We need 5 connections to the joystick.

The connections 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

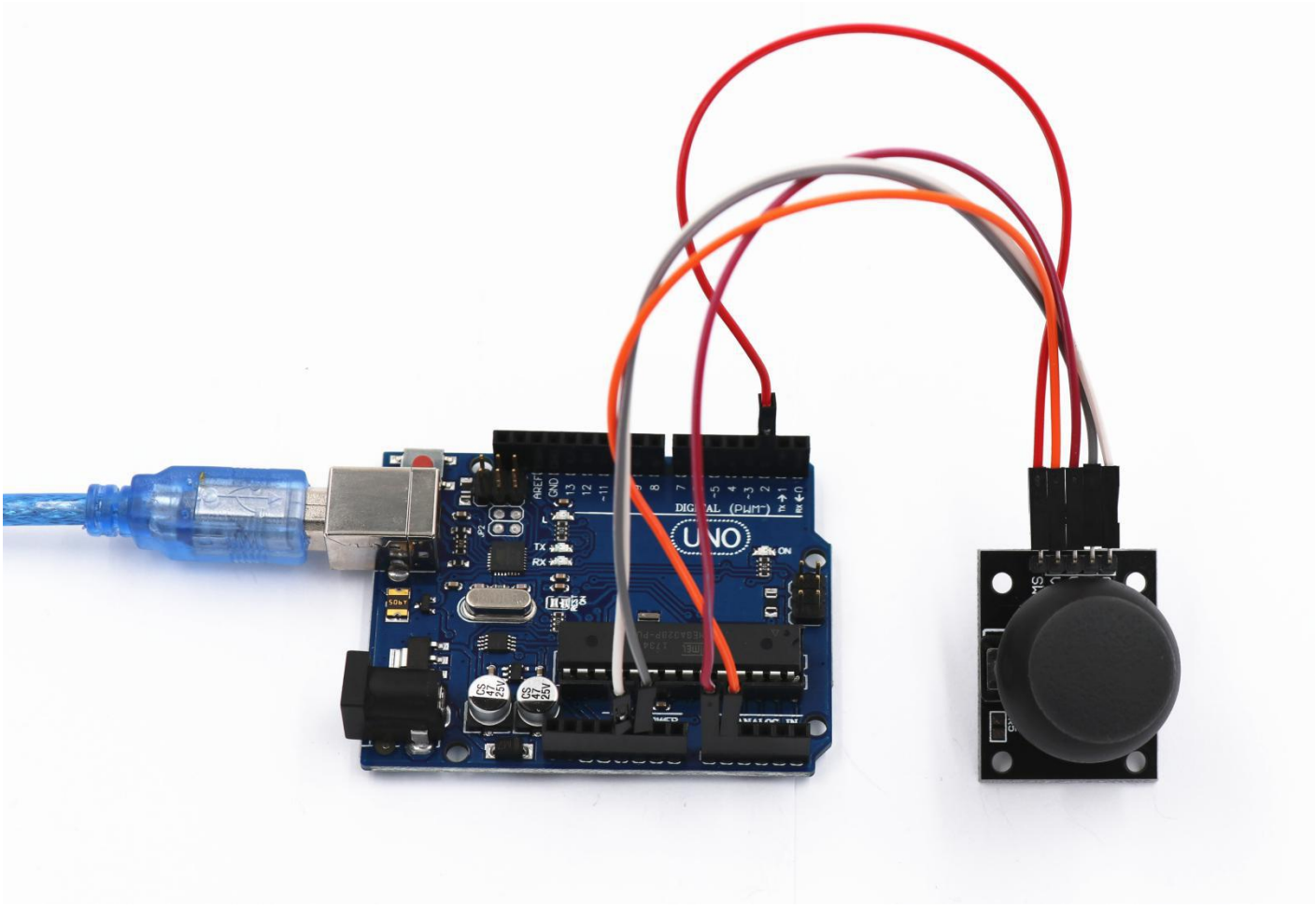
After wiring, please open the program in the code folder- Lesson 12 Analog Joystick Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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.

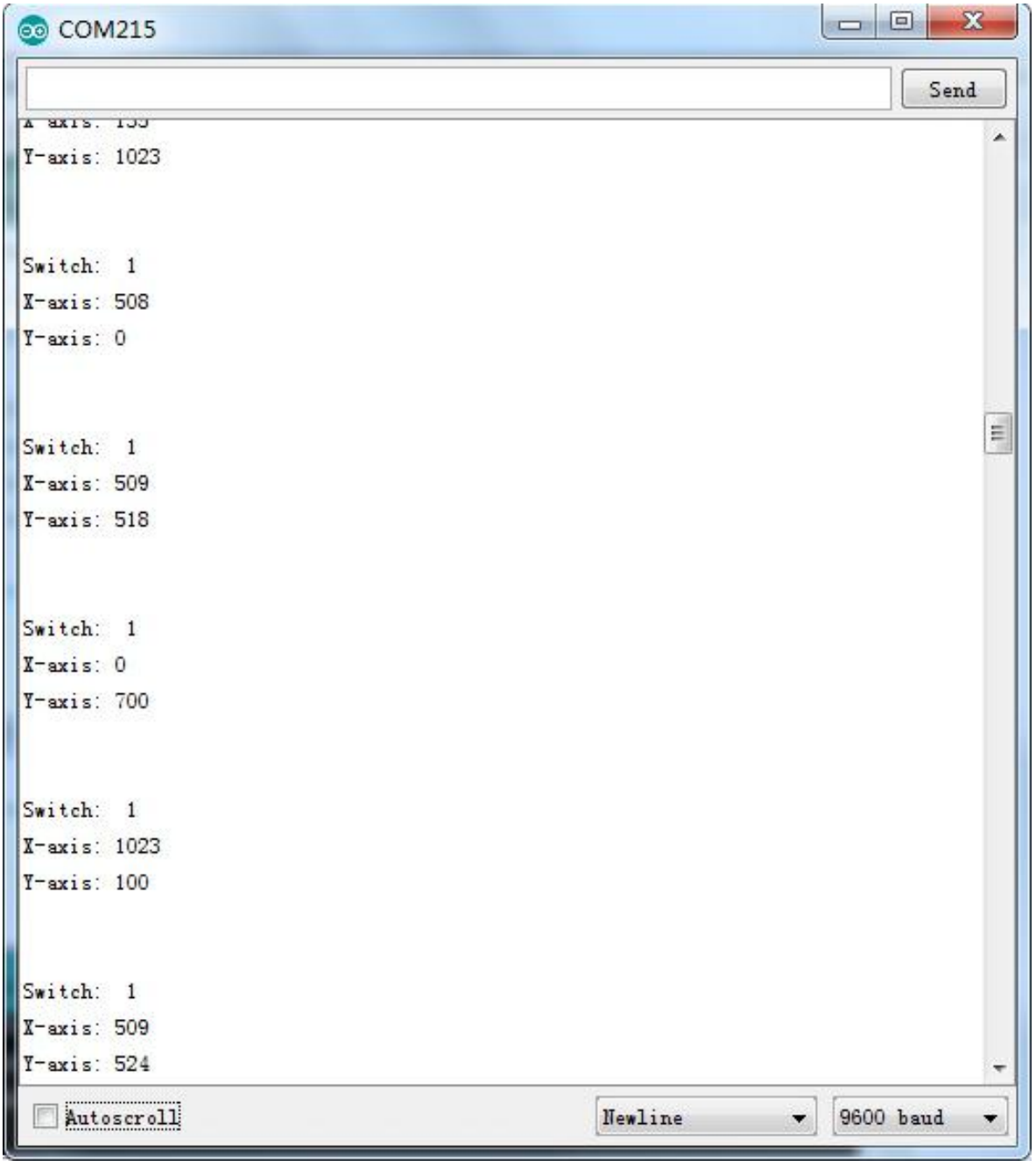
The range of values goes from 0 to 1024.

Example picture



Open the monitor then you can see the data as blow:

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



Lesson 13 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 Uno R3 Board

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

(1) x Water lever detection sensor module



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 and sensor 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

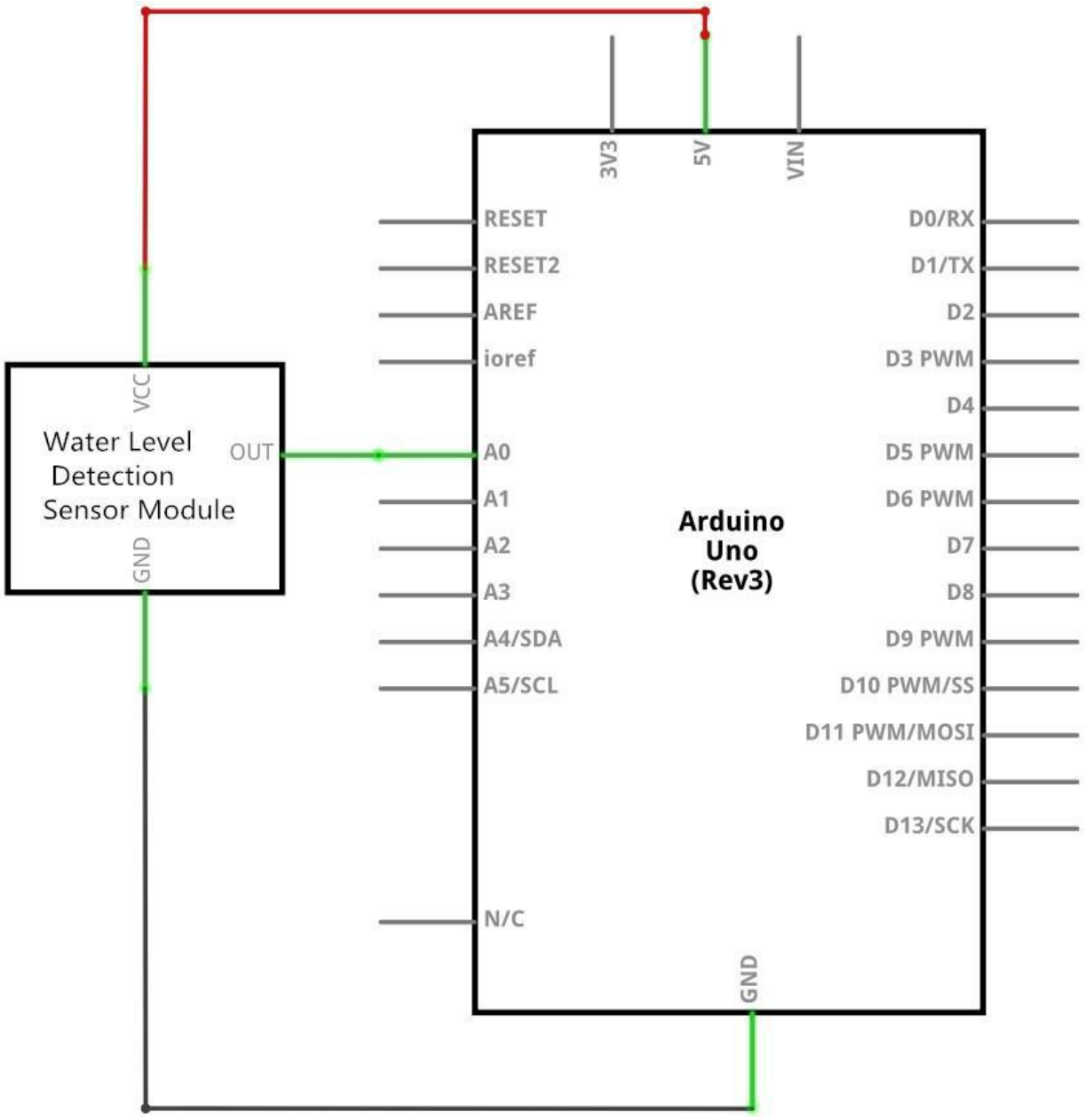
4、Width of detection: 40mm×16mm

5、Working Temperature: 10℃~30℃

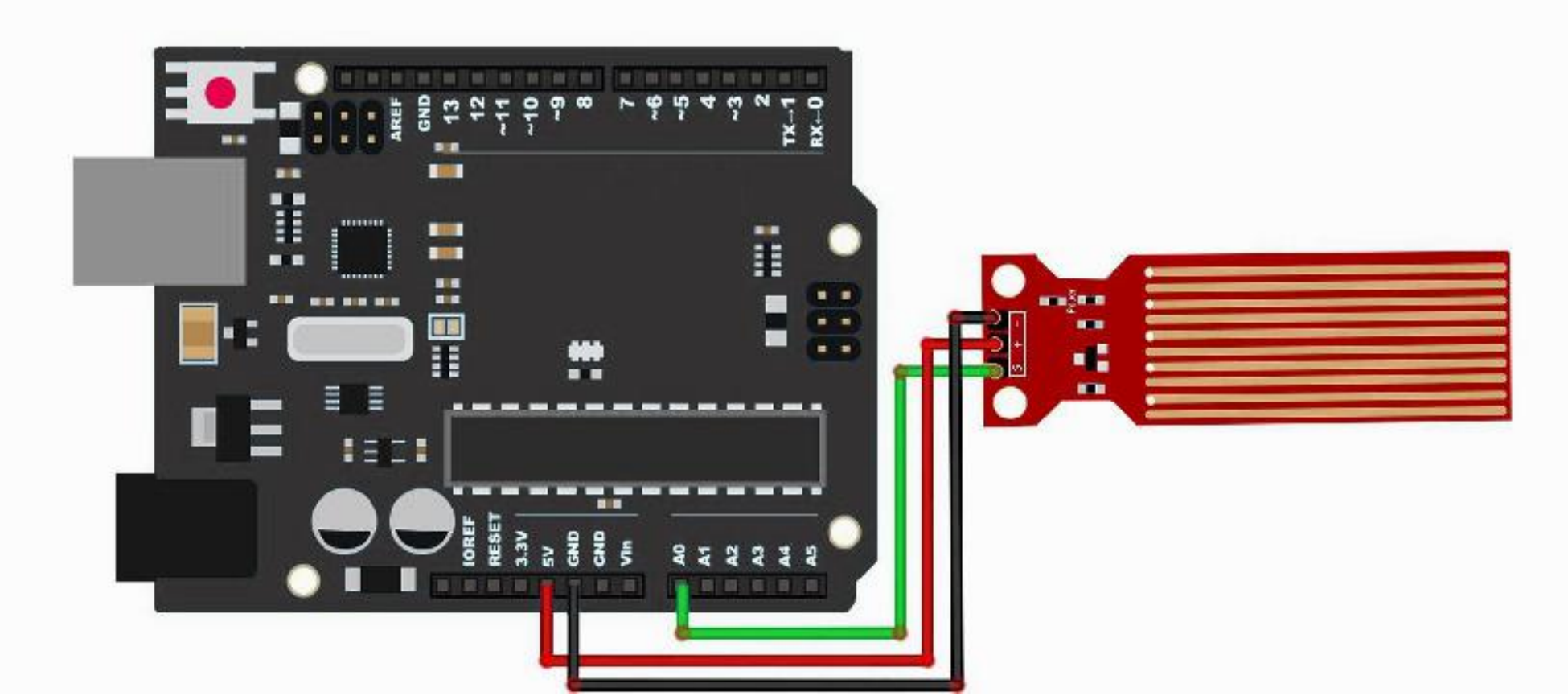
6、Output voltage signal: 0~4.2V

Connection

Schematic



Wiring diagram

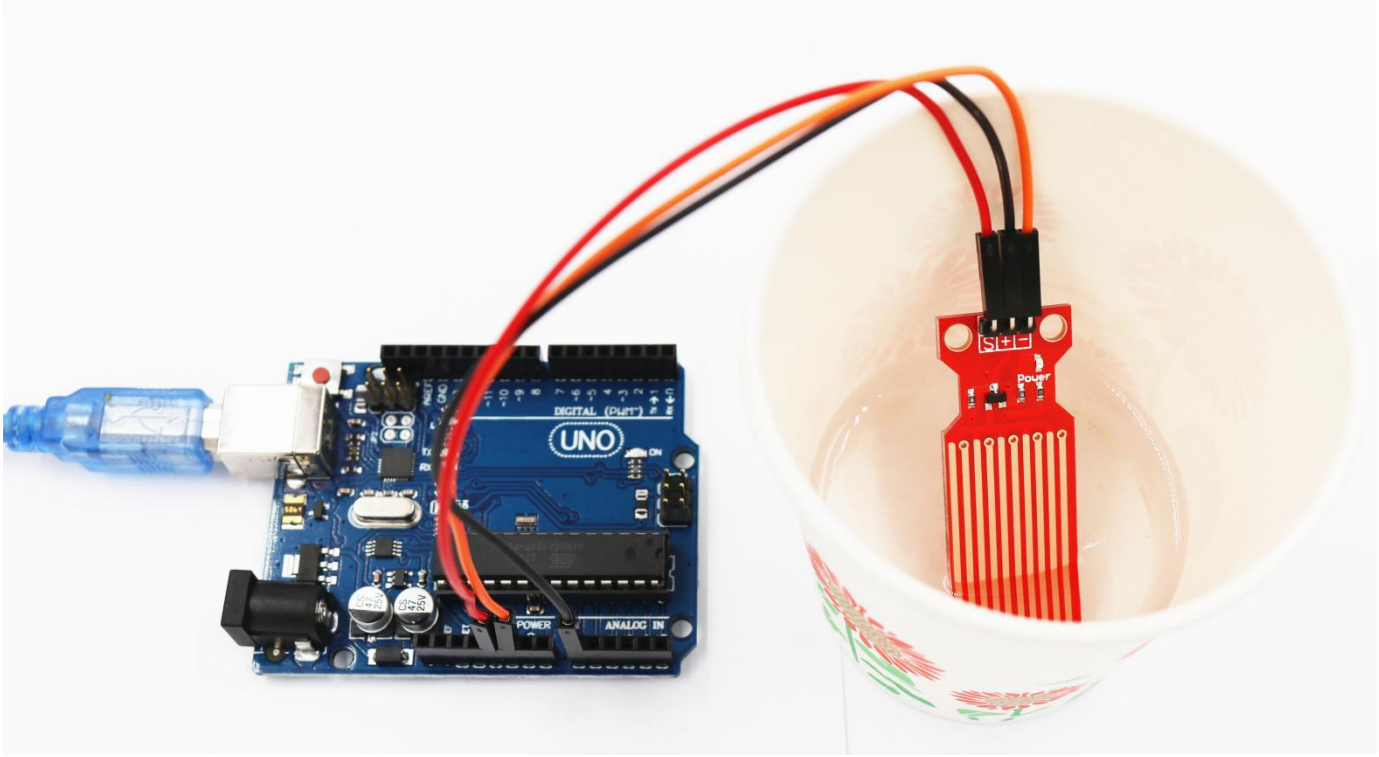


Wiring tips: Power supply (+) is connected to 5V of UNO R3 board, ground electrode (-) is connected to GND. Signal output (S) is connected to the ports (A0-A5) which have function of inputting analog signal in UNO R3 board, random one is OK, but it should define the same demo code as the routine.

Code

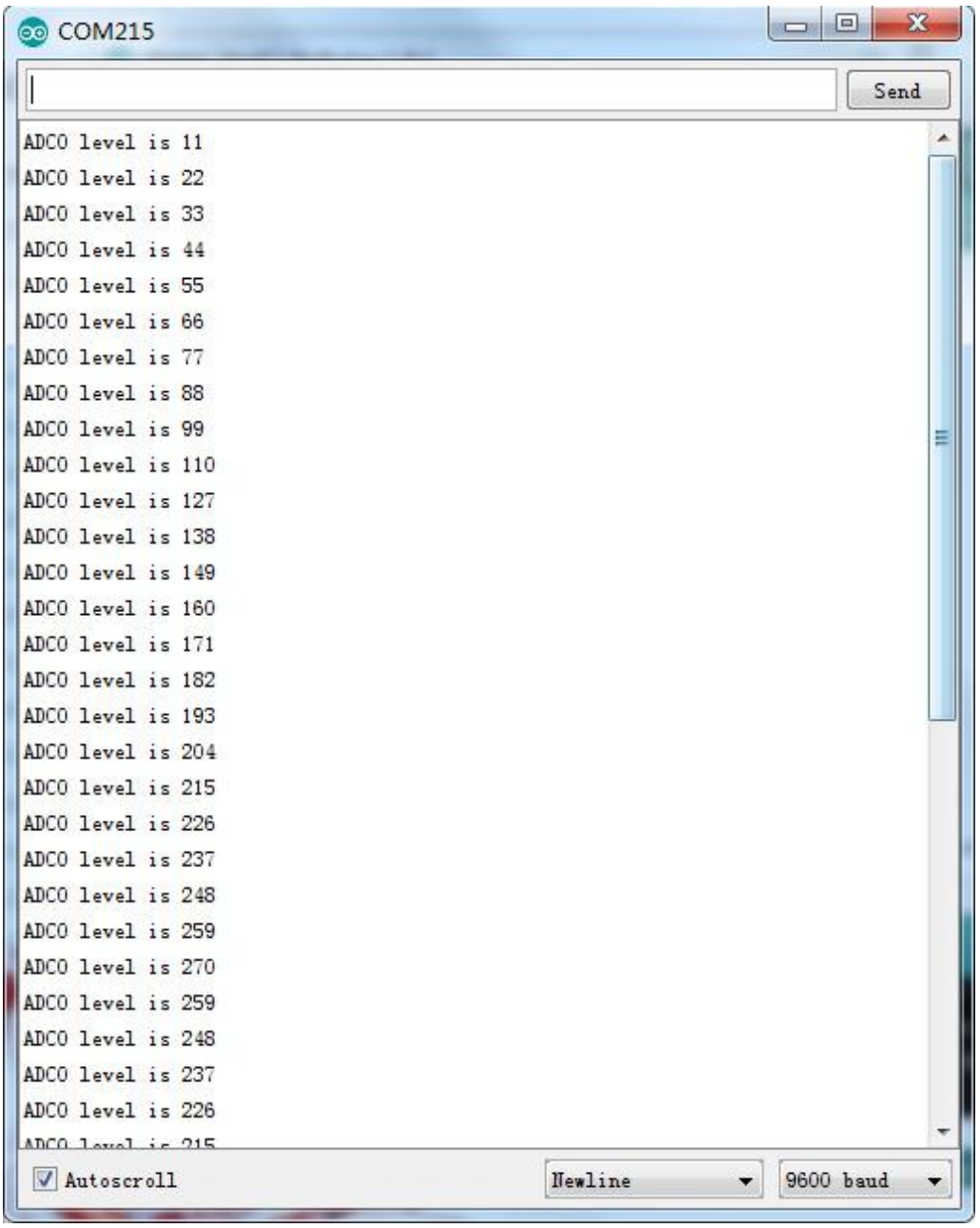
After wiring, please open the program in the code folder- Lesson 13 Water Level Detection Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Open the monitor then you can see the data as below:

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



Lesson 14 Ultrasonic Sensor Module

Overview

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

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

Component Required:



(1) x Uno R3 Board

(1) x Ultrasonic sensor module

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

Component Introduction

Ultrasonic sensor

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

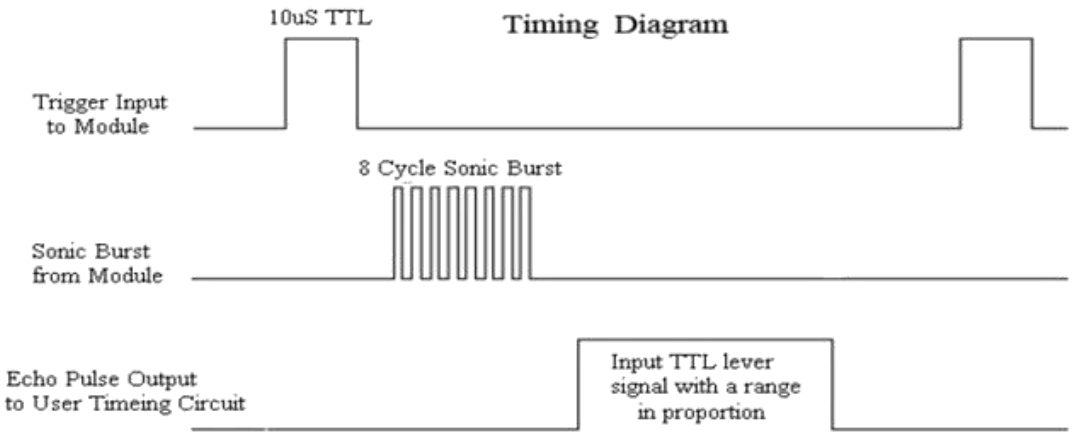
(1) Using IO trigger for at least 10us high level signal,

(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.

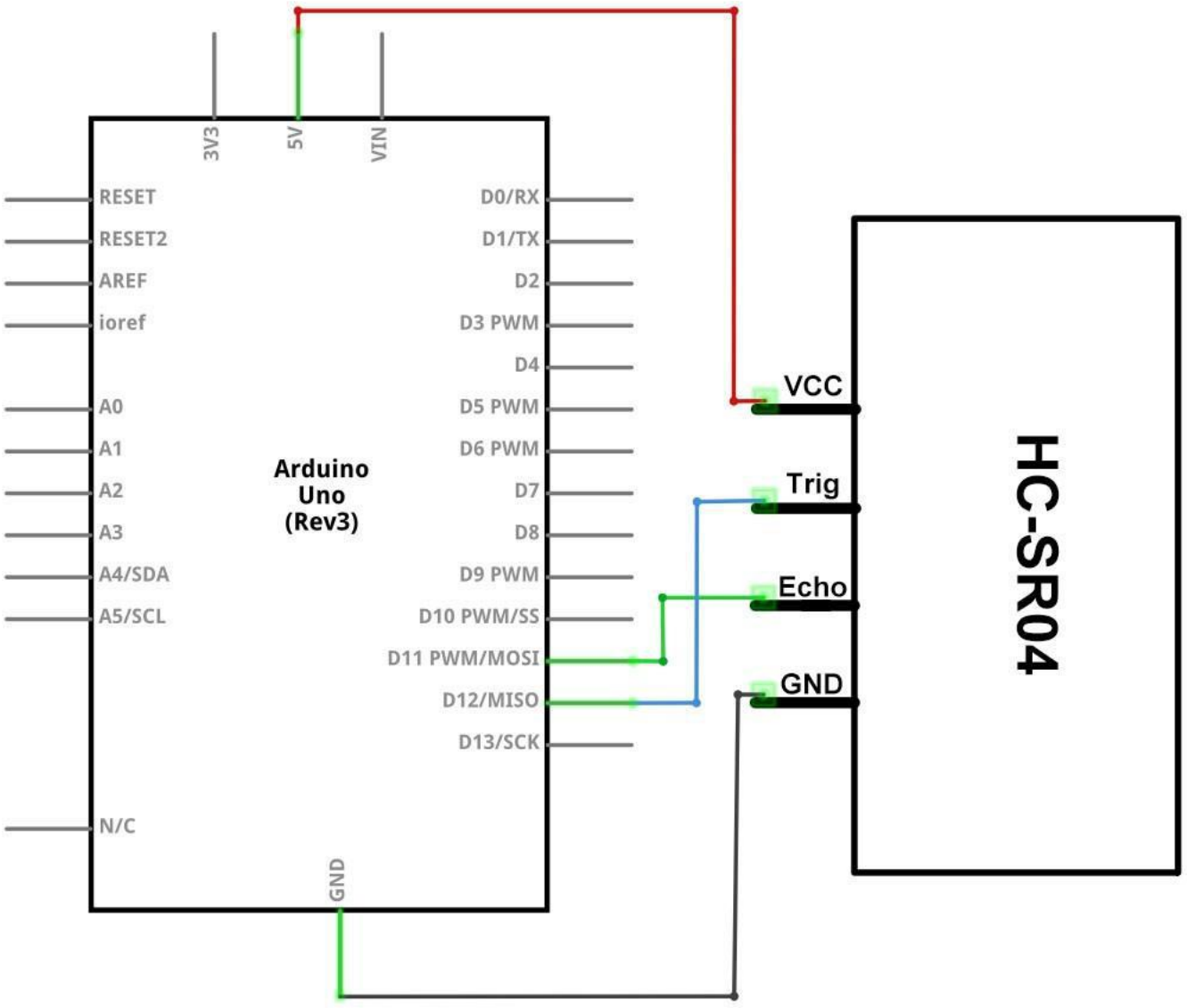
Test distance = (high level time × velocity of sound (340m/s) /2

The Timing diagram is shown below. You only need to supply a short 10us pulse 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 over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

