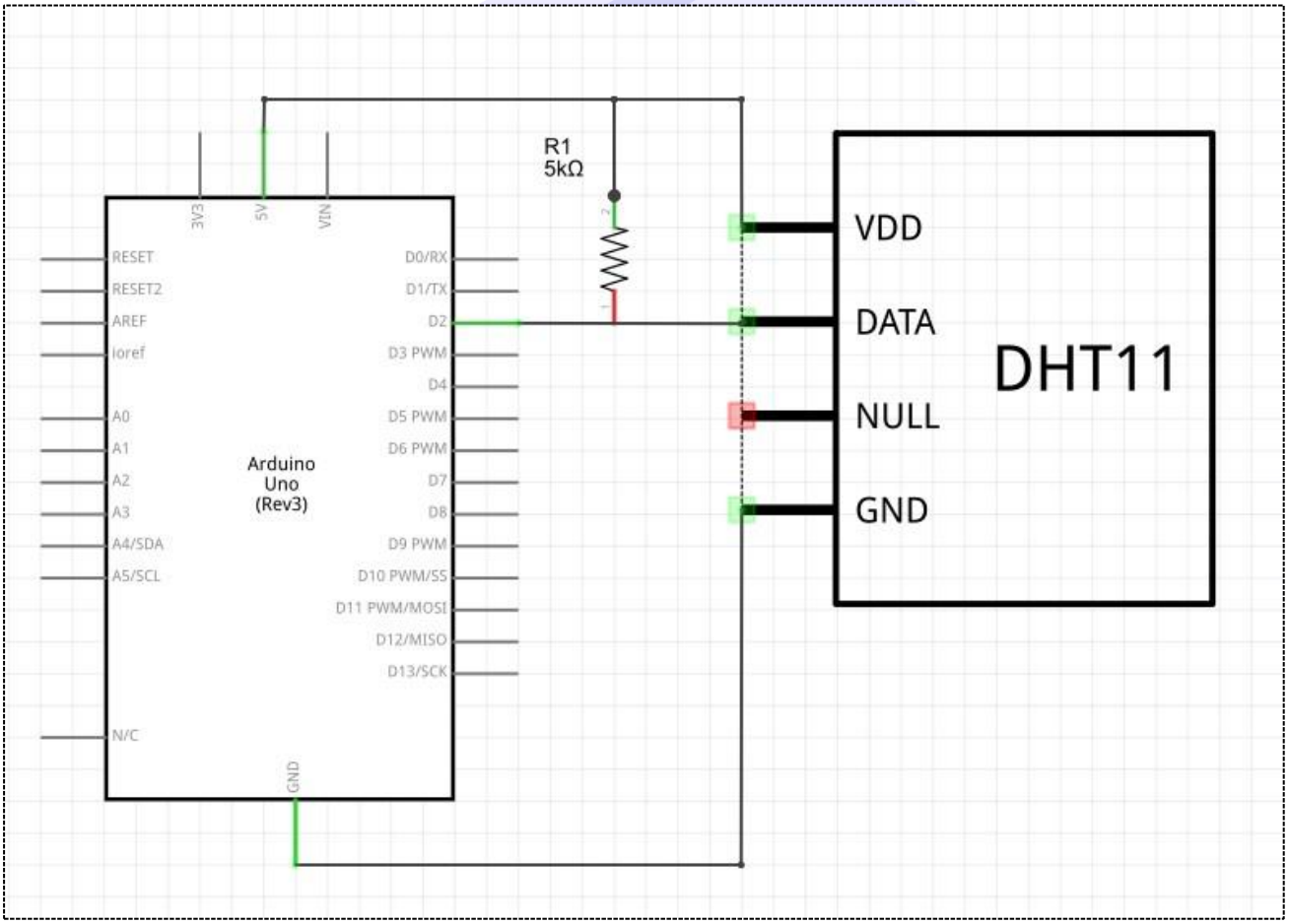
4, GND ground, the negative power

Typical Application

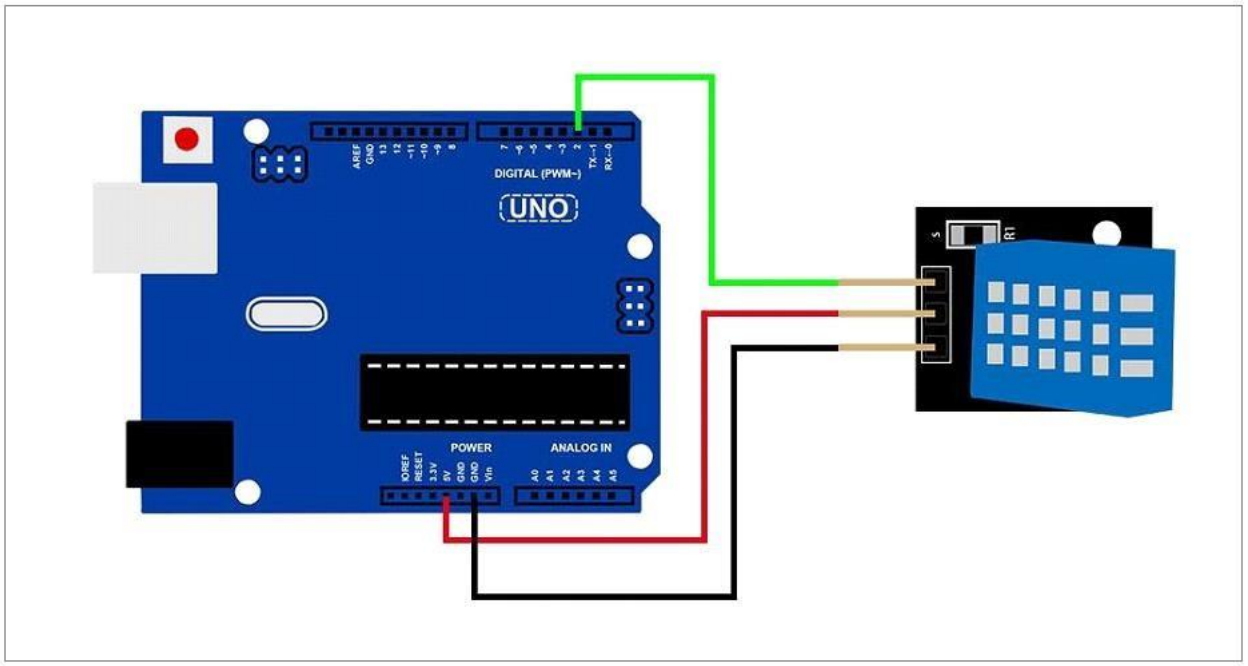


Connection

Schematic



wiring diagram



As you can see we only need 3 connections to the sensor, since one of the pin is not used.

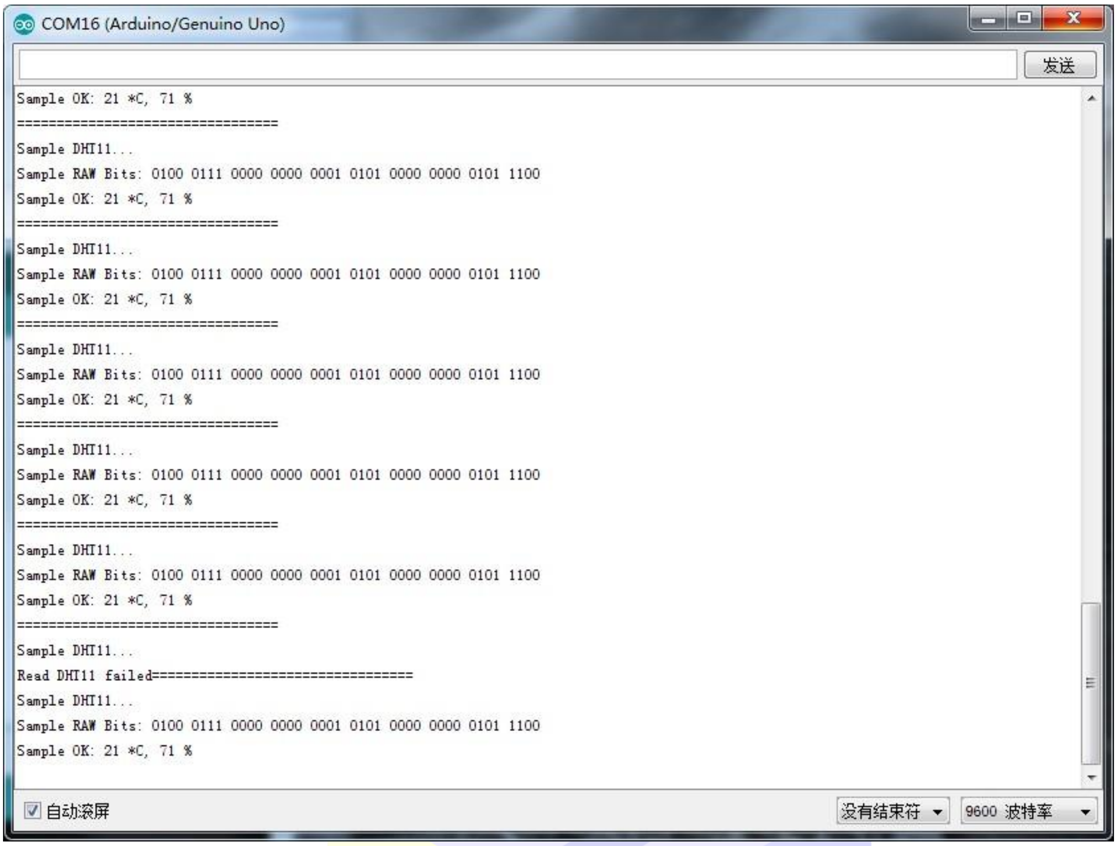
The connection are : Voltage, Ground and Signal which can be connected to any Analog Pin on our UNO.

Code

Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the <simpleDHT> library. Or you need to install again. If you do not do this, your code won't work.

Upload the program then open the monitor, we can see the data as below:(It shows the temperature of the environment, we can see it is 21 degree)


COM16 (Arduino/Genuino Uno)
发送
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
================================
Sample DHT11...
Read DHT11 failed================================
Sample DHT11...
Sample RAW Bits: 0100 0111 0000 0000 0001 0101 0000 0000 0101 1100
Sample OK: 21 *C, 71 %
自动滚屏
没有结束符
9600 波特率

Lesson 23 Analog Joystick Module

Overview

Analog joysticks are a great way to add some control in your projects.

In this tutorial we will learn how to use the analog joystick module.

Component Required:

(1) x Quaduino Uno R3

(1) x Joystick module

(5) x F-M wires

Component Introduction

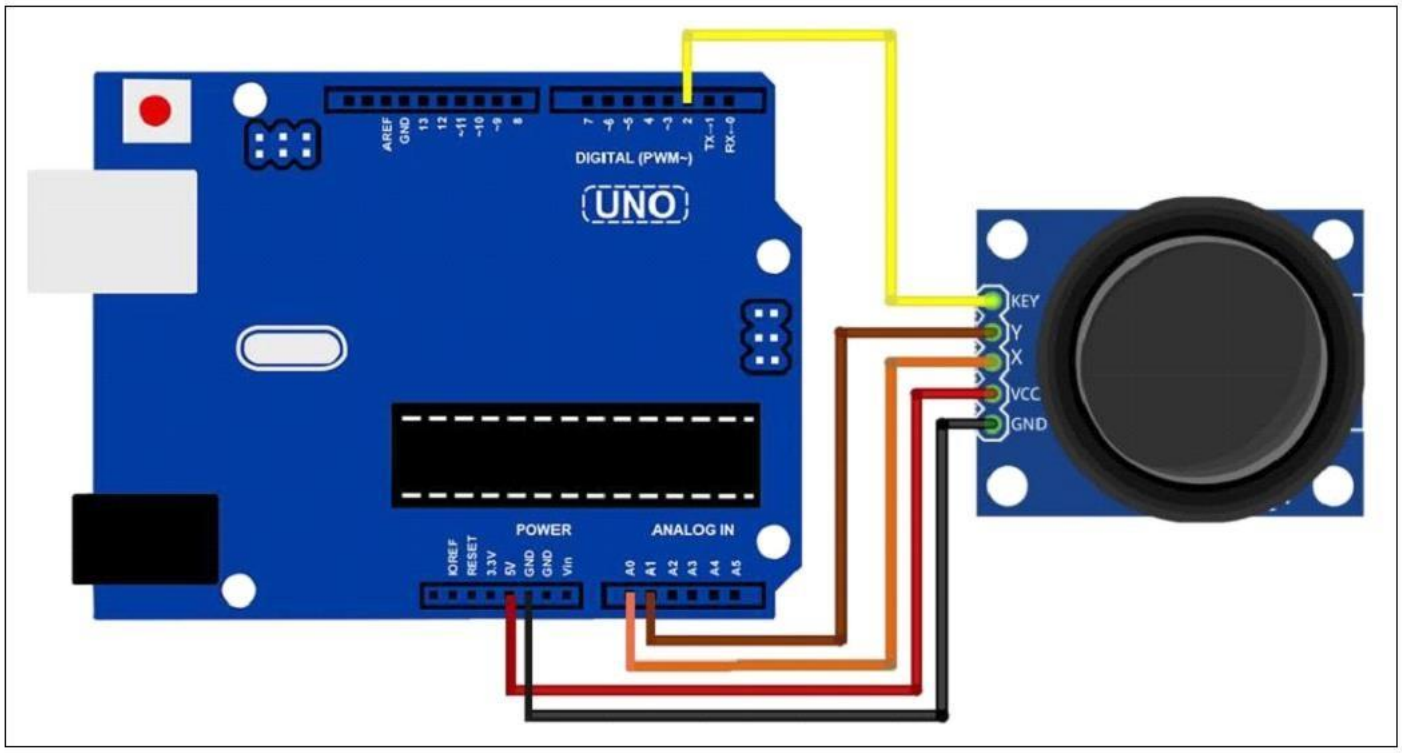
Joystick

The module has 5 pins: Vcc, Ground, X, Y, Key. Note that the labels on yours may be slightly different, depending on where you got the module from. The thumb stick is analog and should provide more accurate readings than simple „directional" joysticks tact use some forms of buttons, or mechanical switches. Additionally, you can press the joystick down (rather hard on mine) to activate a „press to select" push-button.

We have to use analog Arduino pins to read the data from the X/Y pins, and a digital pin to read the button. The Key pin is connected to ground, when the joystick is pressed down, and is floating otherwise. To get stable readings from the Key /Select pin, it needs to be connected to Vcc via a pull-up resistor. The built in resistors on the Arduino digital pins can be used. For a tutorial on how to activate the pull-up resistors for Arduino pins, configured as inputs

Connection

wiring diagram



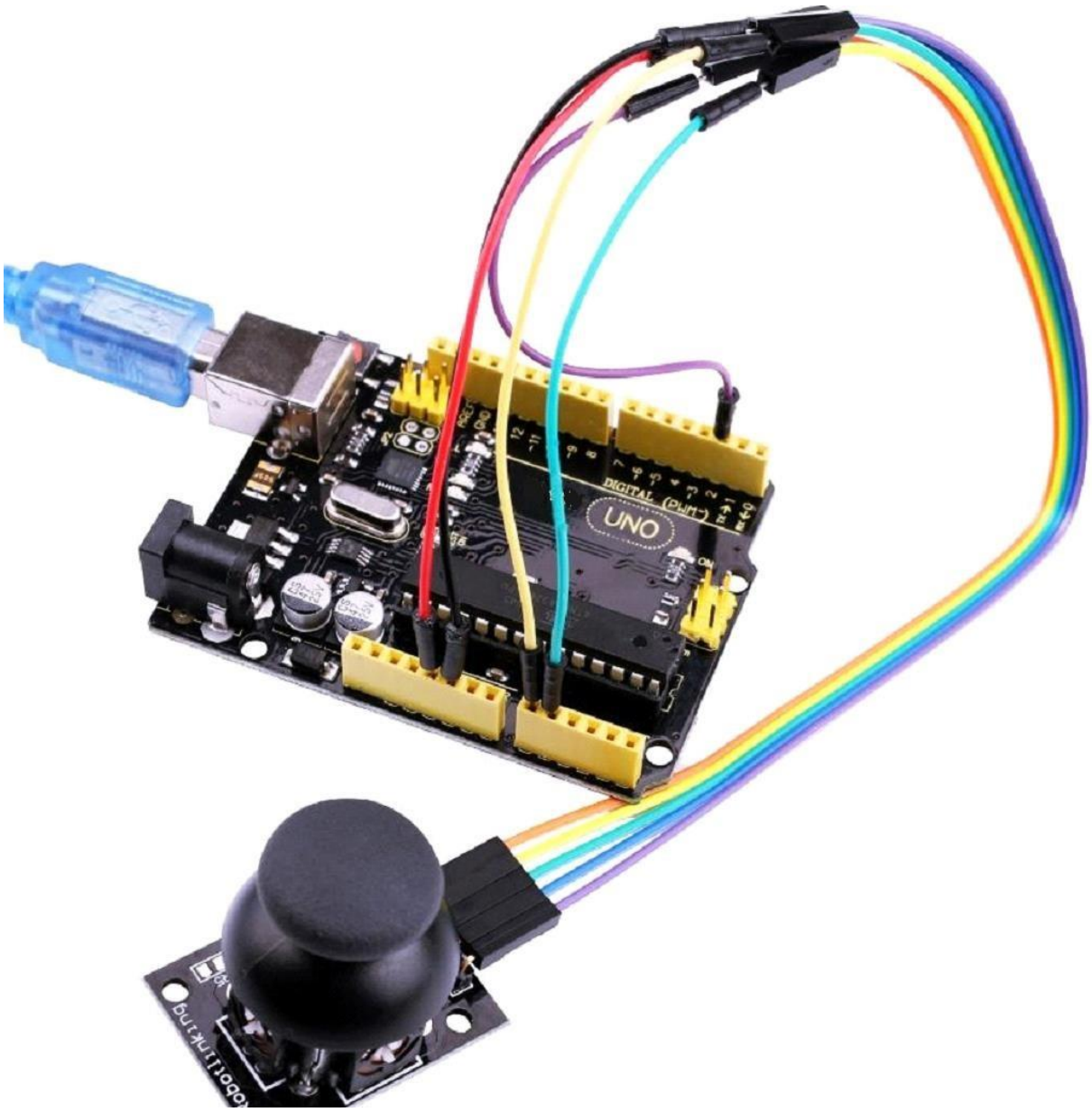
We need 5 connections to the joystick.
The connection are : Key, Y, X, Voltage and Ground.
"Y and X" are Analog and "Key" is Digital. If you don"t need the switch then you can use only 4 pins.

