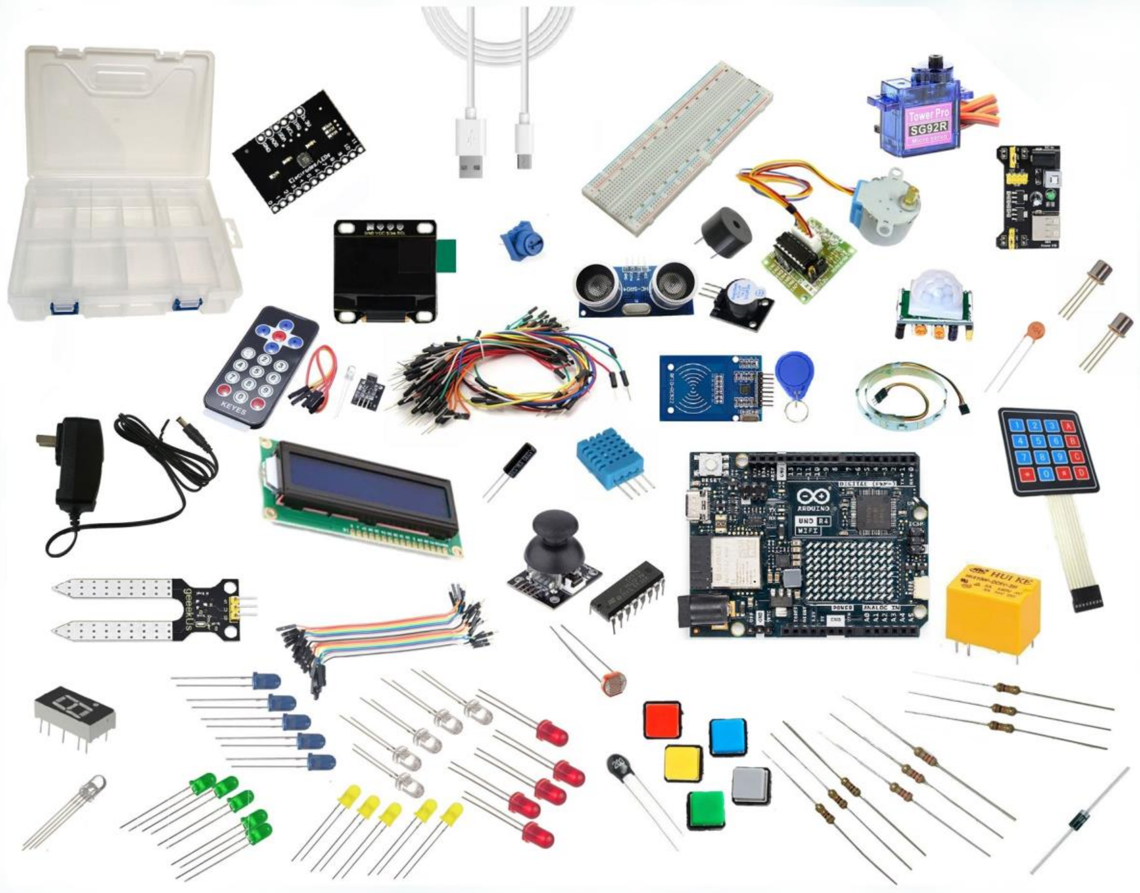
ABRA ARDUINO UNO R4 KIT

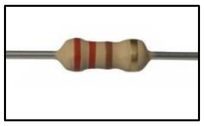
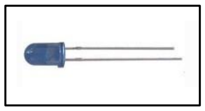
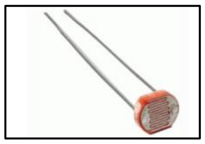
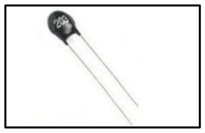
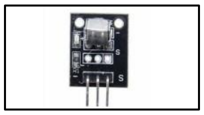
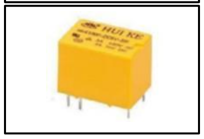
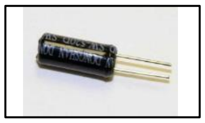
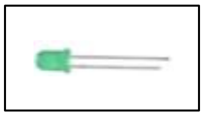
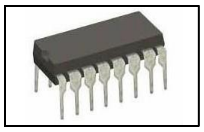
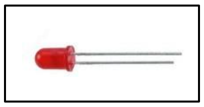
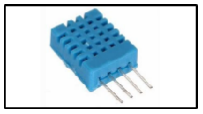
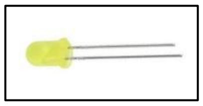
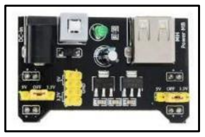
Includes original Arduino UNO R4 WiFi

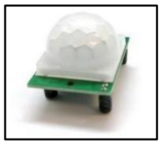


ARD-UNO-KIT-R4-WIFI

www.abra-electronics.com

Parts List

	10x 220Ω Resistor		5x White LED
	10x 1kΩ Resistor		5x Blue LED
	3x 10kΩ Resistor		1x RGB LED
	1x 1N4007 Diode		5x Push Button Switches
	1x 2N2222A NPN Transistor		1x 104pF Disc Capacitor
	1x 2N2907 PNP Transistor		1x Photoresistor
	1x 10kW Potentiometer		1x 10kΩ NTC Thermistor
	1x Active Buzzer		1x IR Receiver Module
	1x Passive Buzzer		1x Remote Control
	1x 5VDC 3A Relay		1x SW-520D Tilt Switch
	5x Green LED		1x 74HC595
	5x Red LED		1x DHT11 Humidity Sensor
	5x Yellow LED		1x Power Supply Module


1x PIR Motion Sensor

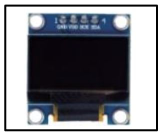
1x Joystick Module

1x HC-SR04P Ultrasonic Sensor

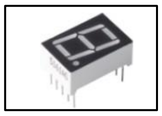
1x MFRC522 RFID Module

1x I2C LCD 1602

1x Arduino R4 Wi-Fi


1x OLED Screen

1x MPR121 Touch Sensor

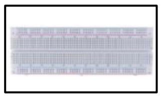

1x 7-Segment Display

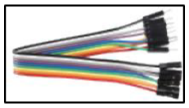
1x Soil Moisture Module Sensor

1x Stepper Motor & ULN2003 Driver

1x 9G Micro Servo

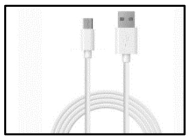
1x 4x4 Keypad


1x Breadboard


1x Set of 20 M-F Wires

1x WS2812 RGB Strip

1x Set of 65 Male Wires

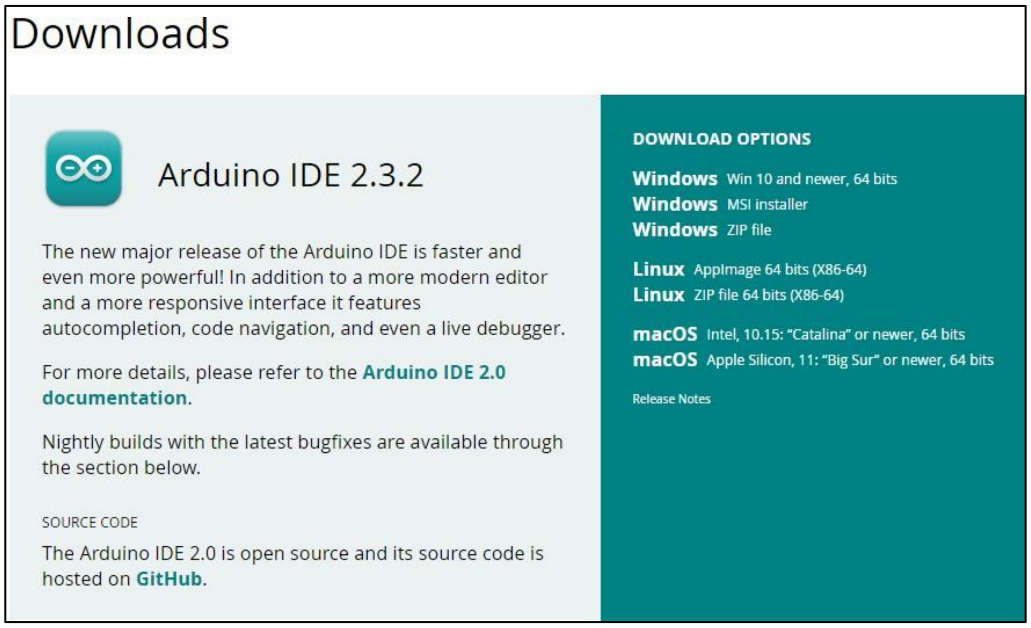

1x Type-C USB 2.0 Cable

1x 9V Power Adapter

Installing Arduino IDE

Introduction

Simply put, the Arduino Integrated Development Environment (IDE) is an application that provides a simple way to write and upload code to Arduino devices. It's free and available for Windows, Mac and Linux. The installer is accessible on the Arduino website.



Downloads

Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

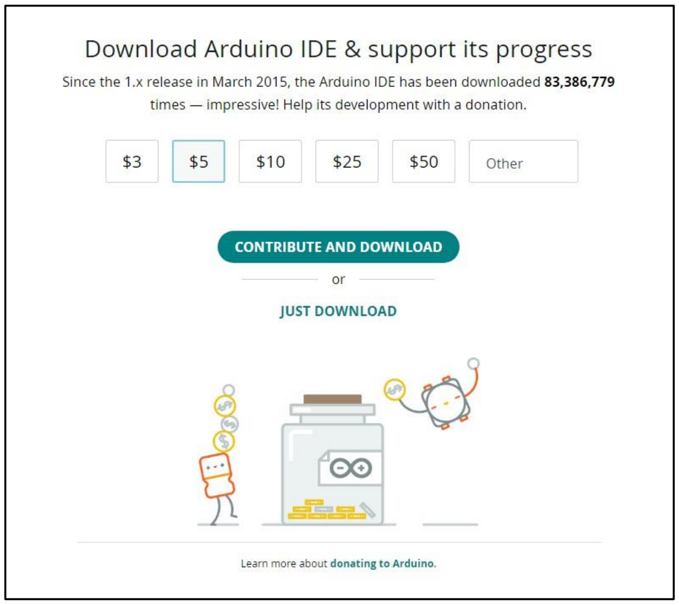
Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

As of the time of this manual's existence, version 2.3.2 of the Arduino IDE will be installed. Select the download option compatible with your system. The following tutorial will be for selecting the first Windows option "Win 10 and newer, 64 bits".

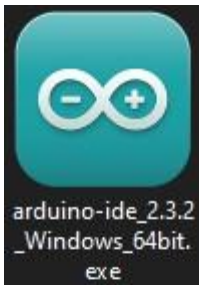


Download Arduino IDE & support its progress

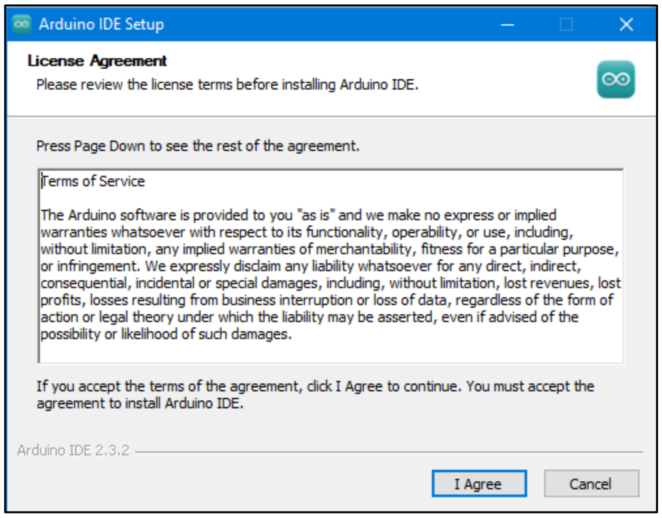
Since the 1.x release in March 2015, the Arduino IDE has been downloaded **83,386,779** times — impressive! Help its development with a donation.