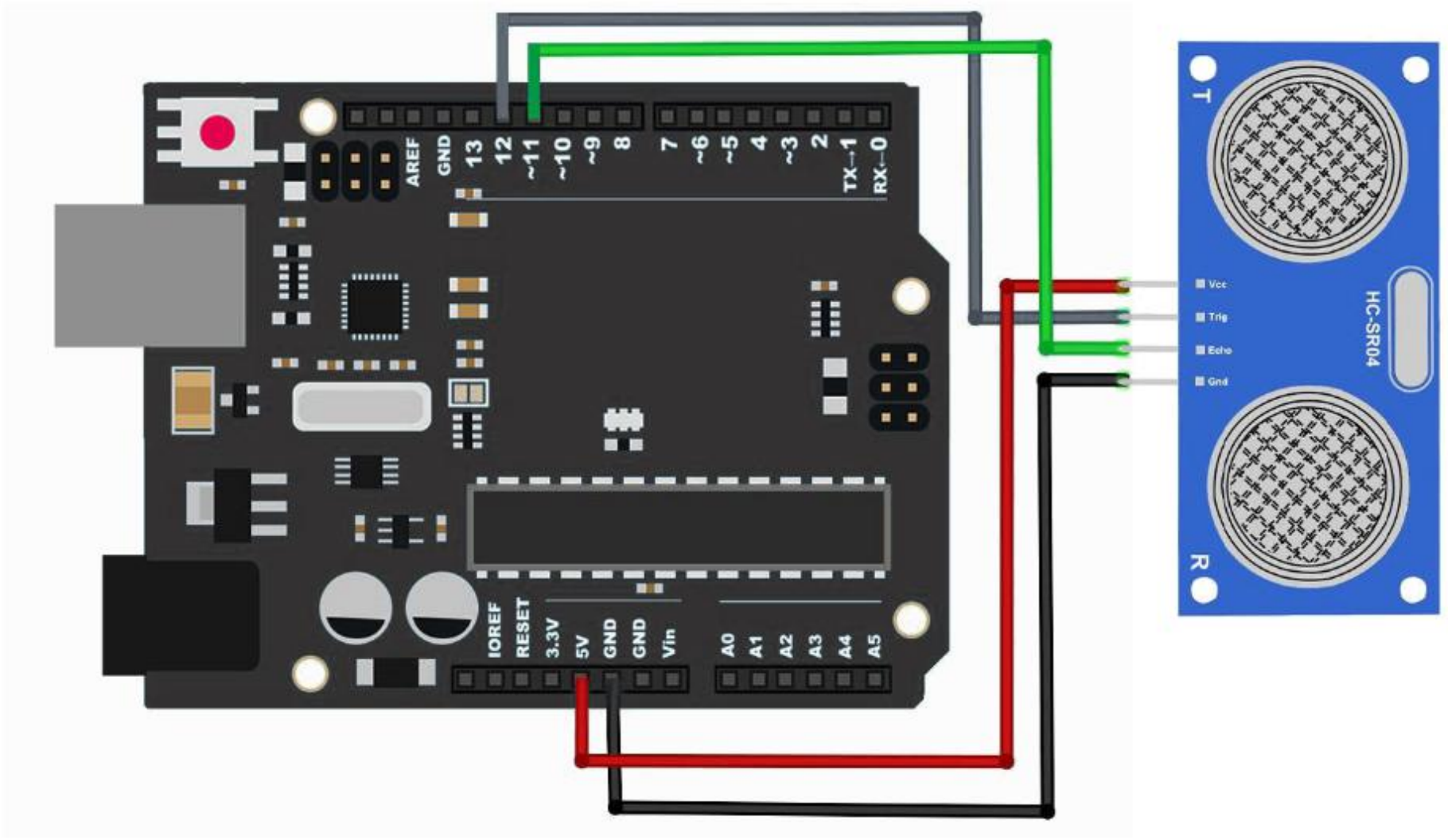


Connection

Schematic



Wiring diagram



Code

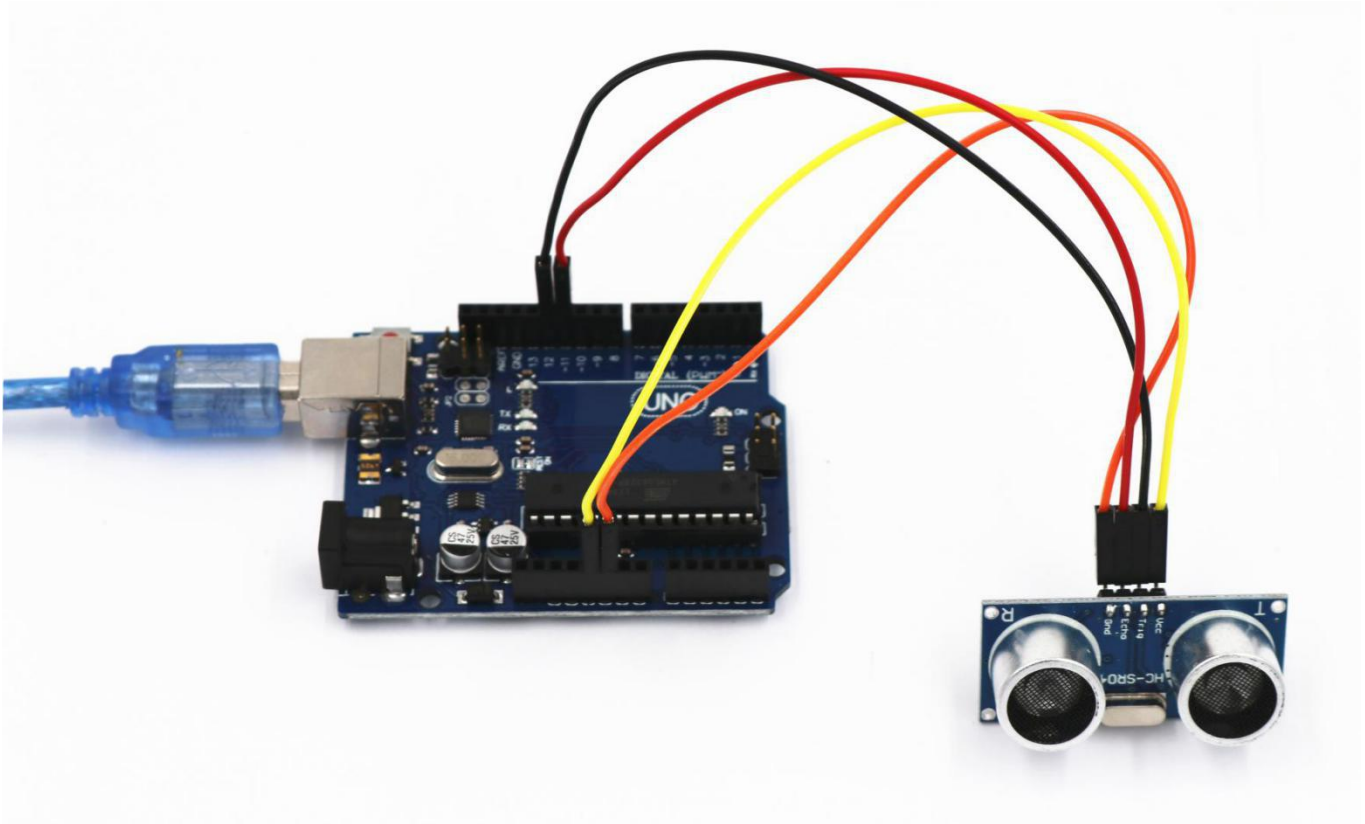
Using a Library designed for these sensors will make our code short and simple. We include the library at the beginning of our code, and then by using simple commands we can control the behavior of the sensor.

After wiring, please open the program in the code folder- Lesson 14 Ultrasonic Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < HC-SR04> library or re-install it, if necessary. Otherwise, your code won't work.

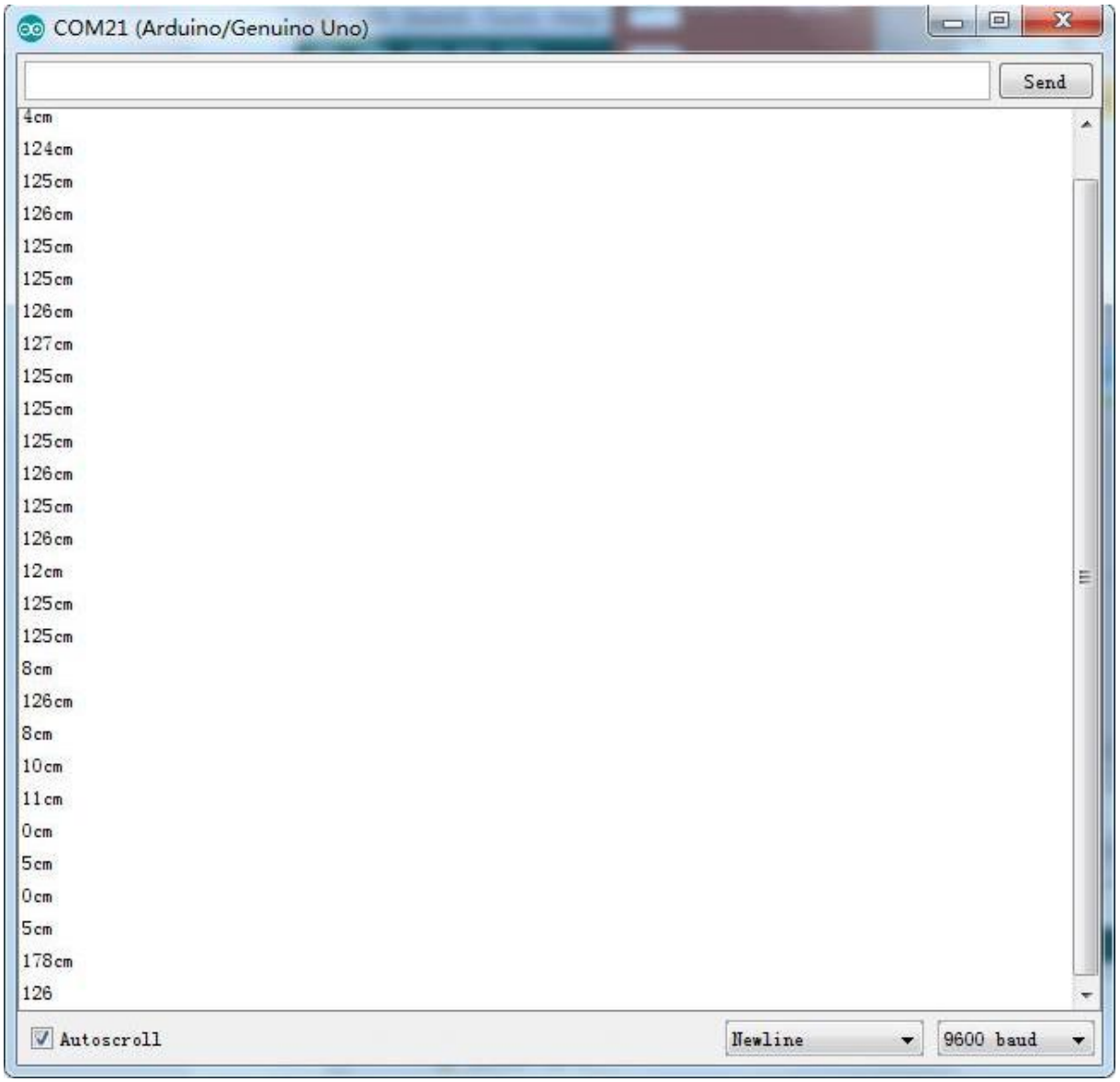
For details about loading the library file, see Lesson 1.

Example picture



Open the monitor then you can see the data as blow:

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



Lesson 15 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.

To make sure the microphone can detect your voice normally, please try to change its sensitivity by turning the blue precise potentiometer on the module. Given to its preciseness, it takes at least 10 circles for you to get some response.



1.DO:digital output
2.VCC: 3.3V-5V DC
3.GND:ground
4.AO:analog output

Component Required:

(1) xUno R3 Board

(1) x Sound sensor module

(4) x F-M wires (Female to Male DuPont 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.



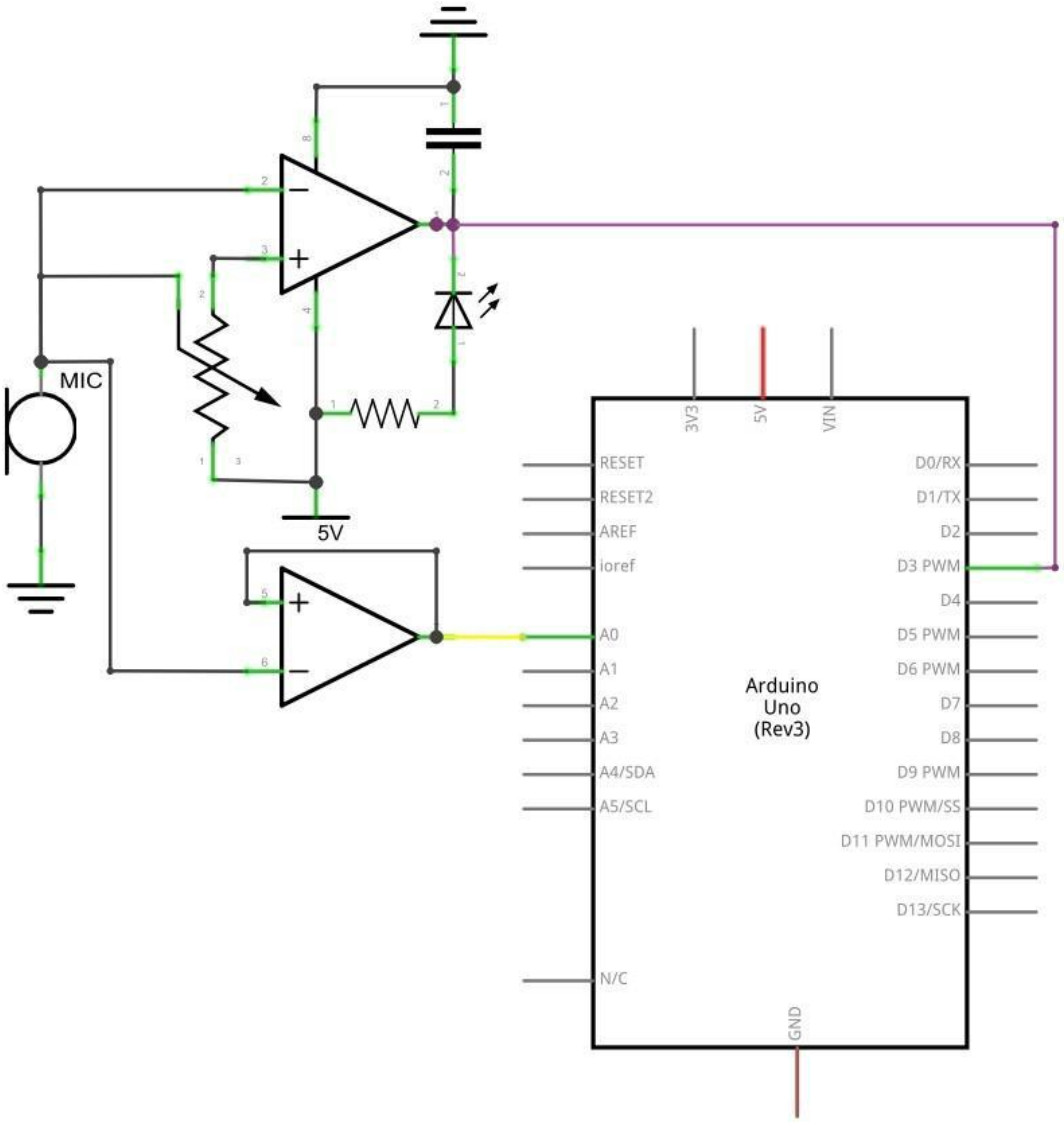
These microphones are widely used in electronic circuits to detect minor sounds or air vibrations which in turn are converted to electrical signals for further use. The two legs as shown in the image above are used to make electrical connection with the circuit.



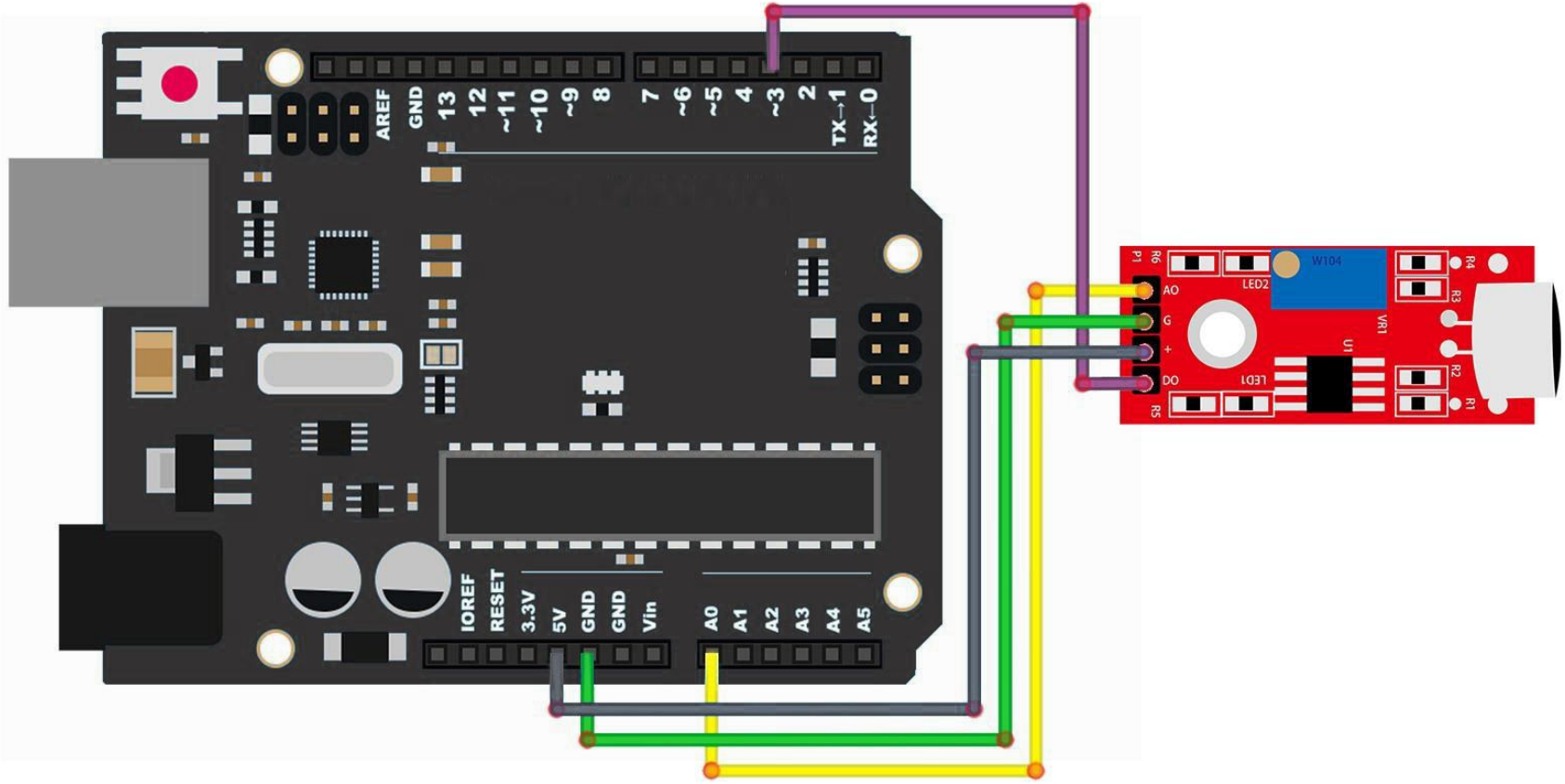
A solid conducting metal body encapsulates the various parts of the microphone. The top face is covered with a porous material with the help of glue. It acts as a filter for the dust particles. The sound signals/air vibrations passes through the porous material and falls on the diaphragm through the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

Connection

Schematic



Wiring diagram

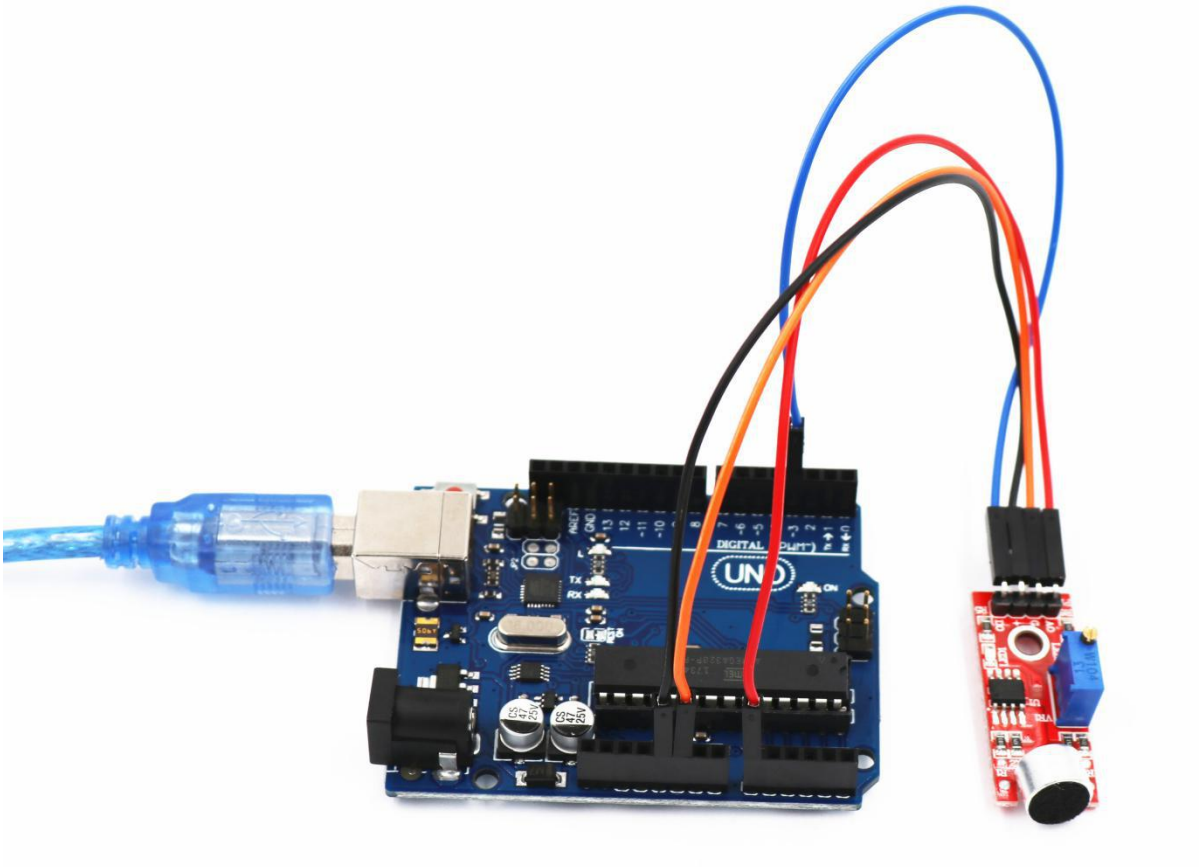


The code

After wiring, please open the program in the code folder- Lesson 15 Sound Sensor Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

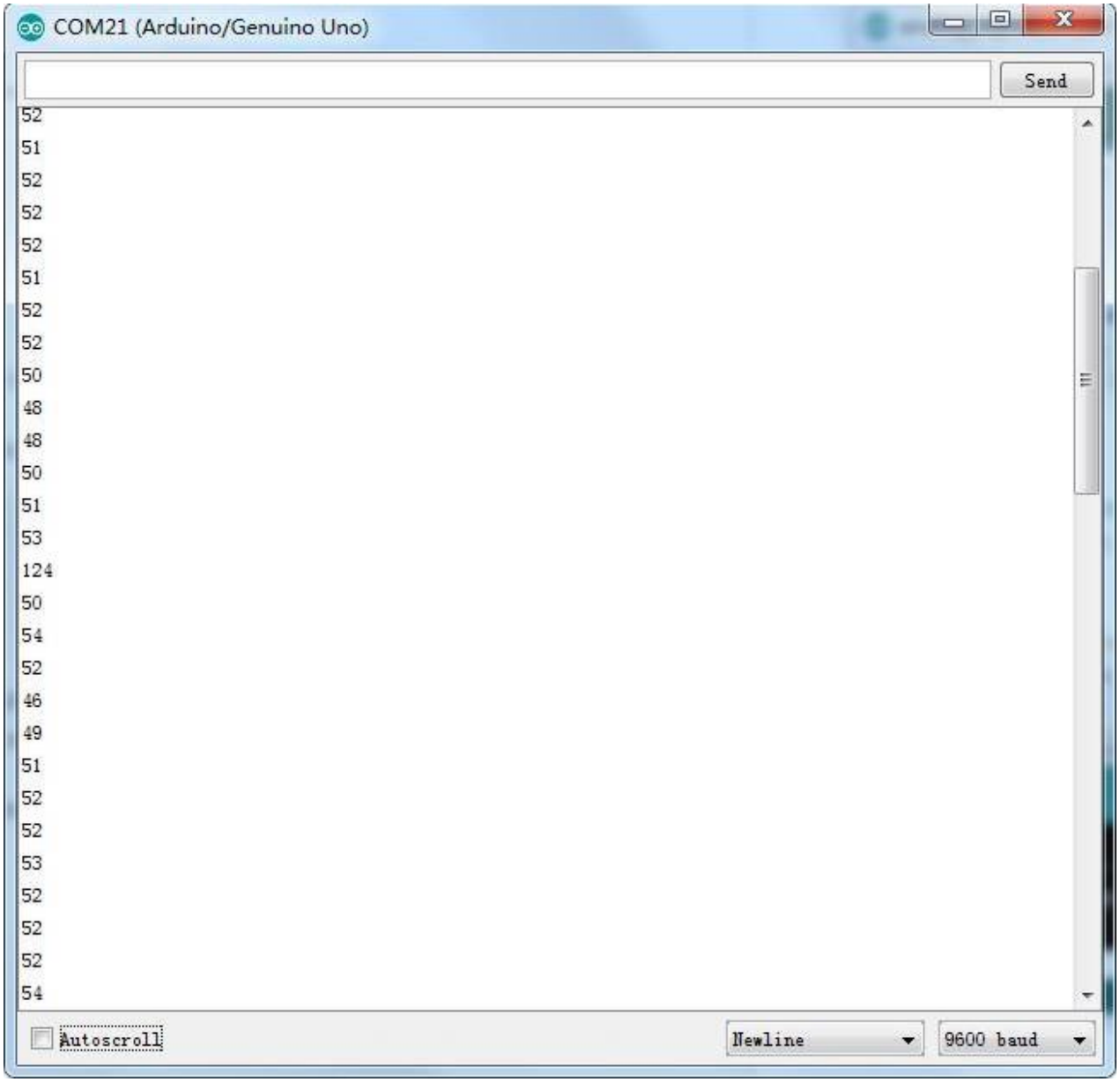
This module provides two signal output modes, for which we wrote two codes: digital_signal_output and analog_signal_output. The code of digital_signal_output works when the voice reaches a certain value, it will trigger a digital signal and the dig #11 pin on Arduino will output a high level and the indicator L will be lit up at the same time. This triggering value may be changed according to the sensitivity adjustment method mentioned above. The code of analog_signal_output will read the analog value of the module and directly display it on the serial monitor, likewise, this value can also be changed according to the sensitivity adjustment method mentioned above.