Code

Analog joysticks are basically potentiometers so they return analog values.

When the joystick is in the resting position or middle, it should return a value of about 512.

The range of values go from 0 to 1024.



Lesson 24 IR Receiver Module

Overview

Using an IR Remote is a great way to have wireless control of your project.

Infrared remote is simple and easy to use. In this tutorial we will be connecting the IR receiver to the UNO, and then use a Library that was designed for this particular sensor.

In our sketch we will have all the IR Hexadecimal codes that are available on this remote, we will also detect if the code was recognized and also if we are holding down a key.

Component Required:

(1) x Quaduino Uno R3

(1) x IR receiver module

(1) x IR remote

(3) x F-M wires

Component Introduction

IR RECEIVER SENSOR:

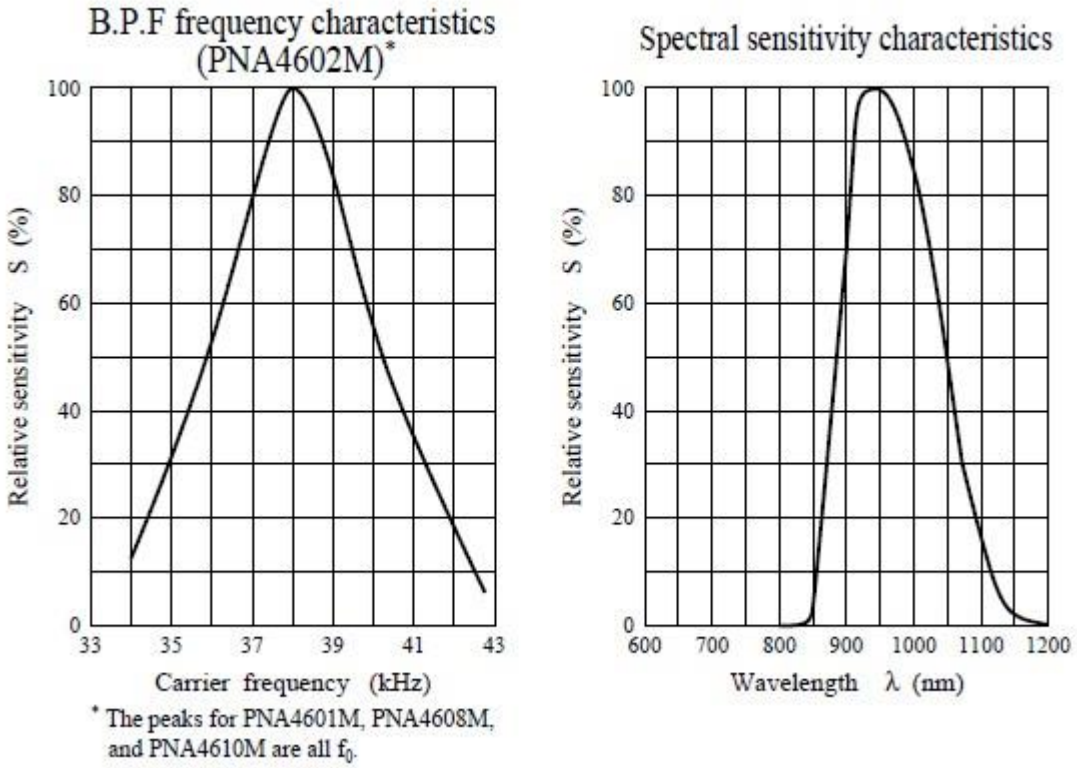
IR detectors are little microchips with a photocell that are tuned to listen to infrared light. They are almost always used for remote control detection - every TV and DVD player has one of these in the front to listen for the IR signal from the clicker. Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels. IR light is not visible to the human eye, which means it takes a little more work to test a setup.

There are a few difference between these and say a CdS Photocells:

IR detectors are specially filtered for Infrared light, they are not good at detecting visible light. On the other hand, photocells are good at detecting yellow/green visible light, not good at IR light.

- IR detectors have a demodulator inside that looks for modulated IR at 38 KHz. Just shining an IR LED won't be detected, it has to be PWM blinking at 38KHz. Photocells do not have any sort of demodulator and can detect any frequency (including DC) within the response speed of the photocell (which is about 1KHz)
- IR detectors are digital out - either they detect 38KHz IR signal and output low (0V) or they do not detect any and output high (5V). Photocells act like resistors, the resistance changes depending on how much light they are exposed to.

What You Can Measure

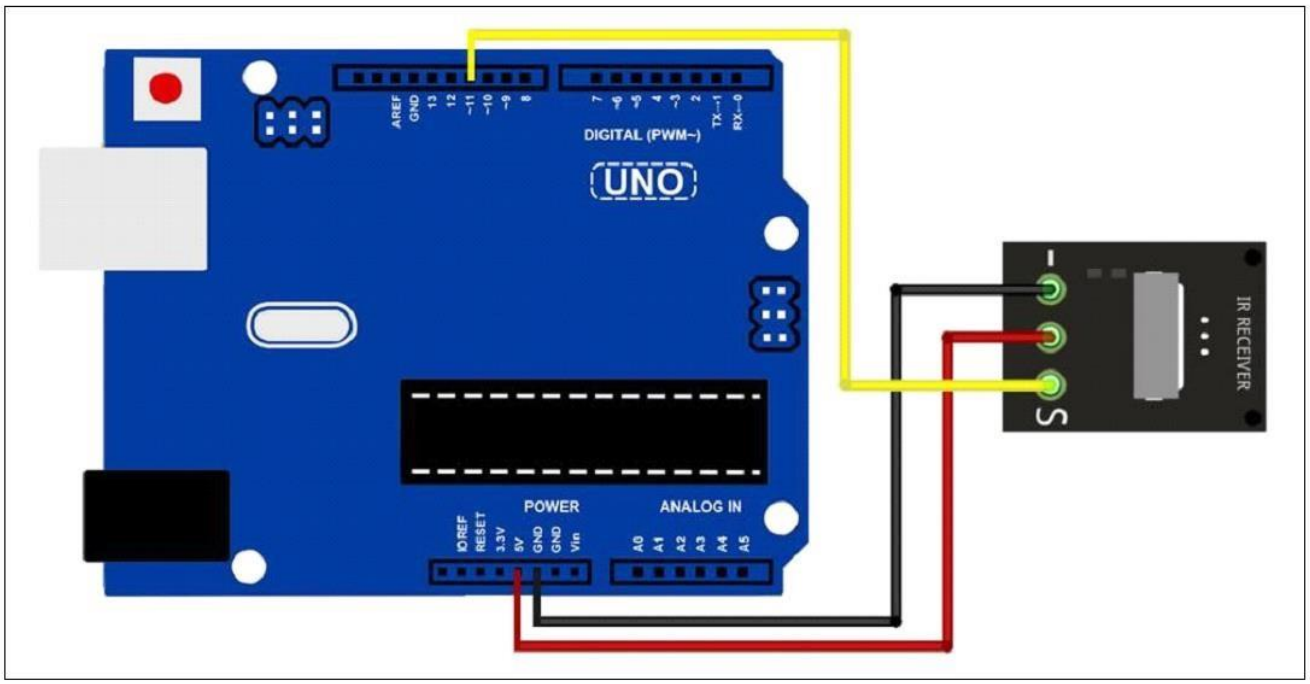


As you can see from these datasheet graphs, the peak frequency detection is at 38 KHz and the peak LED color is 940 nm. You can use from about 35 KHz to 41 KHz but the sensitivity will drop off so that it won't detect as well from afar. Likewise, you can use 850 to 1100 nm LEDs but they won't work as well as 900 to 1000nm so make sure to get matching LEDs! Check the datasheet for your IR LED to verify the wavelength.

Try to get a 940nm - remember that 940nm is not visible light (its Infra Red)!

Connection

wiring diagram



There are 3 connections to the IR Receiver.

The connections are : Signal, Voltage and Ground.

The "-" is the Ground, "S" is signal, and middle pin is Voltage 5V.

Code

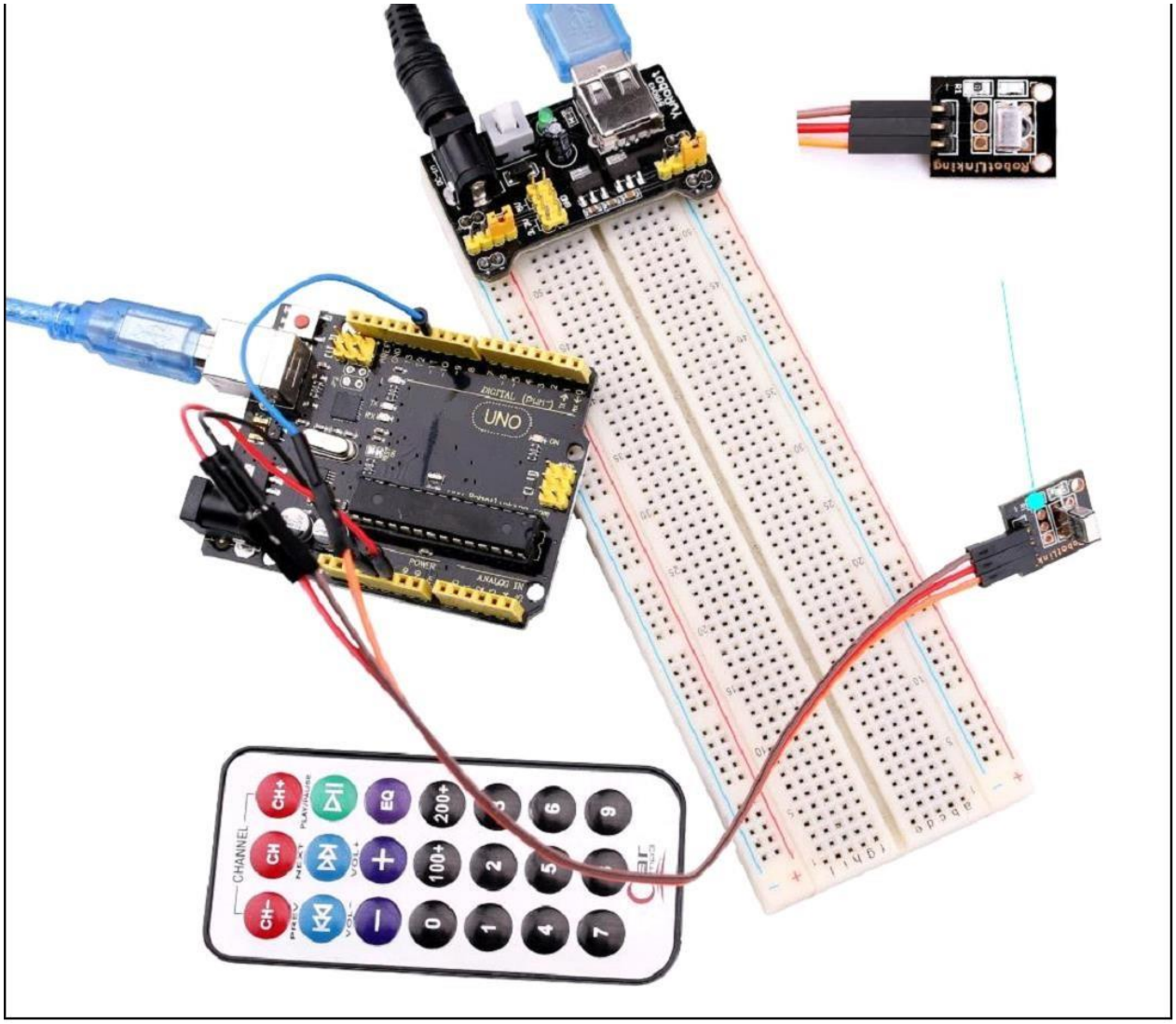
Now that we have the physical setup, all we need now is the code.

Before you can run this, you have to make sure that you have install the <IRremote> library.

Or you need to install again. If you do not do this, your code won't work.

Next we will move the <RobotIRremote> out of the Library folder, we do this because that library conflicts with the one we will be using. You can just drag it back inside the library folder once you are done programming your microcontroller.

Once you have installed the Library, just go ahead and restart your IDE Software.



Lesson 25 Water Level Detection Sensor Module

Overview

In this lesson, you will learn how to use a water level detection sensor module.

This module can perceive the depth of water and the core component is an amplifying circuit which is made up of a transistor and several pectinate PCB routings. When put into the water, these routings will present a resistor that can change along with the change of the water"s depth. Then, the signal of water"s depth is converted into the electrical signal, and we can know the change of water"s depth through the ADC function of UNO R3.

Component Required:

(1) x Quaduino Uno R3

(1) x Water lever detection sensor module

(3) x F-M wires

Component Introduction

Water sensor:

A water sensor brick is designed for water detection, which can be widely used in sensing the rainfall, water level, even the liquate leakage. The brick is mainly composed of three parts: an electronic brick connector, a 1 MΩ resistor, and several lines of bare conducting wires.

This sensor works by having a series of exposed traces connected to ground. Interlaced between the grounded traces are the sense traces.

The sensor traces have a weak pull-up resistor of 1 MΩ. The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace. Believe it or not this circuit will work with the digital I/O pins of your UNO R3 board or you can use it with the analog pins to detect the amount of water induced contact between the grounded andsensor traces.

This item can judge the water level through with a series of exposed parallel wires stitch to

measure the water droplet/water size. It can easily change the water size to analog signal, and output analog value can directly be used in the program function, then to achieve the function of water level alarm.

It has low power consumption, and high sensitivity.

Features:

1、Working voltage: 5V

2、Working Current: <20ma

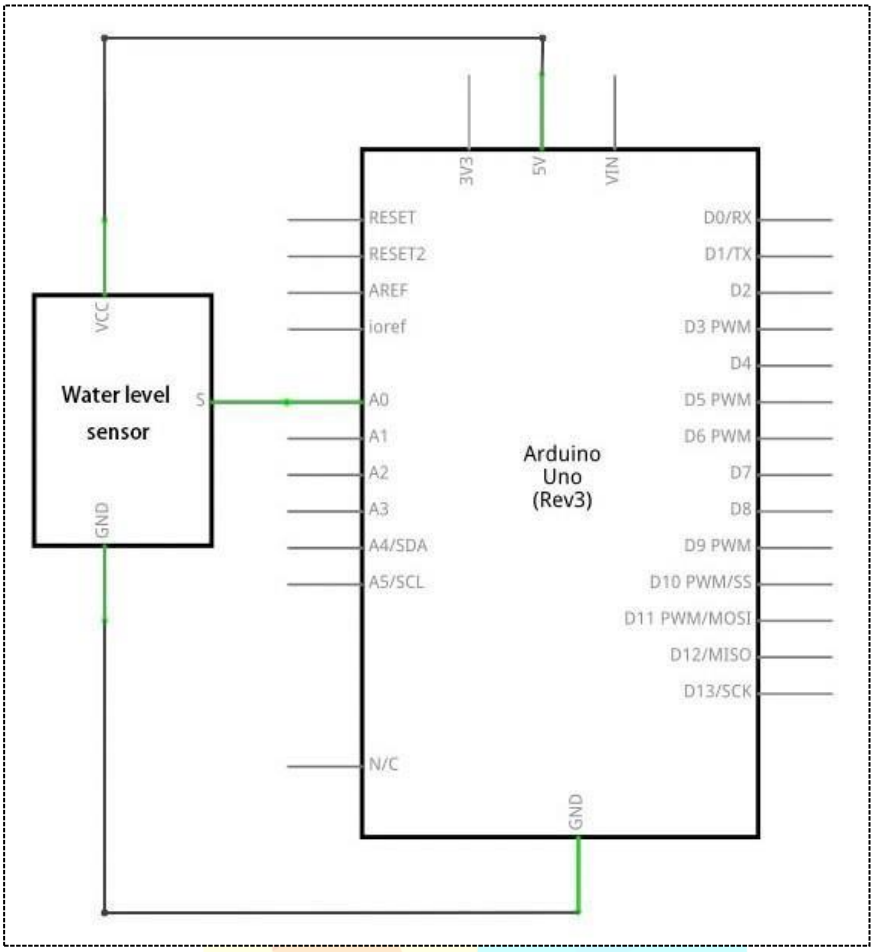
3 、 Interface: Analog

4、Width of detection: 40mm×16mm

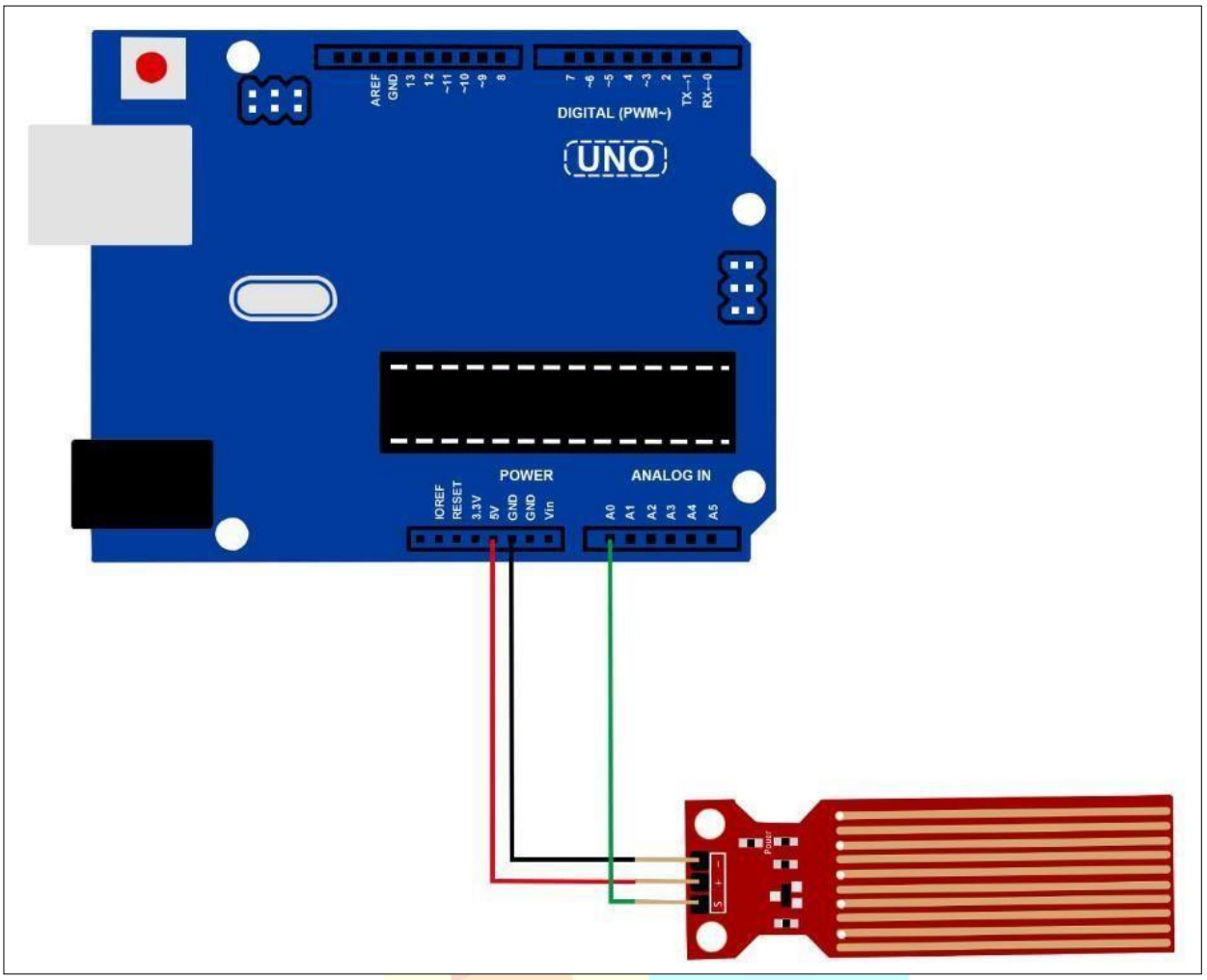
5、Working Temperature: 10℃~30℃

6、Output voltage signal: 0~4.2V

Connection Schematic



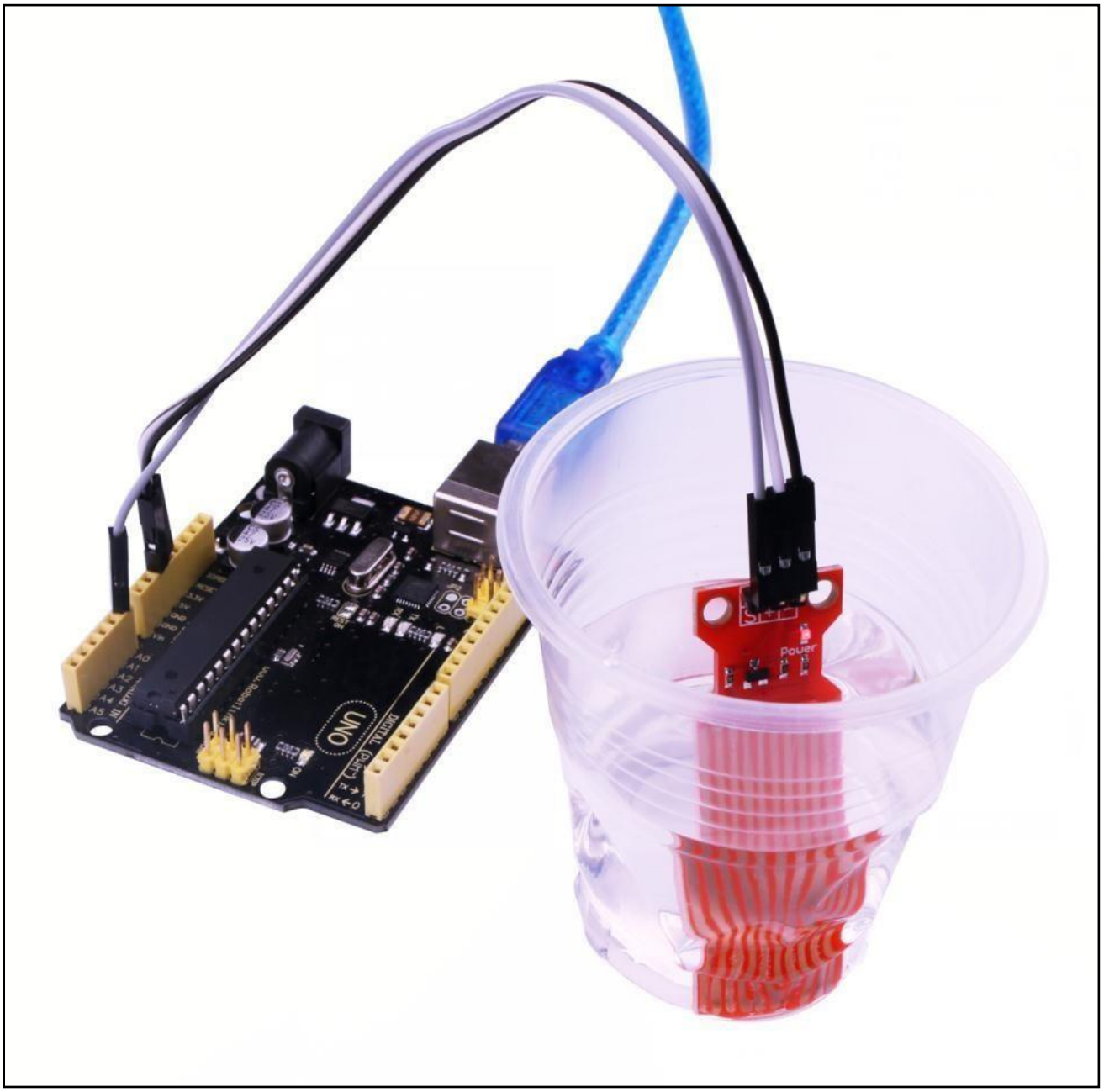
wiring diagram



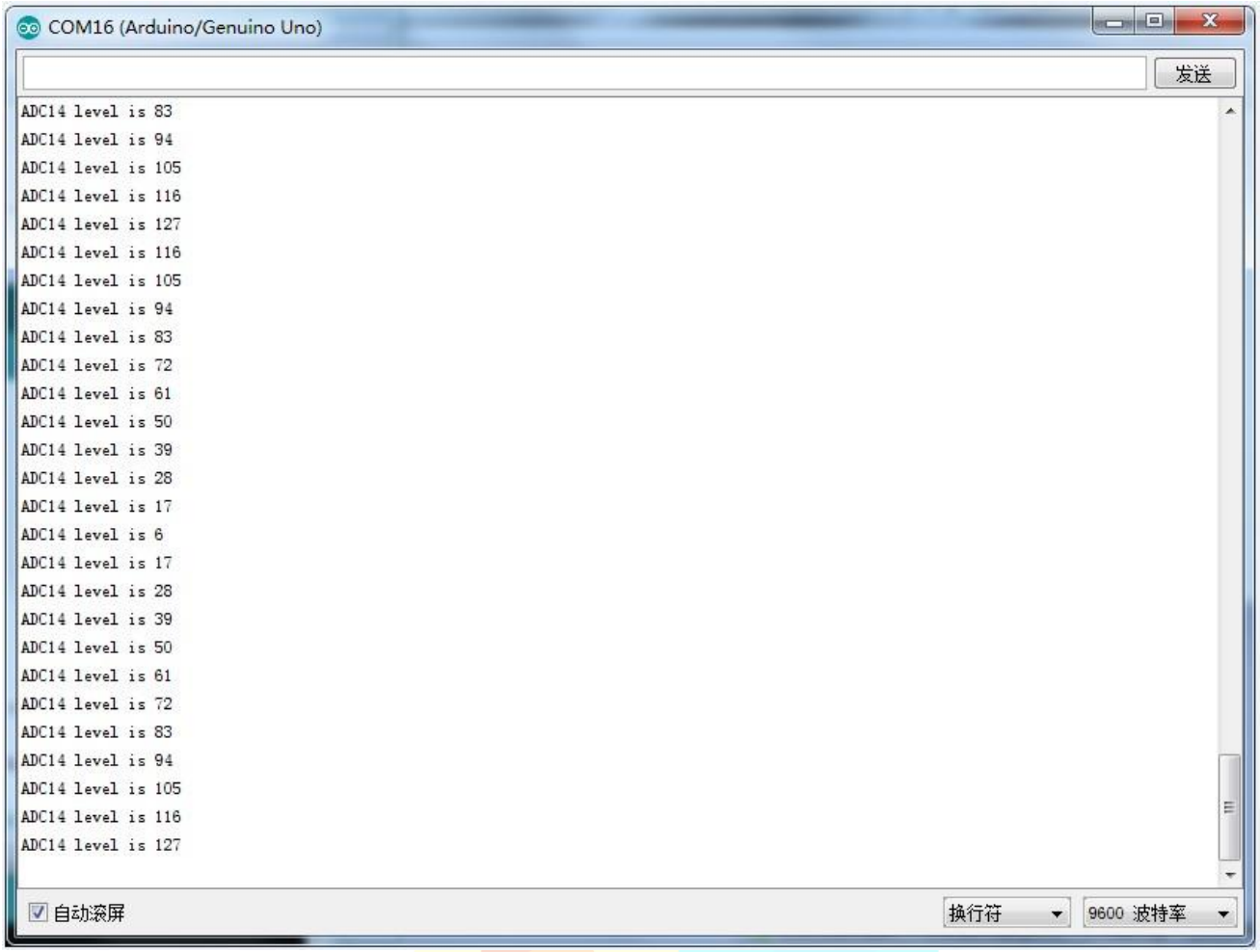
Wiring tips: Power supply (+) is connected to 5V of UNO R3 board, ground electrode (-) is connected to GND. Signal output (S) is connected to the ports (A0-A5) which have function of inputting analog signal in UNO R3 board, random one is OK, but it should define the same demo code as the routine.

Code

See the code file.



open the monitor you can see the data as below:



Lesson 26 Real Time Clock Module

Overview

In this lesson, you will learn how to use the DS3231, clock module that displays the year, month, day, hour, minute, second and week. Support is via a backup battery trickle charger, which can be used unless being connected to UNO with only three data cables.

Component Required:

(1) x Quaduino Uno R3

(1) x DS3231 real time clock module

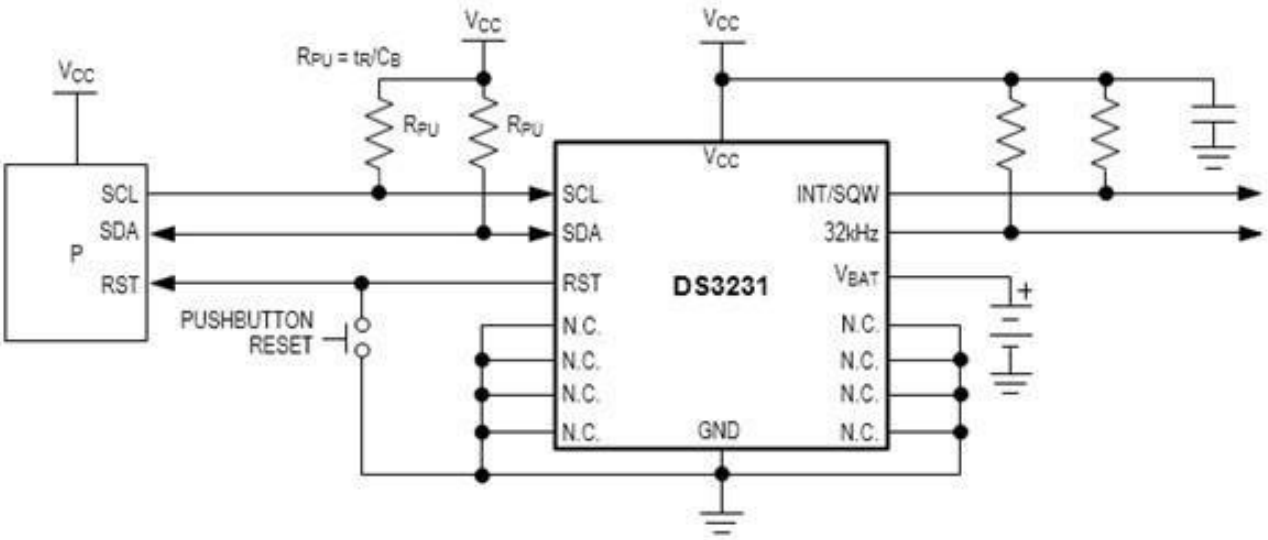
(4) x F-M wires

Component Introduction

DS1302

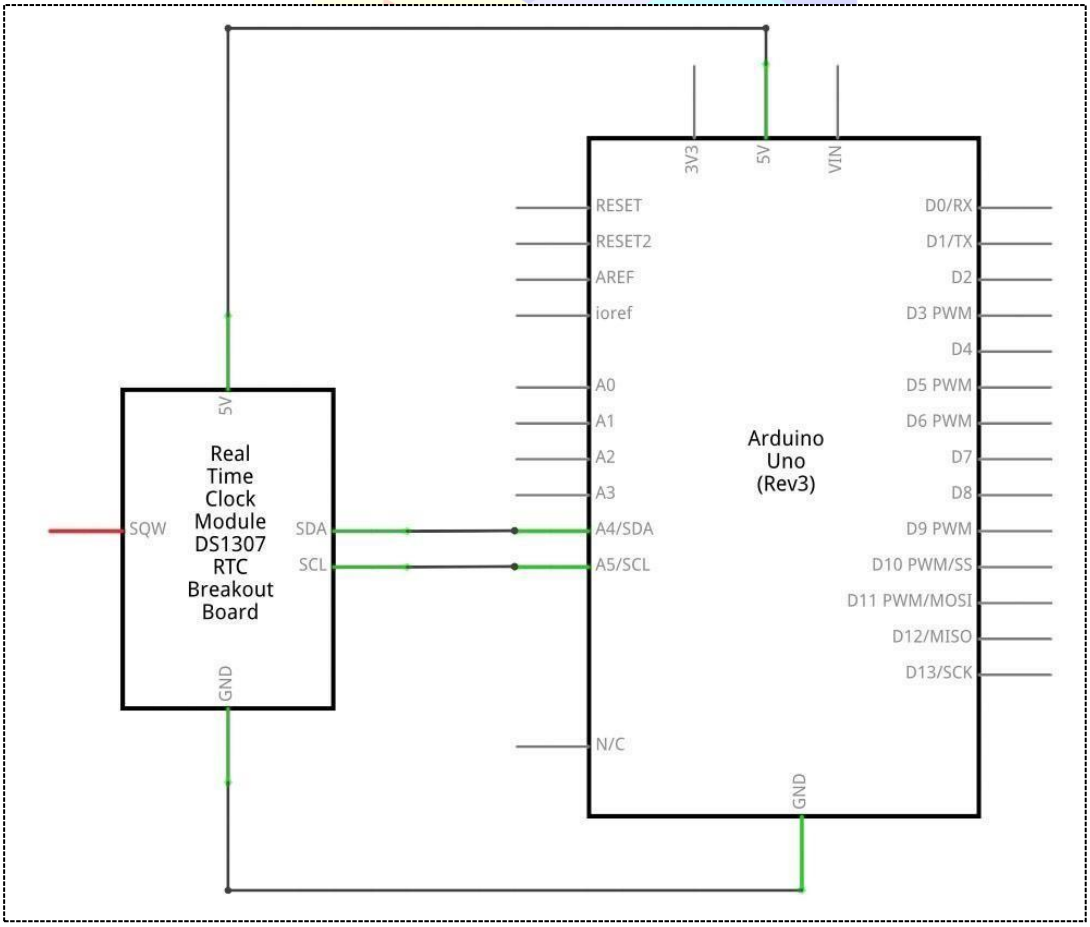
DS3231

The DS3231 is a simple time-keeping chip. It has an integrated battery, so the clock can continue keeping time even when unplugged.