Learn more about **donating to Arduino.**

Proceed to free download by clicking “JUST DOWNLOAD”. If prompted to join newsletter, click “JUST DOWNLOAD” again.



Finally, open the Arduino IDE setup program. It should share a similar name as above.



License Agreement
Please review the license terms before installing Arduino IDE.

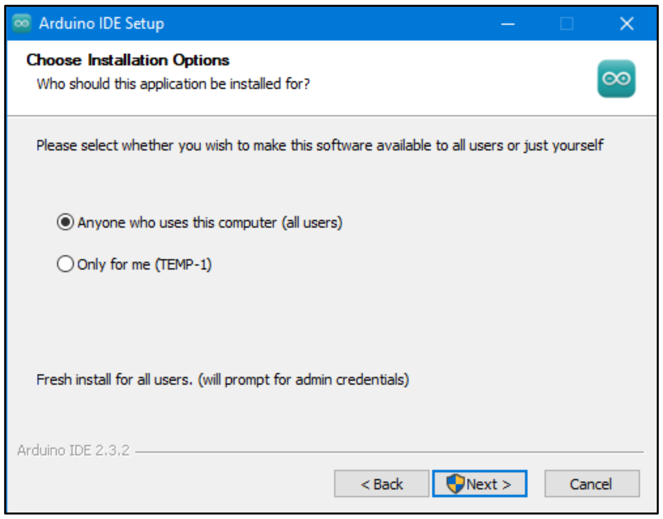
Press Page Down to see the rest of the agreement.

Terms of Service

The Arduino software is provided to you "as is" and we make no express or implied warranties whatsoever with respect to its functionality, operability, or use, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or infringement. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility or likelihood of such damages.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Arduino IDE.

Arduino IDE 2.3.2



Choose Installation Options
Who should this application be installed for?

Please select whether you wish to make this software available to all users or just yourself

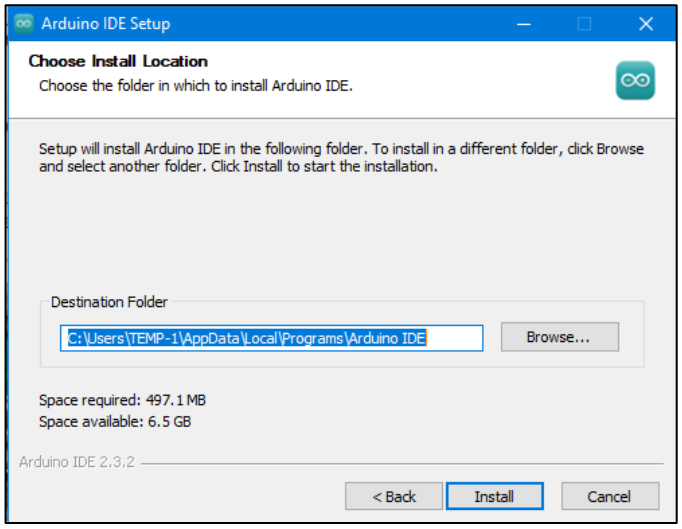
[✓ Anyone who uses this computer (all users)]

Only for me (TEMP-1)

Fresh install for all users. (will prompt for admin credentials)

Arduino IDE 2.3.2

You will be displayed the License Agreement. Upon clicking “I Agree”, the installation options are followed. Choose which is best suited for you. If you do not know, “all users” is fine.



Choose Install Location
Choose the folder in which to install Arduino IDE.

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

Destination Folder
C:\Users\TEMP-1\AppData\Local\Programs\Arduino IDE

Space required: 497.1 MB
Space available: 6.5 GB

Arduino IDE 2.3.2

You may change the destination folder of the installation by clicking “Browse…” if you wish. Click “Install”. After it’s installed on your computer, you may launch it.

Serial Monitor & Installing Libraries

Installation of Arduino Libraries

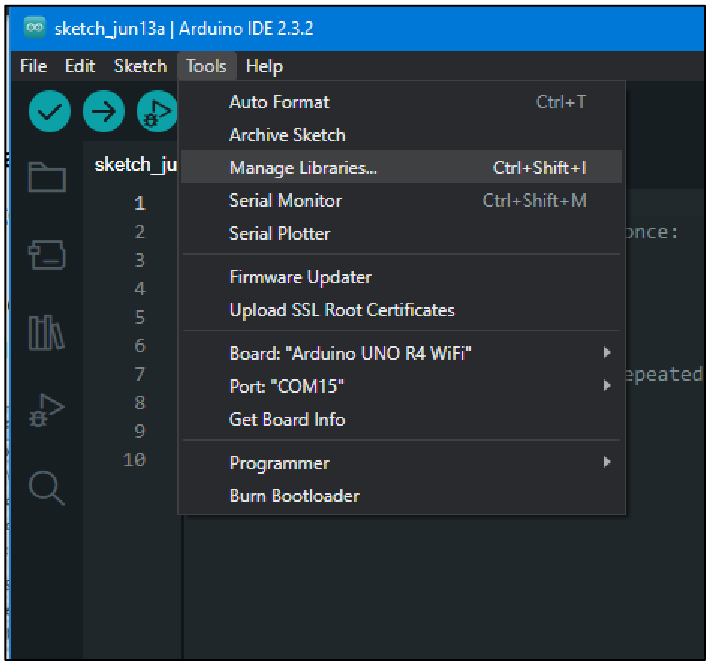
As you're settling in with the program and familiarize yourself with the built-in functions, it's a good idea to learn how to import libraries that are not available in the library manager search.

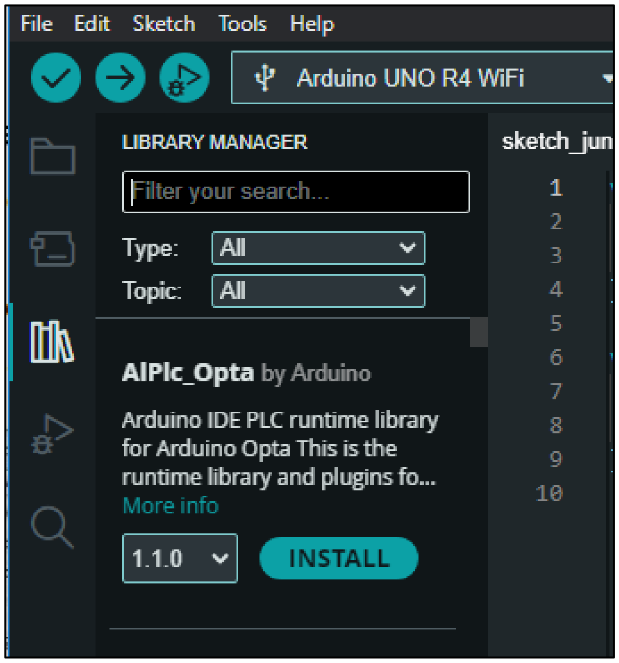
What Are Libraries?

In this context, a library is a collection of files containing code that is responsible for configuring its designated devices to work with your Arduino. For example, importing the LiquidCrystal library allows the user to program the LCD Display to show text they desire through the Arduino.

How to Install a Library Using Library Manager…

Using the Library Manager, you can install most used libraries from there. On the top of the window, click "Tools" and select "Manage Libraries". Alternatively, you could also click on the books and it will show the Library Manager.

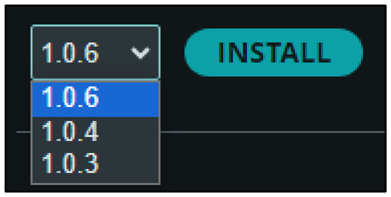

sketch_jun13a | Arduino IDE 2.3.2
File
Edit
Sketch
Tools
Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino UNO R4 WiFi"
Port: "COM15"
Get Board Info
Programmer
Burn Bootloader
sketch_ju



AlPlc_Opta by Arduino

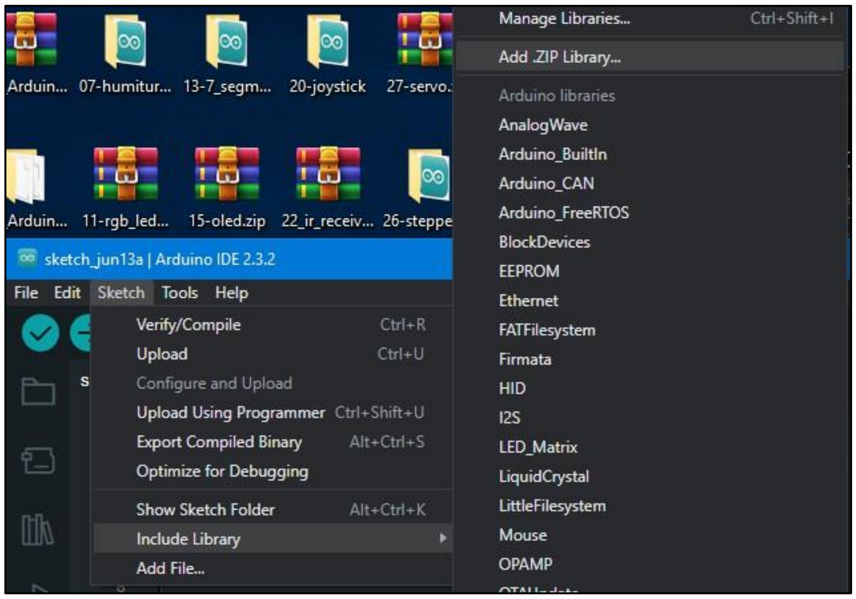
Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo...

From this point, it's advisable to always install the latest version of the desired library (unless specified otherwise by library creator) and you're set to work with it. The first option on the dropdown menu will typically be the latest version of the library.



How to Install a Library through Importing a ZIP File…

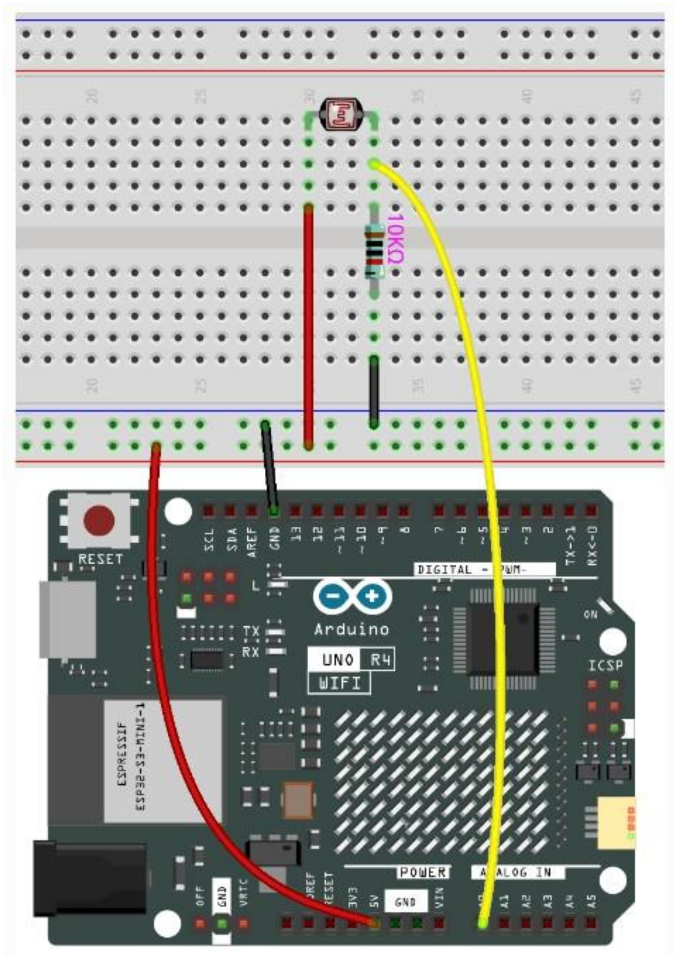
Importing a library in a ZIP file is simple. On top of the screen, select "Sketch", then "Include Library", then "Add .ZIP Library…"



You'll be prompted to select the ZIP file. Once you find it, select it, click "Open" and the library will proceed with its installation process. When it's done, it's ready for use.