Example picture



Open the monitor then you can see the data as below:

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



Lesson 16 Membrane Switch 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 shows up at the Serial Monitor on your computer. Whenever you press a key, it shows up on the Serial Monitor. 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 make 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 Uno R3 Board

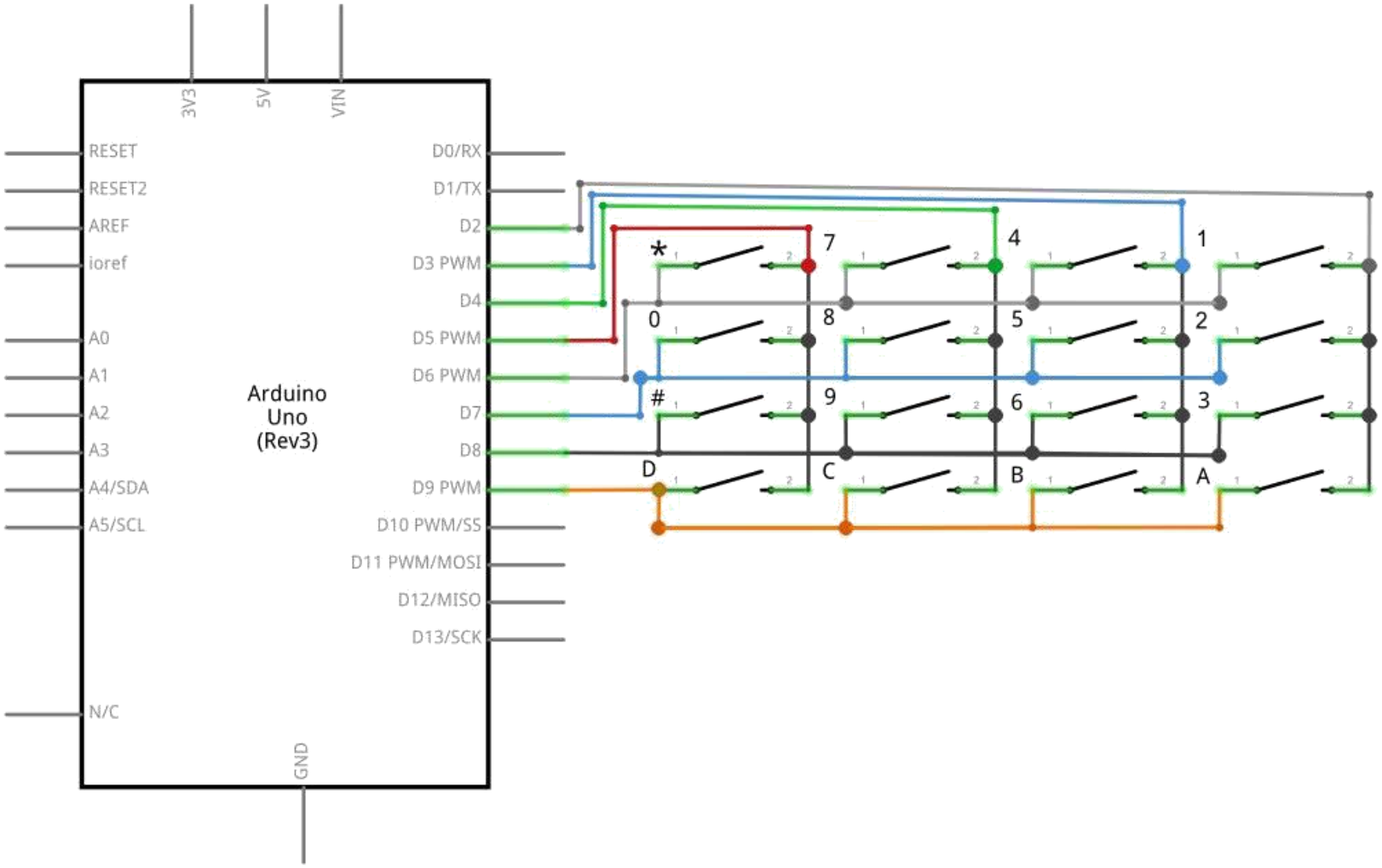
(1) x Membrane switch module

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

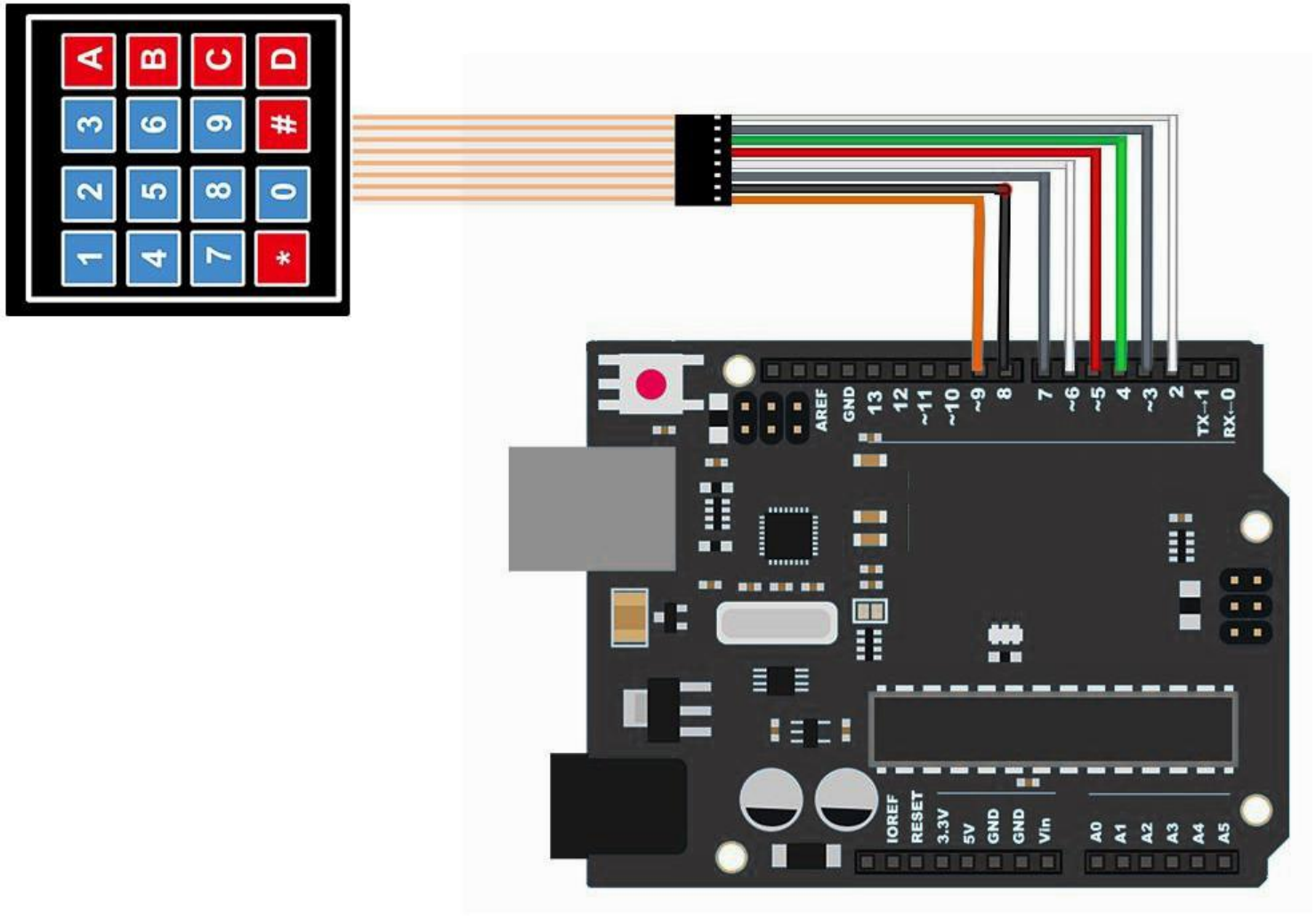


Connection

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

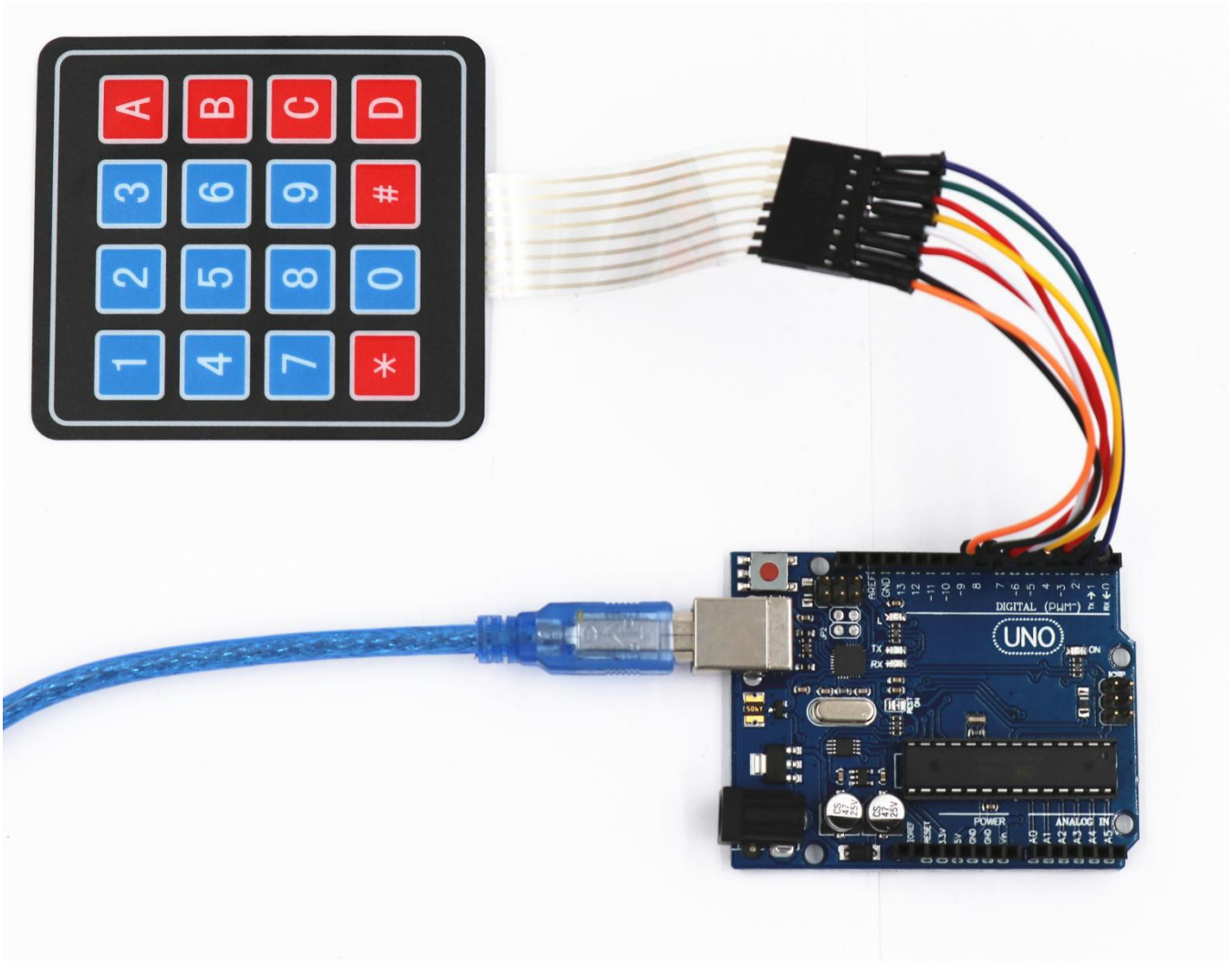
Code

After wiring, please open the program in the code folder- Lesson 16 Membrane Switch Module and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

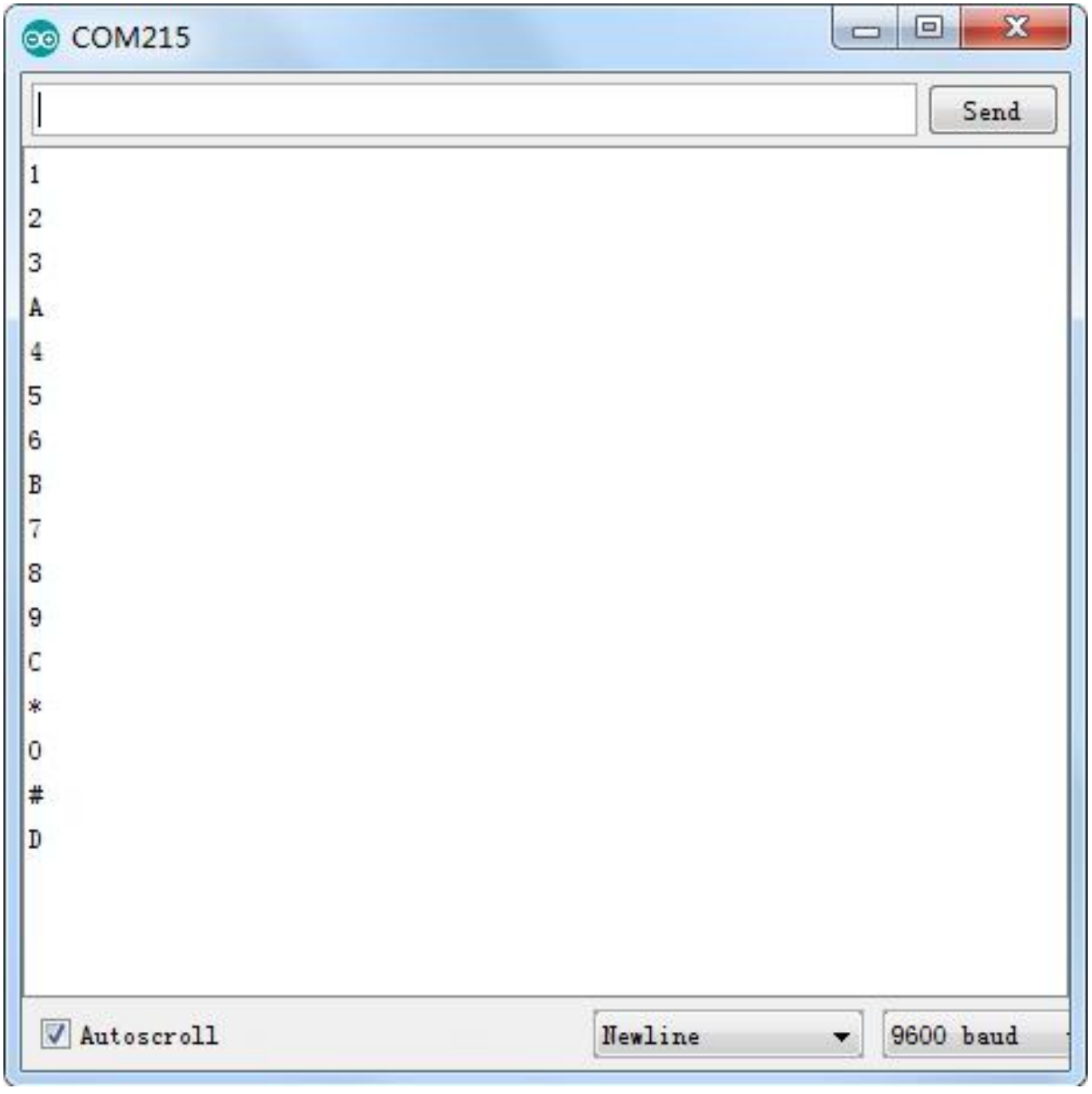
For details about loading the library file, see Lesson 1.

Example picture



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.

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



Lesson 17 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) xUno R3 Board

(1) x 830 tie-points breadboard

(8) x leds

(8) x 220 ohm resistors

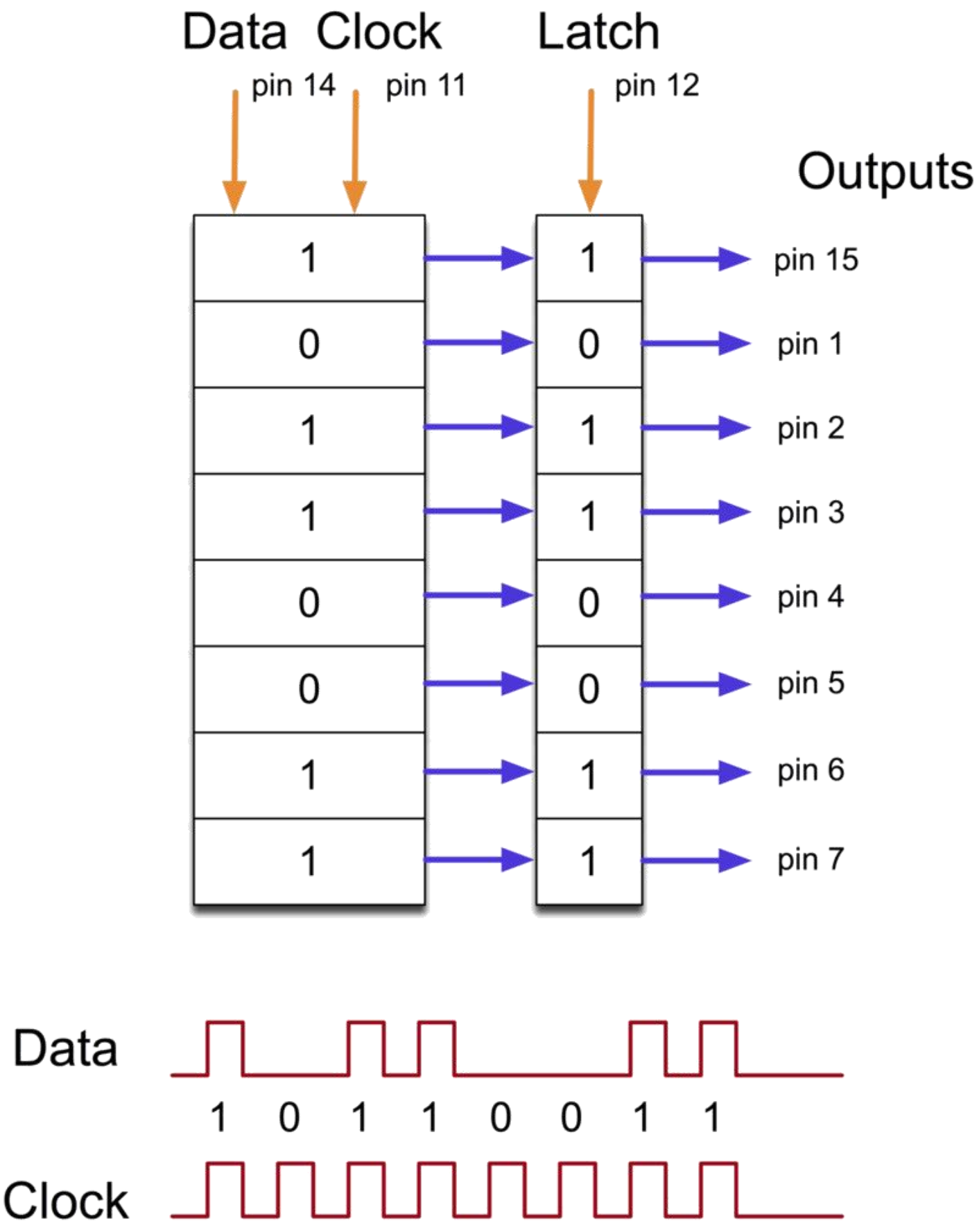
(1) x 74hc595 IC

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

Component Introduction

74HC595 Shift Register:

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

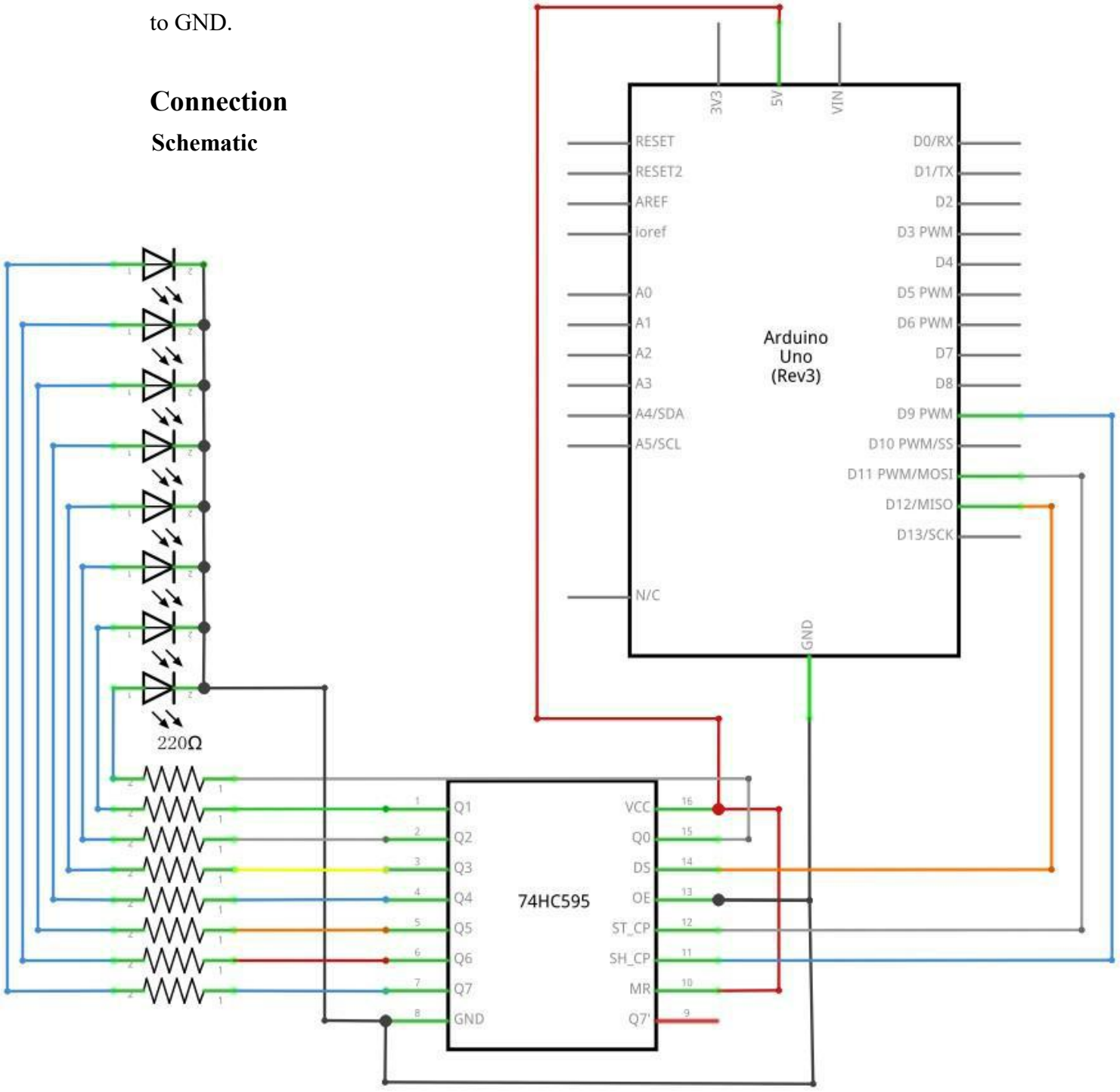

Data
Clock
Latch
pin 14
pin 11
pin 12
Outputs
1
0
1
1
0
0
0
1
1
0
1
1
1
1
1
1
pin 15
pin 1
pin 2
pin 3
pin 4
pin 5
pin 6
pin 7
Data
1 0 1 1 0 0 1 1
Clock

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

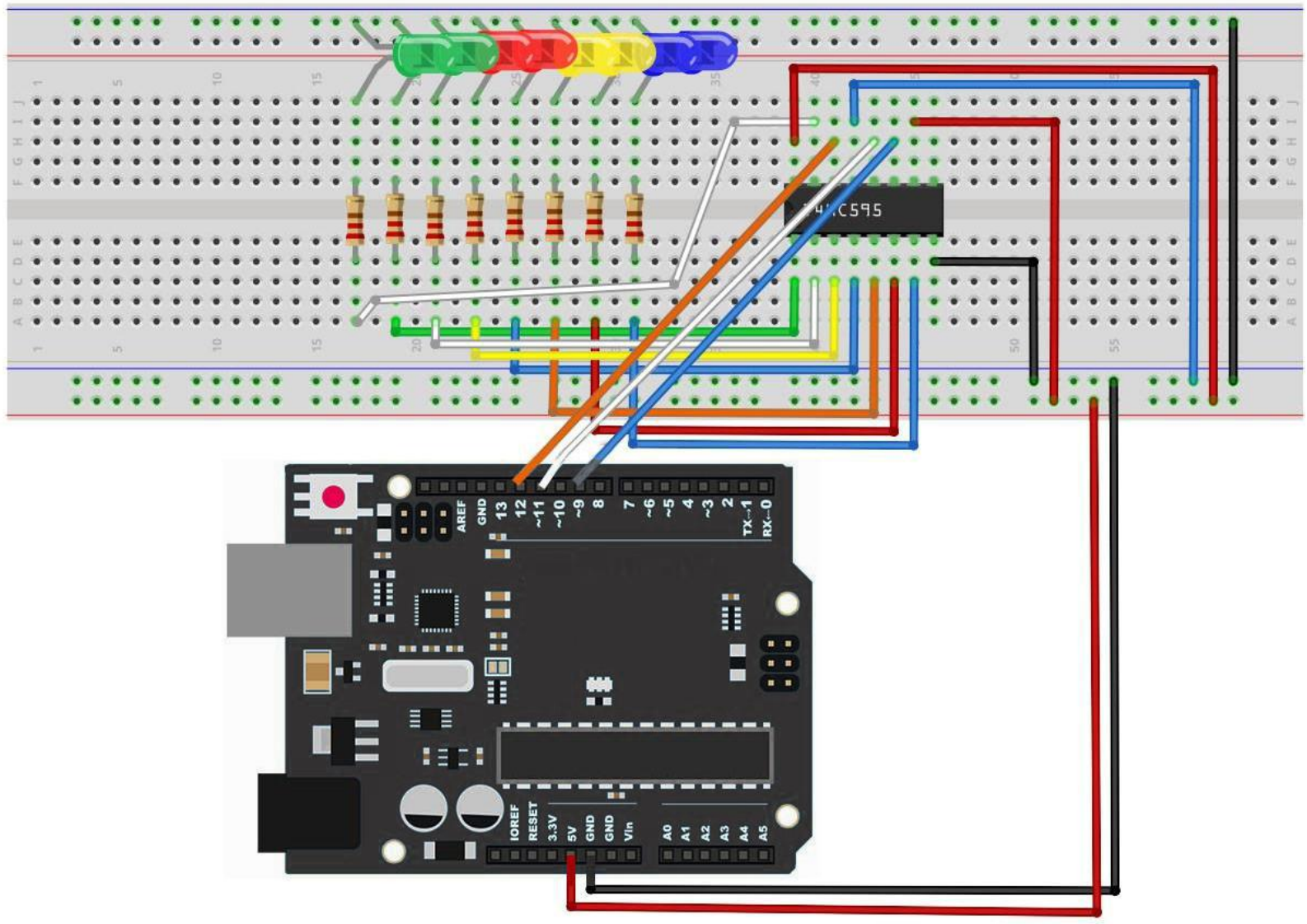
The chip also has an output enable (OE) pin, which is used to enable or disable the outputs all at once. You could attach this to a PWM-capable UNO pin and use 'analogWrite' to control the brightness of the LEDs. This pin is active low, so we tie it to GND.

Connection

Schematic



Wiring diagram



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

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

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

After wiring, please open the program in the code folder- Lesson 17 Eight LED with

74HC595 and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

which of our eight LEDs are on or off.

```
byte leds = 0;
```

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

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

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

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

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

The function 'updateShiftRegister', first of all sets the latchPin to low, then calls the 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.

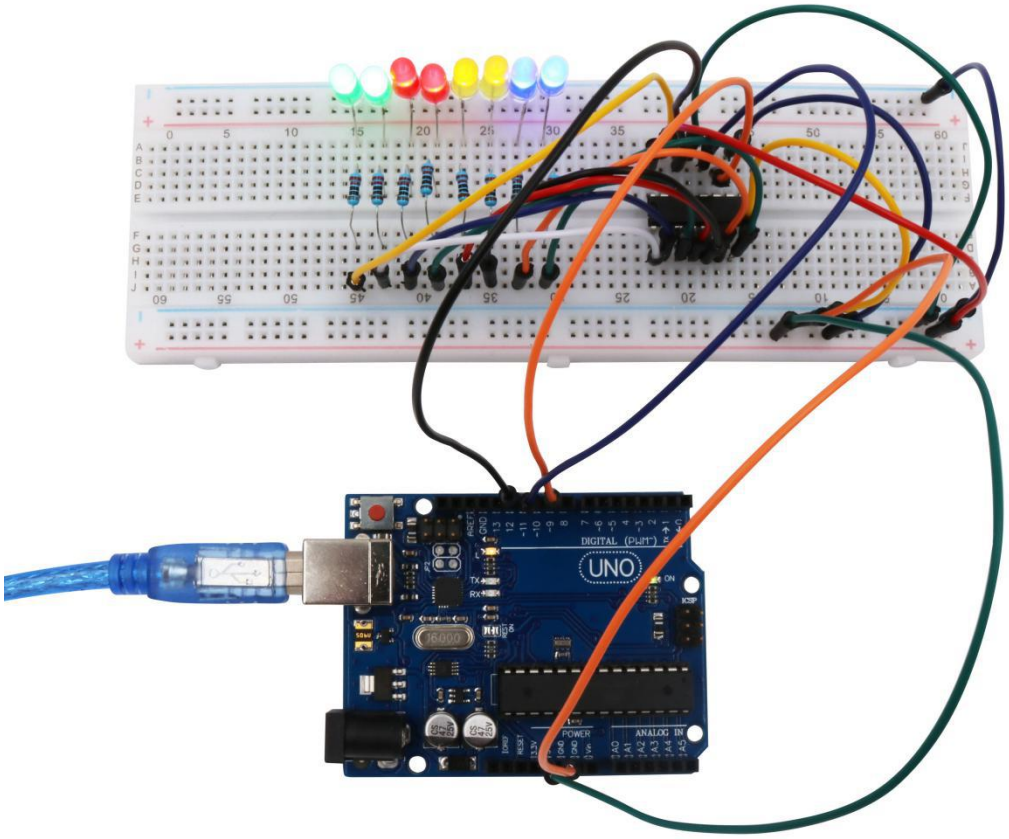
The third parameter specifies which end of the data you want to start at. We are going to start with the right most bit, which is referred to as the 'Least Significant Bit' (LSB).

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.

Example picture



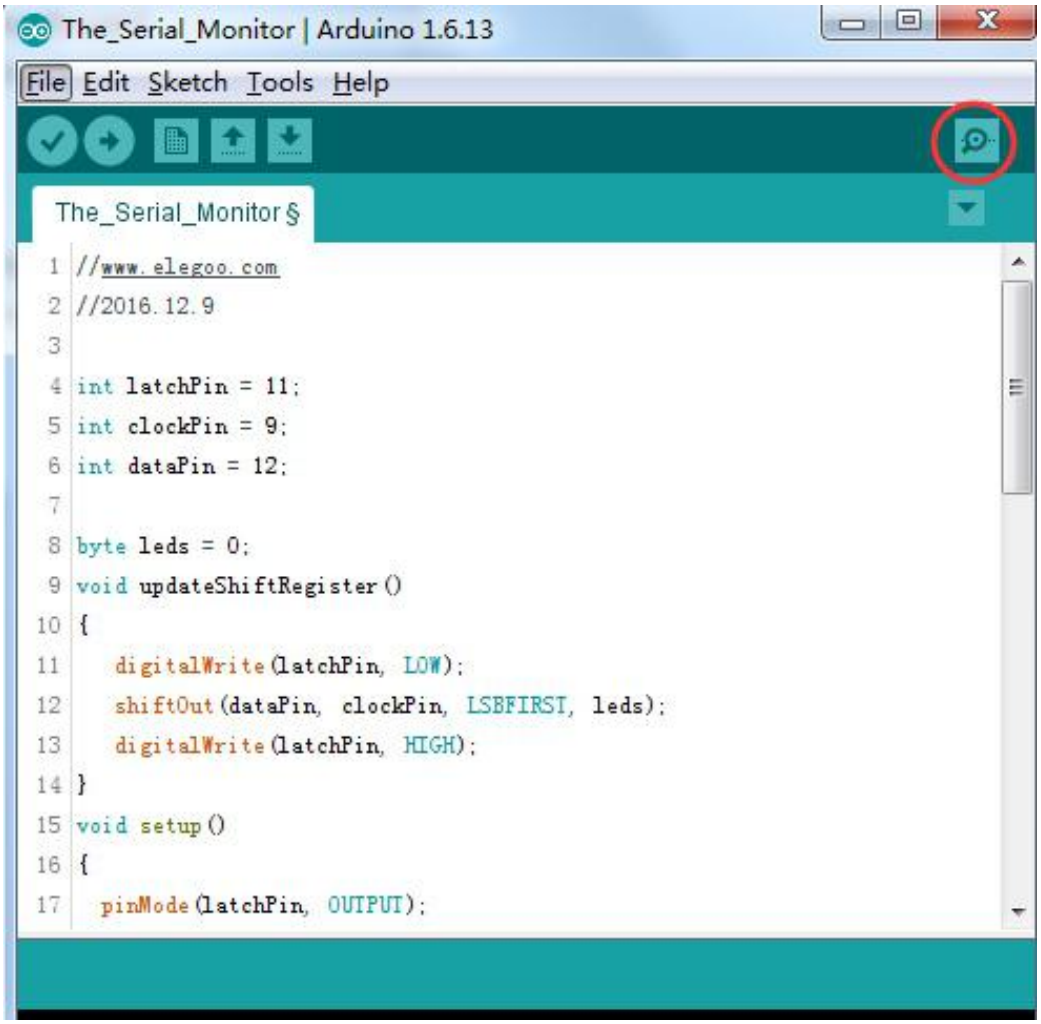
Lesson 18 The Serial Monitor

Overview

In this lesson, you will build on Lesson 17, 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 17. So, if you have not already done so, follow Lesson 17 now.