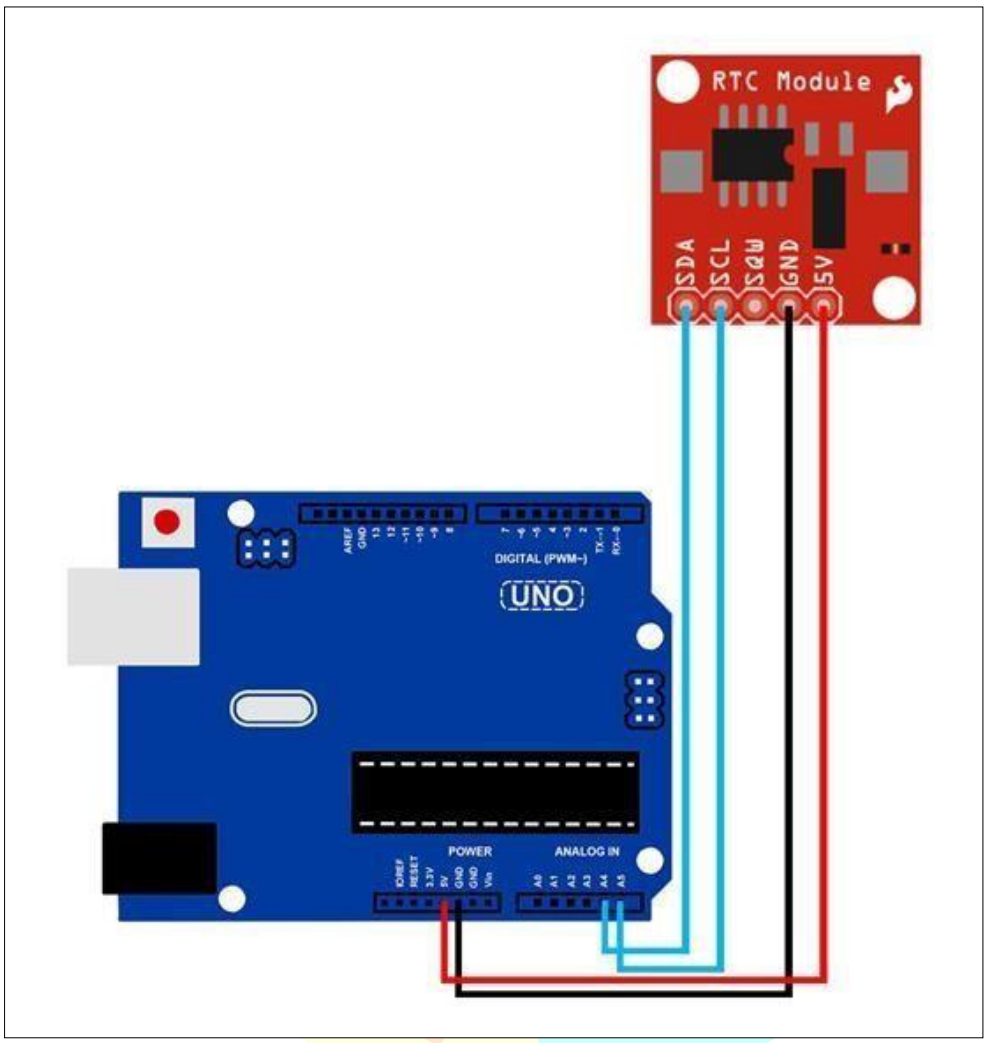


Connection

Schematic



wiring diagram



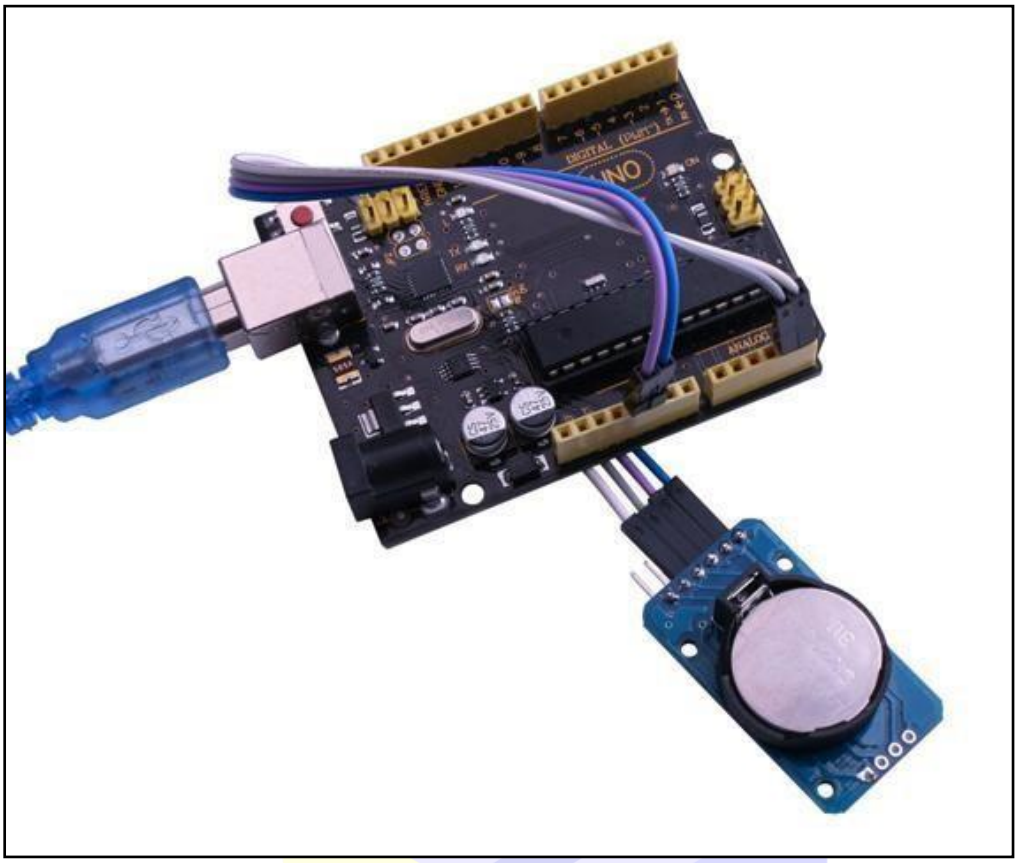
Set up according to the following image.

Ignore the 32K and SQW pins; you will not need them. Plug the SCL pin into your UNO R3 board A5 port, and the SDA pin into the A4 port. The VCC pin plugs into the 5V port, and the GND plugs into the GND port.

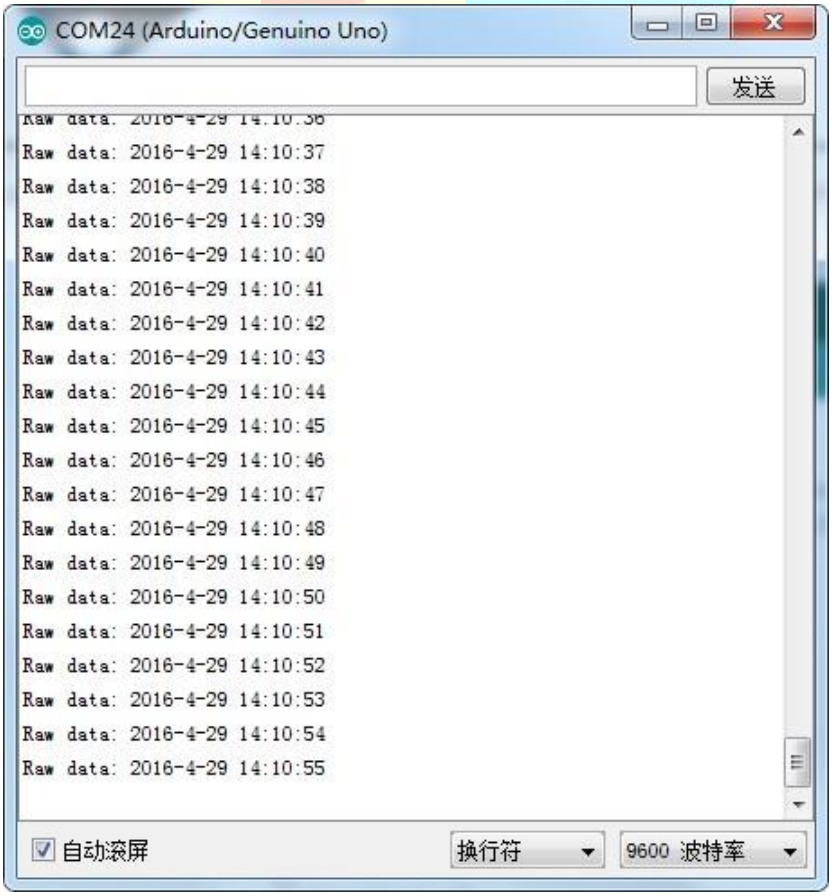
Code

Now that we have the physical setup, all we need now is the code.

Before you can run this, make sure that you have installed the <DS3231> <OneWireKeypad> library or re-install it if needed. Otherwise, your code won't work.



open the monitor you can see the module can read the time as below:



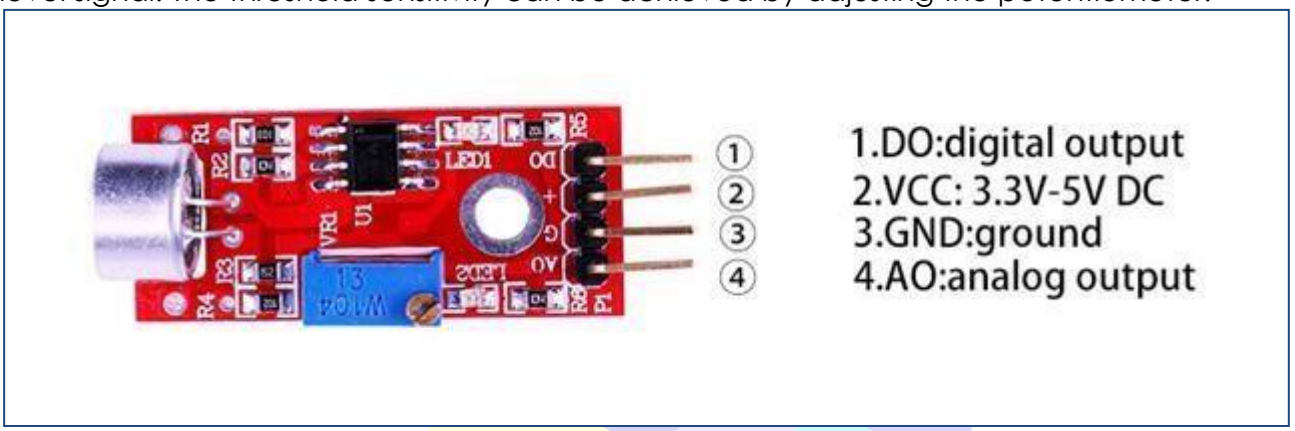
Lesson 27 Sound Sensor Module

Overview

In this lesson, you will learn how to use a sound sensor module. This module has two outputs:

AO: analog output, real-time output voltage signal of microphone

DO: when the intensity of the sound reaches a certain threshold, the output is a high or low level signal. The threshold sensitivity can be achieved by adjusting the potentiometer.



Component Required:

(1) x Quaduino Uno R3

(1) x Sound sensor module

(3) x F-M wires

Component Introduction

Microphone

Transducers are devices which convert energy from one form to other. A microphone is a transducer which converts sound energy to electrical signals. It works opposite to a speaker. Microphones are available in different shapes and sizes. Depending on the application, a microphone may use different technologies to convert sound to electrical signals. Here, we are going to discuss about the electret condenser microphone which is widely used in mobile phones, laptops, etc.

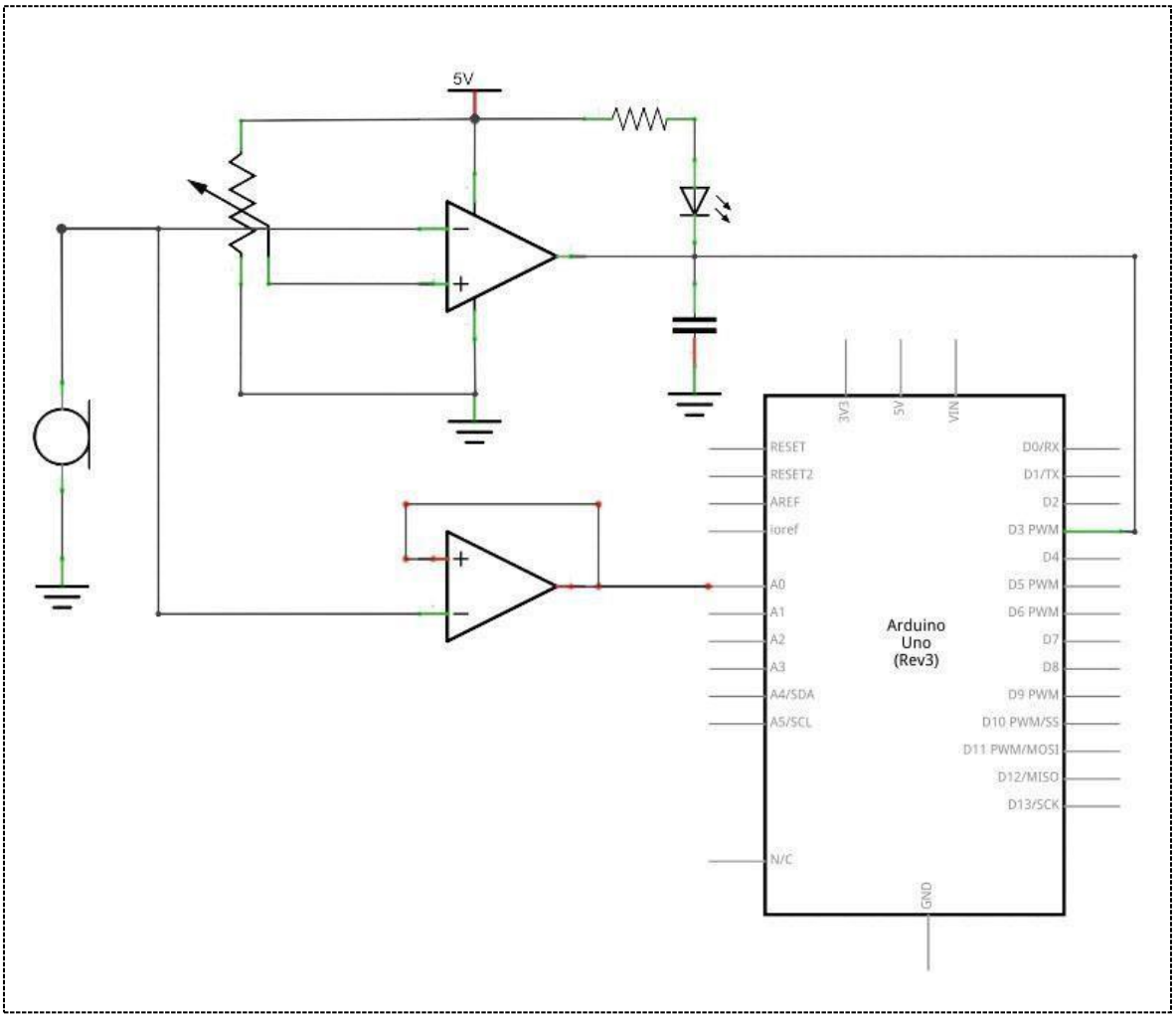
As the name suggests, the electret condenser microphone is a parallel plate capacitor and works on the principle of a variable capacitance. It consists of two plates, one fixed (called the back plate) and the other moveable (called the diaphragm) with a small gap between them. An electric potential charges the plate. When sound strikes the diaphragm it starts moving, thereby changing the capacitance between the plates which in turn results in a variable electric current to flow.

These microphones are widely used in electronic circuits to detect minor sounds or air vibrations which in turn are converted to electrical signals for further use. The two legs as shown in the image above are used to make electrical connection with the circuit.