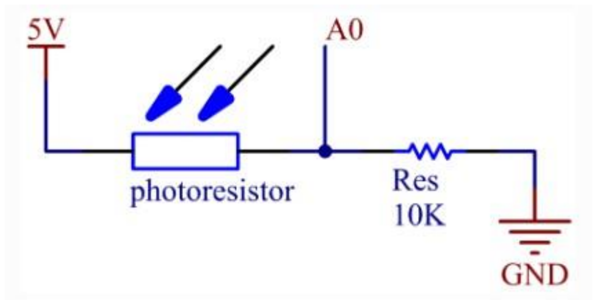
Also, you can open the serial monitor when you need it by clicking "Tools" and "Serial Monitor". The serial monitor displays the output of code when there is any.

Project 1: Photoresistor

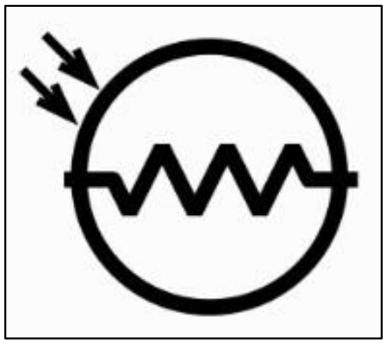


Required Components:

Arduino R4	1
Breadboard	1
10kΩ Resistor	1
Photoresistor	1



This project uses the 01-photoresistor.ino sketch.

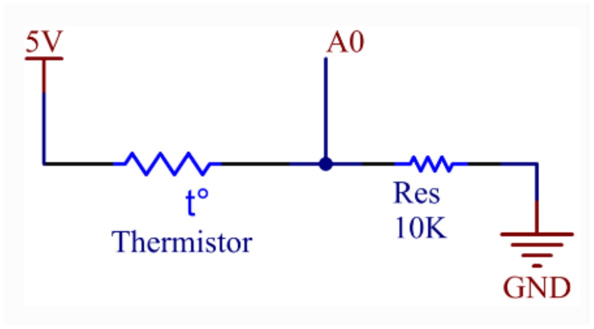


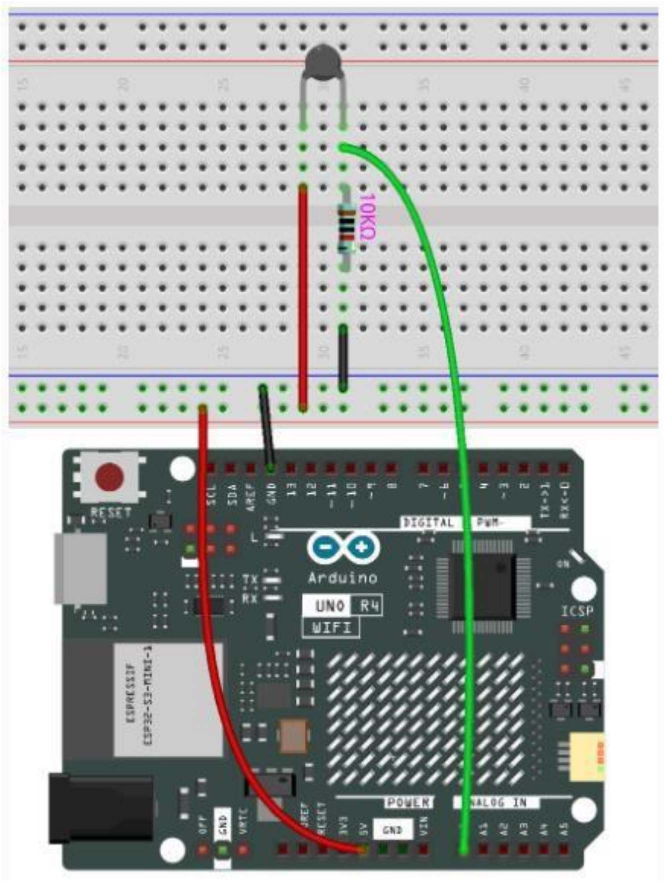
A photoresistor is a variable resistor controlled by light. Its resistance changes with the brightness it is surrounded by, ergo its resistance decreases with higher light intensity and vice versa. The actual resistance value varies on the photoresistor.

In this basic project, the Photoresistor is connected to a 5V pin of the Arduino to supply power. Pin A0 is connected to the photoresistor's other pin to read its value. It is also grounded with a 10kΩ resistor.

After uploading code to your Arduino when you set up your breadboard design, open the Serial Monitor and you'll see the reading range from 0 to 1023. The value will change with the lighting cast on the photoresistor.

Project 2: Thermistor

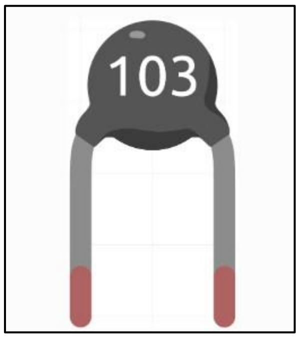


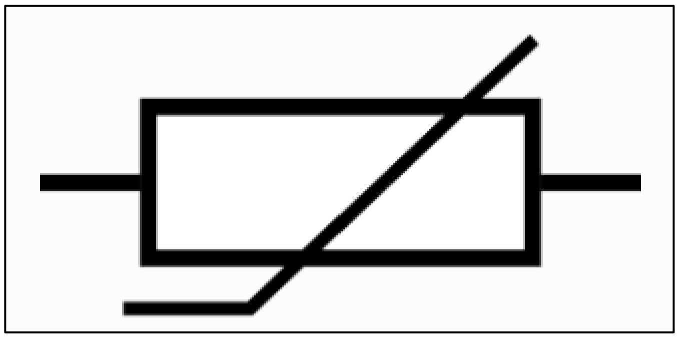


<u>Required Components:</u>

Arduino R4	1
Breadboard	1
10kΩ Resistor	1
Thermistor	1

This project uses the 02-thermistor.ino sketch.





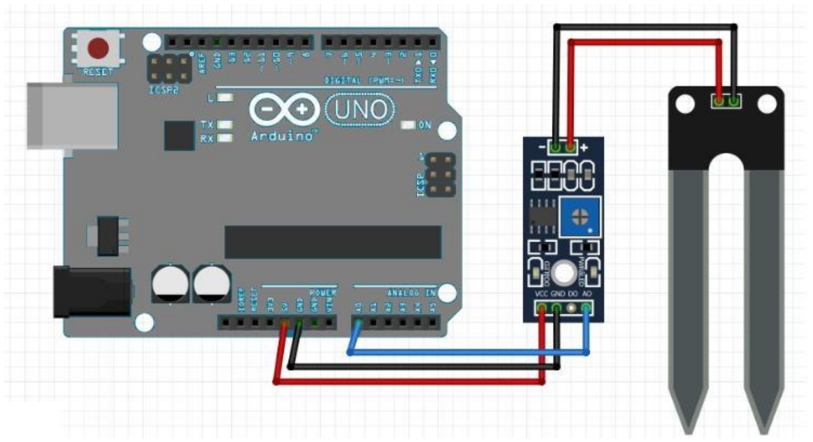
A thermistor is a thermal resistor in which its resistance is heavily dependent on temperature.

There are two kinds of thermistors: PTC (Positive Temperature Coefficient) and NTC (Negative Temperature Coefficient. PTC thermistors' resistance <u>increases</u> with the increase in temperature and are commonly used as "fuses" to protect overcurrent. NTC thermistors' resistance <u>decreases</u> with increase in temperature and are mostly used in temperature sensing and inrush current limiting.

Considering this NTC thermistor is 10kΩ and measured under 25 degrees Celsius, the relation between resistance and temperature can be calculated in the following formula:
RT = RN * expB((1/TK) – (1TN))
RT is total resistance of the thermistor when the temperature is TK.
RN is resistance of the given thermistor, in this case being 10k.
TK is a Kelvin temperature at which its resistance is being calculated, so degrees + 273.15.
TN is the given rated temperature but in Kelvin, so 25 + 273.15.
B is the material-specific constant of 3950.
The relation is an empirical formula where only accurate when the temperature and the resistance are within the effective range.

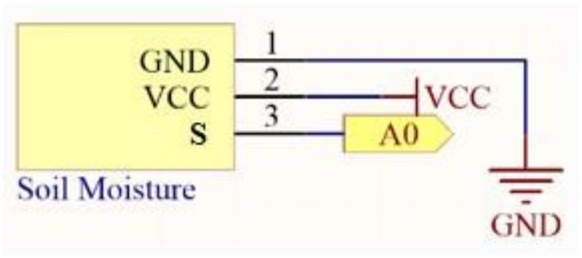
The above calculation solves for the Kelvin temperature, but if you open the serial monitor when you run the code, the output will just display the temperature in Celsius and Fahrenheit.

Project 3: Soil Moisture



Required Components:

Arduino R4	1
Soil Moisture Module Sensor	1

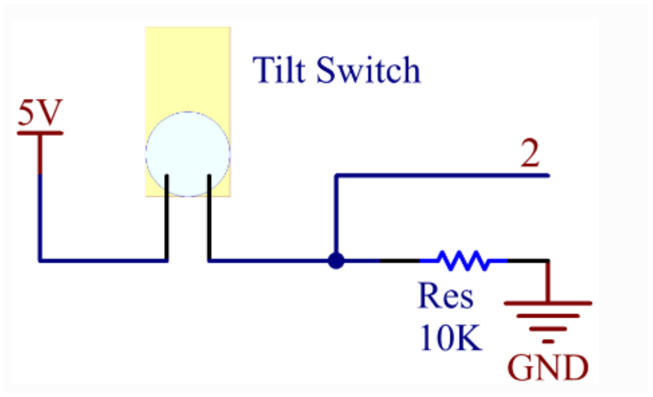


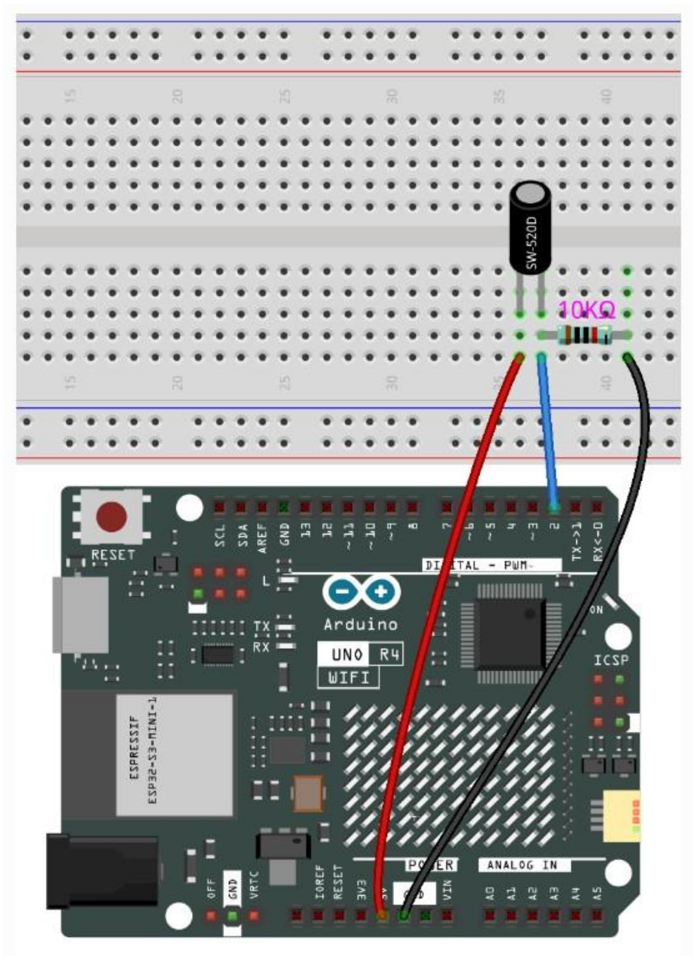
This project uses the 03-moisture.ino sketch.