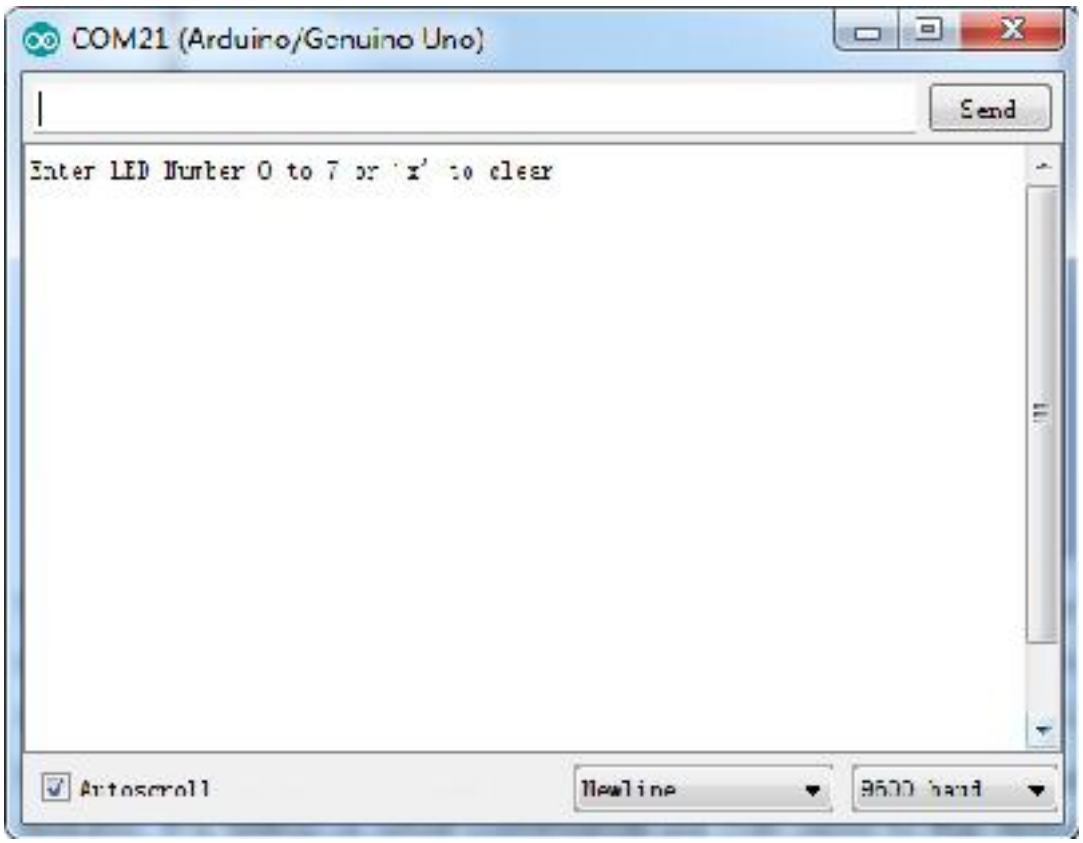
Steps taken

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.

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

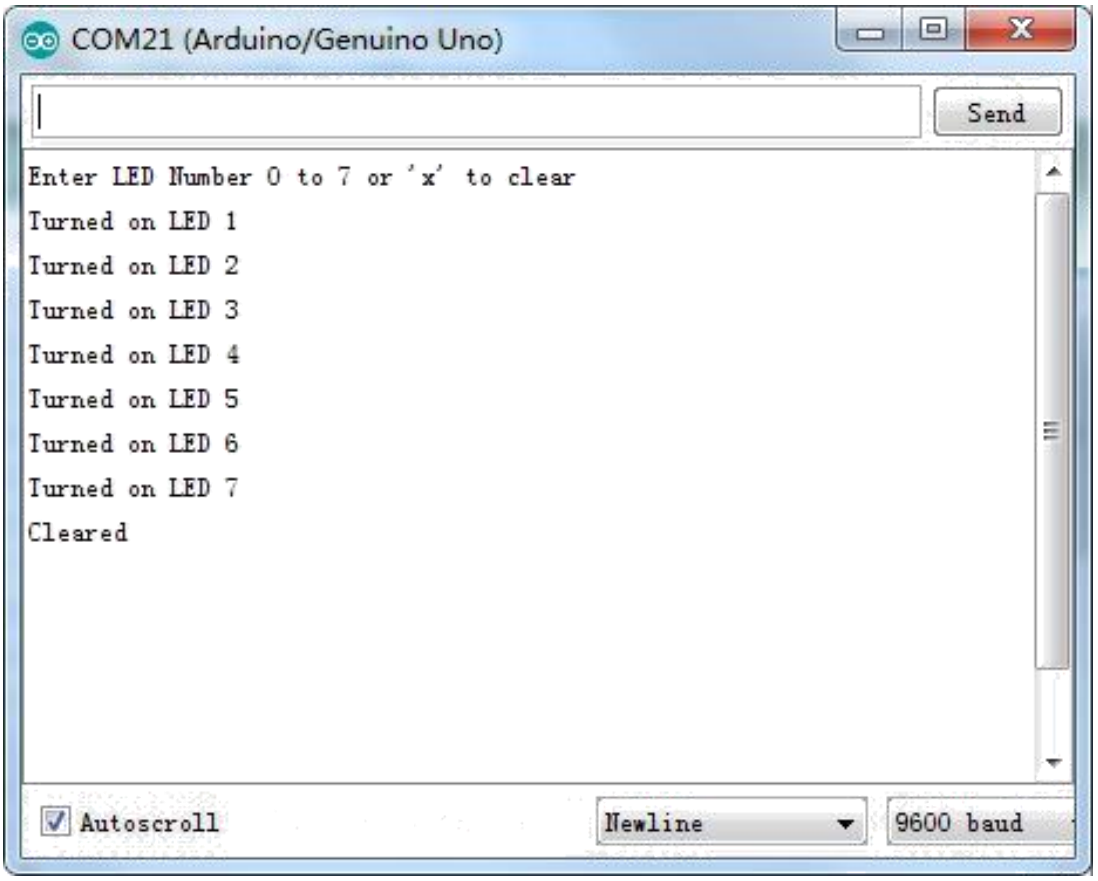


This window is called the Serial Monitor and it is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an UNO board (over USB) and also to receive messages from the UNO.

The message “Enter LED Number 0 to 7or '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

After wiring, please open program in the code folder- Lesson 18 The Serial Monitor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

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

```
  while (! Serial); // Wait until Serial is ready - Leonardo
  Serial.println("Enter LED Number 0 to 7 or 'x' 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 it is 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 19 Photocell

Overview

In this lesson, you will learn how to measure light intensity using an Analog Input. You will build on lesson 17 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) xUno R3 Board

(1) x 830 tie-points breadboard

(8) x leds

(8) x 220 ohm resistors

(1) x 1k ohm resistor

(1) x 74hc595 IC

(1) x Photoresistor (Photocell)

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



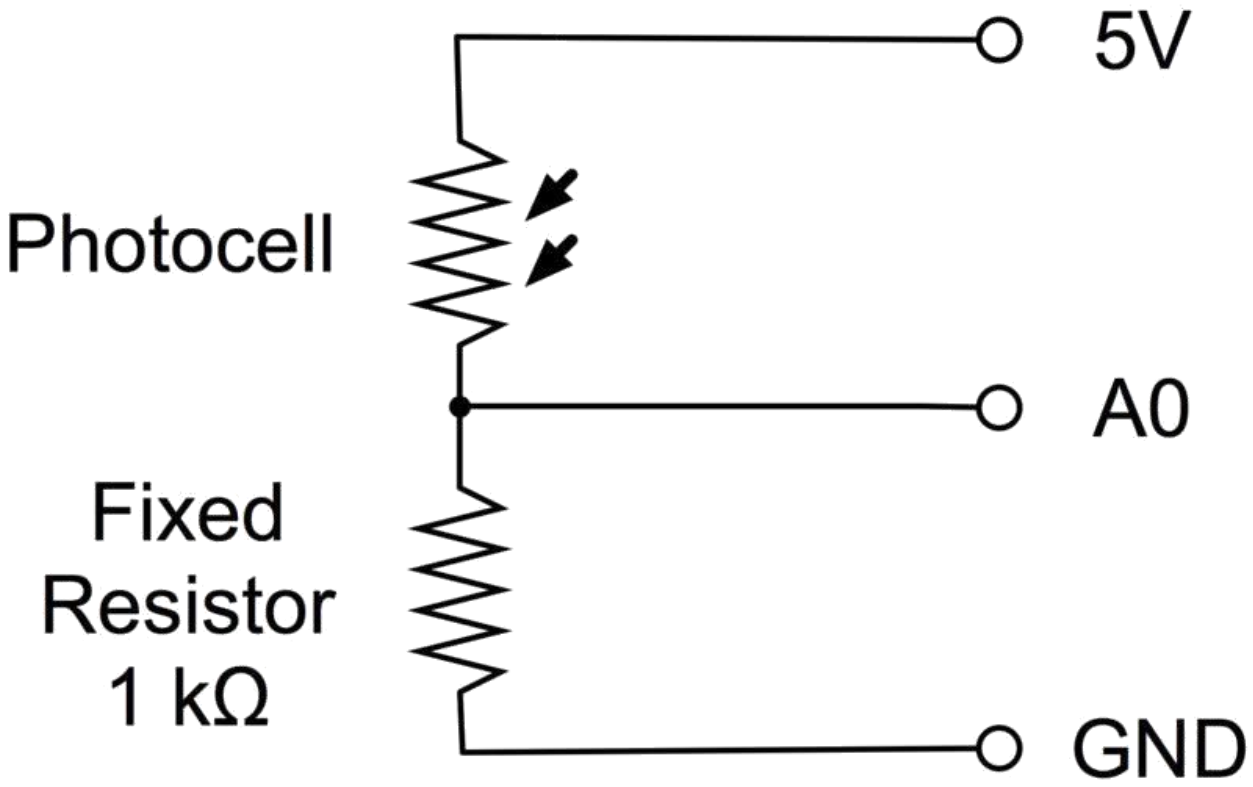
Component Introduction

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 needs to be converted into a voltage.

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



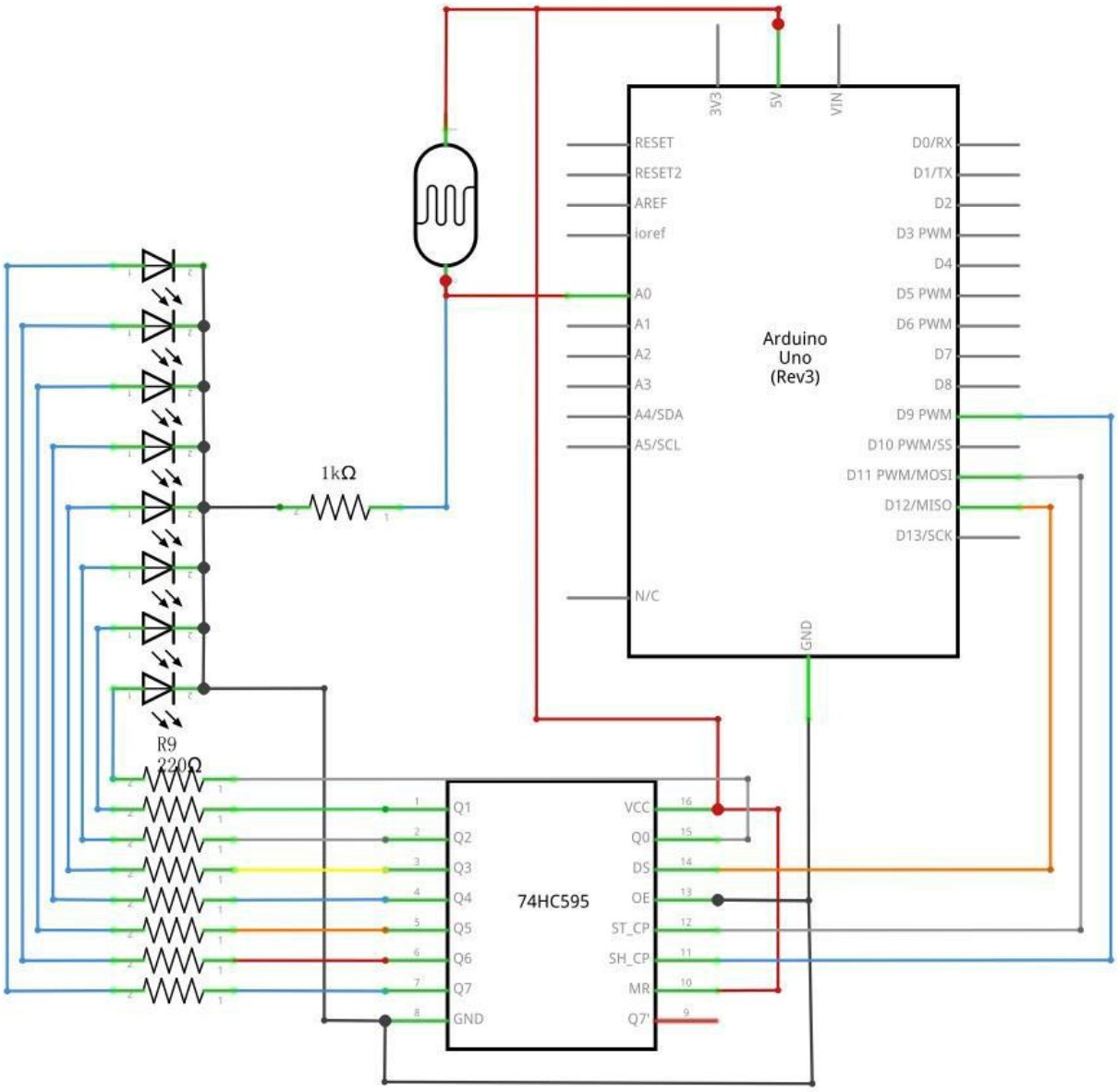
The resistor and photocell together behave like a pot. When the light is very bright, then the resistance of the photocell is very low compared with the fixed value resistor, and so it is as if the pot were turned to maximum.

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

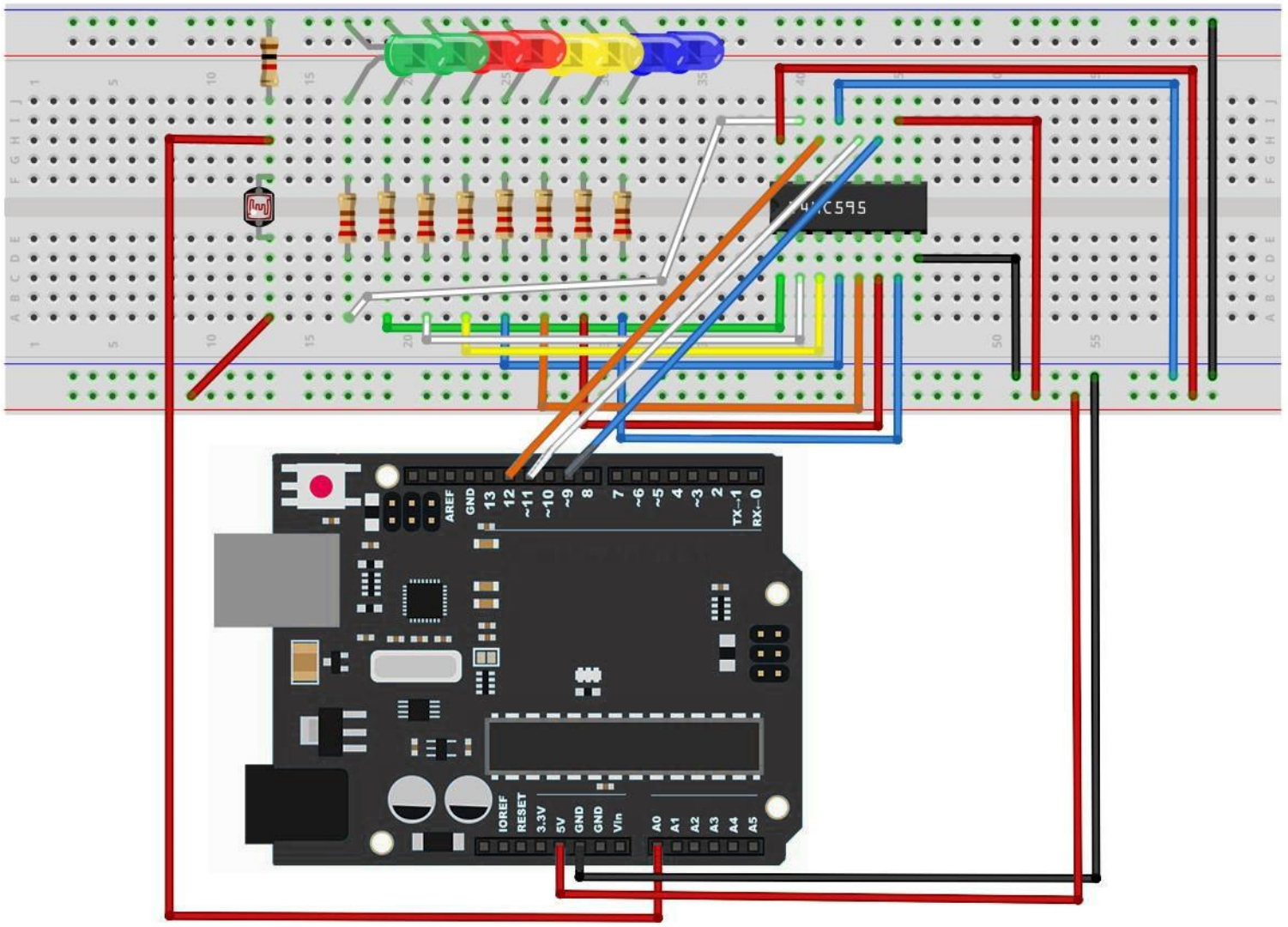
Load up the sketch given in the next section and try covering the photocell with your finger, and then holding it near a light source.

Connection

Schematic



Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 19 Photocell and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

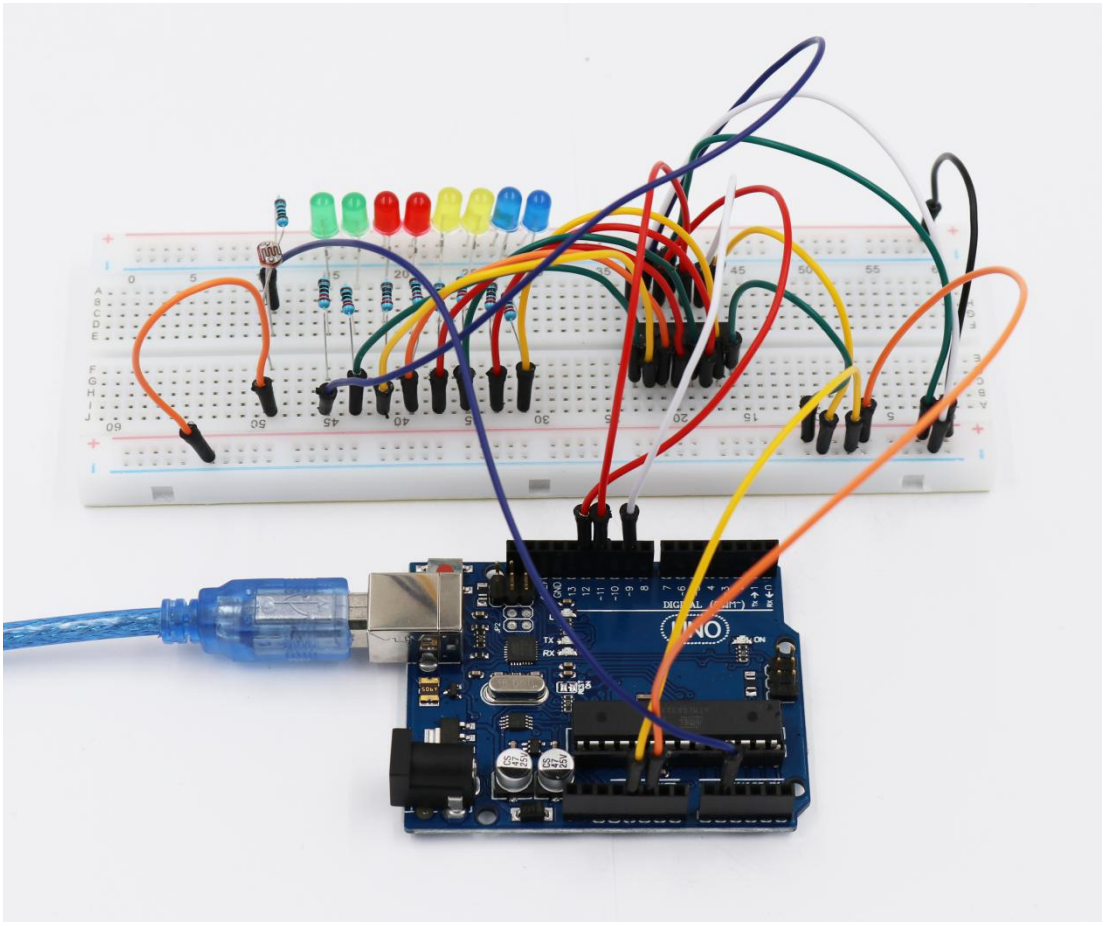
The first thing to note is that we have changed the name of the analog pin to be 'lightPin' rather than 'potPin' since we no longer have a pot connected.

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

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

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

Example picture

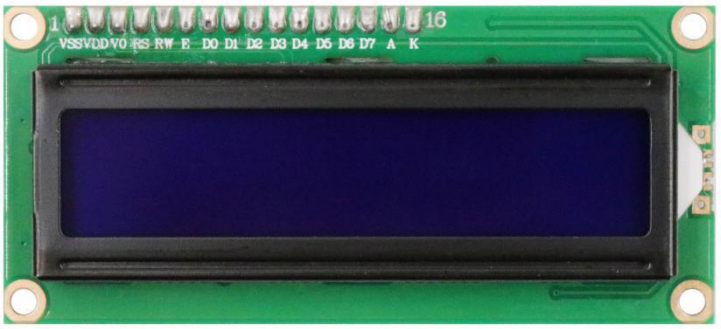


Lesson 20 LCD1602 I2C Module

Overview

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

In this lesson, we will run the Arduino example program for the LCD I2C library, but in the next lesson, we will get our display to show the temperature, using sensors.



Component Required:

(1) x UNO R3 Board

(1) x LCD1602 I2C module

(1) x Potentiometer (10k)

(1) x 830 tie-points Breadboard

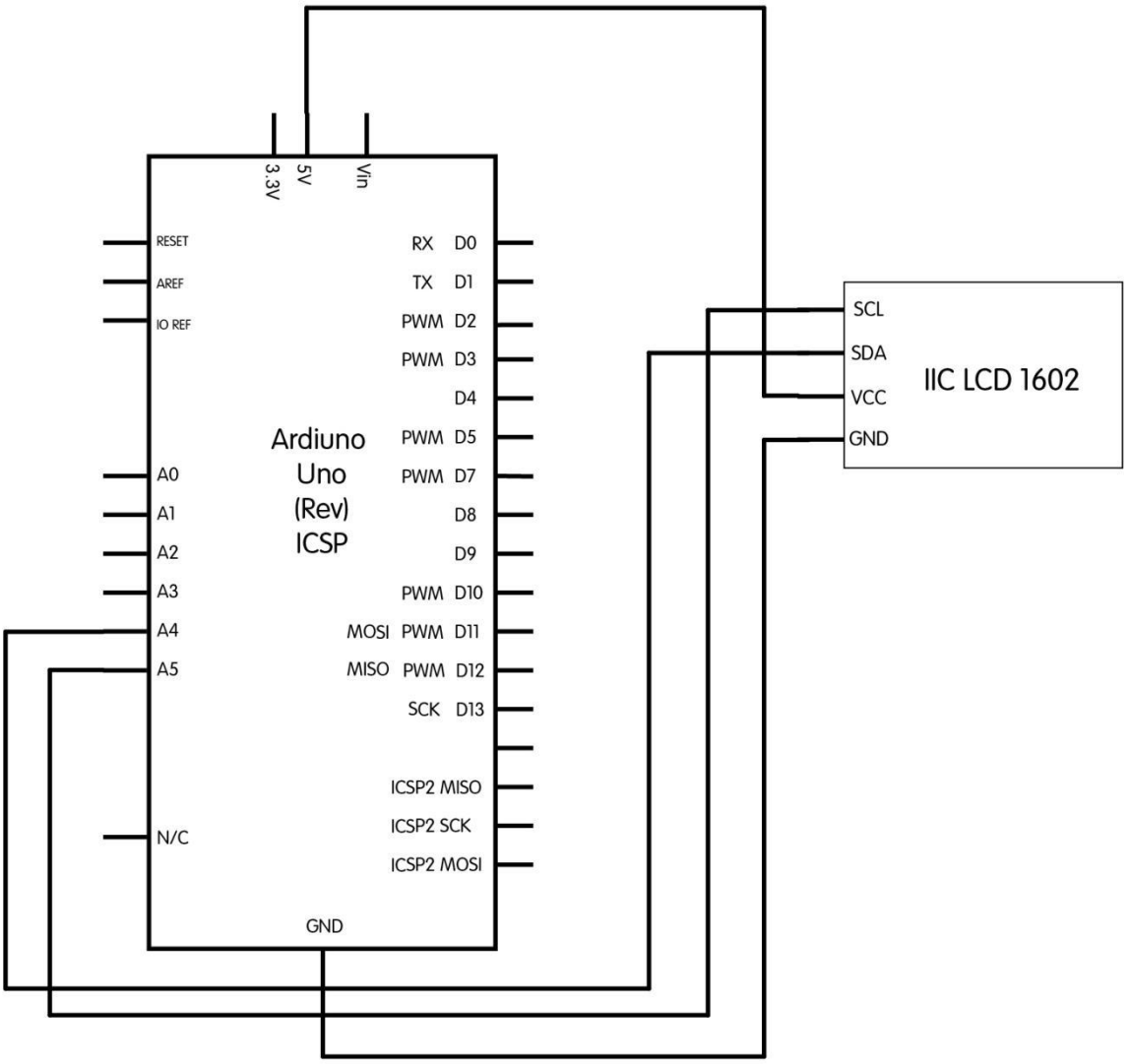
(16) x M-M wires (Male to Male jumper wires)

Introduction:

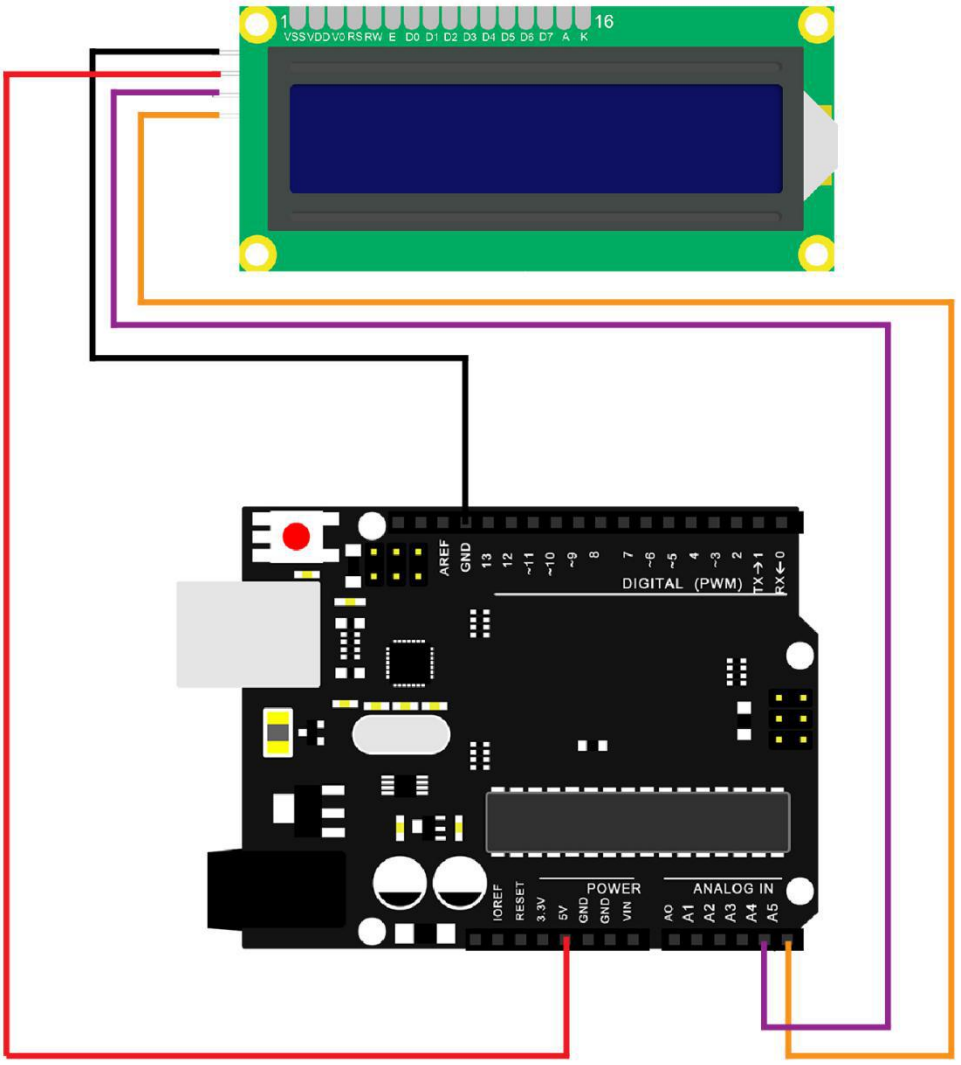
This is great LCD display compatible with arduino. With limited pin resources, your project will quickly run out of resources using normal LCDs. With this I2C interface LCD module, you only need 2 lines (I2C)to display the information.If you already have I2C devices in your project, this LCD module actually cost no more resources at all. The address can be set 0x27.

Connection

Schematic



Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 25 LCD Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < LiquidCrystal_I2C > library or re-install it, if necessary. Otherwise, your code won't work.

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

Upload the code to your Arduino board and you should see the message
“ Hello, world!
LAFVIN”
displayed.

The first thing of note in the sketch is the line:
#include <LiquidCrystal_I2C.h>

This tells Arduino that we wish to use the LiquidCrystal_I2C library.

Next set the LCD address to 0x27 for a 16 chars and 2 line display

LiquidCrystal_I2C lcd(0x27,16,2); //
After uploading this code, make sure the backlight is lit up, and adjust the potentiometer all the way around until you see the text message

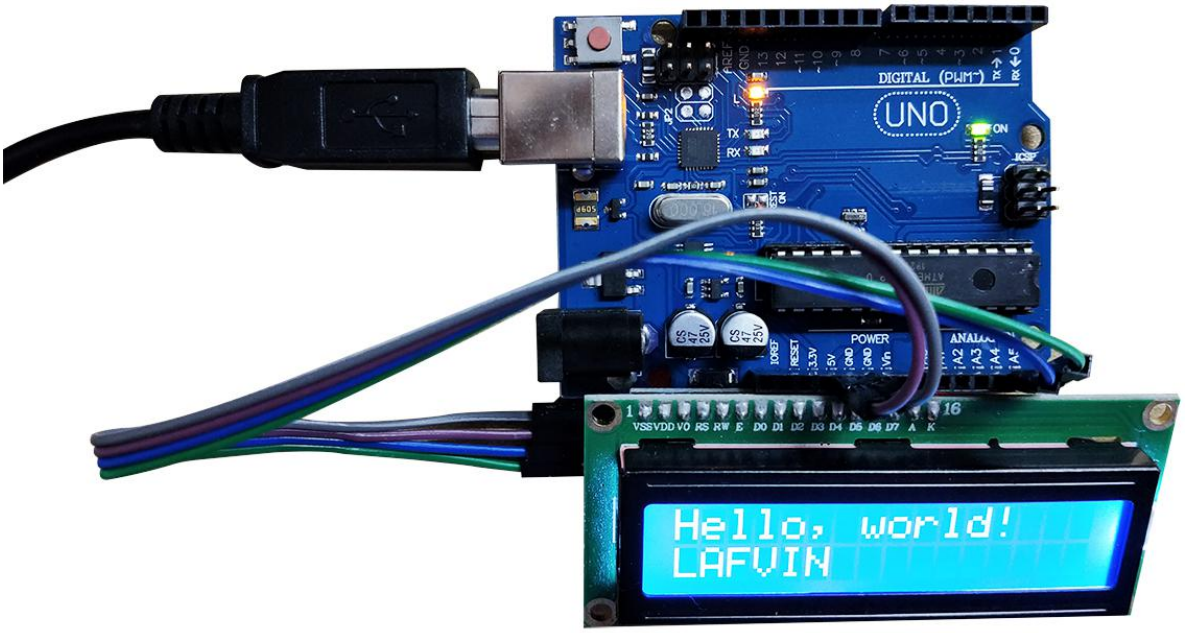
In the 'setup' function, we have four commands:
lcd.setCursor(0,0);
lcd.print("Hello, world!");
lcd.setCursor(0,1);
lcd.print("LAFVIN");

The lcd.setCursor(0,0) tells the Begin displaying the next line of code in the first row and first column.

The lcd.setCursor(0,1) tells the Begin displaying the next line of code in the second row and first column.

lcd.print("Hello, world!");
lcd.print("LAFVIN");
Tell screen display
“ Hello, world!
LAFVIN”

Example picture



Lesson 21 Thermometer

Overview

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

Component Required:

(1) x Uno R3 Board
(1) x LCD1602 I2C Module
(1) x 10k ohm resistor
(1) x Thermistor
(1) x Potentiometer
(1) x 830 tie-points Breadboard
(18) x M-M wires (Male to Male jumper wires)

Component Introduction

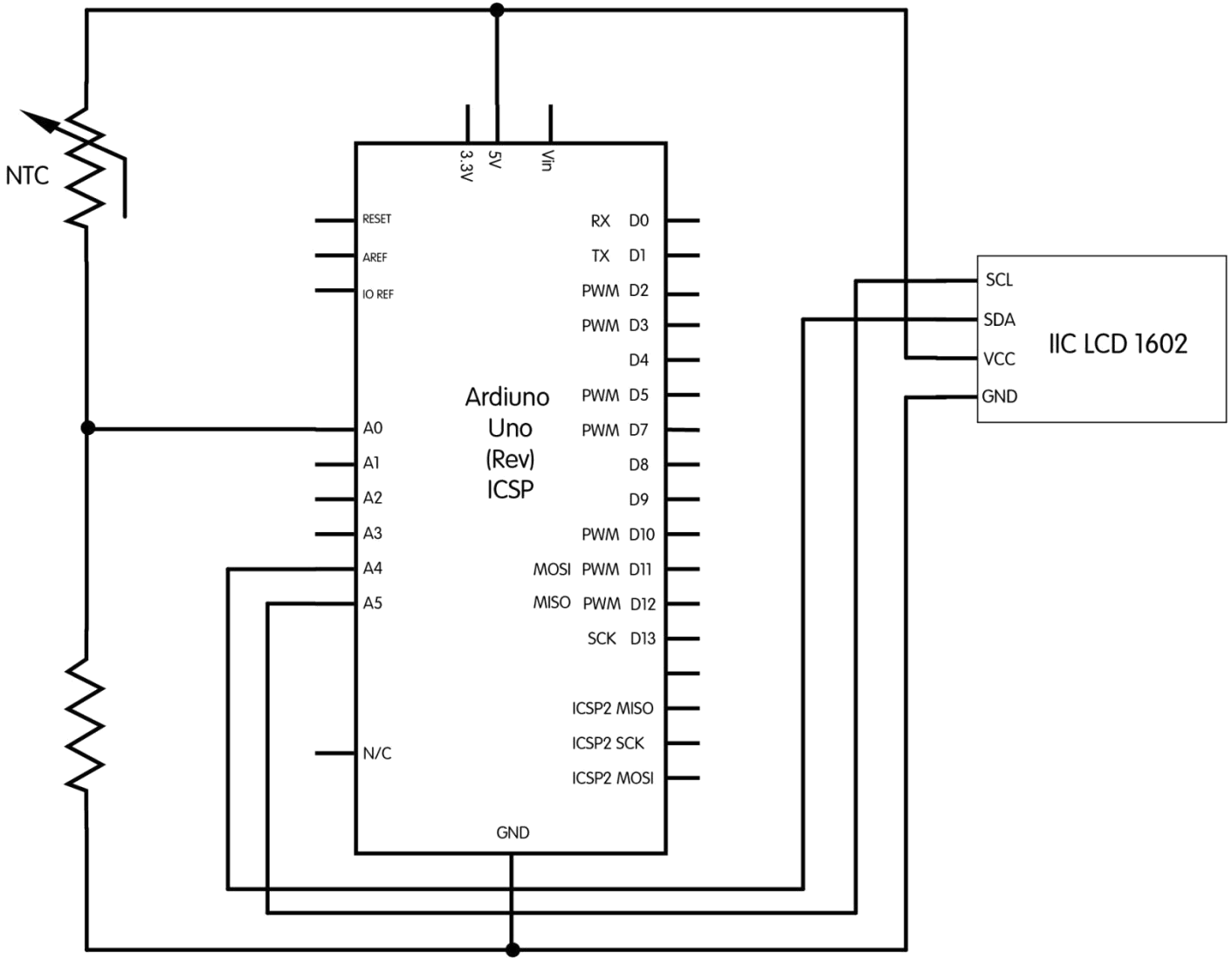
Thermistor

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

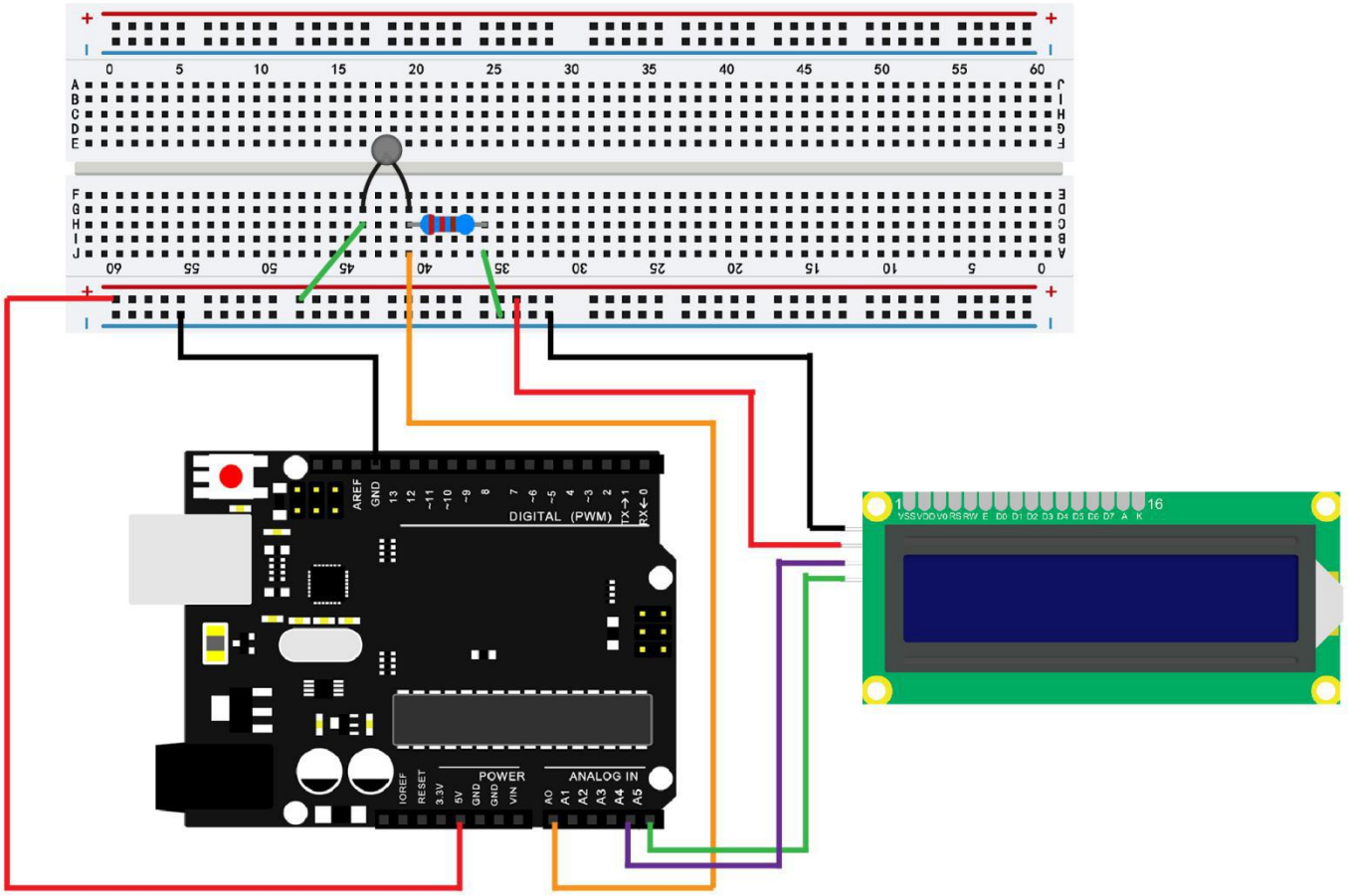
There are two kinds of thermistors, NTC (negative temperature coefficient) and PTC (positive temperature coefficient). In general, you will see NTC sensors used for temperature measurement. PTC's are often used as resettable fuses - an increase in temperature increases the resistance which means that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

Connection

Schematic



Wiring diagram



The breadboard layout is based on the layout from Lesson 25, so it will simplify things if you still have this on the breadboard.

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

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

Code

After wiring, please open the program in the code folder- Lesson 26 Thermometer and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Before you can run this, make sure that you have installed the < LiquidCrystal_I2C > library or re-install it, if necessary. Otherwise, your code won't work.

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

The sketch for this is based on that of lesson 25. Load it up onto your Arduino and you should find that warming the temperature sensor by putting your finger on it will increase the temperature reading.