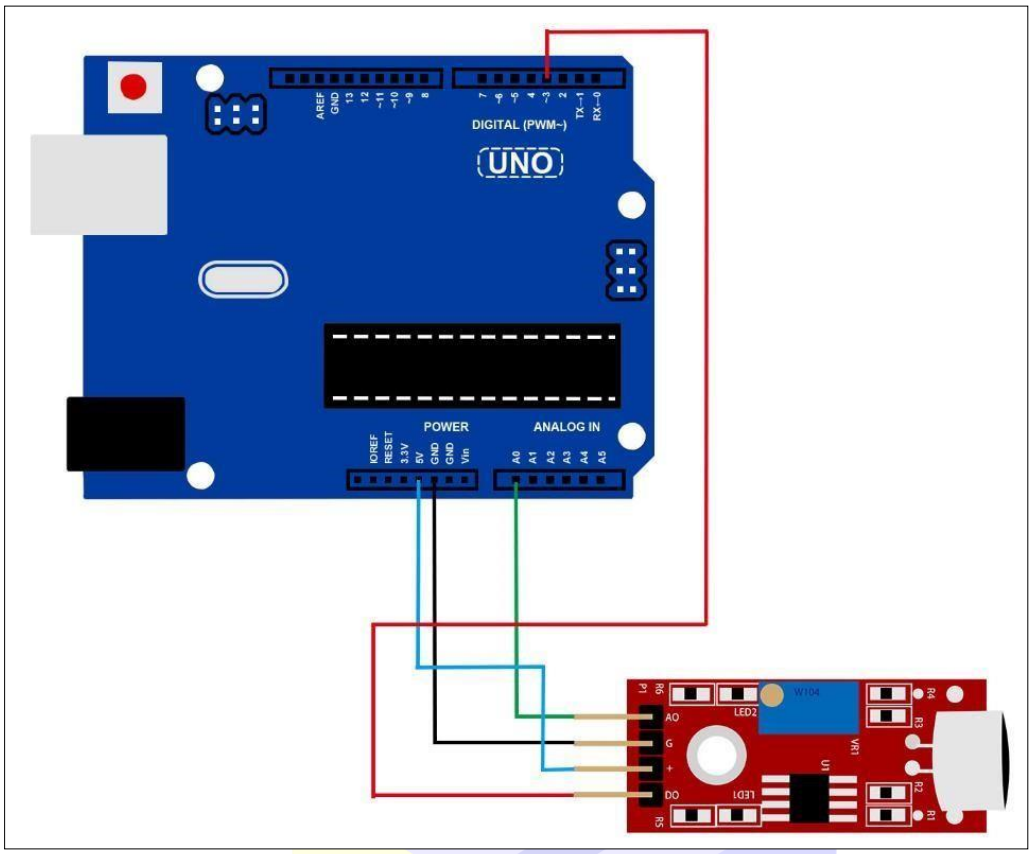
A solid conducting metal body encapsulates the various parts of the microphone. The top face is covered with a porous material with the help of glue. It acts as a filter for the dust particles. The sound signals/air vibrations passes through the porous material and falls on the diaphragm through the hole shown in the image above.

Connection

Schematic

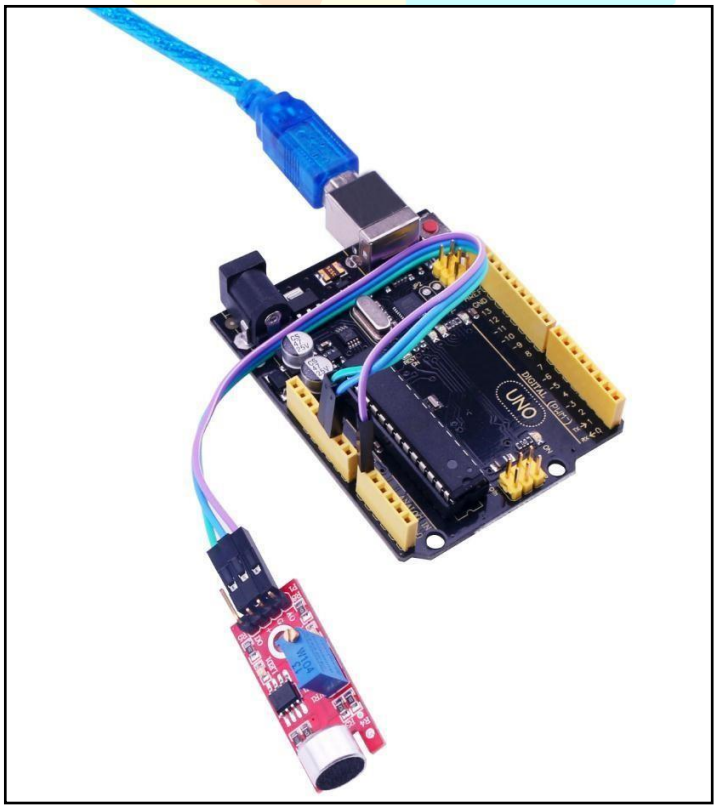


wiring diagram

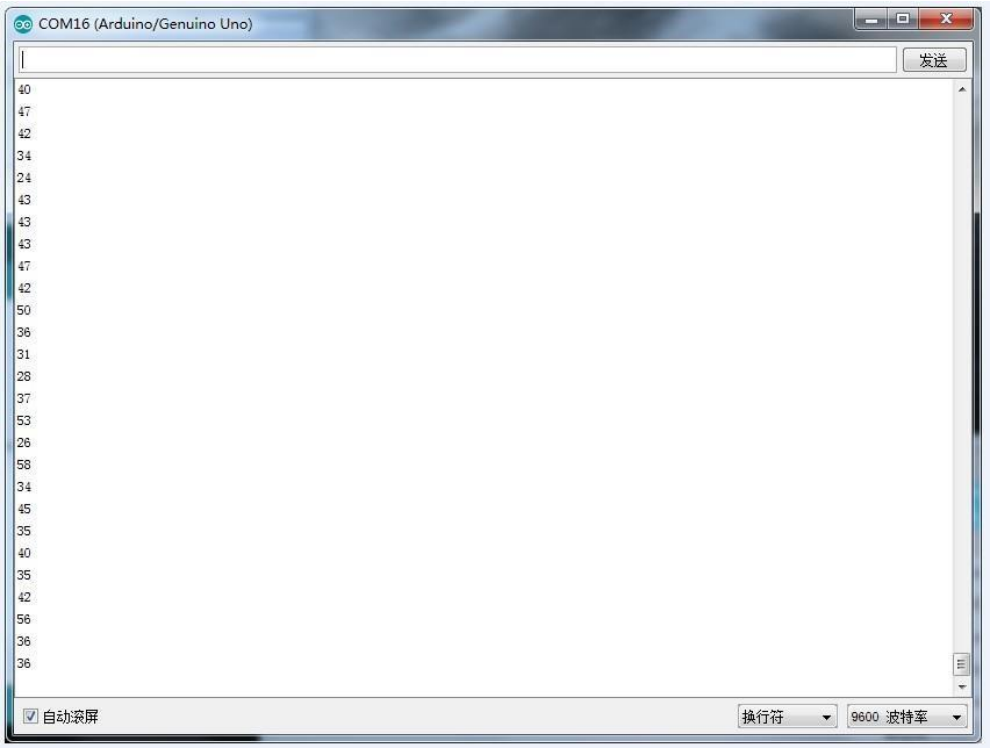


The code

See the code file.



open the monitor you can see the data as below:

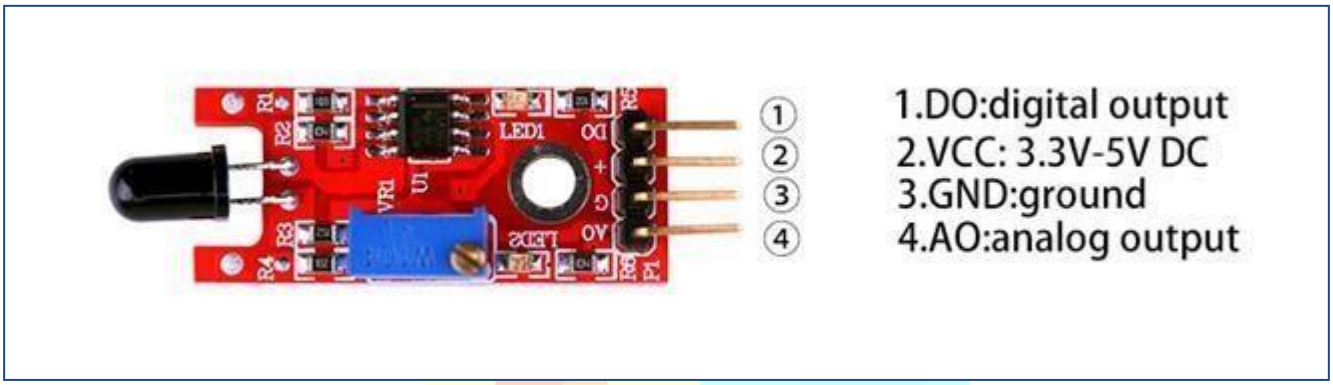

COM16 (Arduino/Genuino Uno)
发送
40
47
42
34
24
43
43
43
47
42
50
36
31
28
37
53
26
58
34
45
35
40
35
42
56
36
36
自动滚屏
换行符
9600 波特率

Lesson 28 FLAME SENSOR MODULE

Overview

In this experiment, we will learn how to use the flame sensor module.

This module is sensitive to the flame and radiation. It also can detect ordinary light source in the range of a wavelength 760nm-1100 nm. The detection distance is up to 100 cm. The Flame sensor can output digital or analog signal. It can be used as a flame alarm or in fire fighting robots.



Component Required:

(1) x Quaduino Uno R3

(1) x USB cable

(1) x Flame sensor module

(x) x F-M wires

Component Introduction

Flame sensor:

- Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm
- Detection distance: 20cm (4.8V) ~ 100cm (1V)
- Detection angle about 60 degrees, it is sensitive to the flame spectrum.
- Comparator chip LM393 makes module readings stable.
- Adjustable detection range.

- Operating voltage 3.3V-5V
- Digital and Analog Output

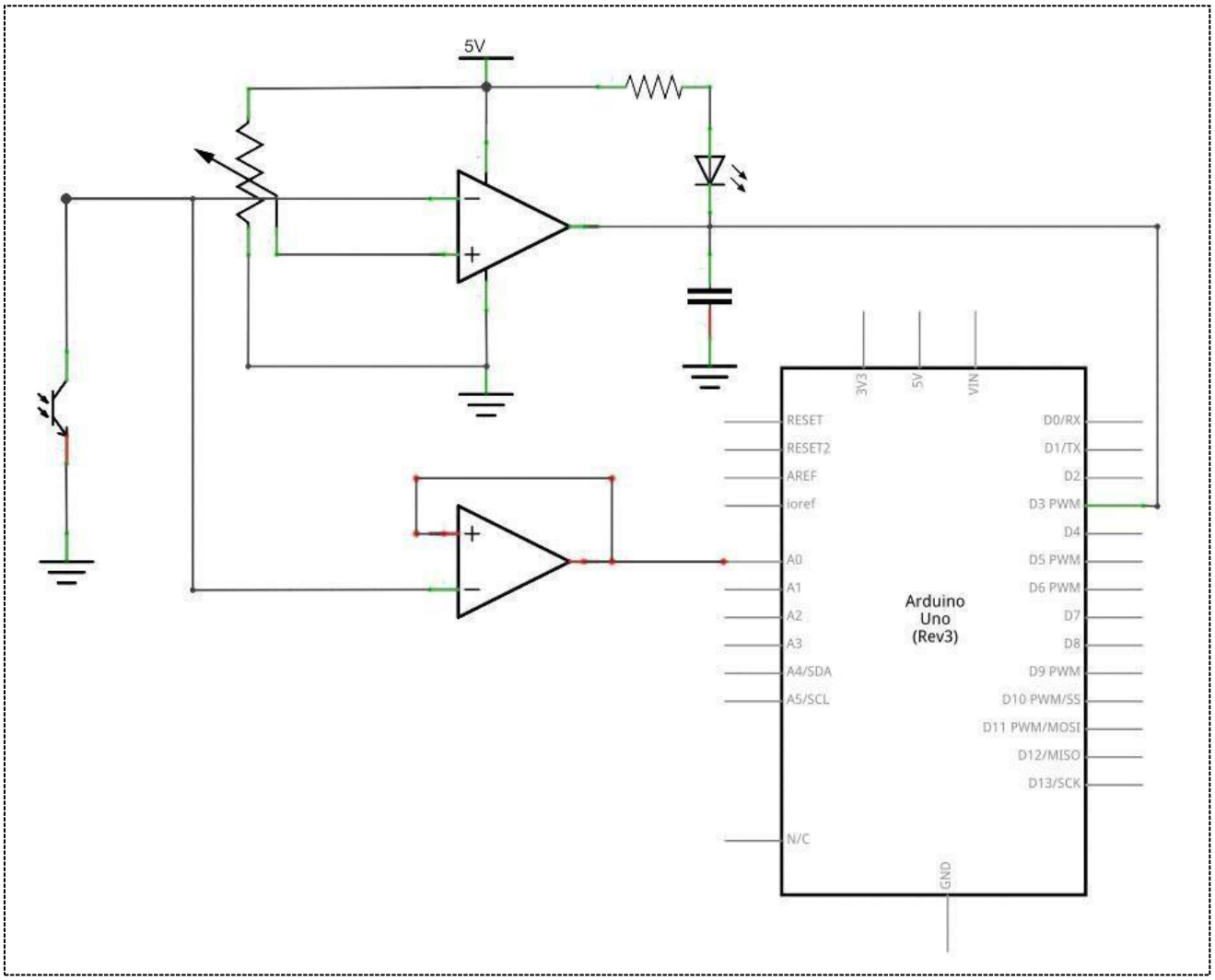
" DO digital switch outputs (0 and 1)

" AO analog voltage output

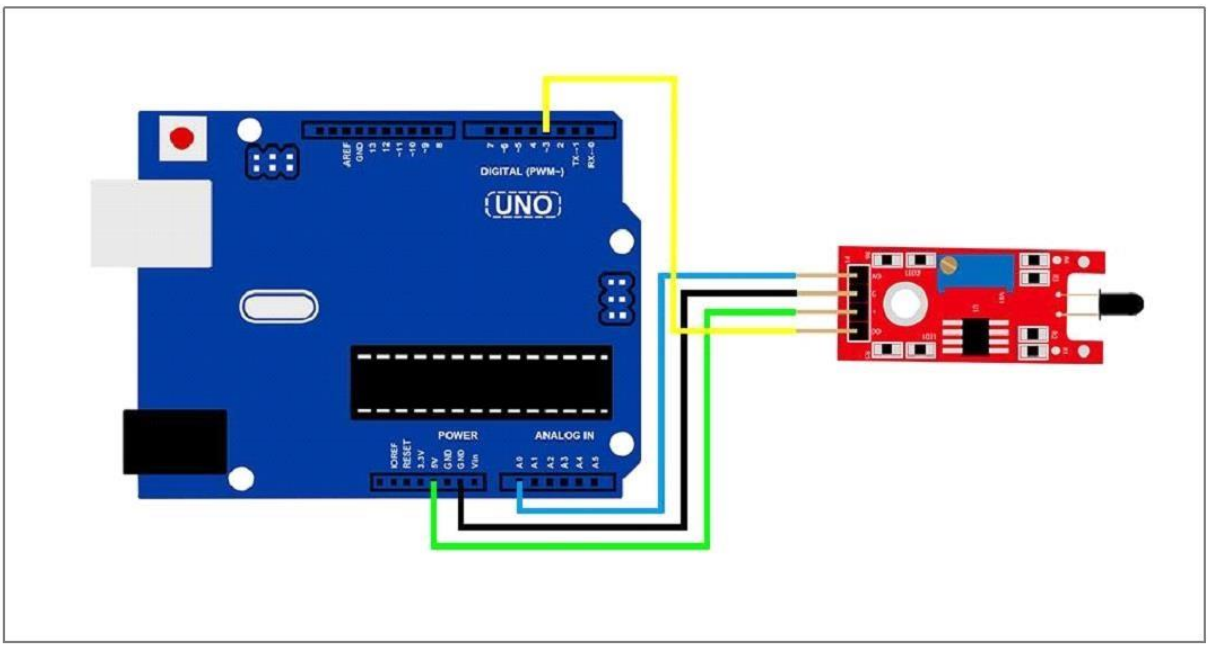
- Power indicator and digital switch output indicator