A soil moisture sensor is a device that measures the water content in soil. It detects the volumetric water content based on changes in the soil. These sensors typically consist of two probes that are inserted into the soil. The data collected by the sensor helps in determining the moisture level, which is crucial for efficient irrigation management, preventing overwatering or underwatering of crops, and optimizing plant growth. They are widely used in agriculture, gardening, and environmental monitoring to ensure the right amount of water is supplied to plants.

Upon uploading the code, the serial monitor will display the soil moisture value. Dip the module into soil, water it, and the value will decrease.

Project 4: Tilt Switch



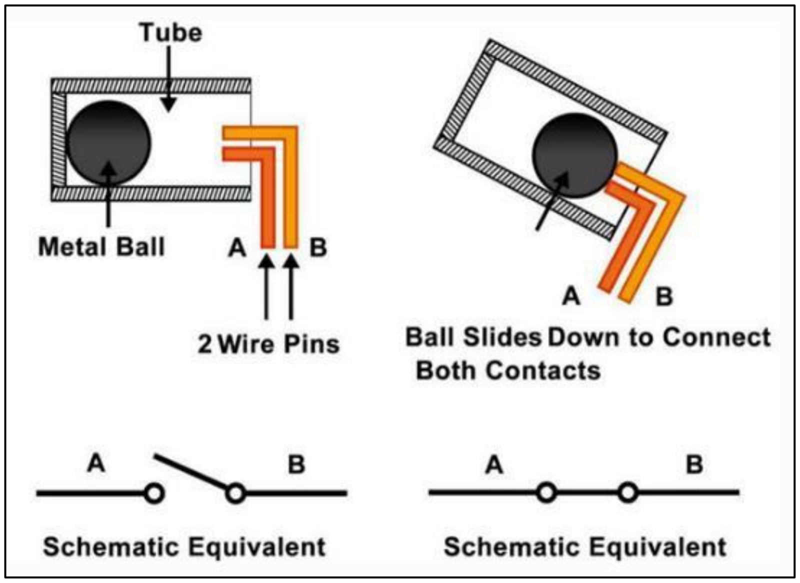


Required Components:

Arduino R4	1
Breadboard	1
10kΩ Resistor	1
Tilt Switch	1

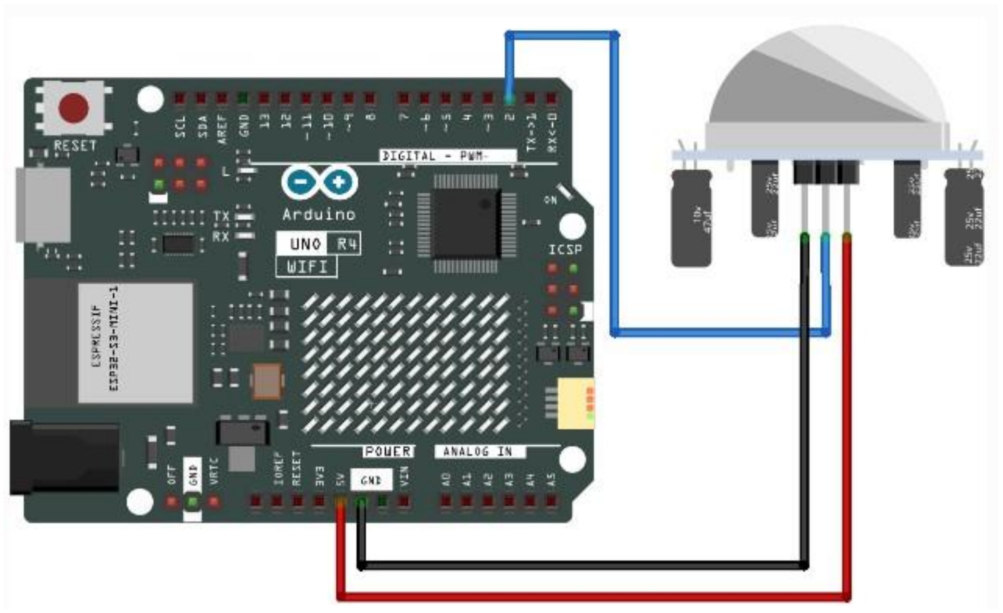
This project uses the 04-tilt_switch.ino sketch.

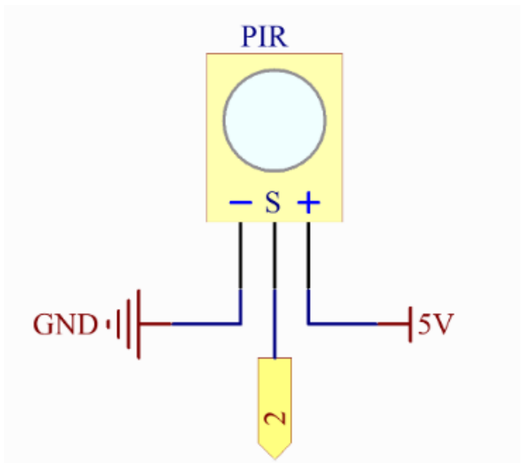
Tilt switches have a metal ball inside of them that is used to connect two pins and serve as a short across the wire. Its functionality is simple: when tilted at a certain angle, the metal ball performs the short, and when tilted the other way, opens the wire. Below is a diagram to show the inside of the switch:



The code will display either a “1” or a “0” on the serial monitor depending on the angle of the tilt switch, whether the ball connects the pins or not.

Project 5: PIR Motion Sensor Module

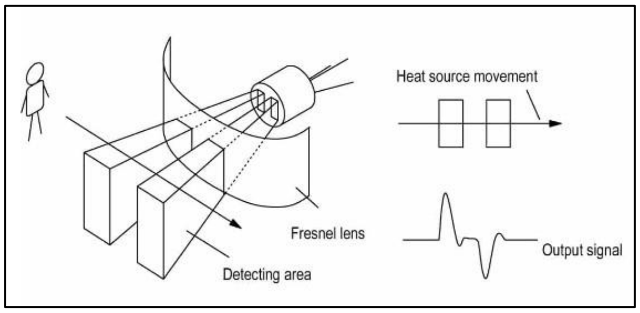




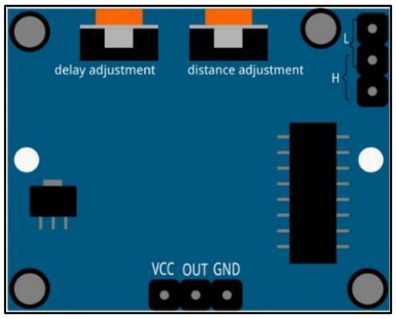
Required Components:

Arduino R4	1
PIR Motion Sensor	1

This project uses the 05-pir_motion_sensor.ino sketch.



The PIR Motion Sensor detects the presence of organisms through infrared heat radiation that they emit. Its method of detection involves both sensor's slots that are connected to a differential amplifier. For instance, when an object is stationary in front of the sensor, both slots tend to receive the same amount of radiation and outputs zero. When an object moves across the sensor's field of view, one slot receives more radiation than the other and the output fluctuates between high and low. This way, the change in output voltage is caused by the motion detection.



Delay Adjustment

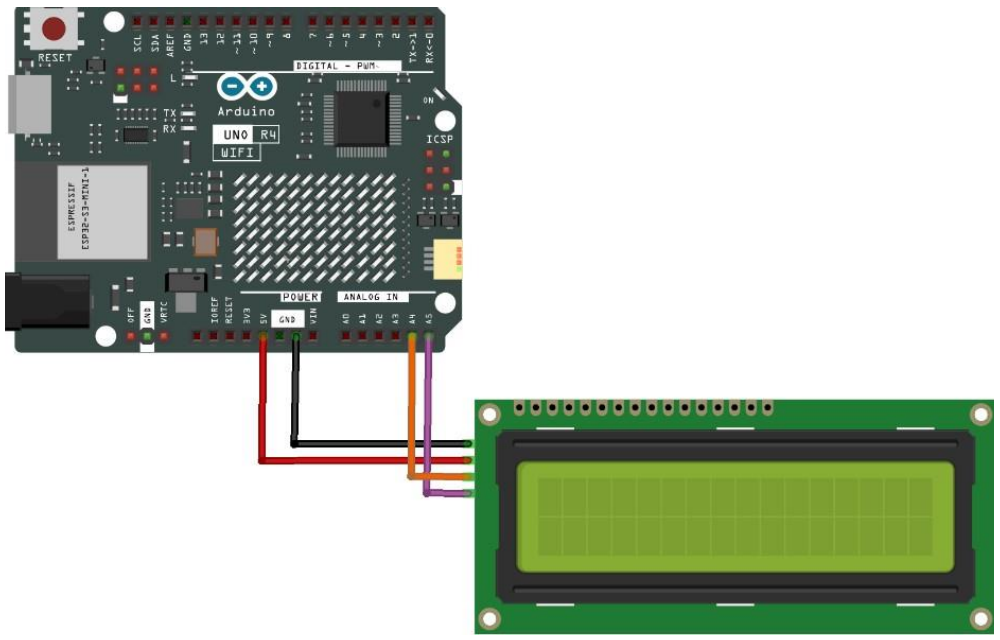
The sensing delay increases up to a maximum 300 seconds upon turning the knob clockwise and decreases down to a minimum of 5 seconds turning the knob counterclockwise.

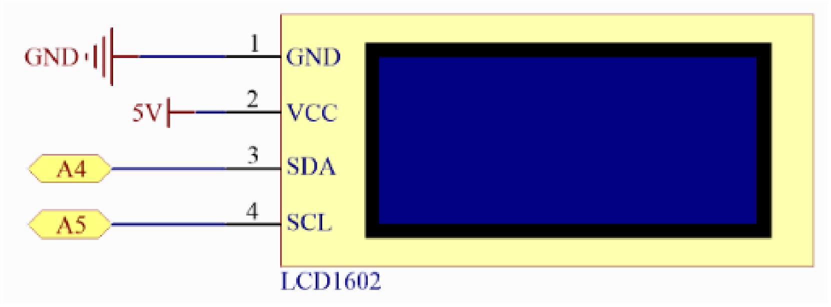
Distance Adjustment

The sensor's range increases up to a maximum of 7 meters upon turning the knob clockwise and decreases down to a minimum of 3 meters turning the knob counterclockwise.

Upon uploading the code to the board, open the serial monitor to observe the sensor's output. "Motion detected!" will display when the module sensor is triggered by motion, and "Monitoring…" will display when there is no motion. The sensor's sensitivity and behavior are dependent on its characteristics and surrounding environment. It's suggested to calibrate the sensor to your preference for best results.