```
LiquidCrystal_I2C lcd(0x27,16,2);  //
```
set the LCD address to 0x27 for a 16 chars and 2 line display

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

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

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

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

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

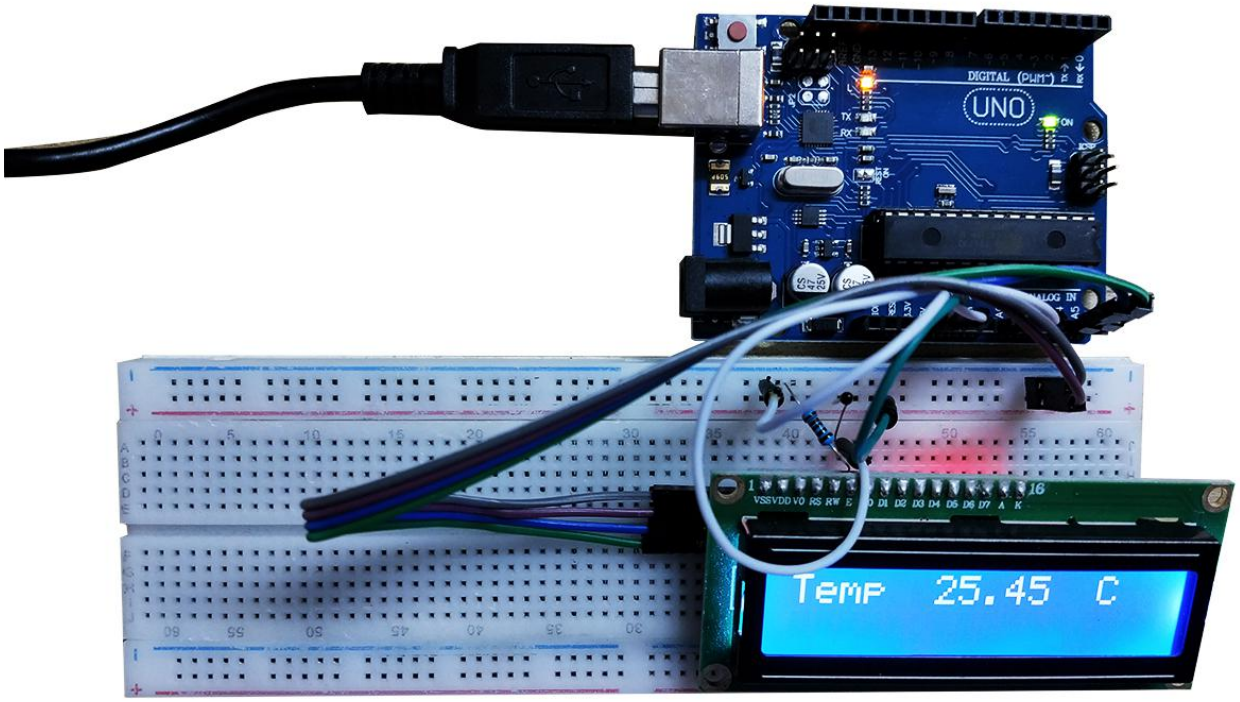
To avoid this, write the whole line of the LCD each time around the loop.

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

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

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

Example picture



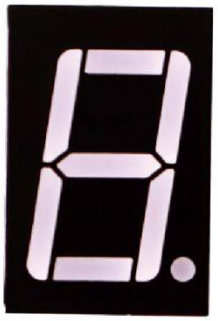
Lesson 22 74HC595 And Segment Display

Overview

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

Component Required:

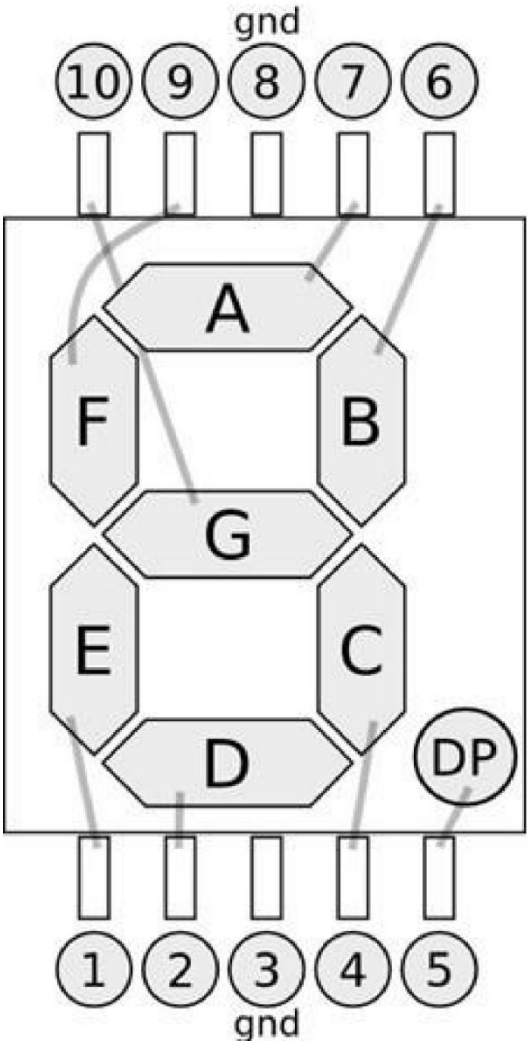
(1) x Uno R3 Board
(1) x 830 tie-points breadboard
(1) x 74HC595 IC
(1) x 1 Digit 7-Segment Display
(8) x 220 ohm resistors
(26) x M-M wires (Male to Male jumper wires)



Component Introduction

Seven segment display

Below is the seven-segment pin diagram.

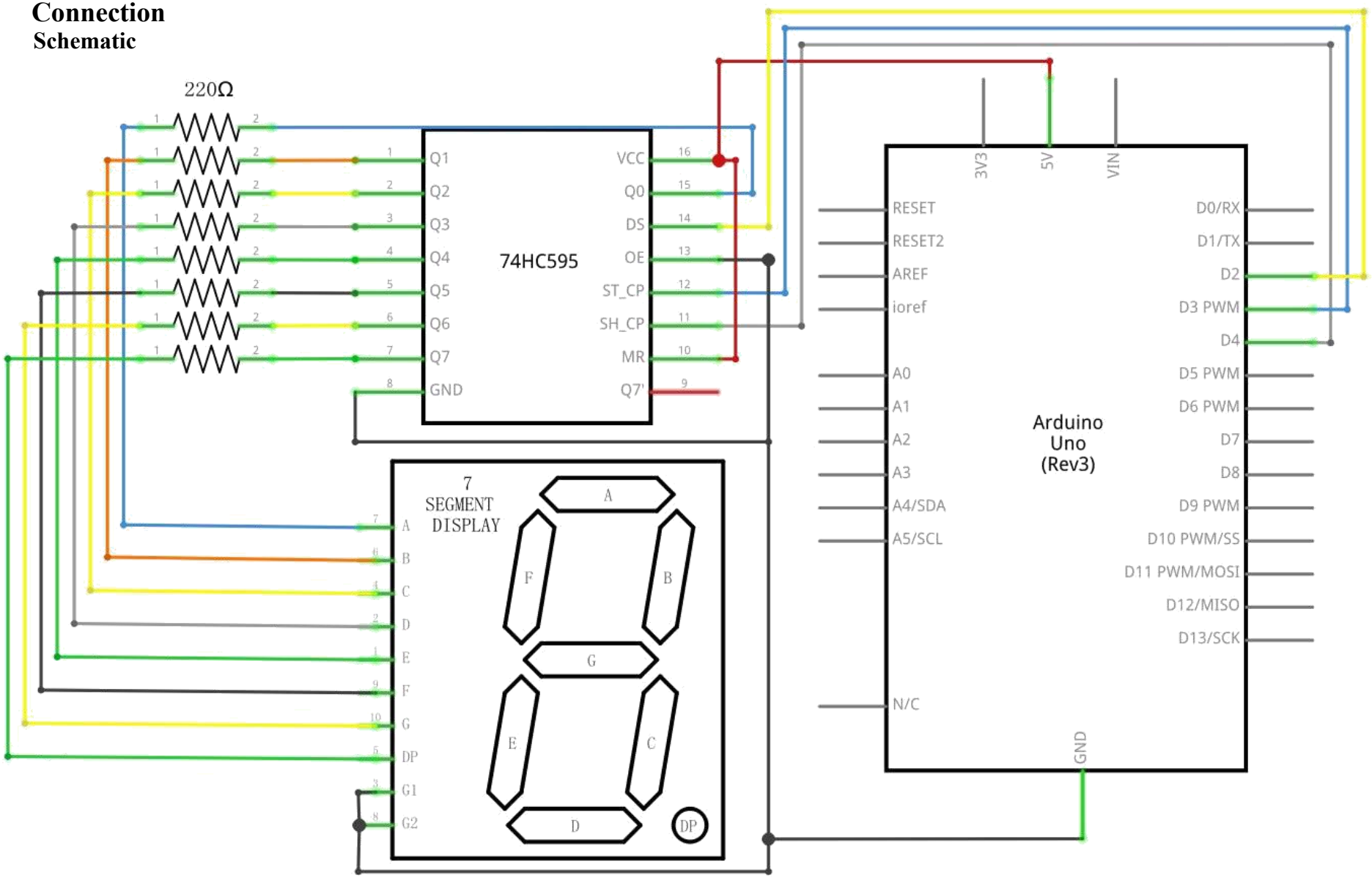


0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device, if you are using a common anode, the table should be replaced every 1 0 0 should all replaced by 1):

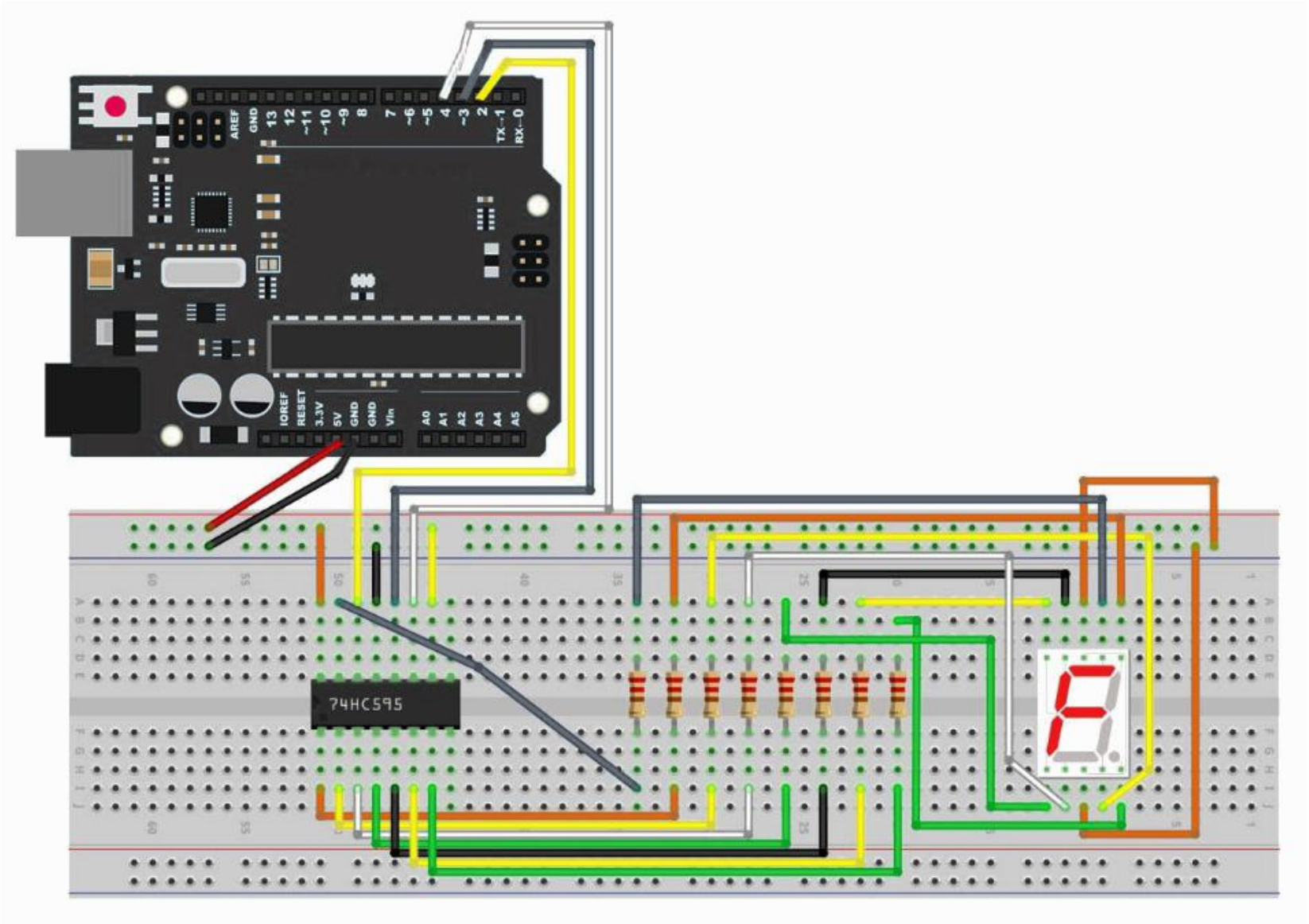
Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

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 2 (the figure below the yellow line)

ST_CP (pin 12, latch pin) connected to UNO R3 board pin 3 (FIG blue line below)

SH_CP (pin 11, clock pin) connected to UNO R3 board pin 4 (the figure below the white line)

Step two: Connect the seven segment display

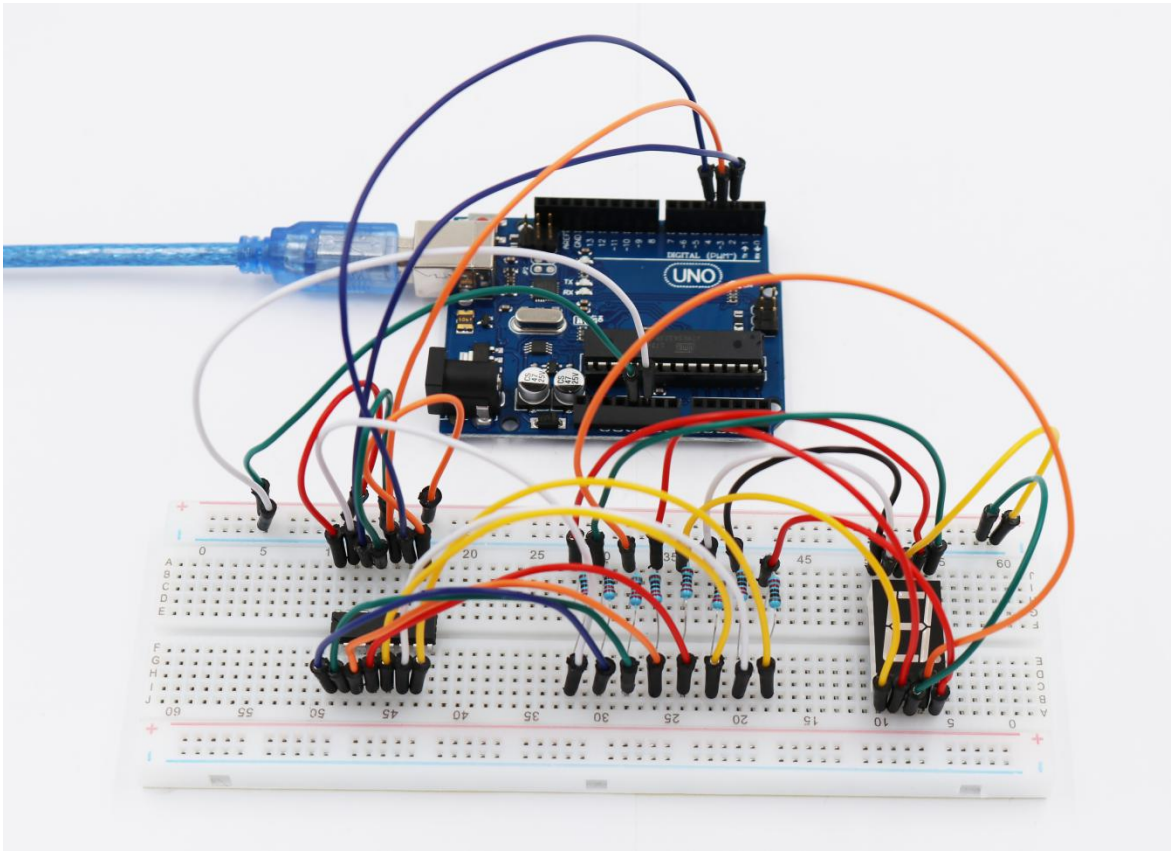
The seven-segment display 3, 8 pin to UNO R3 board GND (This example uses the common cathode, if you use the common anode, please connect the 3, 8 pin to UNO R3 board + 5V)

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

After wiring, please open the program in the code folder- Lesson 22 74HC595 And Segment Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 23 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, please notice that if it is common anode, the common anode pin connects to the power source; if it is common cathode, the common cathode pin connects to 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 that you hardly notice the intervals.

Component Required:

(1) x Uno R3 Board

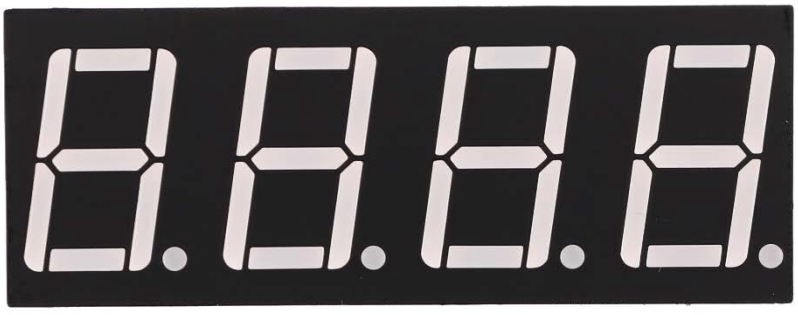
(1) x 830 tie-points breadboard

(1) x 74HC595 IC

(1) x 4 Digit 7-Segment Display

(4) x 220 ohm resistors

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

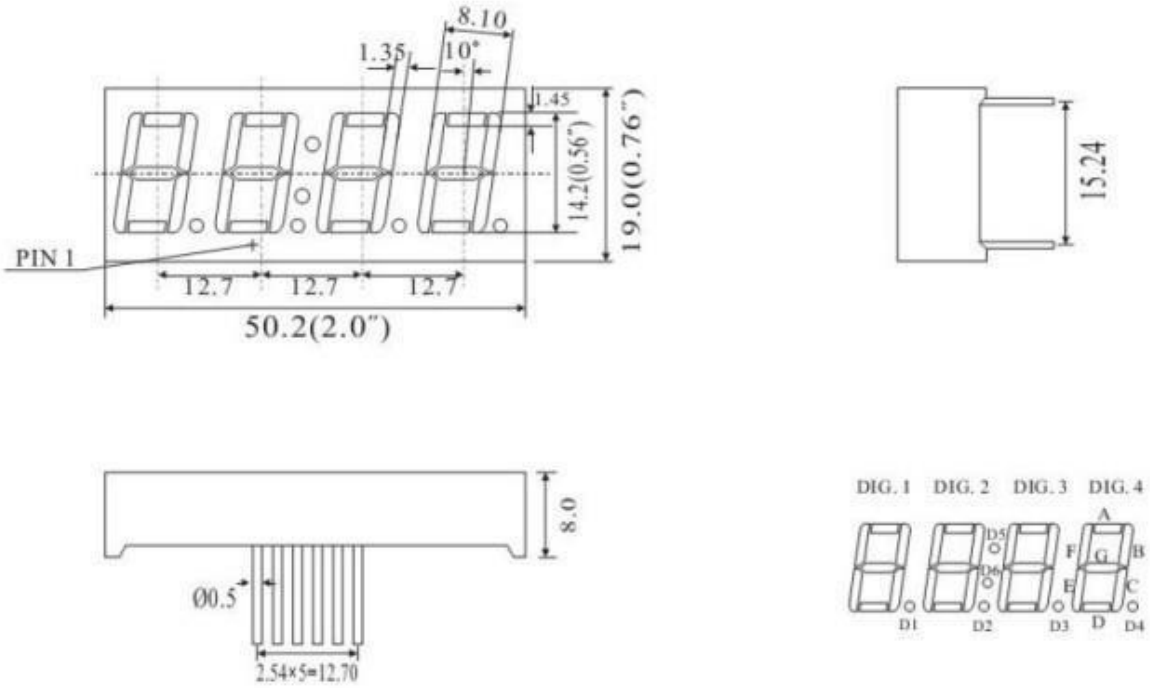


Component Introduction

Four Digital Seven segment display

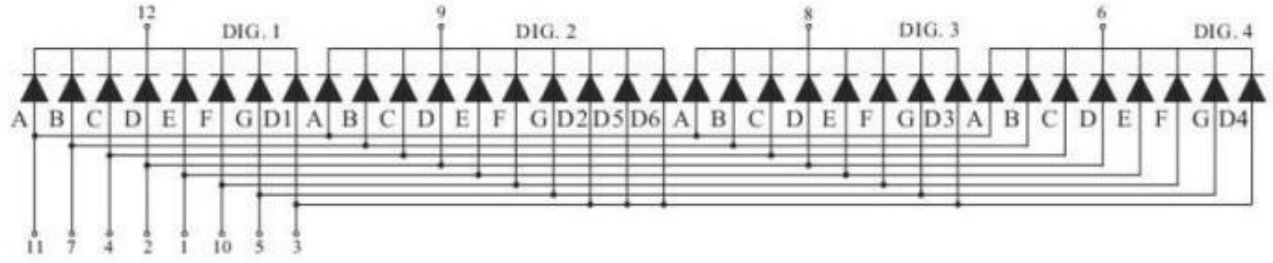
Package Dimensions

CPS05643AB

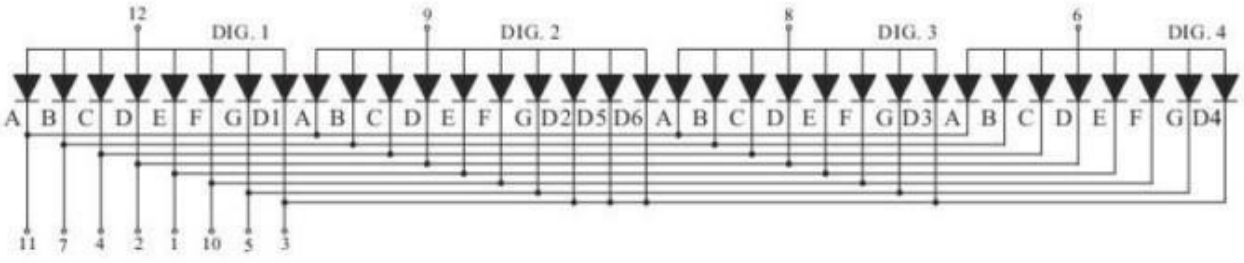


UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

Internal Circuit Diagram



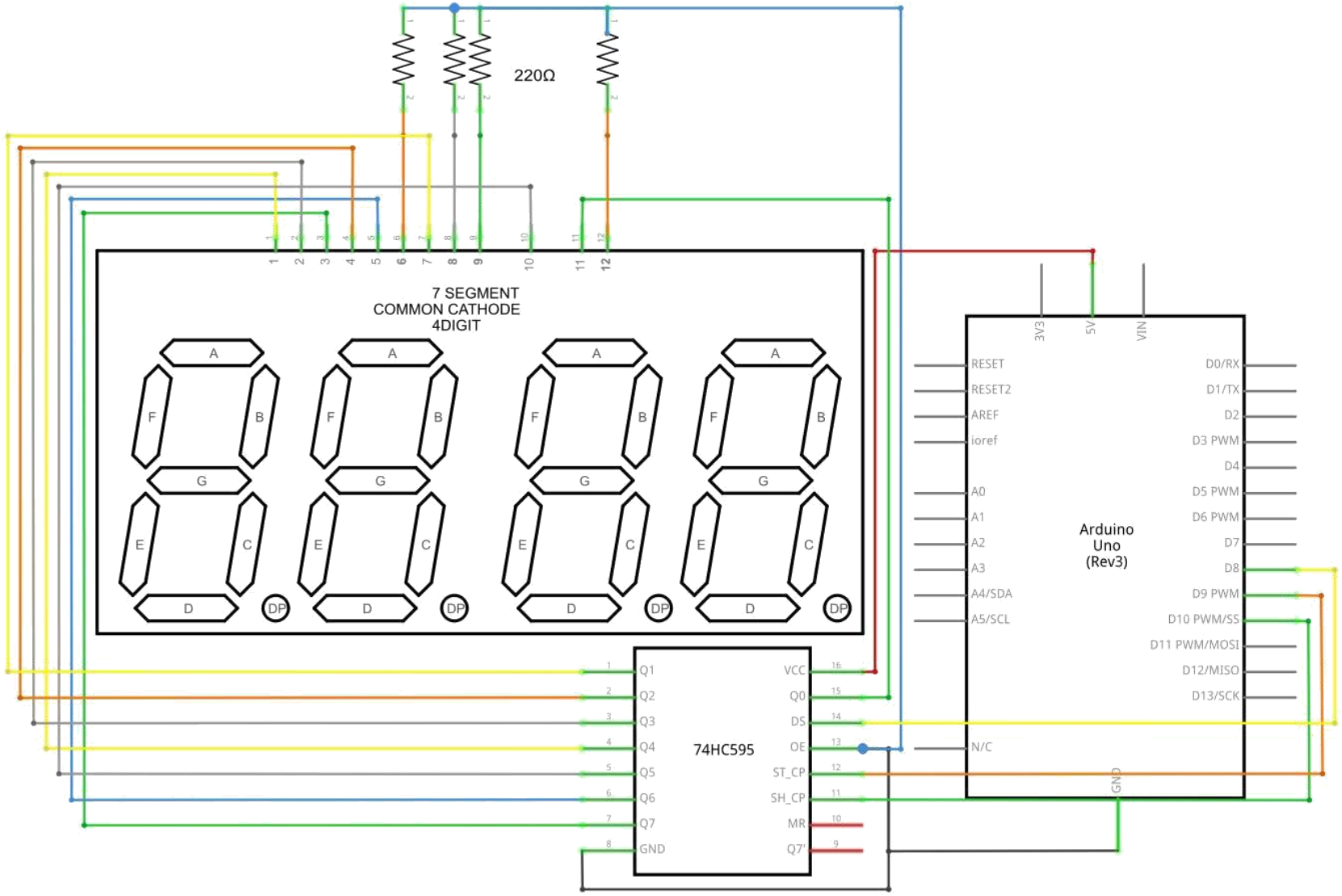
5643A



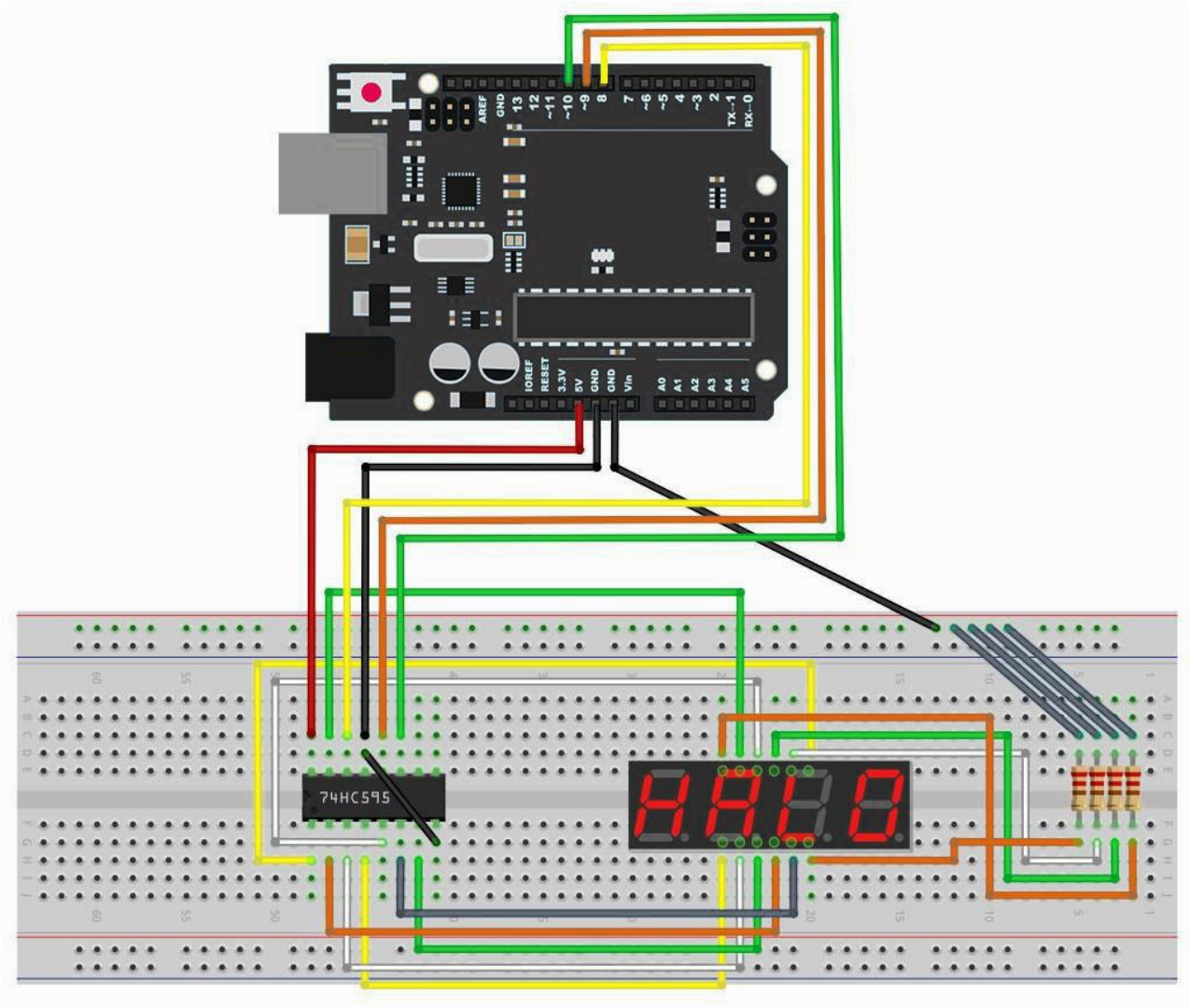
5643B

Connection

Schematic



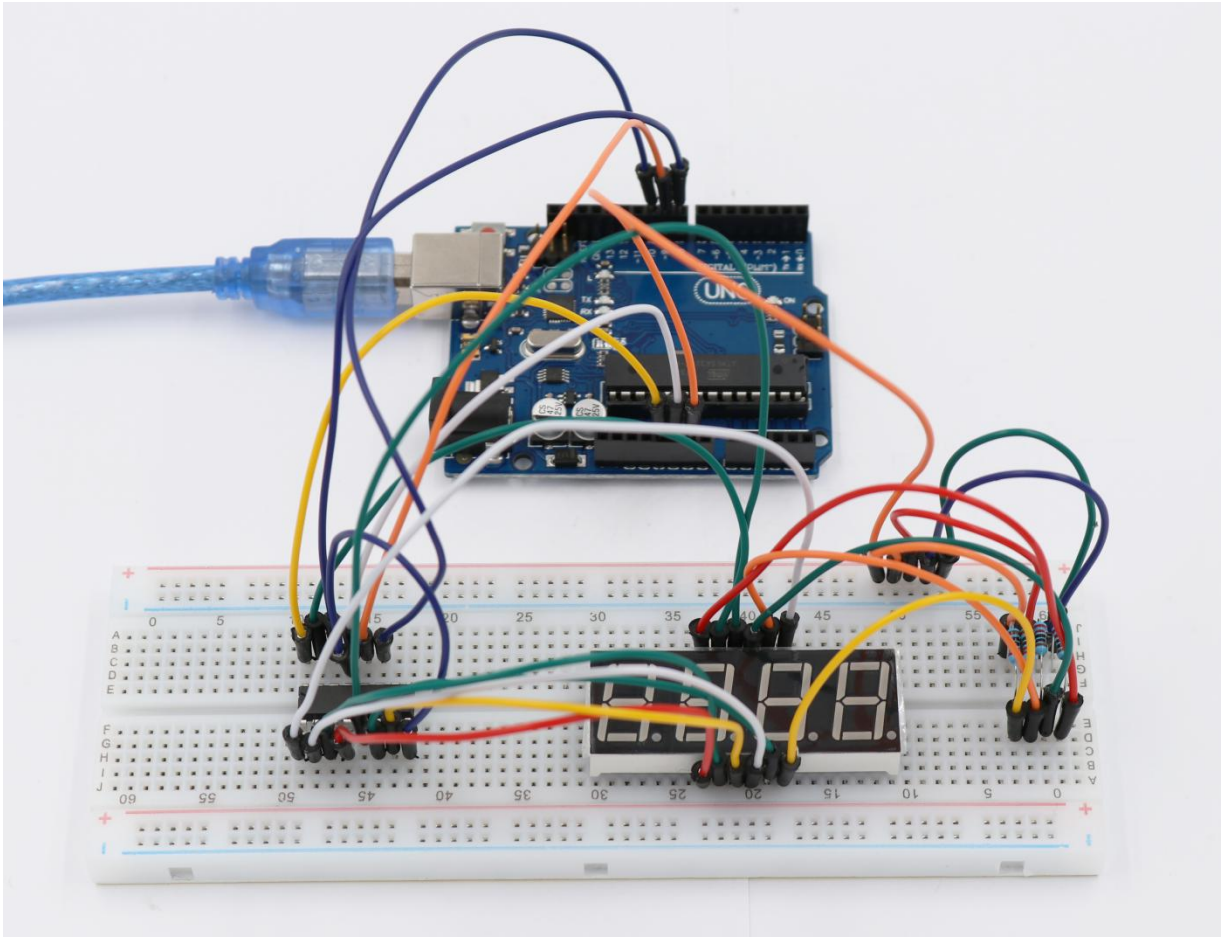
Wiring diagram



Code

After wiring, please open the program in the code folder- Lesson 23 Four Digital Seven Segment Display and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

Example picture



Lesson 24 DC Motors

Overview

In this lesson, you will learn how to control a small DC motor using an UNO R3 and a transistor.

Component Required:

(1) x Uno R3 Board

(1) x 830 tie-points breadboard

(1) x L293D IC