Connection

Schematic

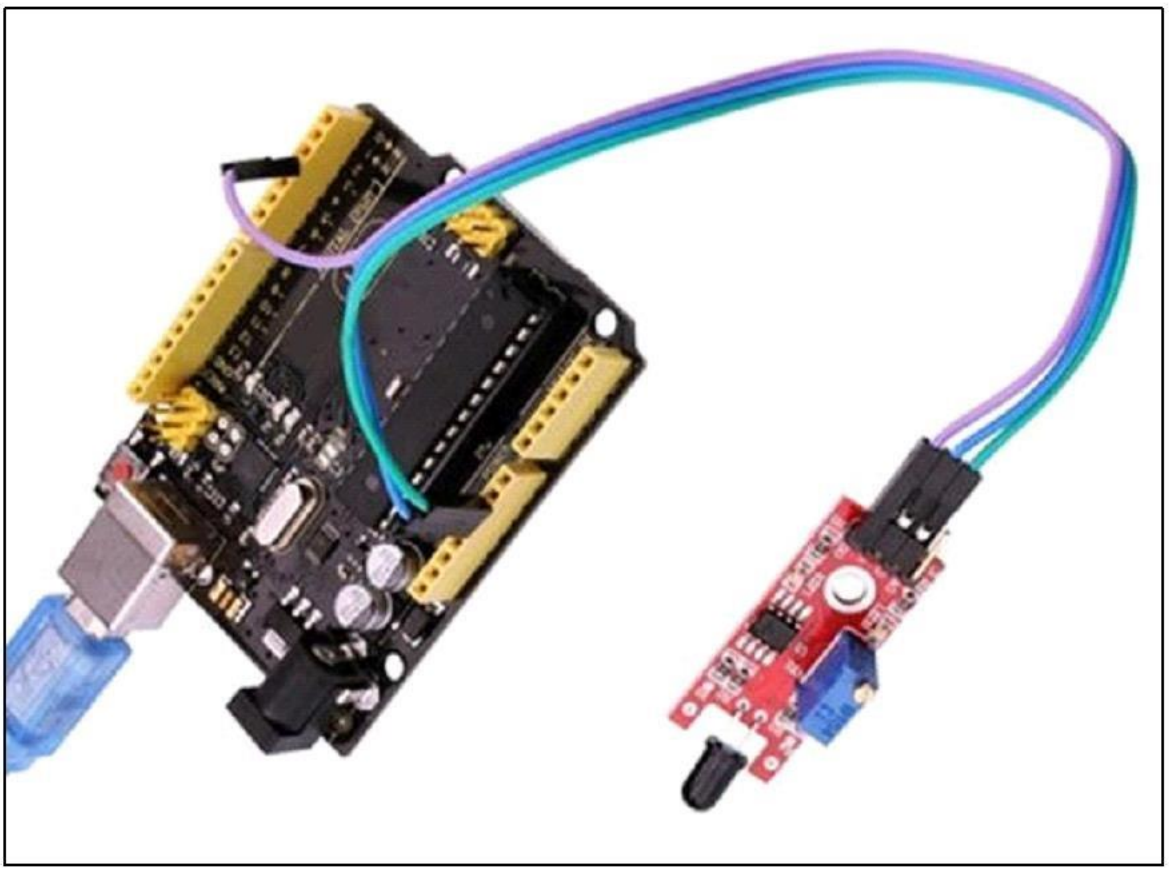


wiring diagram

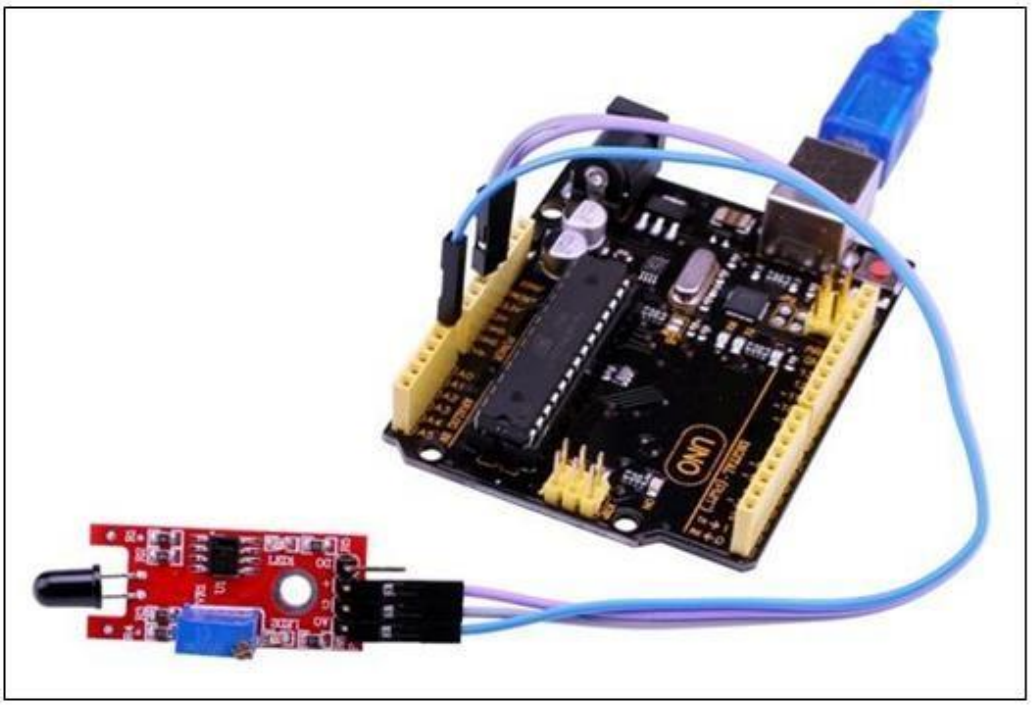


Result

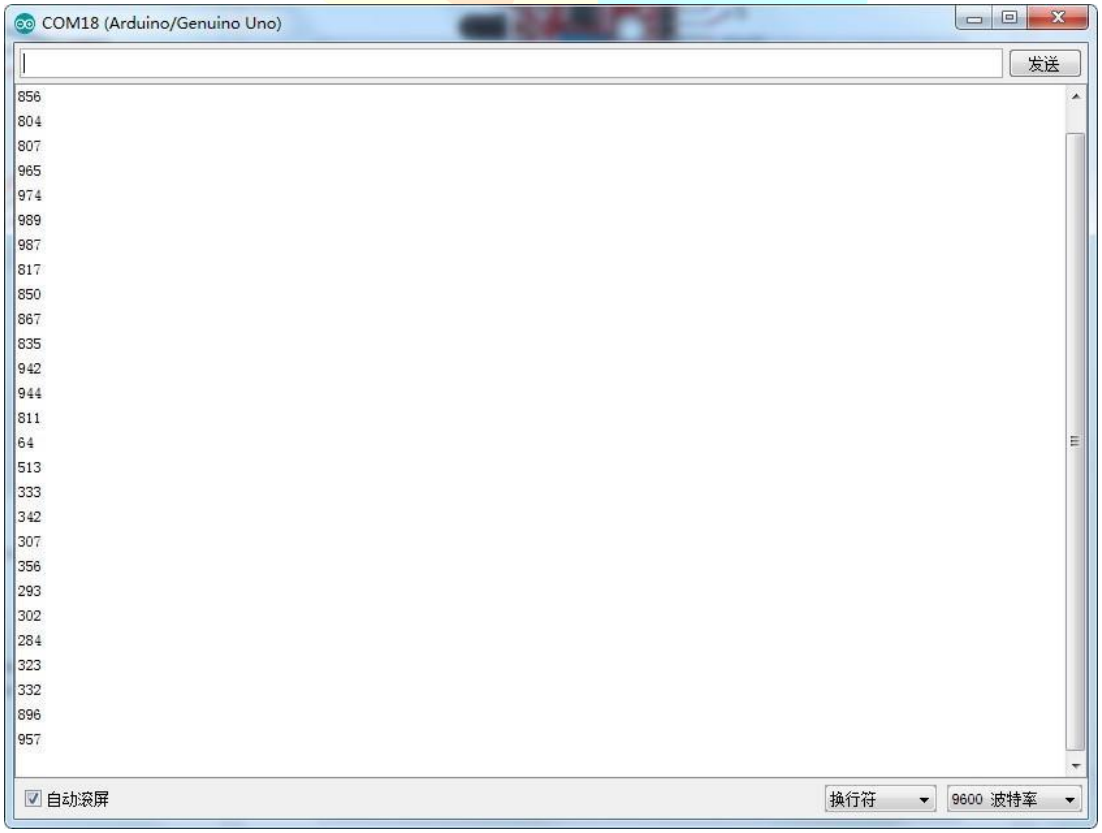
For the flame sensor module, we can choose the output: digital output or analog output. In the following picture, we use the DO port to output. so we can see that if the flame sensor sensing the flame, the light will turn on.



In the following picture, we use the AO port to output. so we can see that if the flame sensor sensing the flame, the module will output a data which reflect the strength of the flame. The number is from 0 to 1023.



Upload the program then open the monitor, we can see the data as below:



Lesson 29 RC522 RFID Module

Overview

In this lesson, you will learn to how to apply the RC522 RFID Reader Module on UNO R3. This module uses the Serial Peripheral Interface (SPI) bus to communicate with controllers such as Arduino, Raspberry Pi, beagle board, etc.

Component Required:

(1) x Quaduino Uno R3

(1) x RC522 module

(3) x F-M wires

Component Introduction

RC522

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO 14443A / MIFARE® mode.

The MFRC522"s internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE®Classic (e.g. MIFARE® Standard) products. The MFRC522 supports contactless communication using MIFARE® higher transfer speeds up to 848 kbit/s in both directions.

Various host interfaces are implemented:

• SPI interface

• serial UART (similar to RS232 with voltage levels according pad voltage supply)

• I2C interface.

The figure below shows a typical circuit diagram, using a complementary antenna connection to the MFRC522.

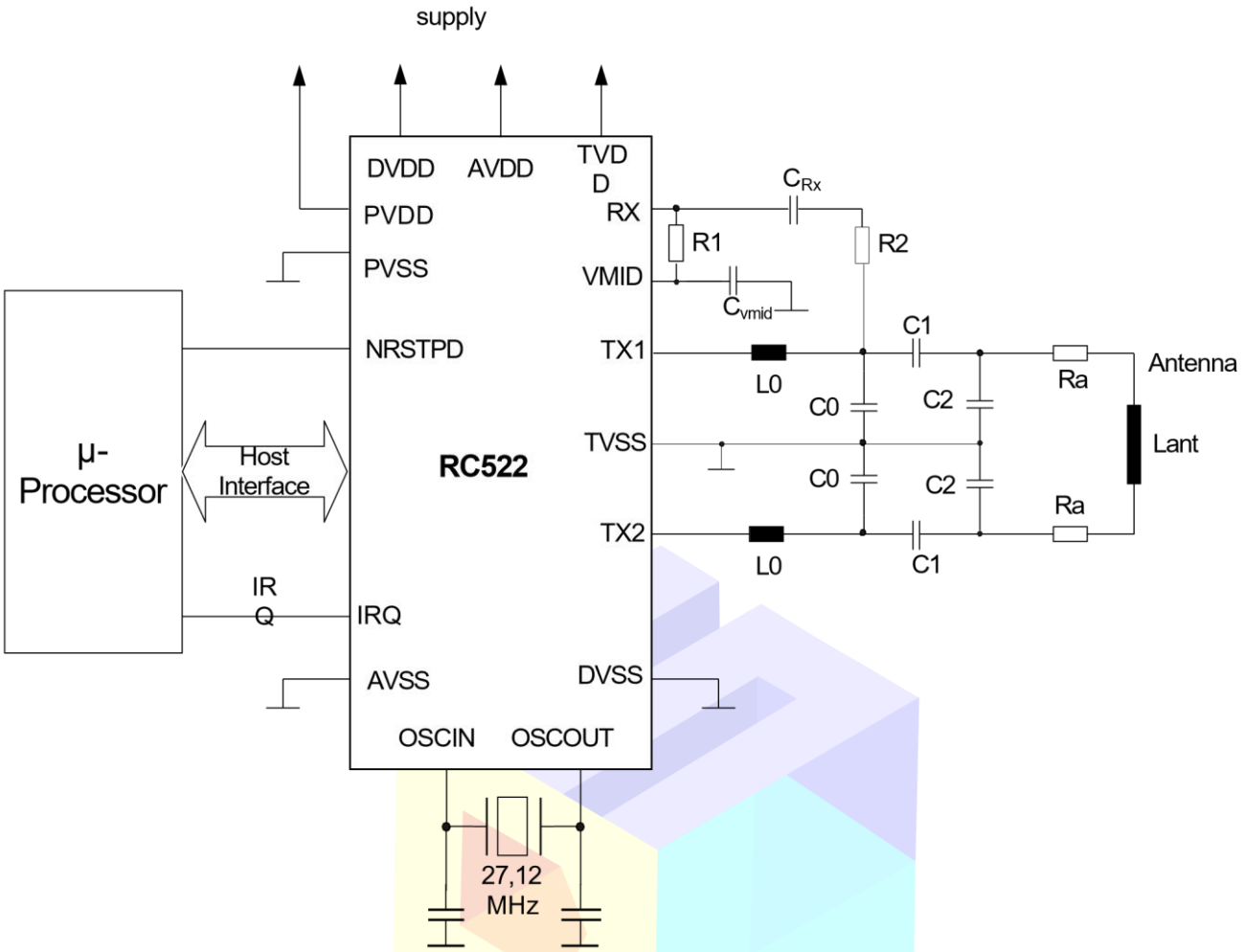
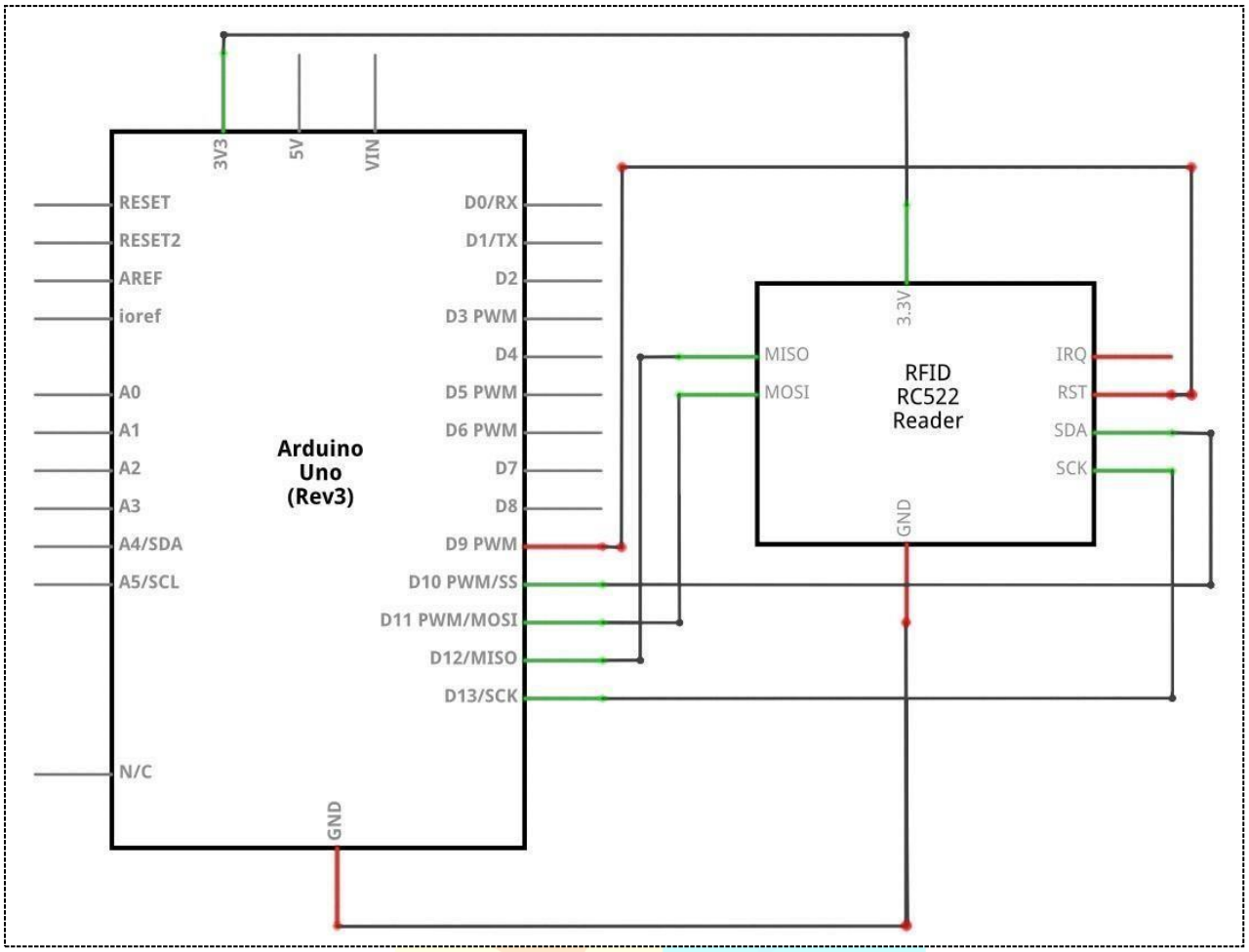
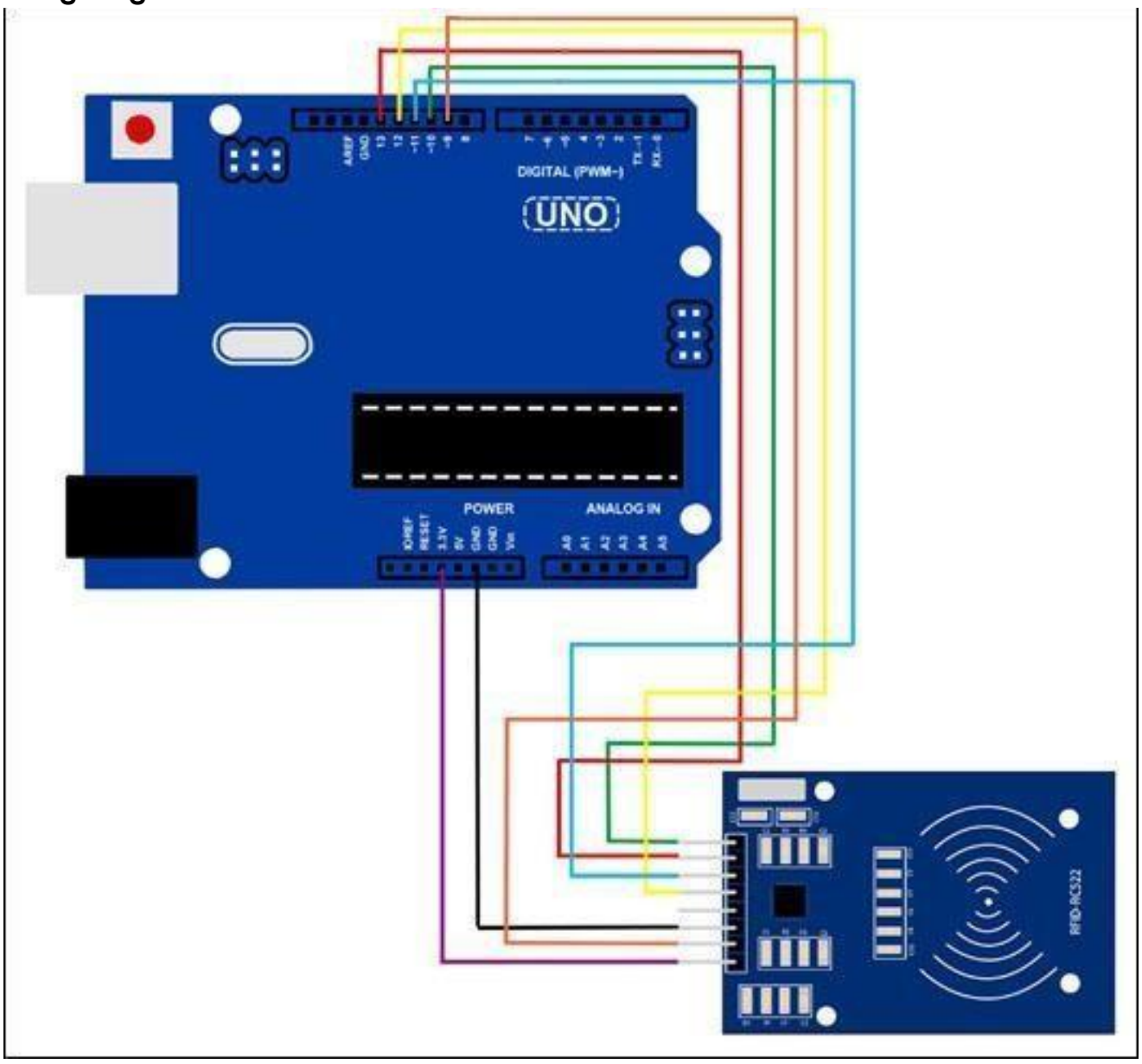