Project 6: I2C LCD1602

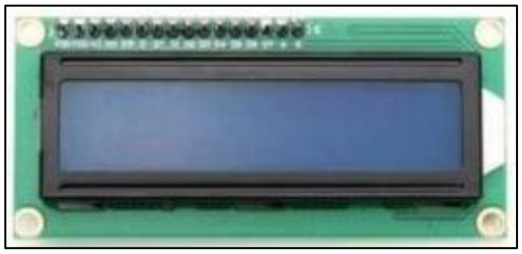


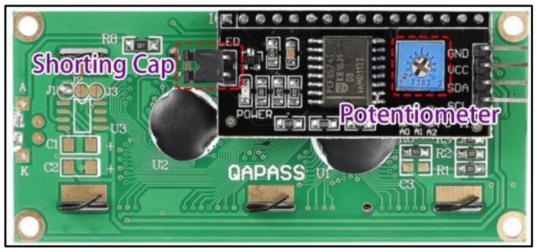


Required Components:

Arduino R4	1
I2C LCD1602	1

This project uses the 06-i2c_lcd.ino sketch.





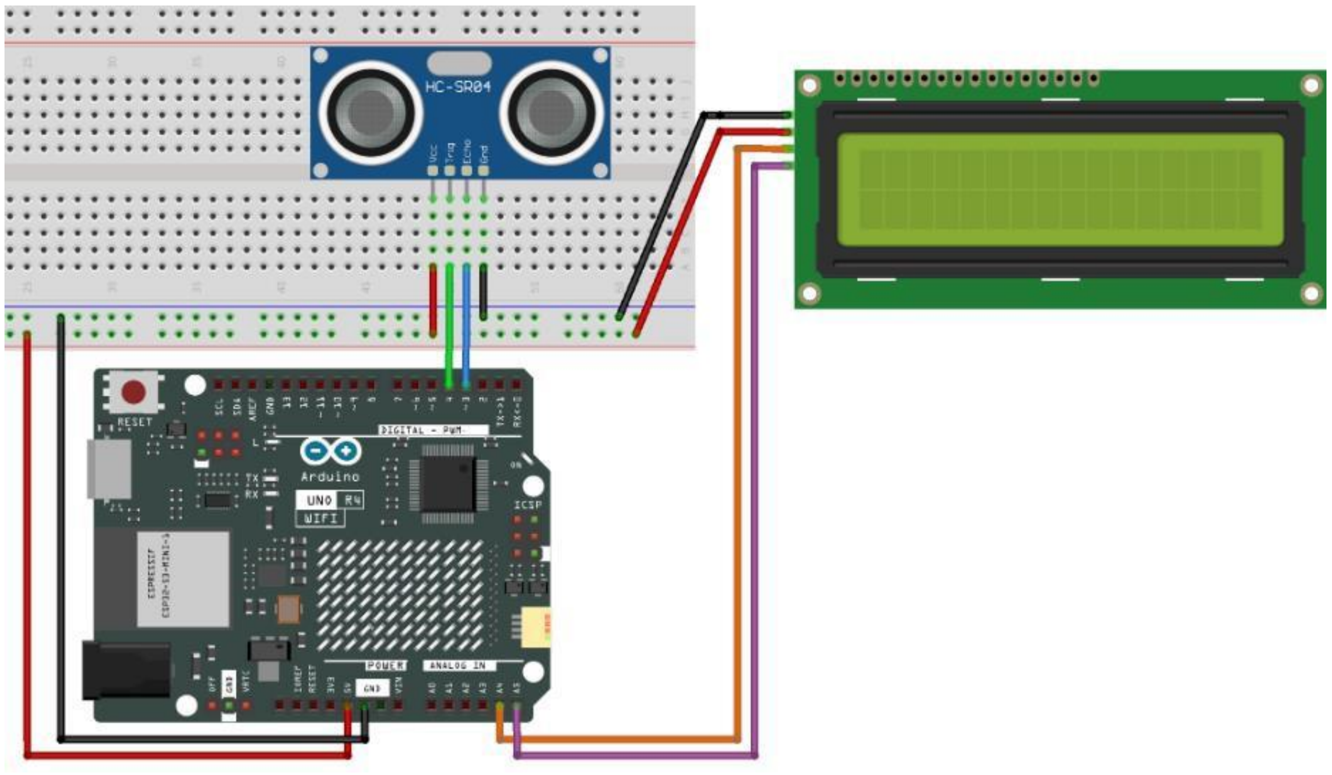
The I2C LCD1602 display is a 16x2 character LCD module that uses the I2C communication protocol for its interfacing. The display consists of 16 columns and 2 rows, allowing it to show up to 32 characters at a time.

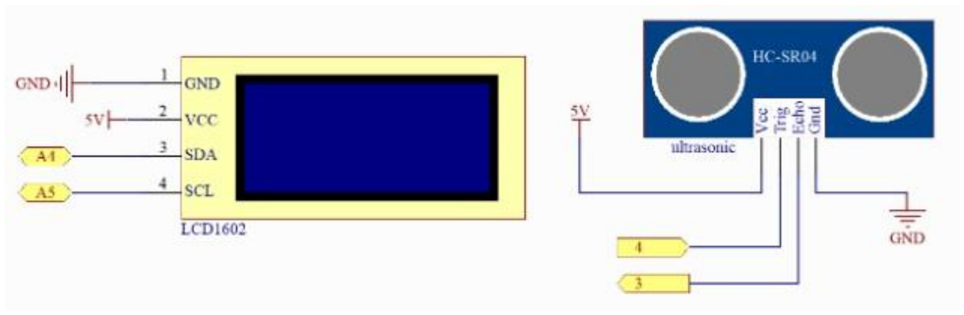
GND: Ground
VCC: 5V Supply
SDA: Serial Data Line
SCL: Serial Clock Line

Shorting Cap (Jumper): The cap shorts two pins responsible for enabling the backlight. It may be removed to disable it.
Potentiometer: Adjusts the contrast. Turn clockwise to increase and counterclockwise to decrease.

When the code is uploaded, the LCD will display “ABRA” on its first line and “Electronics!” on the second line. The text can be changed on the lcd.print() line.

Project 7: Ultrasonic Sensor

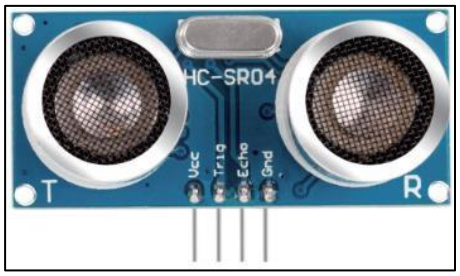




Required Components:

Arduino R4	1
Breadboard	1
Ultrasonic Sensor	1
I2C LCD1602	1

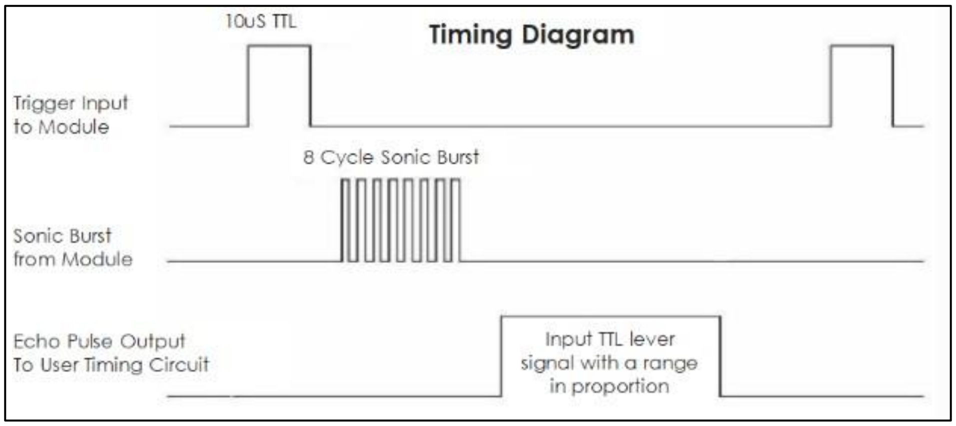
This project uses the 07-ultrasonic.ino sketch.



The ultrasonic sensor measures the distance between itself and the object using ultrasonic soundwaves through its two probes. One sends the ultrasonic waves and the other receives them and transforms the sending and receiving time into the distance. The sensor's detecting range is between 2 cm and 450 cm. The module uses ultrasonic transmitters, receiver and a control circuit. Its functionality is as follows:

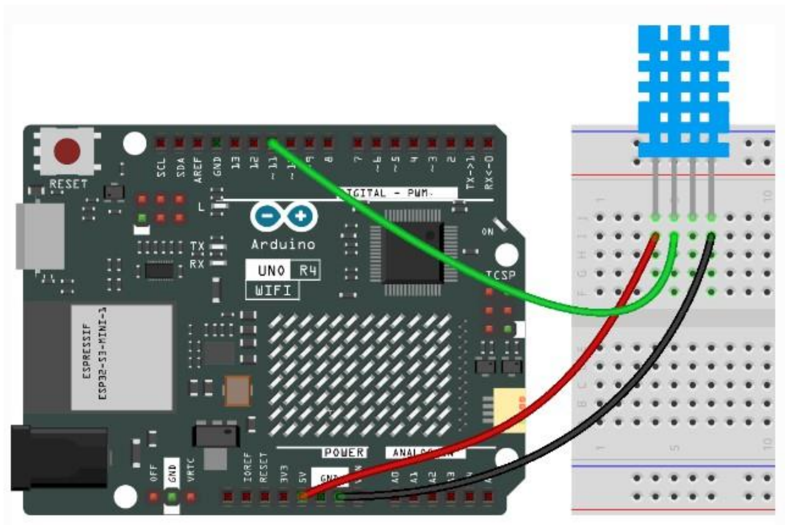
1. IO flip-flop processes a high-level signal of at least 10 µs.
2. The module automatically sends eight 40kHz pulses and detects if there is a pulse return.
3. If there is a pulse return, the duration of the high-level signal from the IO is the time from the transmission of the wave to its return.

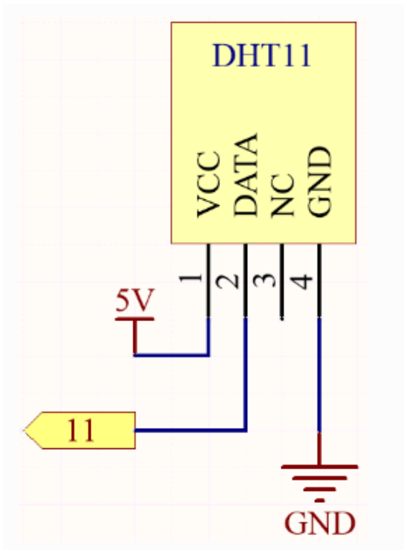
Therefore, Distance = ((high time x speed of sound) / 2) where speed of sound is equal to 343m/s.


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

Upon uploading the code, the distance between an object and the ultrasonic sensor will be displayed on the I2C LCD Display.