(1) x Fan blade and 3-6v motor

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

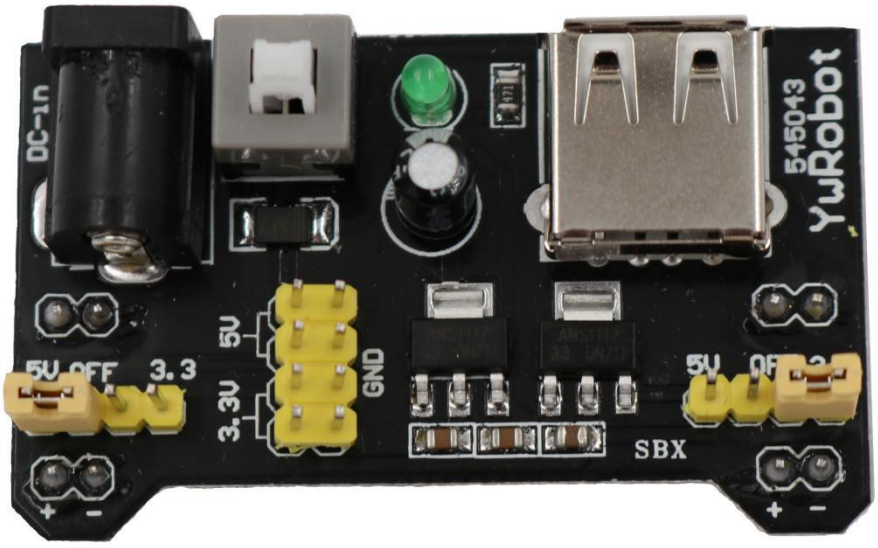
(1) x Power Supply Module

(1) x 9V1A adapter

Component Introduction

Breadboard Power Supply

The small DC motor is likely to use more power than an UNO R3 board digital output can handle directly. If we tried to connect the motor straight to an UNO R3 board pin, there is a good chance that it could damage the UNO R3 board. So we use a power supply module provides power supply



Product Specifications:

Locking On/Off Switch

LED Power Indicator

Input voltage: 6.5-9v (DC) via 5.5mm x 2.1mm plug

Output voltage: 3.3V/5v

Maximum output current: 700 mA

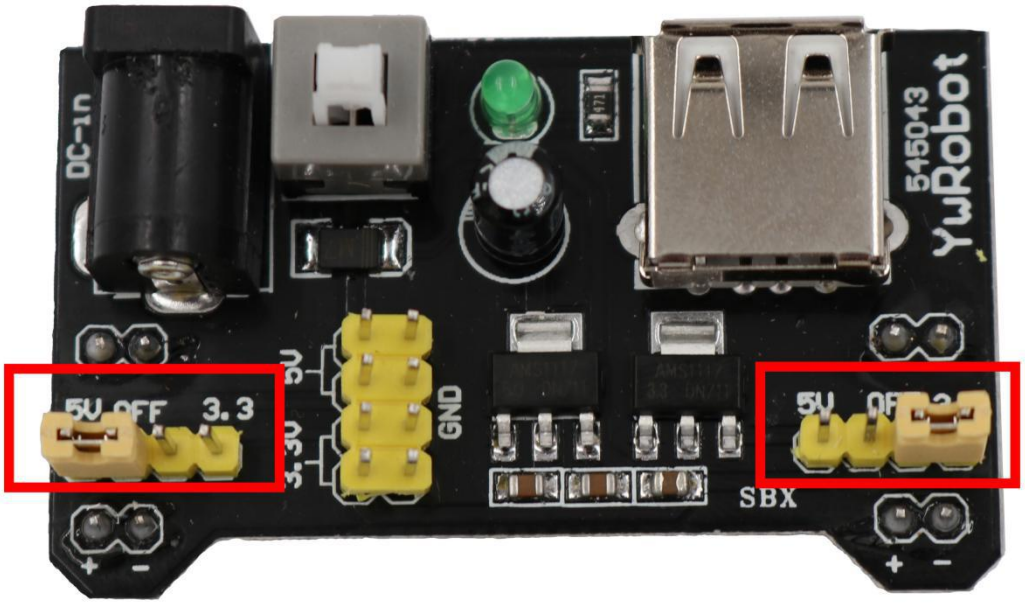
Independent control rail output. 0v, 3.3v, 5v to breadboard

Output header pins for convenient external use

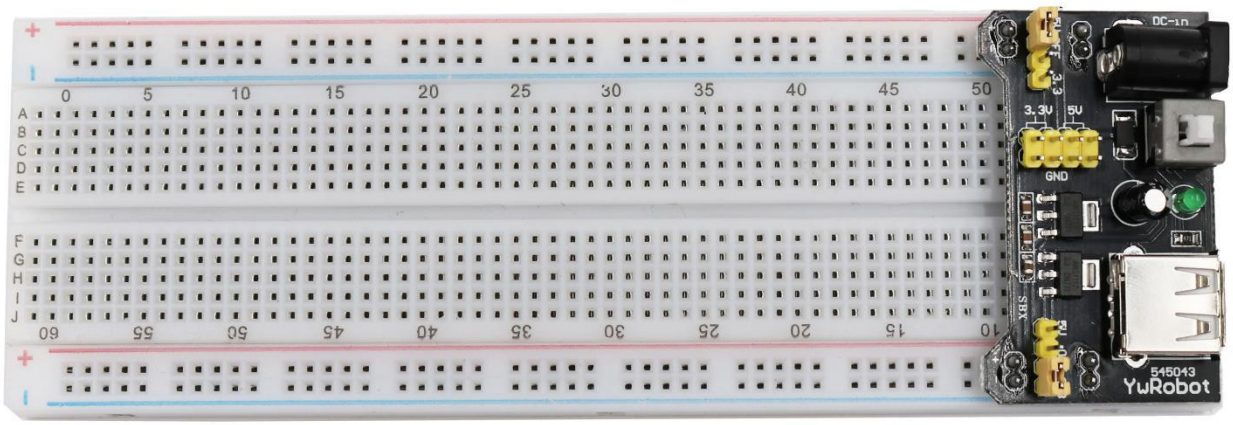
Size: 2.1 in x 1.4 in

USB device connector onboard to power external device

Setting up output voltage:



The left and right voltage output can be configured independently. To select the output voltage, move jumper to the corresponding pins. Note: power indicator LED and the breadboard power rails will not power on if both jumpers are in the “OFF” position.

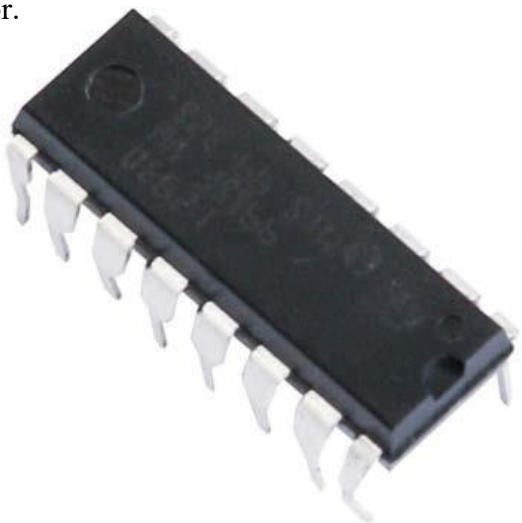


Important note:

Make sure that you align the module correctly on the breadboard. The negative pin(-) on module lines up with the blue line(-) on breadboard and that the positive pin(+) lines up with the red line(+). Failure to do so could result in you accidently reversing the power to your project

L293D

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor.



Product Specifications:

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

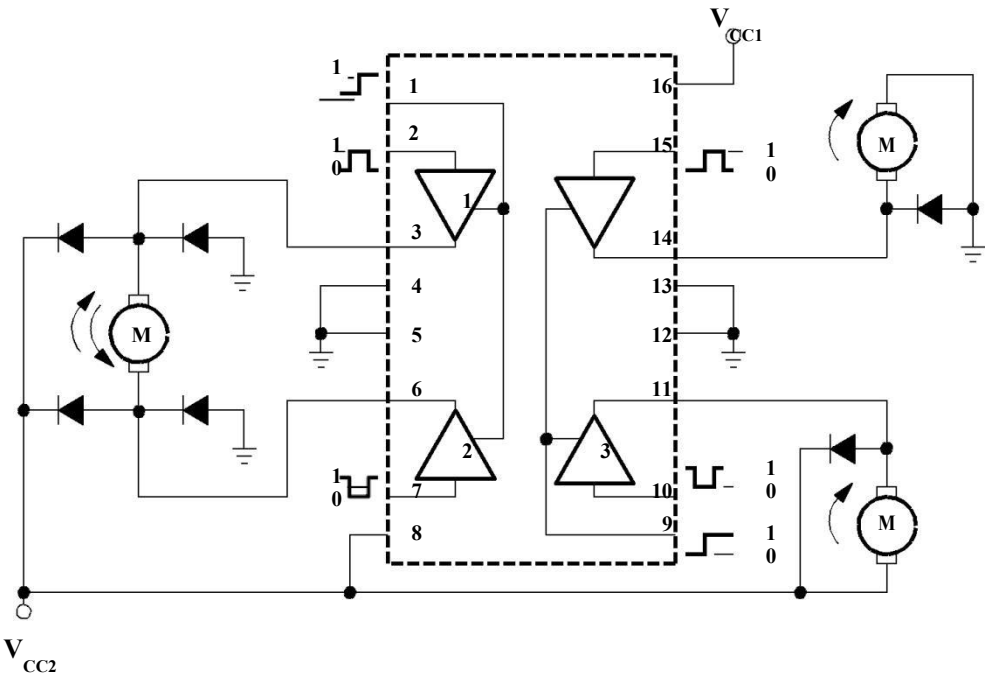


Description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

Block diagram



I got fed up with indecipherable pinout diagrams within datasheets, so have designed my own that I think gives more pertinent information.

There are 3 wires connected to the Arduino, 2 wires connected to the motor, and 1 wire connected to a battery.



To use this pinout:

The left hand side deals with the first motor, the right hand side deals with a second motor. Yes, you can run it with only one motor connected.

Arduino Connections

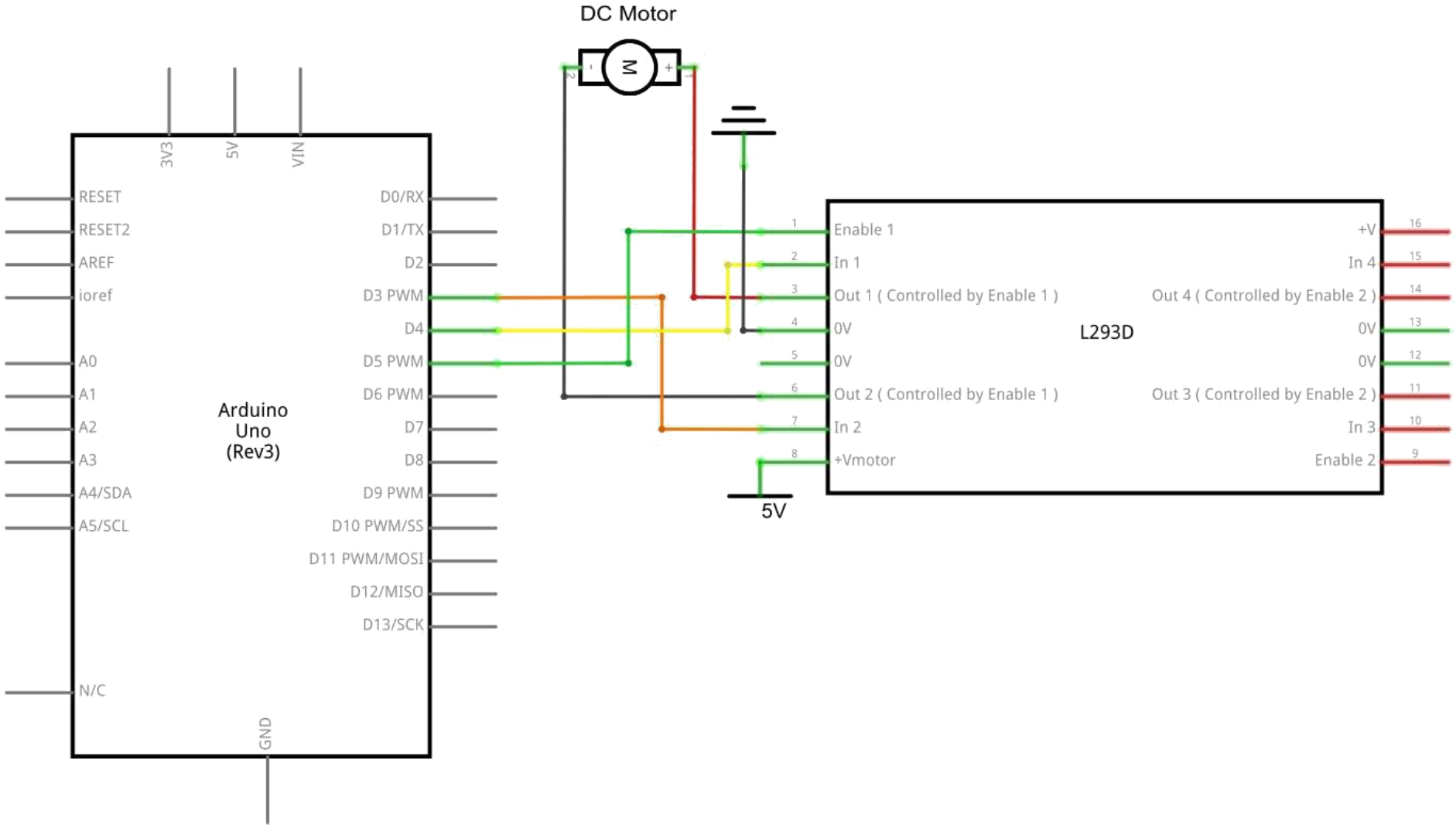
M1 PWM - connect this to a PWM pin on the Arduino. They're labelled on the Uno, pin 5 is an example. Output any integer between 0 and 255, where 0 will be off, 128 is half speed and 255 is max speed.

M1 direction 0/1 and M1 direction 1/0 - Connect these two to two digital Arduino pins. Output one pin as HIGH and the other pin as LOW, and the motor will spin in one direction.

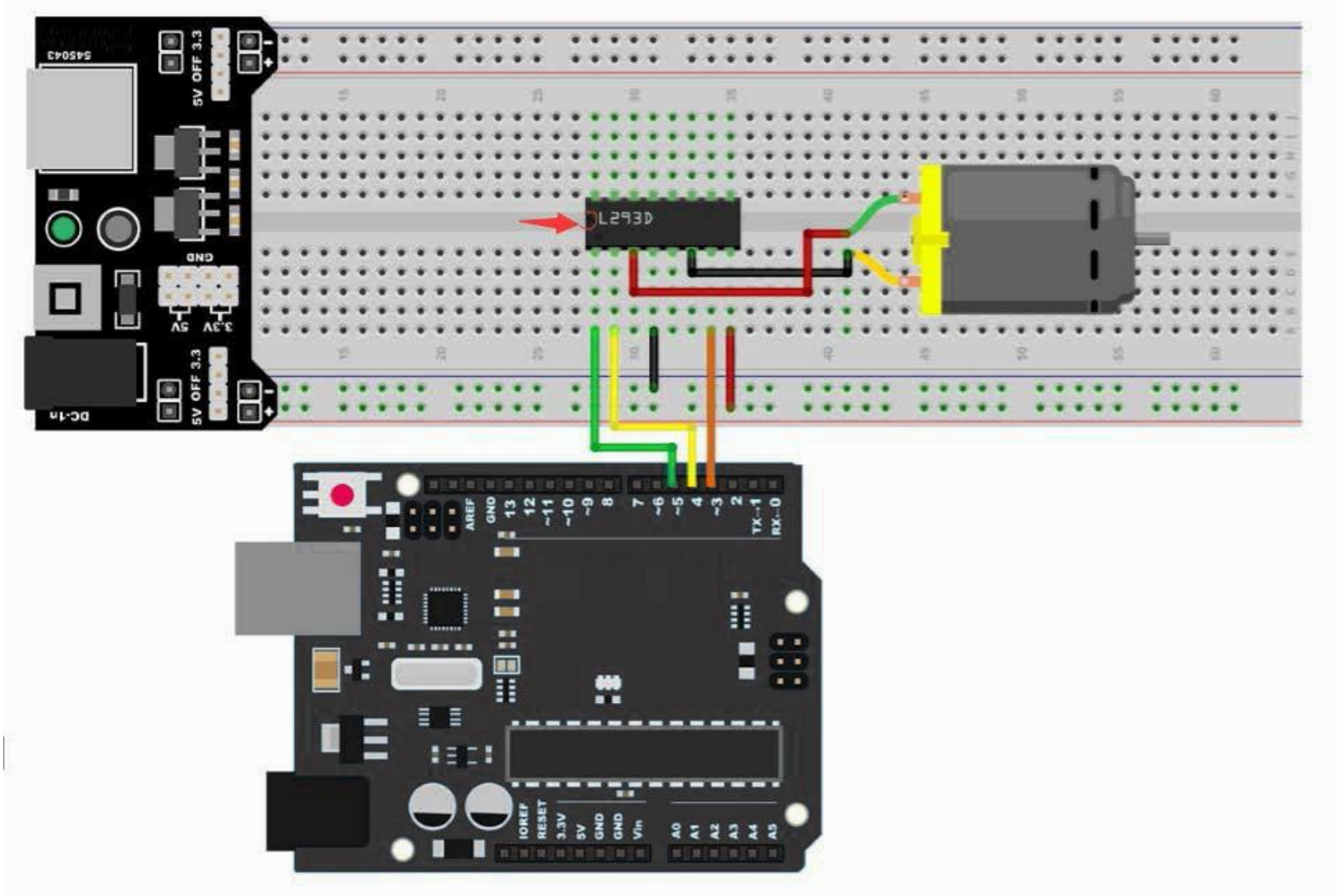
Reverse the outputs to LOW and HIGH, and the motor will spin in the other direction.

Connection

Schematic

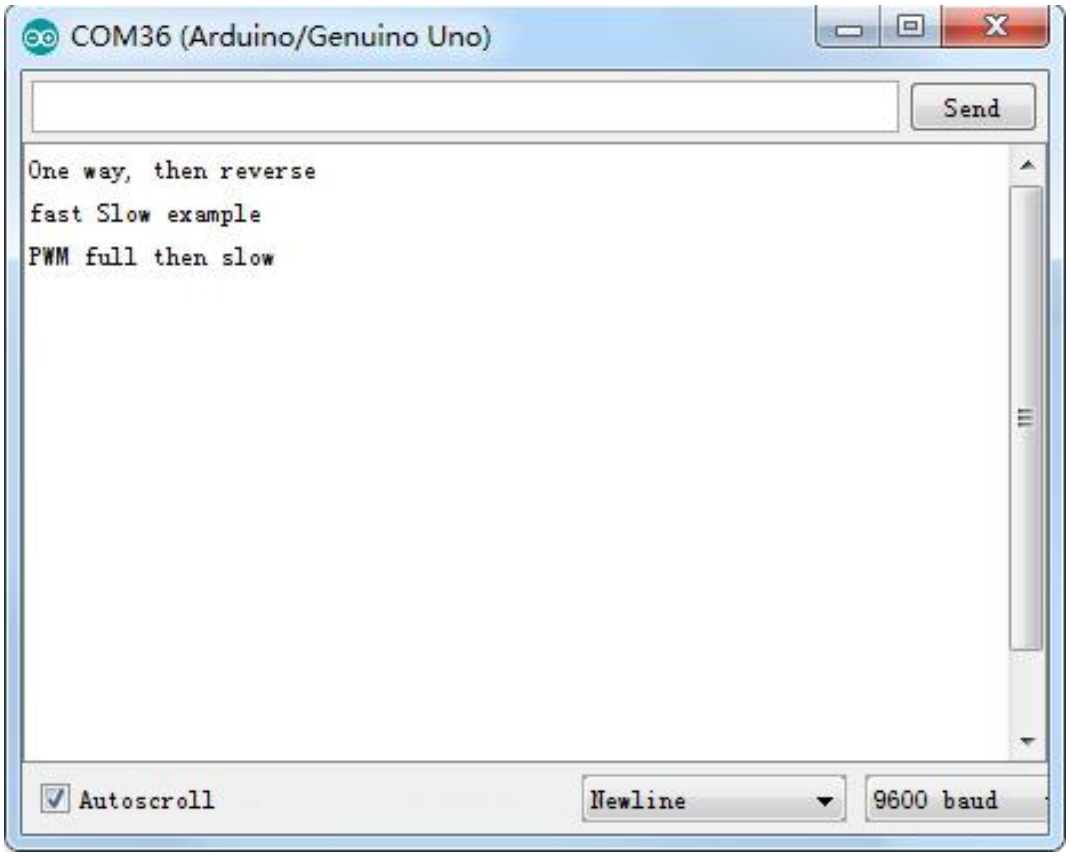


Wiring diagram



The code below does not use a separate power supply (ie a battery), it uses instead the 5v power from the Arduino. Note that this would be risky without the L293D controlling it.

You should _never_ connect a motor directly to the Arduino, because when you switch a motor off you get an electrical feedback. With a small motor, this will damage your Arduino, and with a large motor, you can watch an interesting flame and sparks effect.

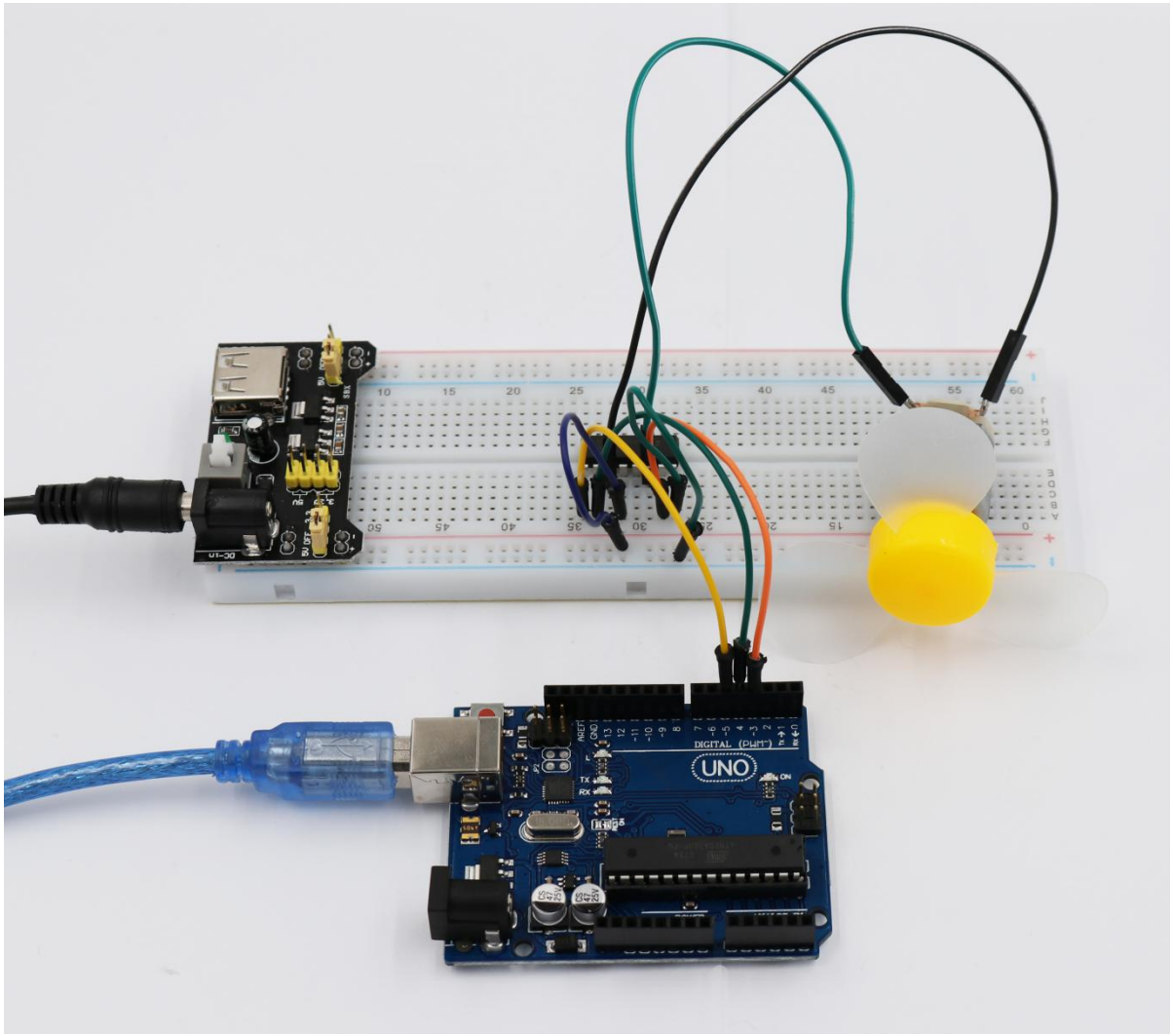


Code

After wiring, please open the program in the code folder- Lesson 24 DC Motors and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