Fig 32. Typical Application CircuitDiagram

Connection Schematic



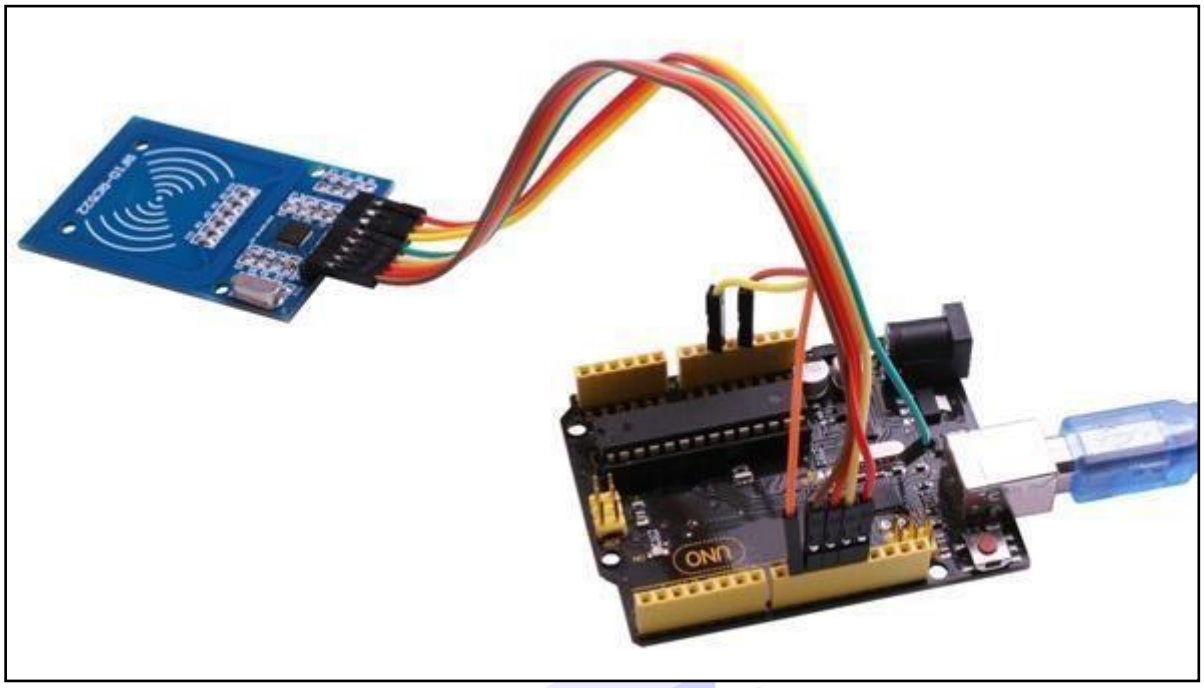
wiring diagram



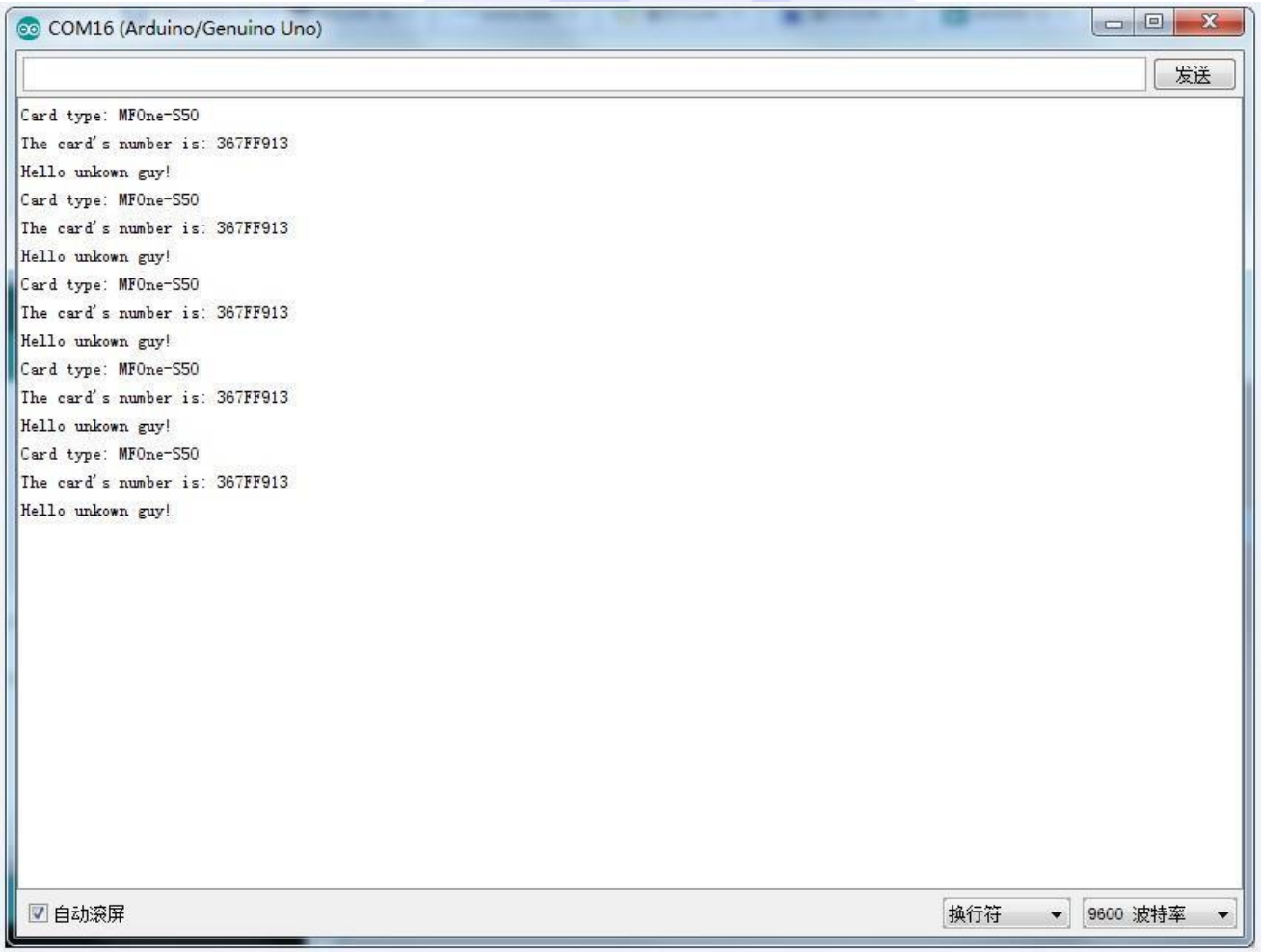
Code

Now that we have the physical setup, all we need now is the code.

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



open the monitor you can see the data as blow:



Lesson 30 LM35 TEMPERATURE SENSOR

Overview

UNO R3 board can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the programming language". Using the UNO R3 board I will show you how to get the analog input from the LM35 temperature sensor and display the information in the serial monitor.

Component Required:

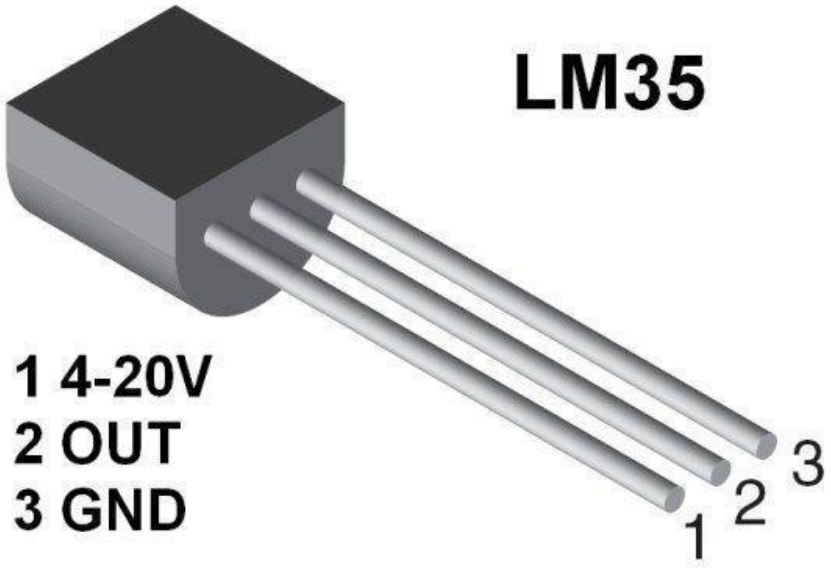
(1) x Quaduino Uno R3

(1) x lm35

(3) x F-M wires

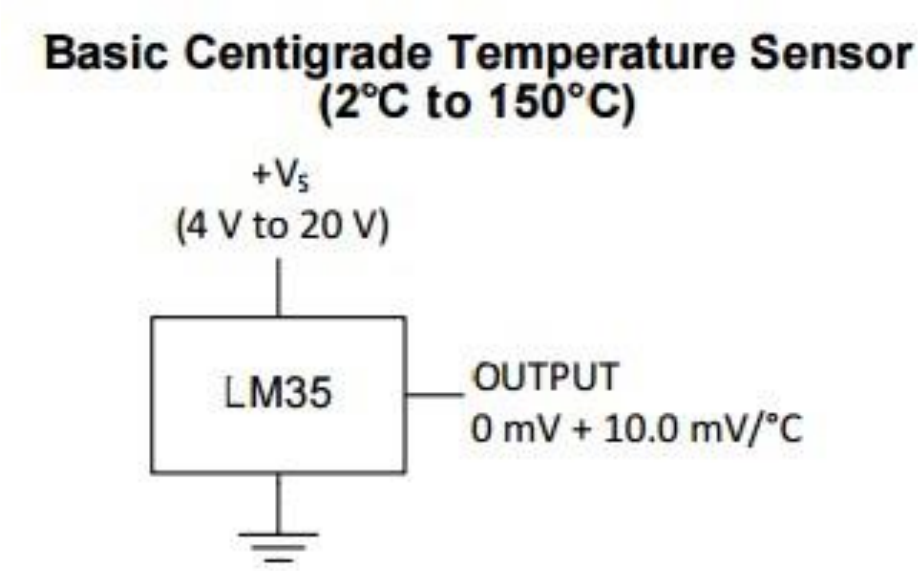
Component Introduction

LM35

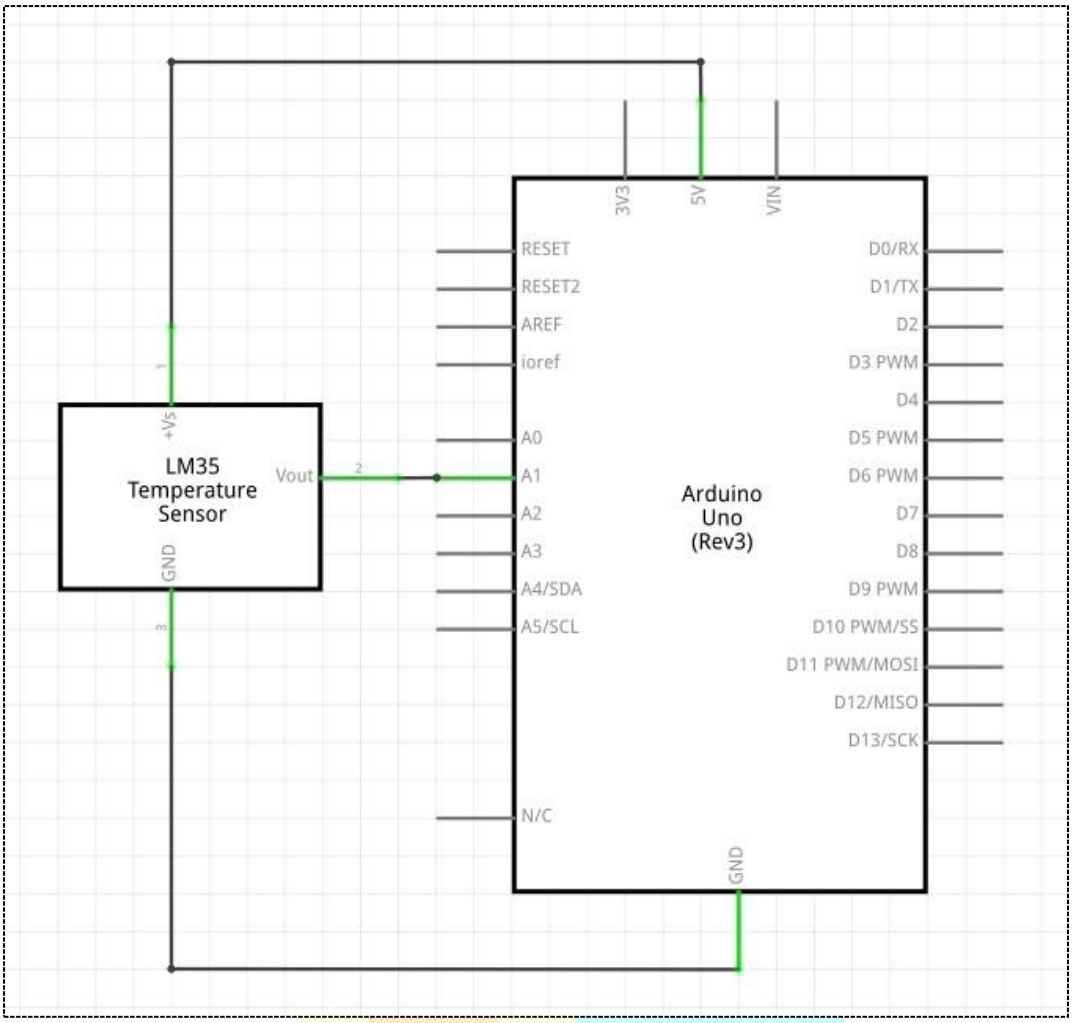


The LM35-series devices are precision integrated-circuit temperature sensors, with an output voltage linearly proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The

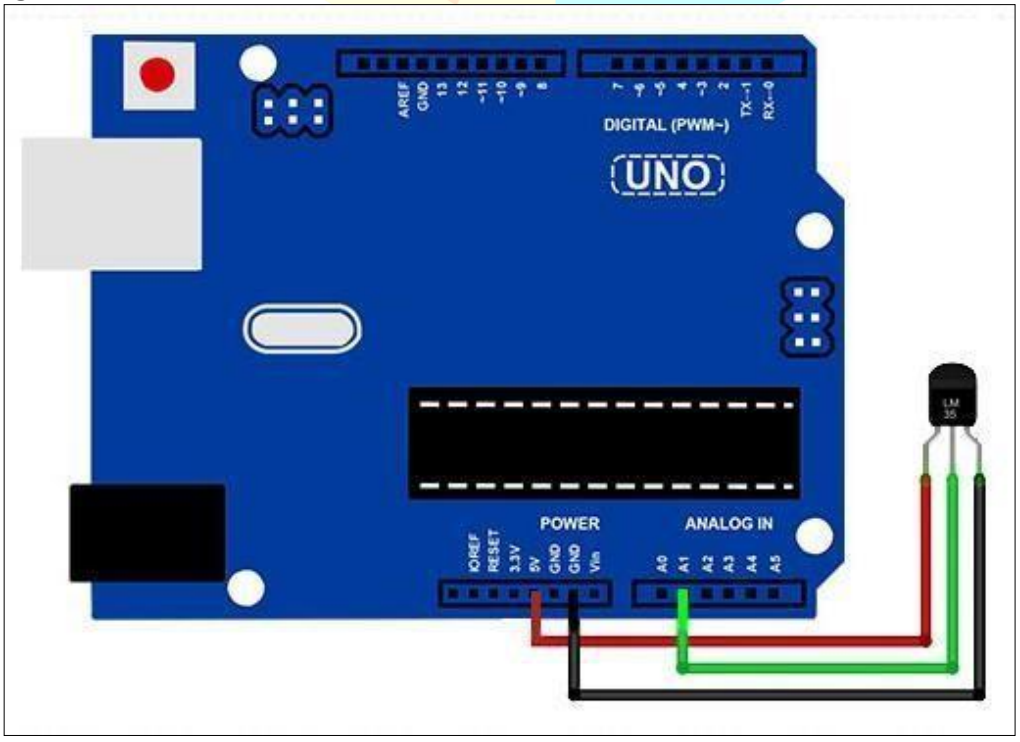
LM35 device does not require any external calibration or trimming to provide typical accuracies of ± ¼ °C at room temperature and ± ¾ °C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The temperature-sensing element is comprised of a delta-V BE architecture. The temperature-sensing element is then buffered by an amplifier and provided to the VOUT pin. The amplifier has a simple class A output stage with typical 0.5-Ω output impedance as shown in the Functional Block Diagram. Therefore the LM35 can only source current and it's sinking capability is limited to 1 µA.