Project 8: Humiture Sensor Module

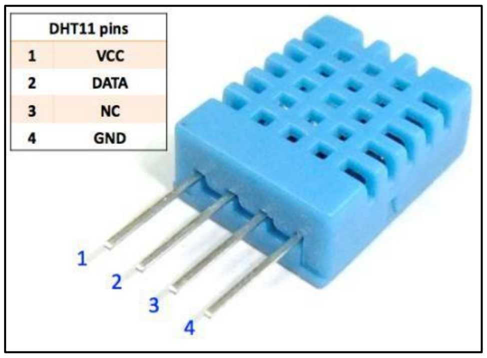




Required Components:

Arduino R4	1
Breadboard	1
DHT11 Humidity Sensor	1

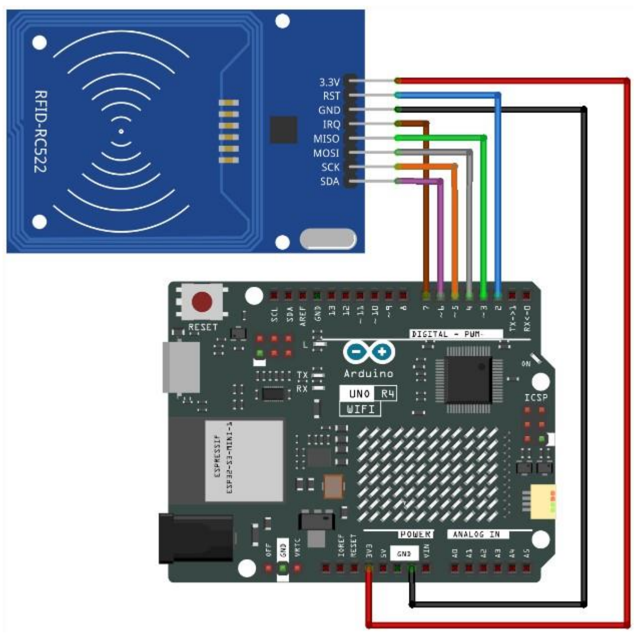
This project uses the 08-humiture_sensor.ino sketch.

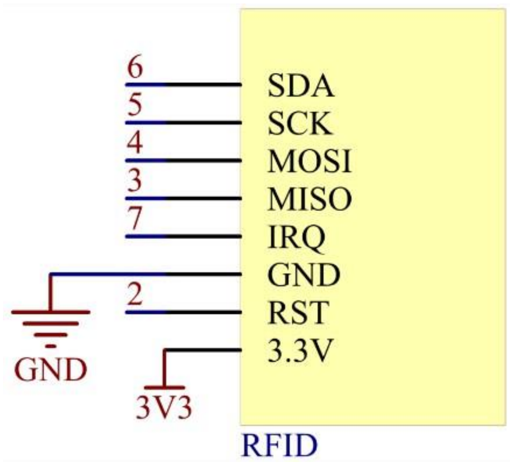


The DHT11 Humidity Sensor contains a calibrated digital signal output of temperature and humidity. The communication process initiates with the DATA line sending signals to the sensor that returns a different signal. The host receives the answer signal along with a 40-bit humiture data (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum).

Upon uploading the code, open up the serial monitor, and you'll be able to read the humidity and temperature values in your location.

Project 9: RFID-RC522 Module





Required Components:

Arduino R4	1
MFRC522 RFID Module	1

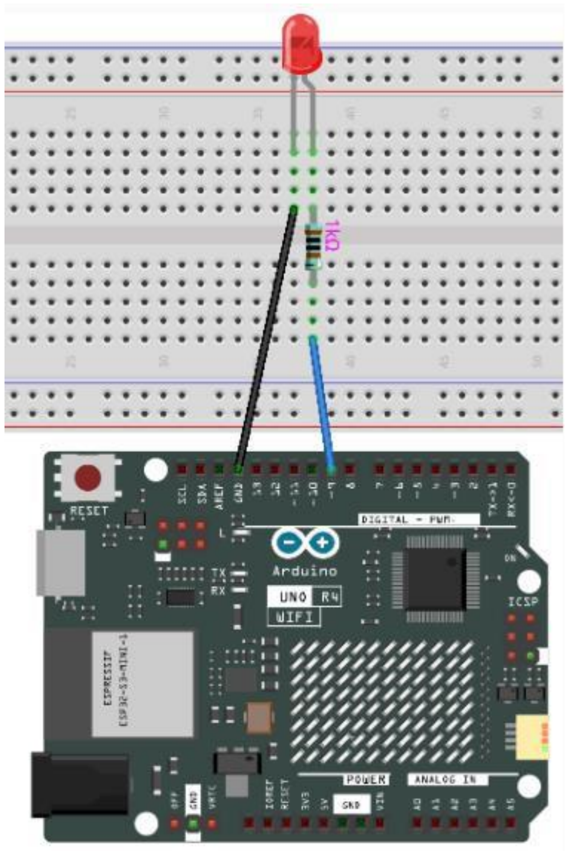
This project uses the 09-mfrc522.ino sketch.

Radio Frequency Identification (RFID) encompasses technologies that use wireless communication between a tag and a reader to automatically identify and track objects. The transmission range of the tag is typically limited to a few centimeters from the reader. Most RFID tags have at least one integrated circuit (IC) and an antenna. The IC stores information and manages radio frequency (RF) communication with the reader.

Upon uploading the code to the Arduino, you will see the RFID Card's information be displayed on the serial monitor as you bring it close to the RFID Reader.

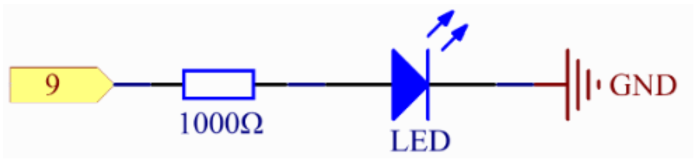
Note: Chances are that you will have to import the RFID.zip library in this project's folder to get the code to compile and upload. See *Serial Monitor & Installing Libraries*.

Project 10: LED



Required Components:

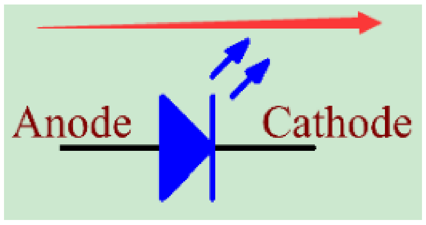
Arduino R4	1
Breadboard	1
LED	1
1kΩ Resistor	1



This project uses the 10-led.ino sketch.

A semiconductor light-emitting diode (LED) is a component that converts electrical energy into light energy through PN junctions. Based on wavelength, it can be classified into laser diodes, infrared LEDs, and visible LEDs, commonly referred to as LEDs.

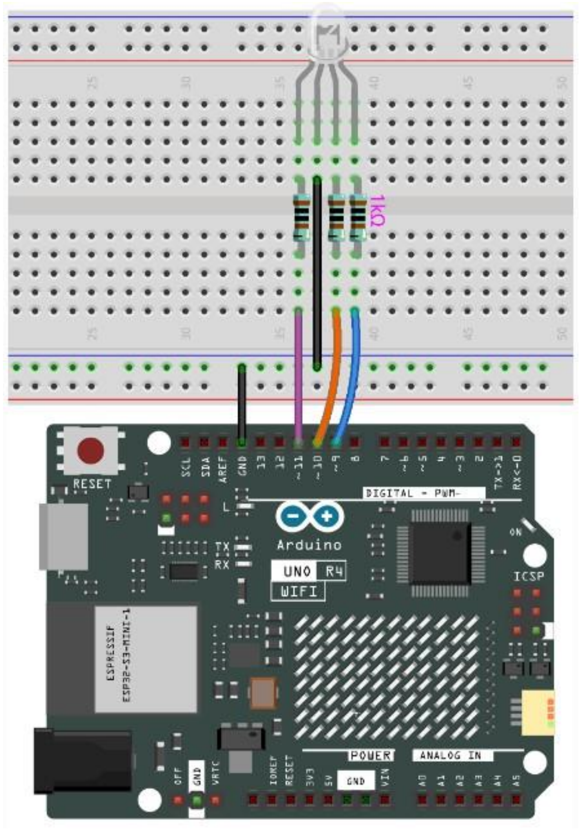
Diodes exhibit unidirectional conductivity, meaning current flows in the direction indicated by the arrow in the circuit symbol. To light up an LED, the anode must be connected to a positive power supply, with resistor, and the cathode to a negative power supply.



An LED has two pins: the longer pin is the anode, and the shorter pin is the cathode. It is crucial not to connect them incorrectly. LEDs have a fixed forward voltage drop and cannot be directly connected to a power source, as the supply voltage might exceed this drop and damage the LED. The forward voltage for red, yellow, and green LEDs is 1.8V, while for white LEDs, it is 2.6V. Most LEDs can handle a maximum current of 20mA, so a current-limiting resistor is needed in series to protect the LED.

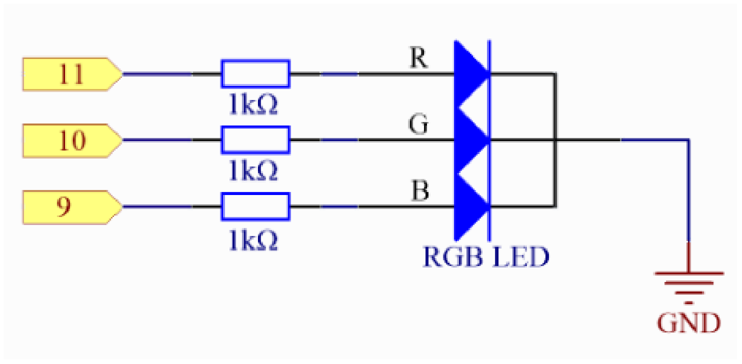
Upon uploading the code, the LED will start blinking at a rate of 2Hz (every half a second it turns on and turns off repeatedly).

Project 11: RGB LED

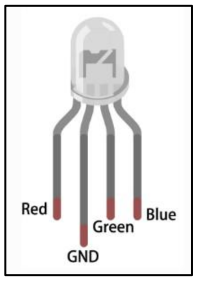


Required Components:

Arduino R4	1
Breadboard	1
RGB LED	1
1kΩ Resistor	3



This project uses the 11-rgb_led.ino sketch.

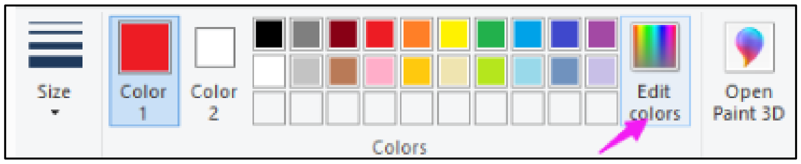


RGB LEDs can emit light in many colors. An RGB LED combines three LEDs—red, green, and blue—into a transparent or semi-transparent plastic casing. By adjusting the input voltage to its three pins, it can display various colors, theoretically producing up to 16,777,216 different shades.

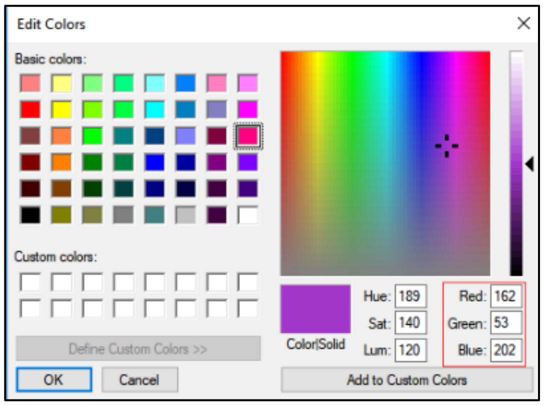
There are two types of RGB LEDs: common anode and common cathode. This kit includes the common cathode type, where the cathodes of the three LEDs are connected. By connecting the common cathode to GND and adjusting the input to the three pins, the LED will display the corresponding colors.

Upon uploading the code, the RGB LED will start to flash seven different colors.

Tip: If you'd like to add or change colors, you will need to provide the color() function your desired color's red, green and blue values (in that exact order). The simplest way to do this is to open Paint on your computer (assuming you're on Windows) and click Edit colors.