After program loading, turn on all the power switches. The motor will slightly rotate clockwise and anticlockwise for 5 times. Then, it will continue to dramatically rotate clockwise. After a short pause, it will dramatically rotate anticlockwise. Then the controller board will send PWM signal to drive the motor, the motor will slowly reduce its maximum RPM to the minimum and increase to the maximum again. Finally, it comes to a stop for 10s until the next cycle begins.

Example picture



Lesson 25 Relay

Overview

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

Component Required:

(1) x Uno R3 Board

(1) x 830 tie-points breadboard

(1) x Fan blade and 3-6v dc motor

(1) x L293D IC

(1) x 5v Relay

(1) x Power Supply Module

(1) x 9V1A Adapter

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



Component Introduction

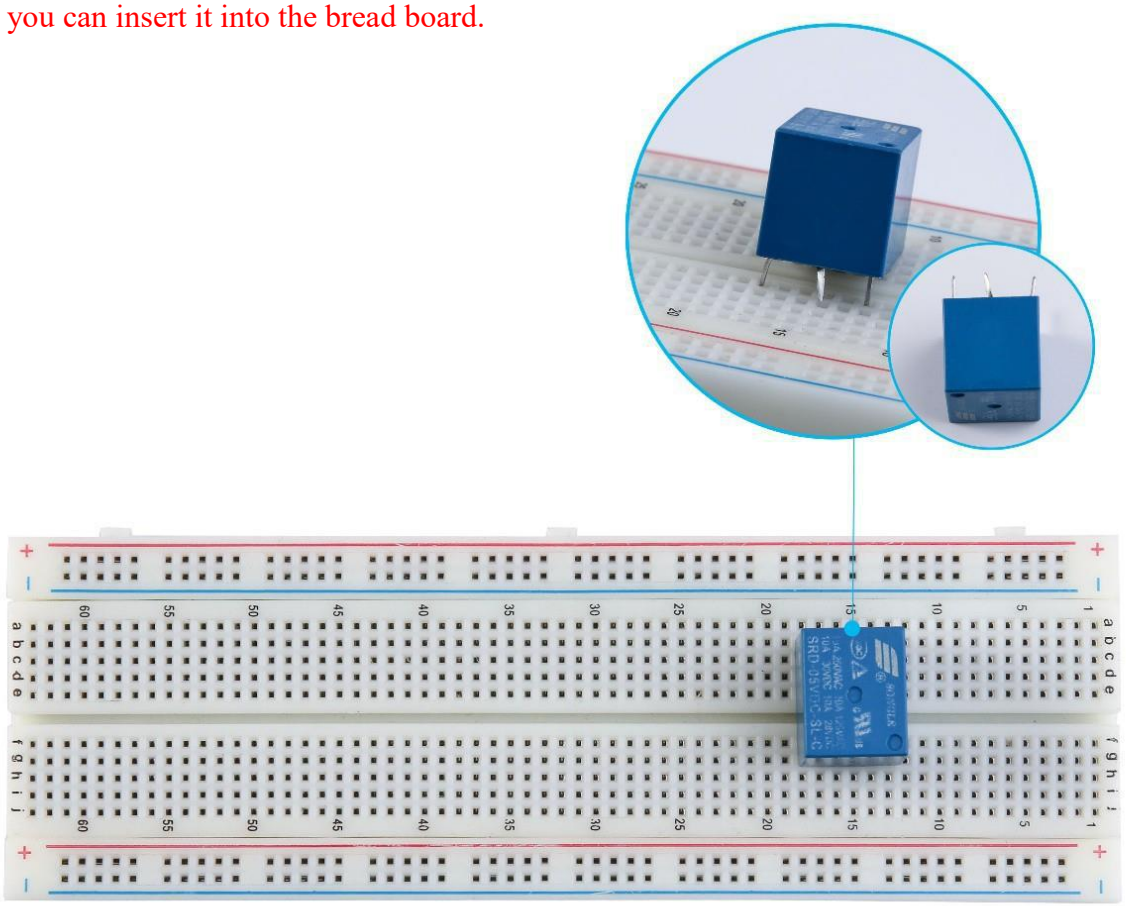
Relay:

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

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

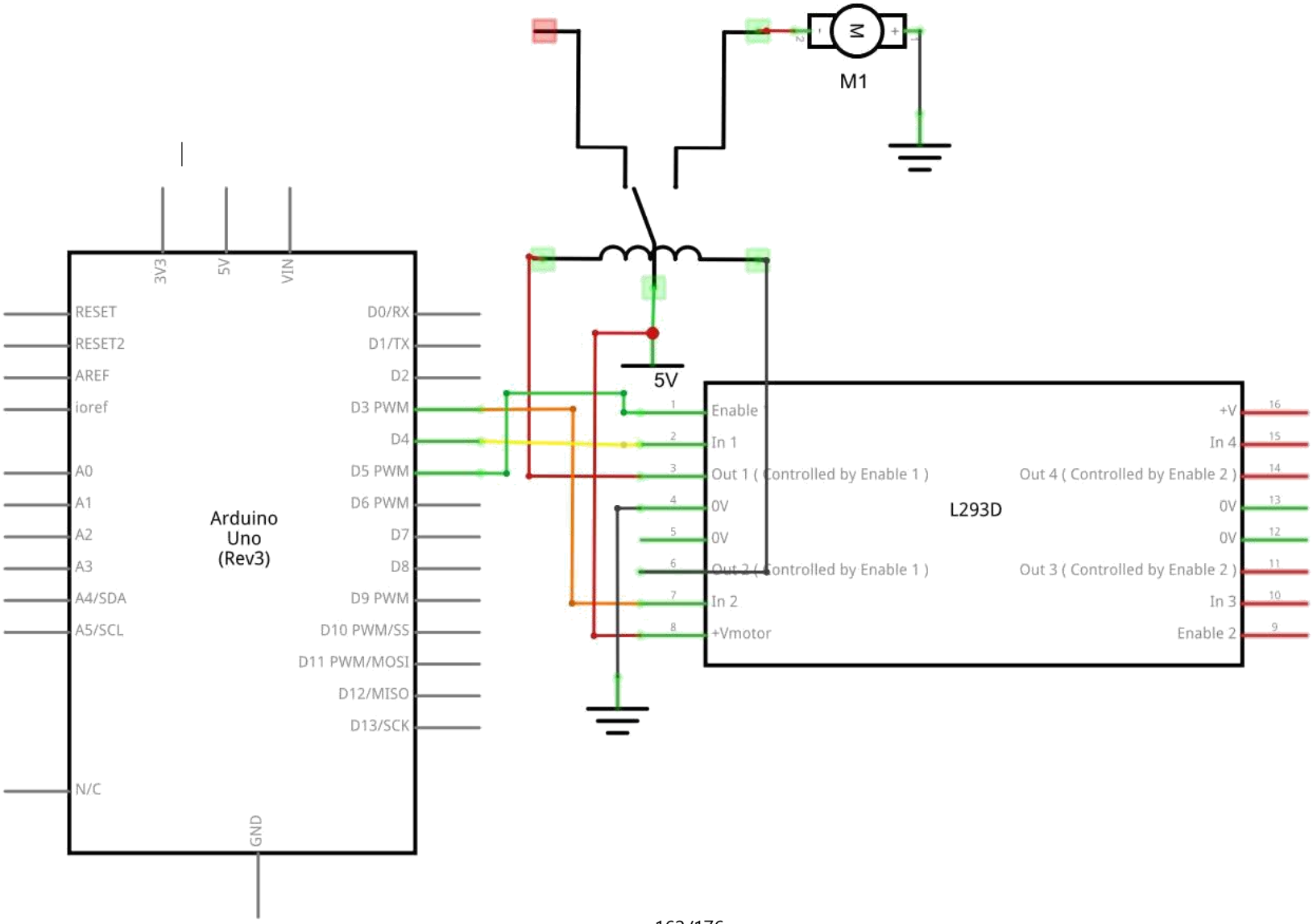
Below is the schematic of how to drive relay with Arduino.

You may be confused about how to insert the relay into the bread board. As the picture below shows, you will have to bend one of the pins of the relay slightly then you can insert it into the bread board.

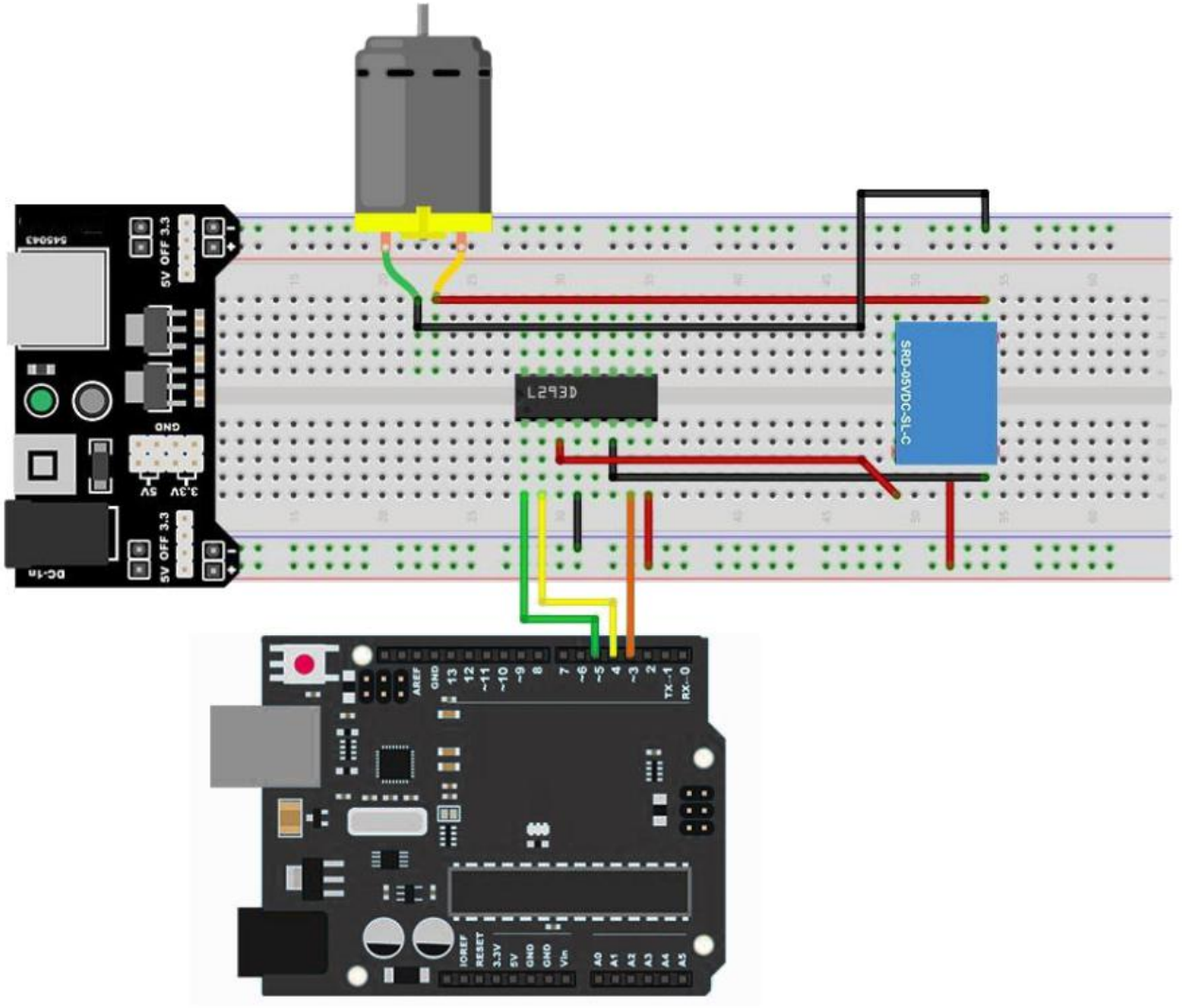


Connection

Schematic



Wiring diagram

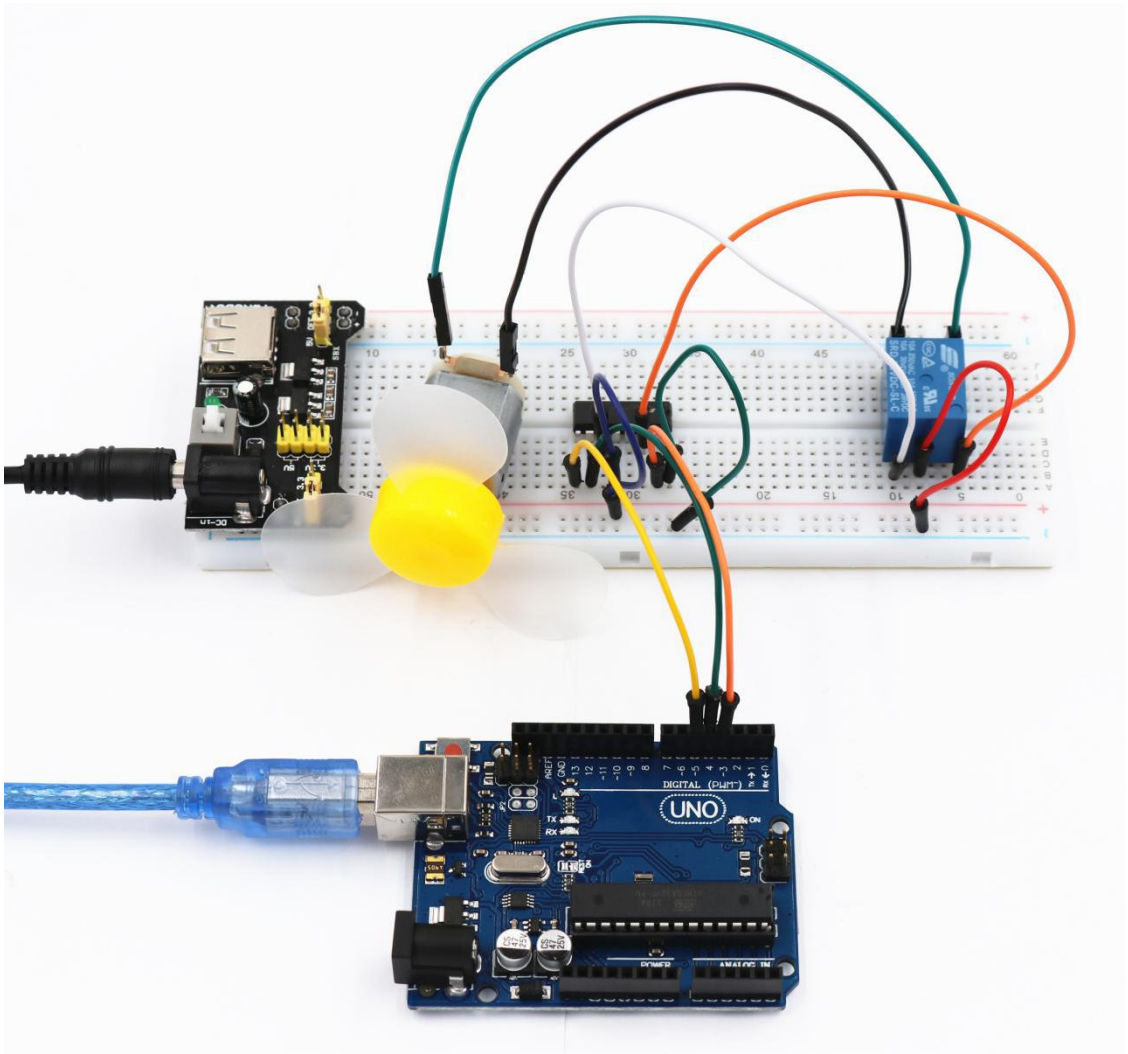


Code

After wiring, please open the program in the code folder- Lesson 25 Relay and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

After program loading, turn on all the power switches. The relay will pick up with a ringing sound. Then, the motor will rotate. After a period of time, the relay will be released, and the motor stops.

Example picture



Lesson 26 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 Uno R3 Board

(1) x 830 tie-points breadboard

(1) x ULN2003 stepper motor driver module

(1) x Stepper motor

(1) x 9V1A Adapter

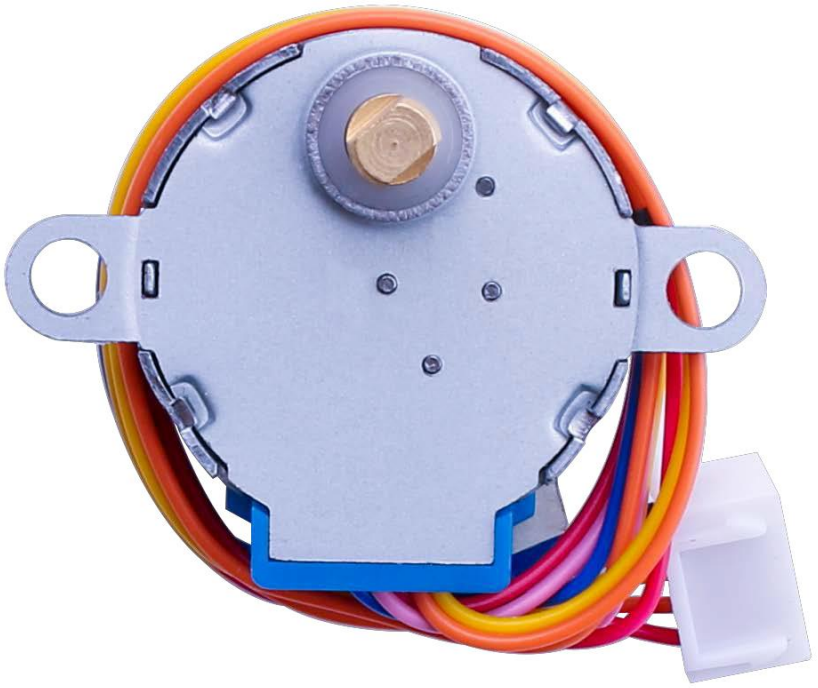
(1) x Power supply module

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

(1) x M-M wire (Male to Male jumper wire)

Component Introduction

Stepper Motor



A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.

Stepper motor 28BYJ-48 Parameters

Model: 28BYJ-48

Rated voltage: 5VDC

Number of Phase: 4

Speed Variation Ratio: 1/64

Stride Angle: 5.625° /64

Frequency: 100Hz

DC resistance: 50Ω±7%(25℃)

Idle In-traction Frequency: > 600Hz

Idle Out-traction Frequency: > 1000Hz

In-traction Torque >34.3mN.m(120Hz)

Self-positioning Torque >34.3mN.m

Friction torque: 600-1200 gf.cm

Pull in torque: 300 gf.cm

Insulated resistance >10MΩ(500V)

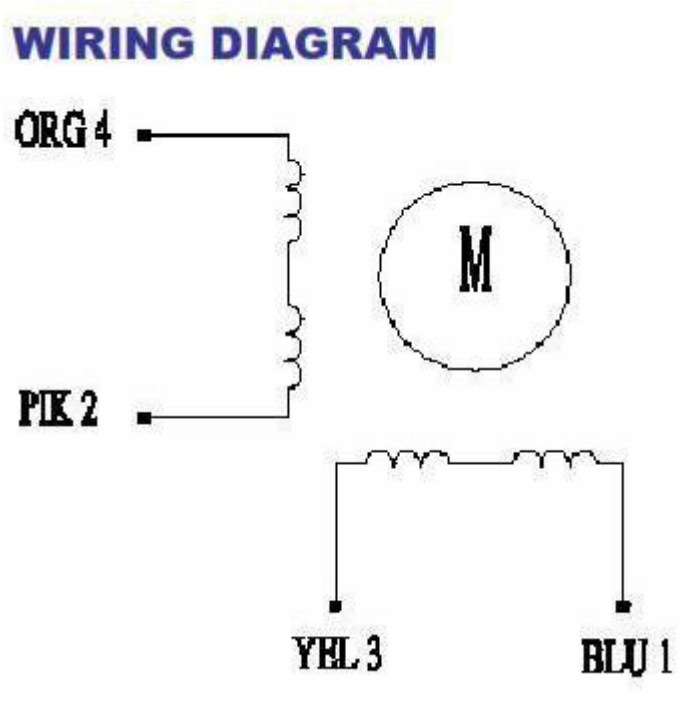
Insulated electricity power：600VAC/1mA/1s

Insulation grade：A

Rise in Temperature <40K(120Hz)

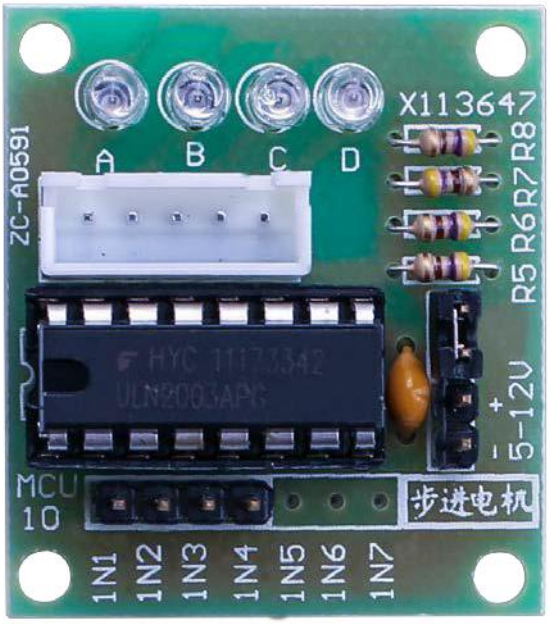
Noise <35dB(120Hz,No load,10cm)

Interfacing circuits



The bipolar stepper motor usually has four wires coming out of it. Unlike unipolar steppers, bipolar steppers have no common center connection. They have two independent sets of coils instead. You can distinguish them from unipolar steppers by measuring the resistance between the wires. You should find two pairs of wires with equal resistance. If you've got the leads of your meter connected to two wires that are not connected (i.e. not attached to the same coil), you should see infinite resistance (or no continuity).

ULN2003 Driver Board



Product Description

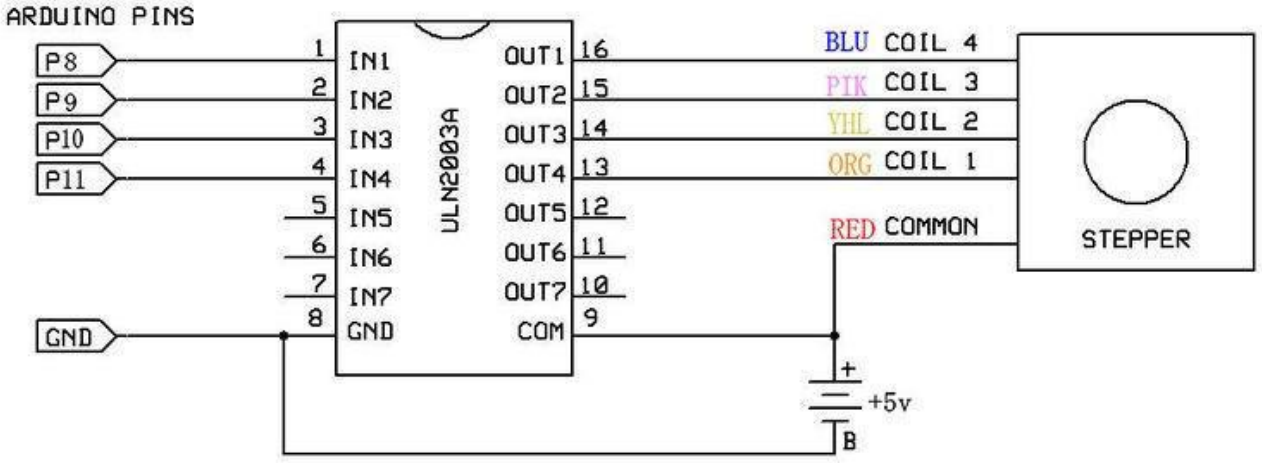
- Size: 42mmx30mm
- Use ULN2003 driver chip, 500mA
- A. B. C. D LED indicating the four phase stepper motor working condition.