Connection Schematic

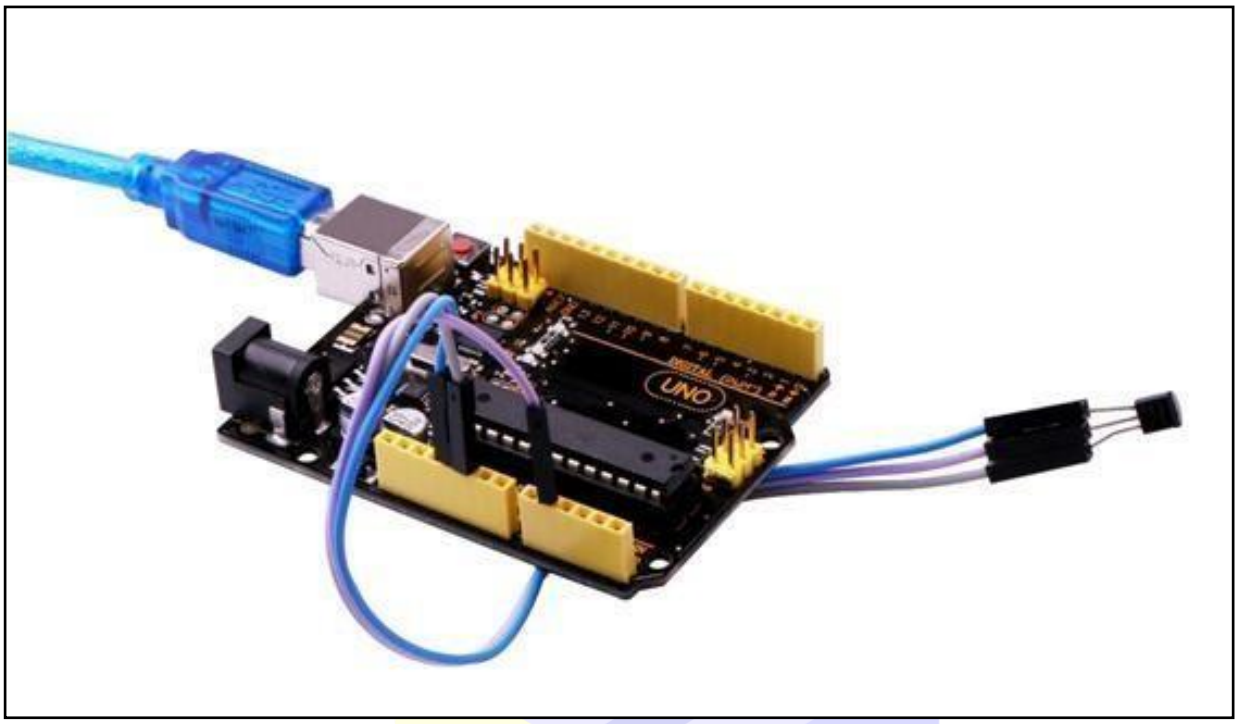


wiring diagram

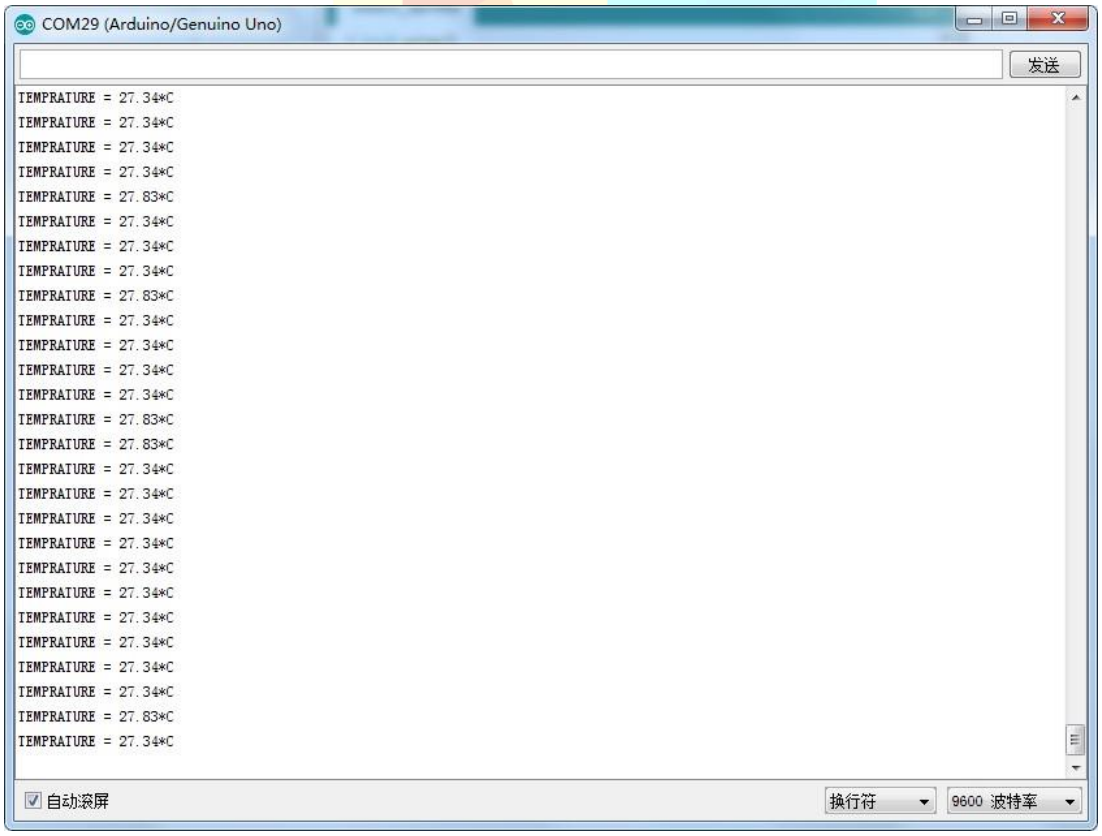


The code

See the code file.



open the monitor you can see the data as blow:



Lesson-31: Bluetooth4.0 with Arduino

Bluetooth4.0 connection with Arduino

1. Overview:

In this tutorial we explain, how to send and receive data from Arduino to an Android app.

For this, we have taken a simple example of LED. We will send command to get LED on and off using an Android App.

1.1 Default Setting of BT Module:

The default setting of Bluetooth4.0 - HM10 are as below:

Name - BT05 (This is the name of the Bluetooth device you will see in Android apps and phones bluetooth settings)

Password- 123456789 (Not required but just incase if you need to know)

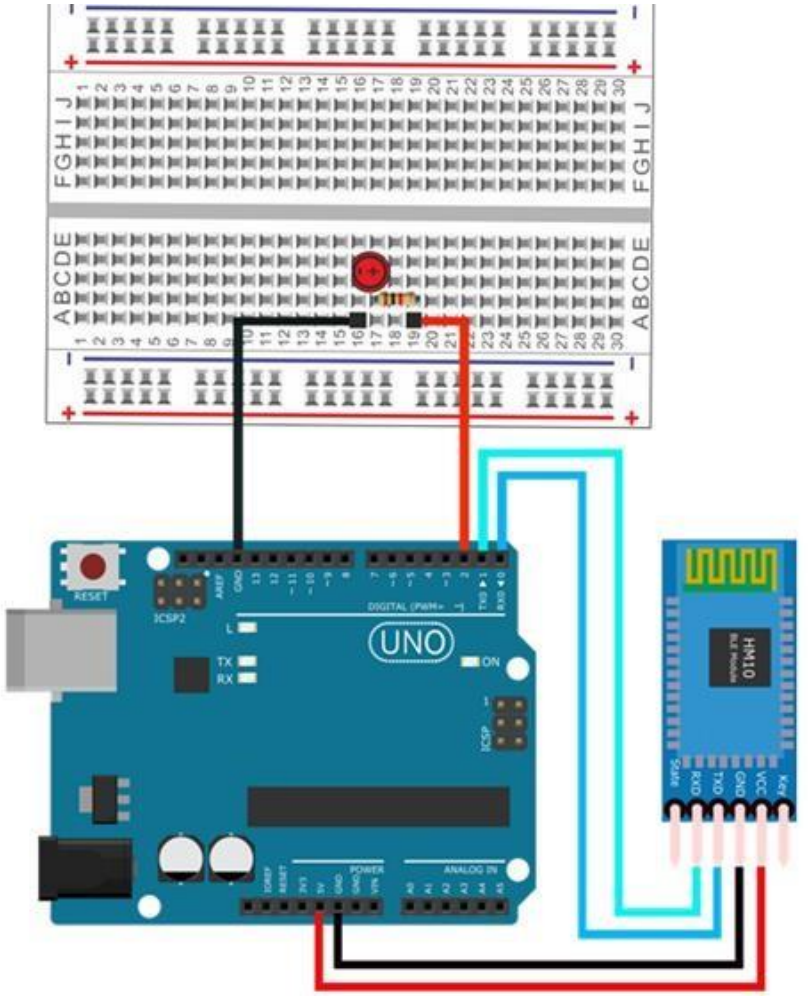
Baud Rate- 9600

All the modules have the same baud rate by default. Make sure to set Baud Rate to 9600 in your code. If you want to change the baud rate in your code, then you also need to change the baud rate of device by using AT Commands. For now you do not have to alter the code if you using our default settings.

2. Components required

1x Uno R3 and USB Cable
1x Small breadboard
6x Jumper wires
1x resistor 220ohm 1x LED

3. Building Circuit



Make sure you setup the exact circuit as shown in the figure.

4. Programming:

Once the circuit part is done, Arduino is needed to be programmed.

Very Important Step to be followed:

1. Before uploading below code, remove the RX and TX wires from Arduino's TX and RX pins.
2. Upload the code.
3. Connect the Tx and Rx pins to the arduino board once again.
4. You should see the small red led indicator in the Bluetooth 4.0 module blinking slowly. At this moment the Bluetooth is NOT paired with your android app and hence it will blink. Once its paired to the android app the red led will glow steadily and will not blink.
5. Follow the next section to pair the android app and the Bluetooth device.

```
//Hardware Required: BLE HM10 & Arduino
//www.quadstore.in

byte LED = 2;           // device to control char BT_input=' ';       // to store input character received via BT.
```

```
void setup()
{
  Serial.begin(9600);    // default baud rate of module
  pinMode(LED, OUTPUT);   // device to control
  while (!Serial)
  {

  }

}

void loop()
{
 if (Serial.available())
 {
  BT_input = Serial.read();
  if ( BT_input== 48)      //ascii code for 0 is dec 48
  {
   digitalWrite(LED, LOW);
   Serial.println(BT_input);
   Serial.println("LED is OFF");
  }
  if ( BT_input== 49)
  {
  digitalWrite(LED, HIGH);
  Serial.println(BT_input);
  Serial.println("LED is ON");  //ascii code for 1 is dec 49
  }
 }
}
```

5. Mobile app

You can download the Android control App on Playstore. Go to Playstore and search for ***"Bluetoth 4.0 BLE for arduino"****.*

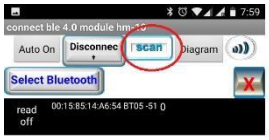
1. Download ***Bluetoth 4.0 BLE for arduino"*** App and install in your phone.

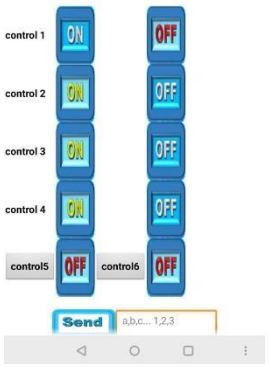
2. Turn ON Bluetooth in your phone settings. NO NEED TO PAIR.

3. Read this step carefully. This is the **most common mistake** many will try to do. DO NOT go to your phones Bluetooth setting and scan for new device and try to pair with BT05. It will give you an error message “Paring Rejected by BT05”.

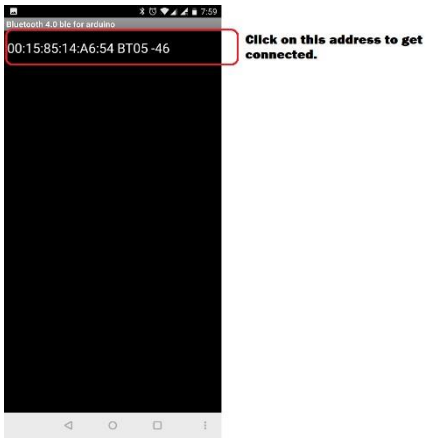
Please note Bluetooth4.0 is advanced module and hence it does not require manual paring. So kindly ignore this step as it gets automatically paired.

4. Now, open the android app “***Bluetoth 4.0 BLE for arduino”*** and click SCAN button in the top middle. It will take some time to scan the device so please be patient. Kindly allow few seconds.





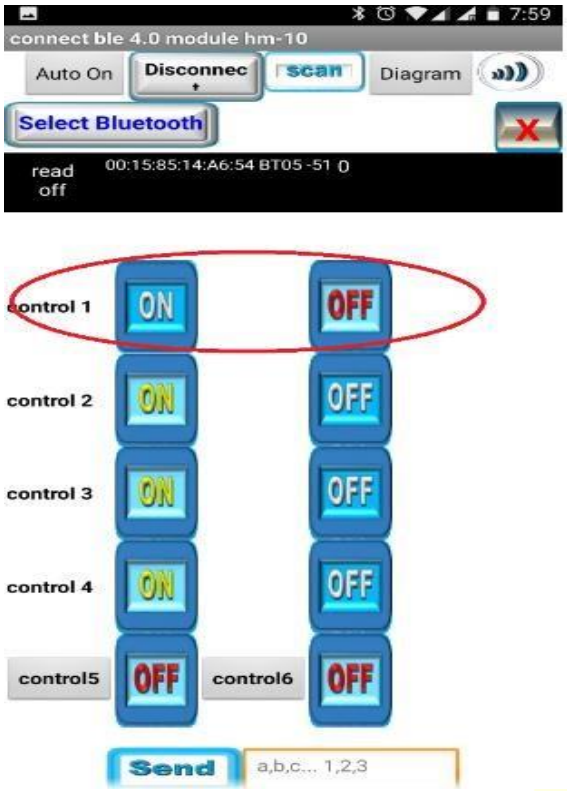
4. Now click on “Select Bluetooth” Button and it will show the address followed by BT05-46. The last two digits followed by BT05 will change according to the devices. Click on address to connect the Bluetooth module.

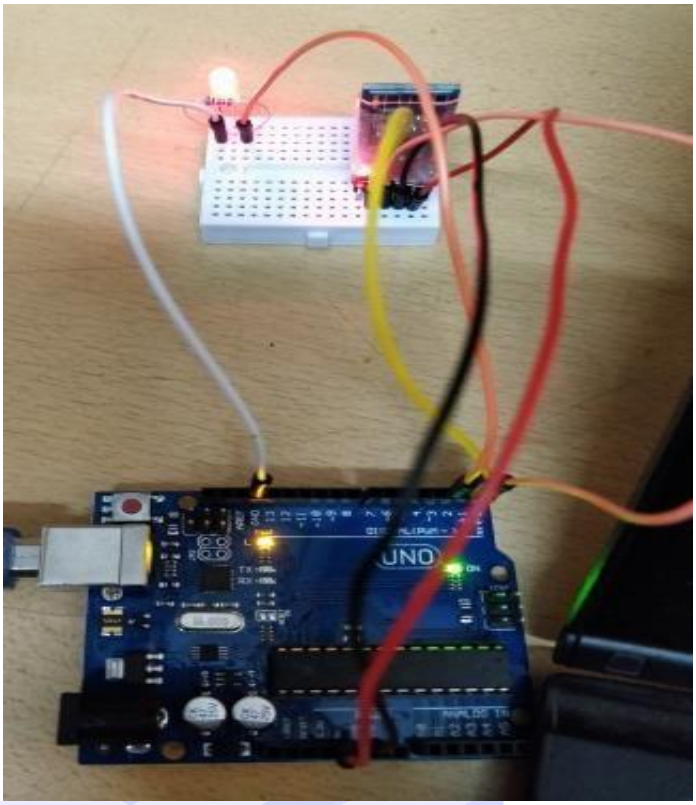


After Connection, the blinking red LED on the Bluetooth MODULE will remain ON.

6. Output:

Press on the button control 1 "ON" to turn on the connected LED. Click OFF to turn OFF the LED.





Lesson-32 : WiFi with Arduino.

Kindly visit the below link to get a detailed tutorial on usage of Wifi with arduino.

http://www.instructables.com/id/Arduino-UNO-ESP8266-WiFi-Module/