- White jack is the four phase stepper motor standard jack.
- Power pins are separated
- We kept the rest pins of the ULN2003 chip for your further prototyping.

The simplest way of interfacing a unipolar stepper to Arduino is to use a breakout for ULN2003A transistor array chip. The ULN2003A contains seven Darlington transistor drivers and is somewhat like having seven TIP120 transistors all in one package. The ULN2003A can pass up to 500 mA per channel and has an internal voltage drop of about 1V when on. It also contains internal clamp diodes to dissipate voltage spikes when driving inductive loads. To control the stepper, apply voltage to each of the coils in a specific sequence.

The sequence would go like this:

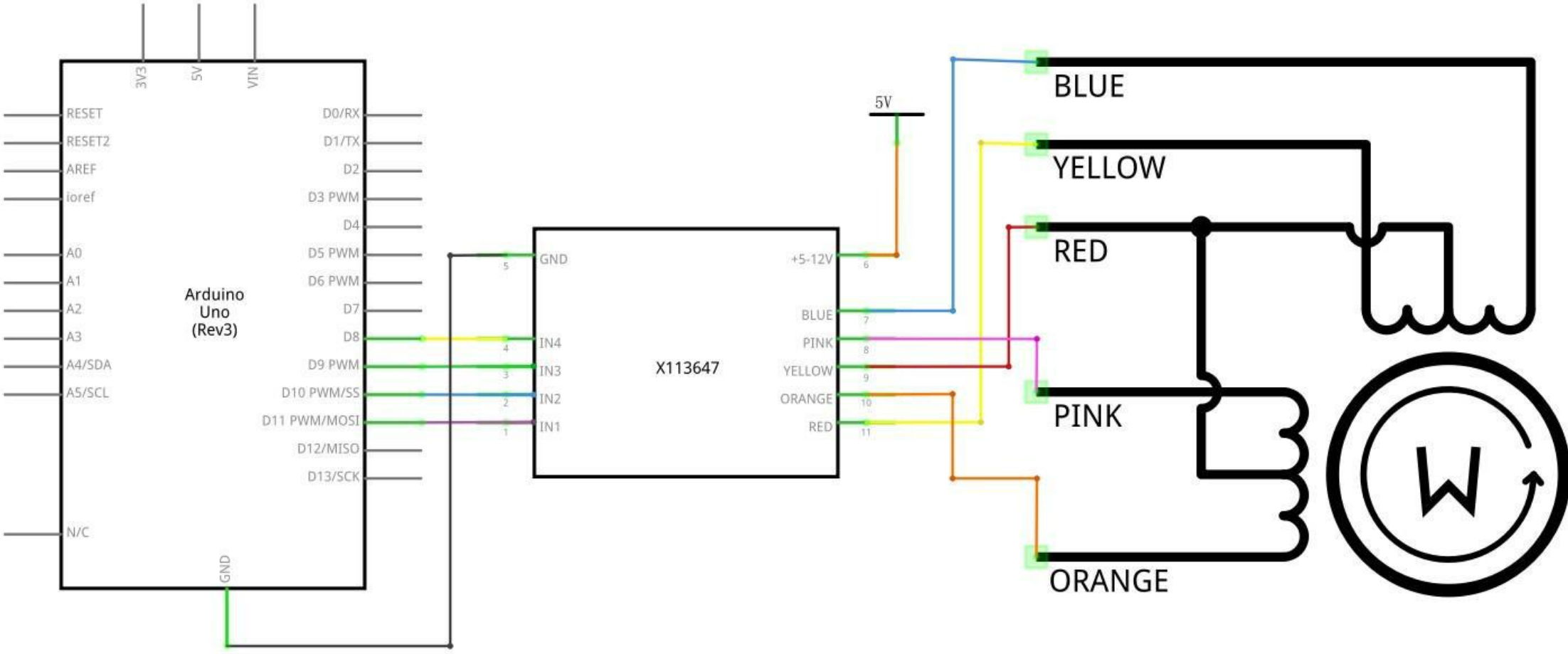
Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 ORG	-	-						-
3 YEL		-	-	-				
2 PIK				-	-	-		
1 BLU						-	-	-

Here are schematics showing how to interface a unipolar stepper motor to four controller pins using a ULN2003A, and showing how to interface using four com

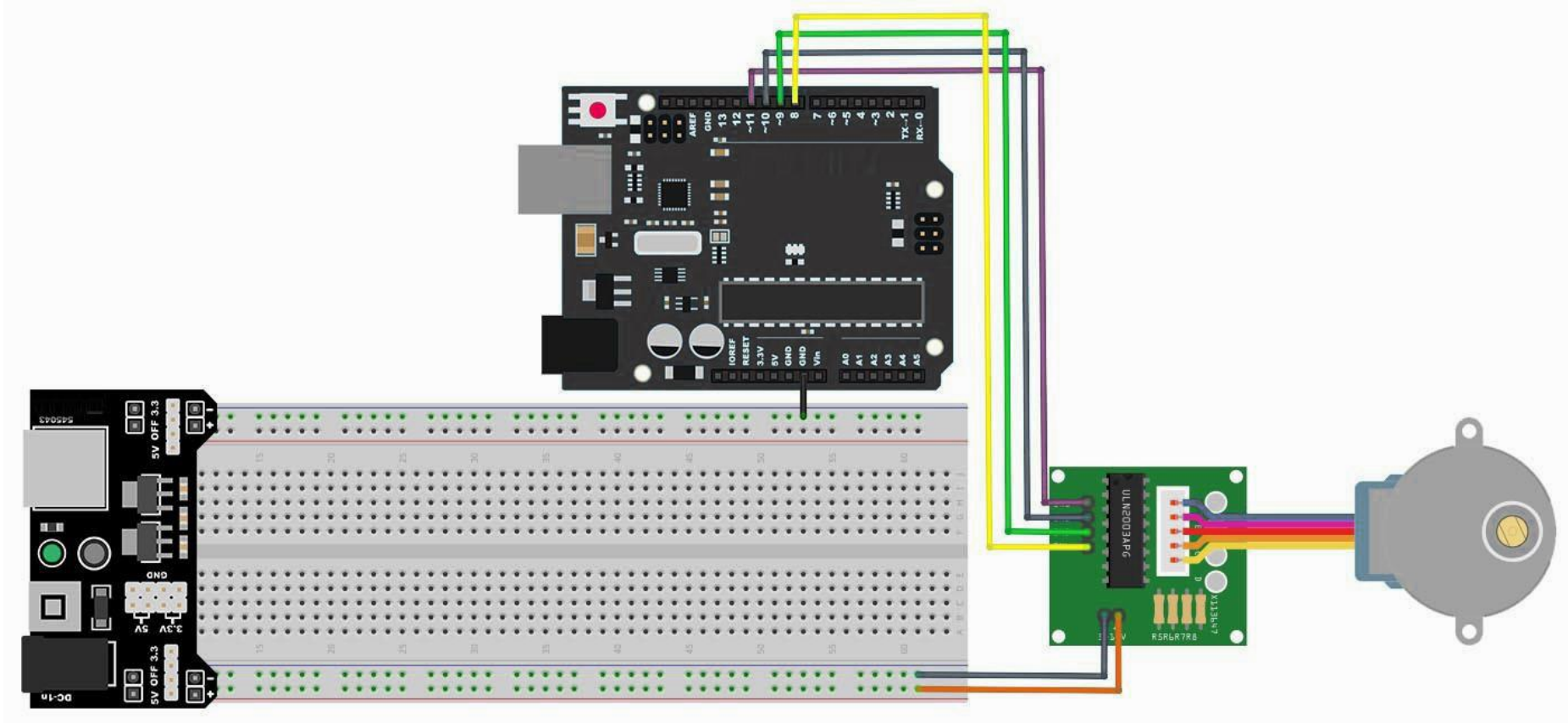


Connection

Schematic



Wiring diagram



We are using 4 pins to control the Stepper.

Pin 8-11 are controlling the Stepper motor.

We connect the Ground from to UNO to the Stepper motor.

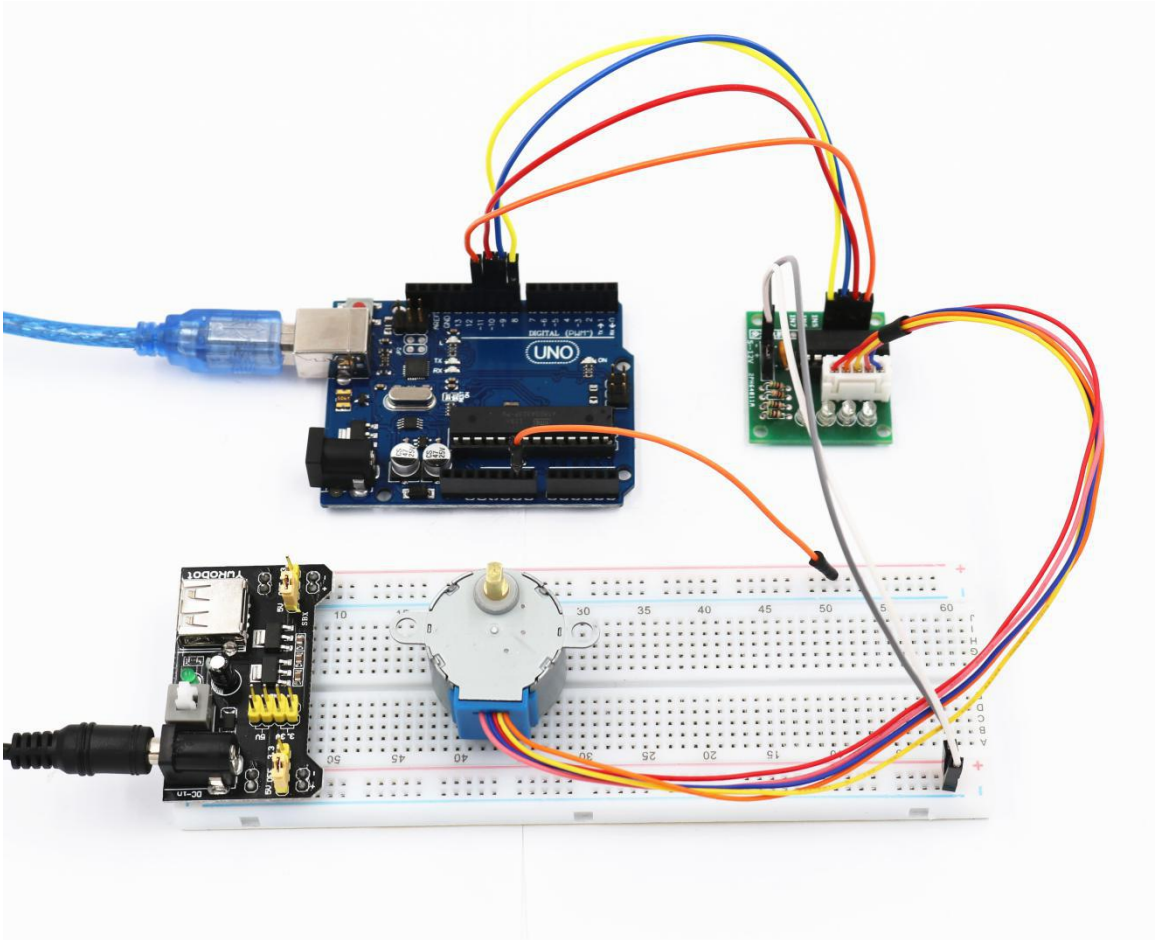
Code

After wiring, please open the program in the code folder- Lesson 26 Stepper Motor and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.

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

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

Example picture



Lesson 27 Controlling Stepper Motor With Remote

Overview

In this lesson, you will learn a fun and easy way to control a stepper motor from a distance using an IR remote control.

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

Since we don't want to drive the motor directly from the UNO, we will be using an inexpensive little breadboard power supply that plugs right into our breadboard and power it with a 9V 1Amp power supply.

The IR sensor is connected to the UNO directly since it uses almost no power.

Component Required:

(1) x Uno R3 Board

(1) x 830 tie-points breadboard

(1) x IR receiver module

(1) x IR remote

(1) x ULN2003 stepper motor driver module

(1) x Stepper motor

(1) x Power supply module

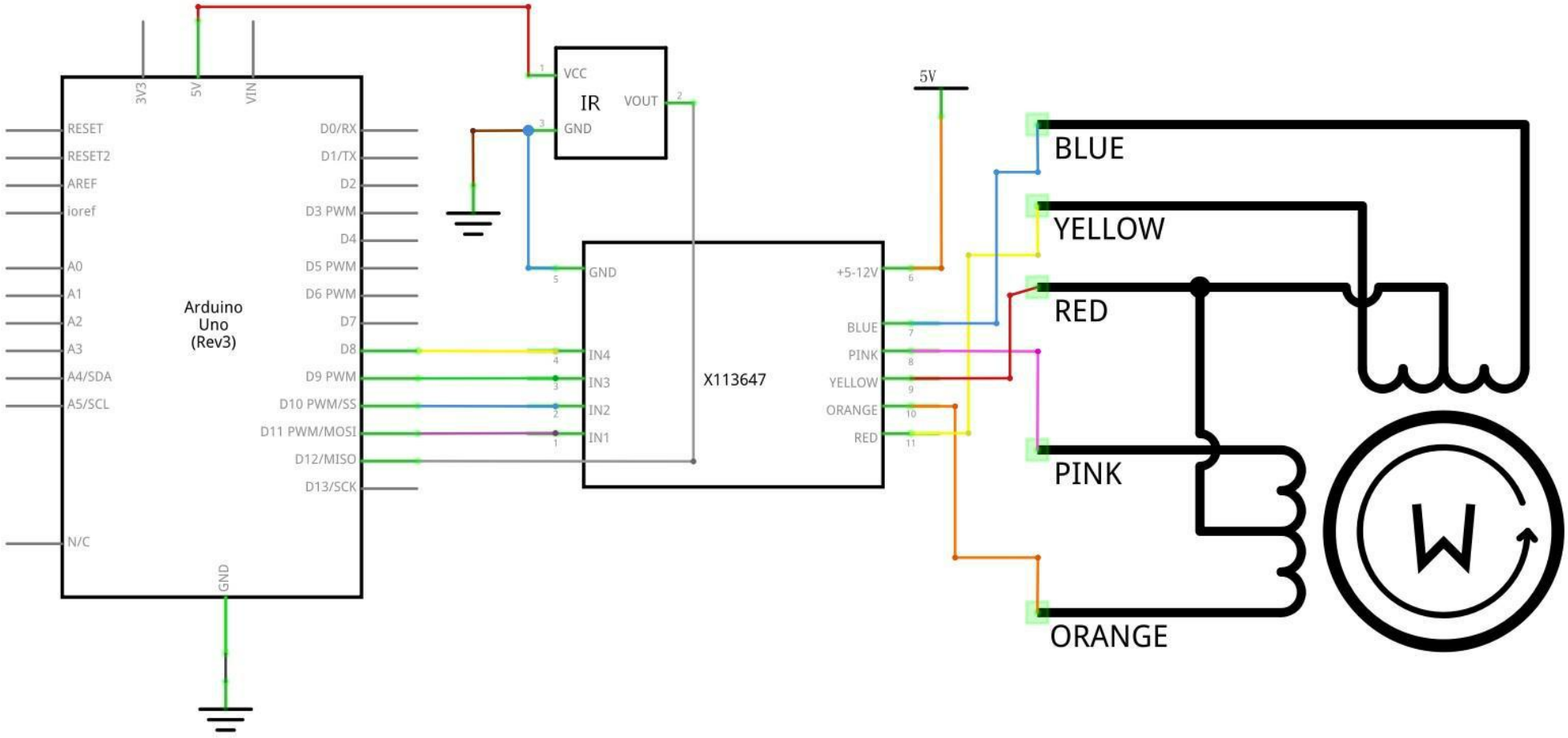
(1) x 9V1A Adapter

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

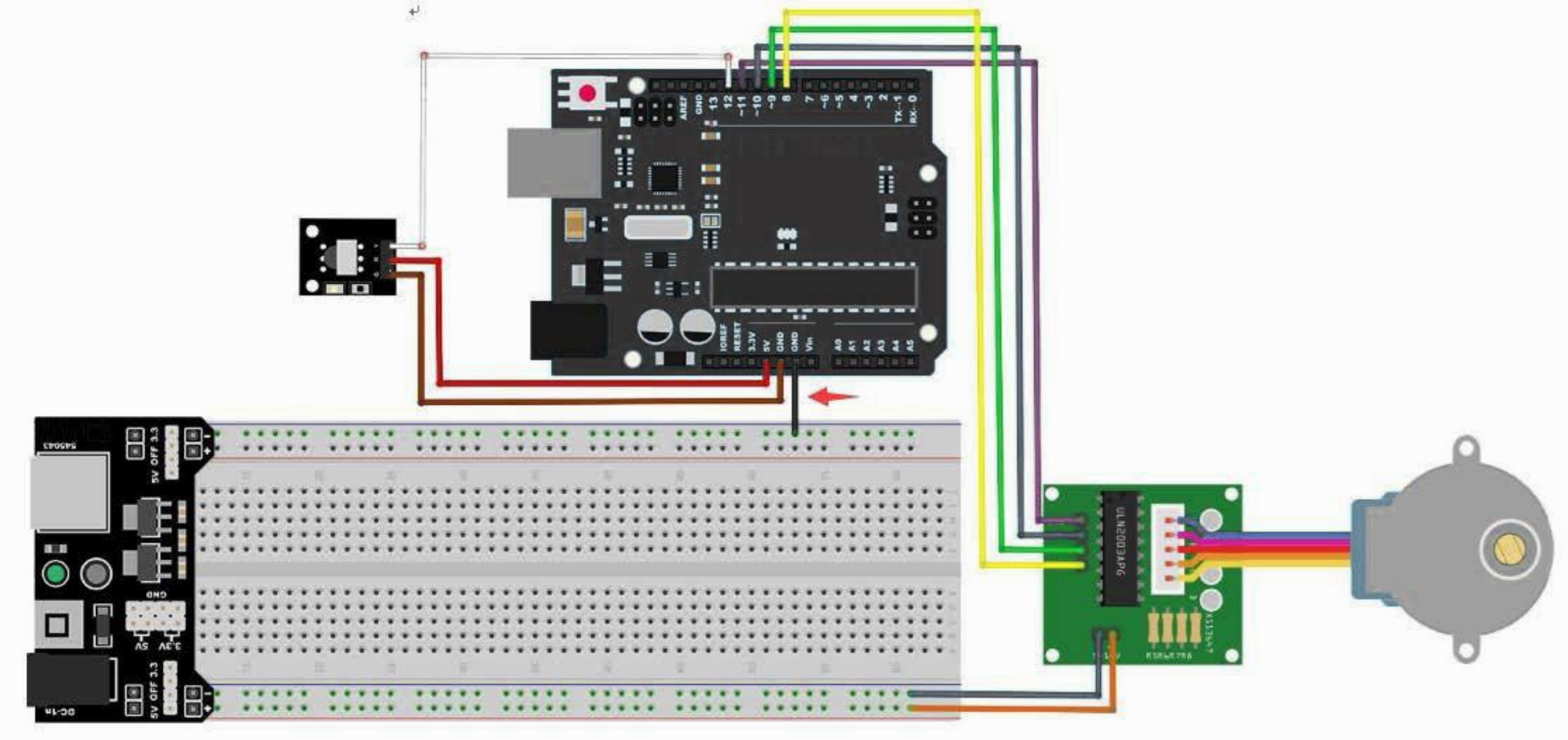
(1) x M-M wire (Male to Male jumper wire)

Connection

Schematic



Wiring diagram



We are using 4 pins to control the Stepper and 1 pin for the IR sensor. Pins 8-11 are controlling the Stepper motor and pin 12 is receiving the IR information.

We connect the 5V and Ground from the UNO to the sensor. As a precaution, use a breadboard power supply to power the stepper motor since it can use more power and we don't want to damage the power supply of the UNO.

Code

After wiring, please open program in the code folder- Lesson 27 Controlling Stepper Motor With Remote and click UPLOAD to upload the program. See Lesson 2 for details about program uploading if there are any errors.
Before you can run this, make sure that you have installed the < IRremote >

< Stepper >library or re-install it, if necessary. Otherwise, your code won't work.
For details about loading the library file, see Lesson 1.

The code only recognize 2 values from the IR Remote control: VOL+ and VOL-.
When VOL+ is pressed on the remote the motor will make a full rotation clockwise.
VOL- will make a full rotation counter-clockwise.

Example picture