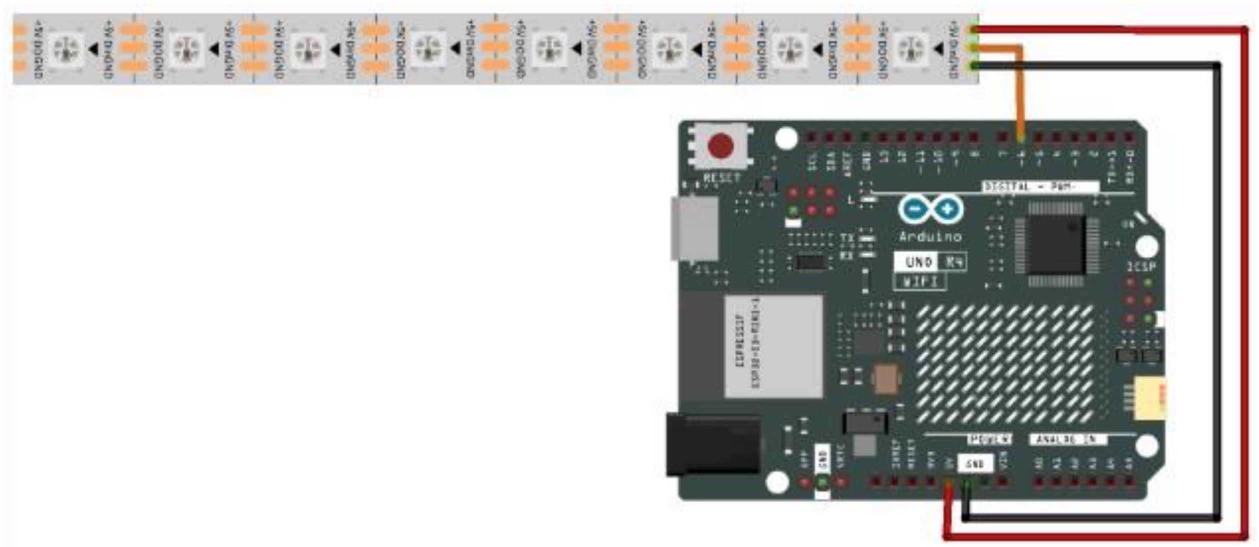


By selecting your desired color, you'll be shown its RGB values. Note the colors displayed on the computer screen may not perfectly replicate the color that will be shown on the LED.



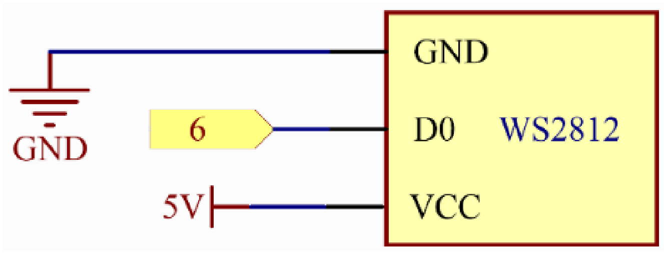
Hue: 189
Sat: 140
Lum: 120
Red: 162
Green: 53
Blue: 202

Project 12: WS2812 RGB LED Strip

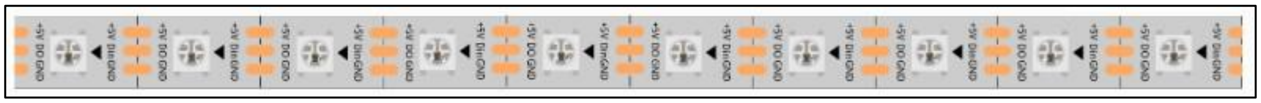


Required Components:

Arduino R4	1
Breadboard	1
WS2812 RGB LED Strip	1



This project uses the 12-ws2812.ino sketch.

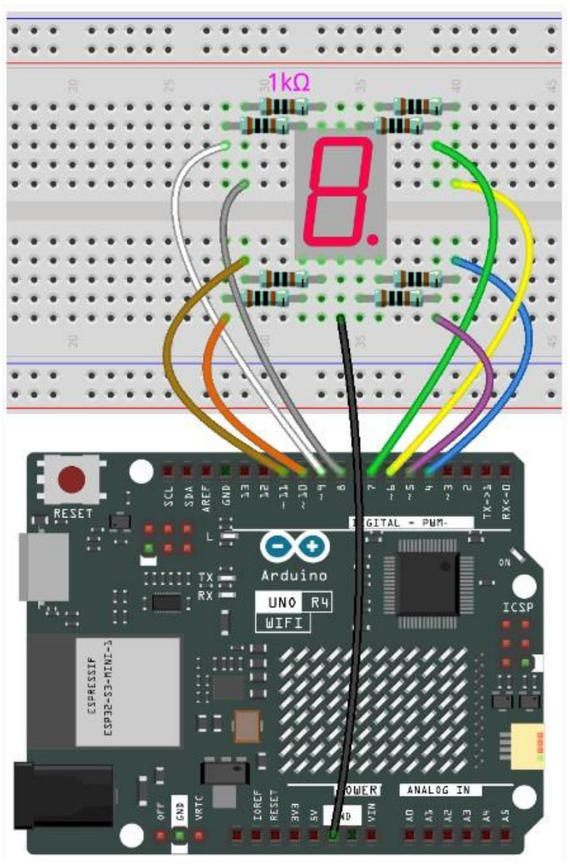


The WS2812 RGB 10 LEDs Strip consists of 10 RGB LEDs, all controllable with a single pin. Each LED includes a WS2812 chip, allowing for independent control. It can achieve 256 levels of brightness and display a full spectrum of 16,777,216 colors. The pixel integrates an intelligent digital interface with a data latch signal shaping amplifier drive circuit, ensuring consistent color accuracy. This strip is flexible, can be easily docked, bent, and cut, and features adhesive tape on the back for mounting on uneven surfaces and installation in narrow spaces.

Be sure to plug the board to the strip as indicated on the diagram. Pay attention to the arrows on the LED strip. Upon uploading the code, the LEDs will begin to flash in a sequential chain.

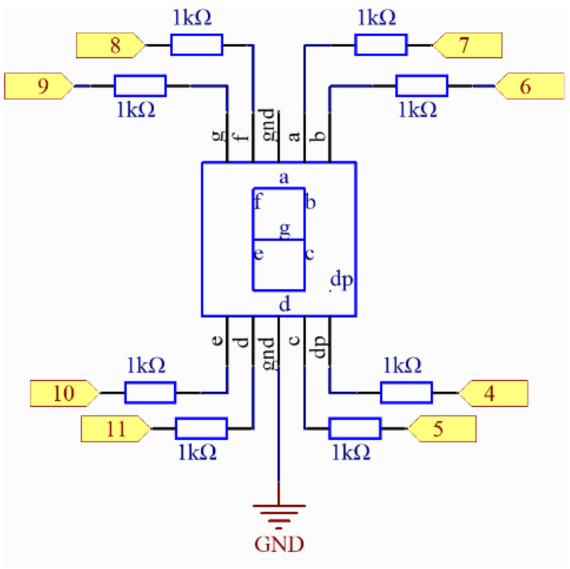
<u>Note: You might have to install or update the FastLED library using the IDE’s Library Manager to successfully upload the code.</u>

Project 13: 7-Segment Display

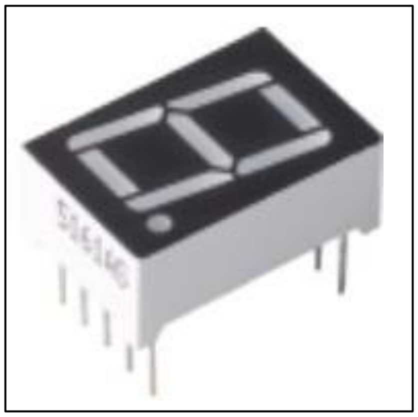


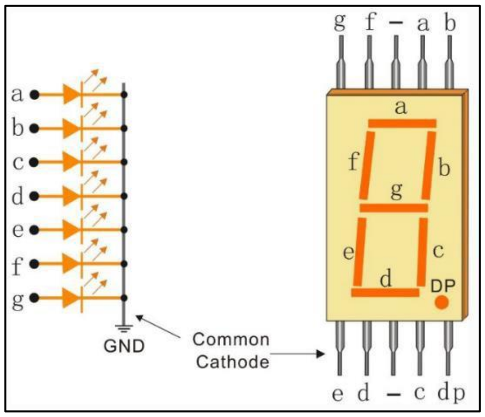
Required Components:

Arduino R4	1
Breadboard	1
7-Segment Display	1
1kΩ Resistor	8



This project uses the 13-7_segment.ino sketch.





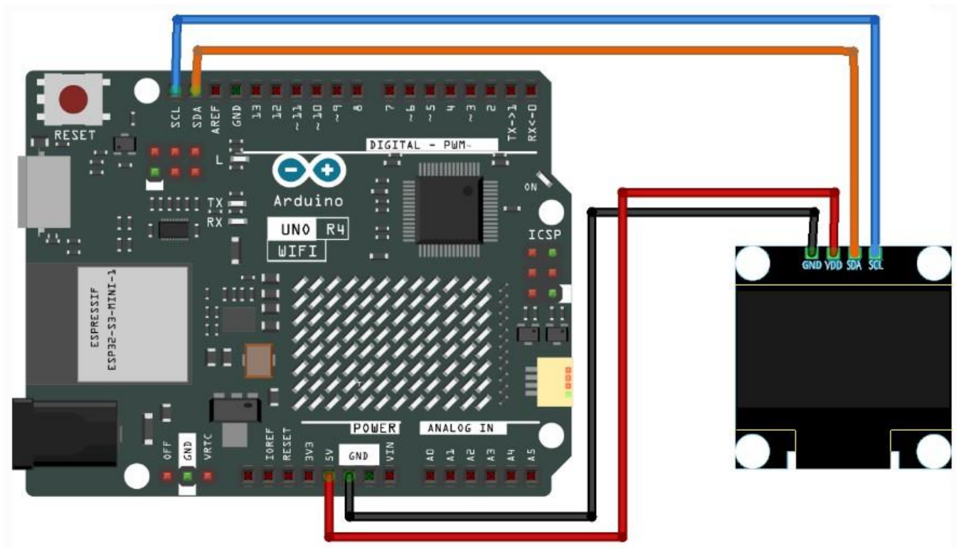
A 7-Segment Display is a component containing 7 LEDs where each is referred to as a segment. The display is programmed to use certain segments to display a letter (A-F) or number (0-9) and its decimal point.

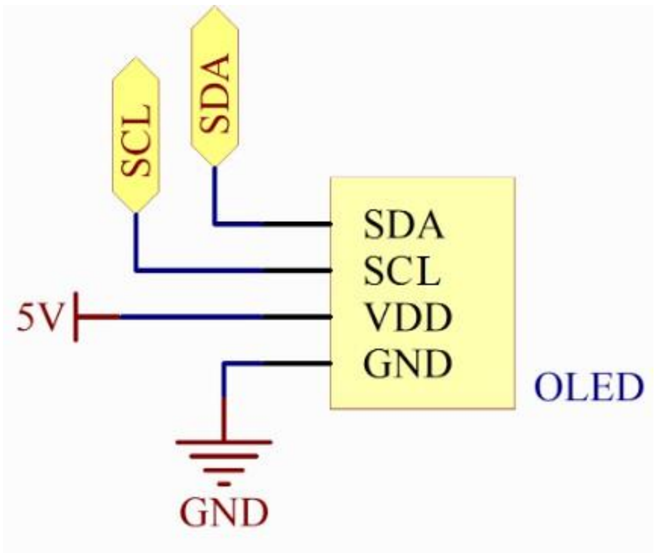
There are two types of segment displays: Common Anode and Common Cathode. This case, the latter is included in the kit where all segments share a common ground. To know how to use the segments to light up a certain character, the below chart explains its binary and hex code. In this case, however, it's simplified with a digitalWrite() command that selects the segments it wants to enable.

Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00000110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00000111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

Upon uploading the code, the 7-Segment Display will initiate counting from 1 to 15 where the double digits are replaced by the sequential letters of the alphabet.

Project 14: OLED

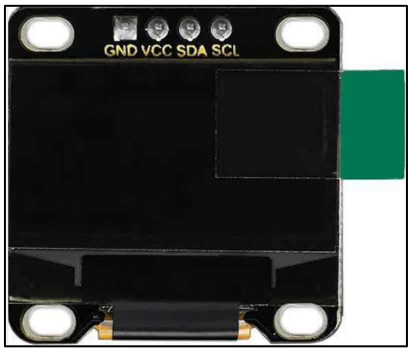




Required Components:

Arduino R4	1
OLED Screen	1

This project uses the 14-oled.ino sketch.

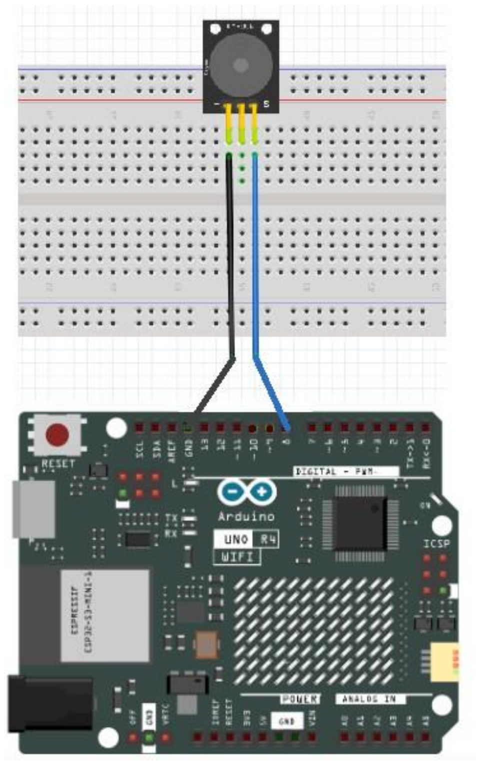


OLED stands for Organic Light-Emitting Diode. It’s a display module that displays text and images on a very thin screen using organic materials that emit light when electric current is passed. Since it doesn’t rely on a backlight, it emitting its own light gives it better contrast, brightness and angle of view in comparison with LCD displays. They’re widely used in smart watches.

Upon upload, the OLED screen will begin displaying various messages and icons to demonstrate its built-in features. Of course, those can be changed in the code as well.

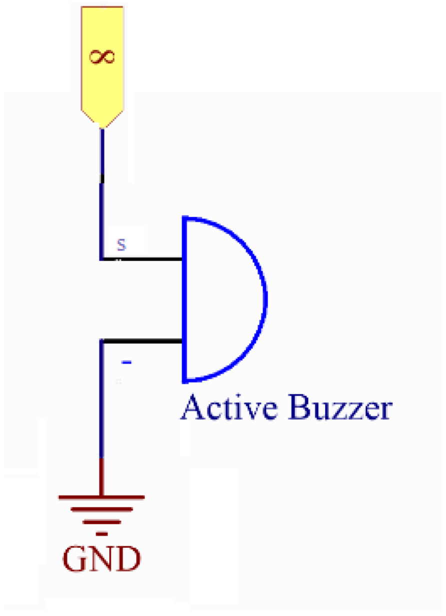
<u>Note: In the Arduino Library Manager, search for **Adafruit SSD1306** and **Adafruit GFX** libraries and install them with their dependencies to successfully upload the code.</u>

Project 15: Active Buzzer



Required Components:

Arduino R4	1
Breadboard	1
Active Buzzer	1

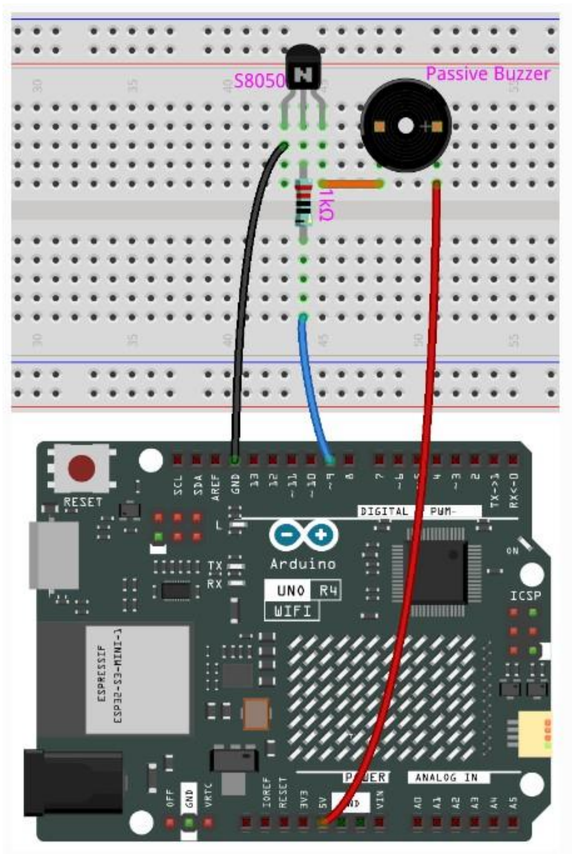


This project uses the 15-active_buzzer.ino sketch.

Buzzers are electronic devices supplied by power widely used in computers, alarms and printers that make sounds upon receiving current. This type of buzzer is active, meaning it contains an internal oscillating circuit that produces the sound, requires DC power supply (or a microcontroller pin that can carry that type of current), and its frequency and tone is fixed.

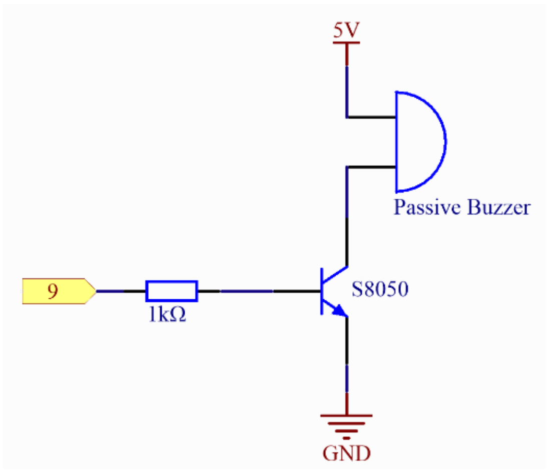
Upon uploading the code, the buzzer will continuously make a sound at a constant frequency.

Project 16: Passive Buzzer

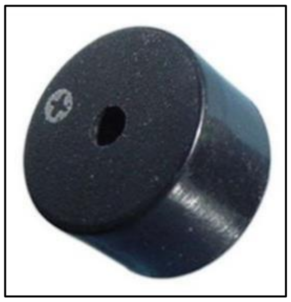


Required Components:

Arduino R4	1
Breadboard	1
Passive Buzzer	1
2N2222A Transistor	1
1kΩ Resistor	1



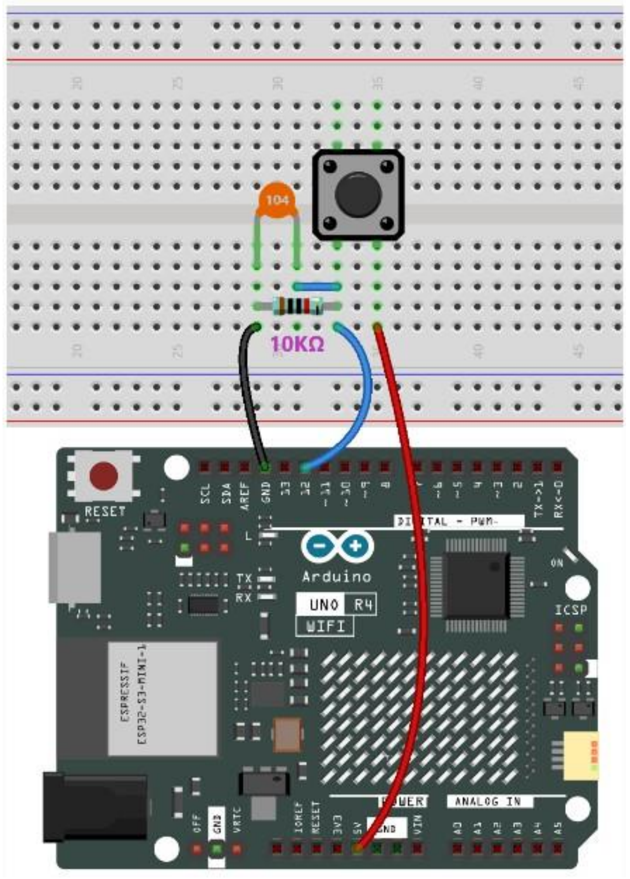
This project uses the 16-passive_buzzer.ino sketch.



Unlike the active buzzer, the passive buzzer does not have an internal oscillating circuit and requires an AC signal to produce sound (a square wave). Its pitch is not fixed and can be controlled by frequency of input signal.

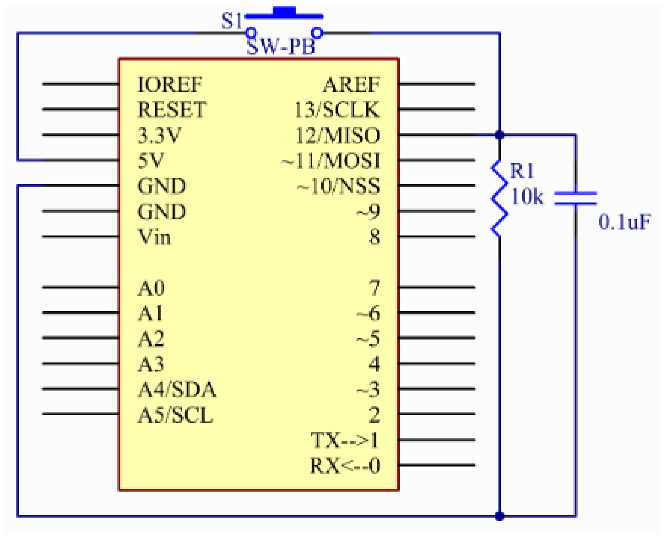
Upon uploading the code, a short melody made up of different pitches will play. The melody() function contains the notes of the melody and noteDurations() contains the steps for each note in chronological order. You can make changes in those functions to create your own melody, but note that you will be required to change a value in the following for loop: for (int thisNote = 0; thisNote < x; thisNote++) where x is the number of notes your melody() function contains. This is important to fully play your custom melody.

Project 17: Button

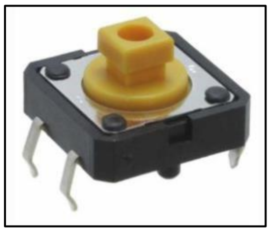


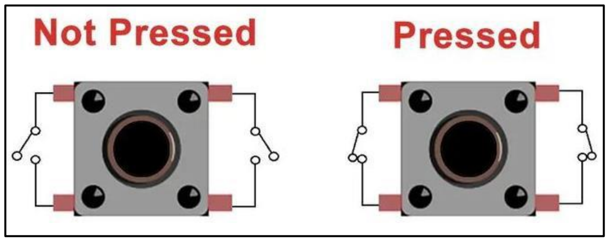
Required Components:

Arduino R4	1
Breadboard	1
Button	1
104pF Disc Capacitor	1
10kΩ Resistor	1



This project uses the 17-button.ino sketch.

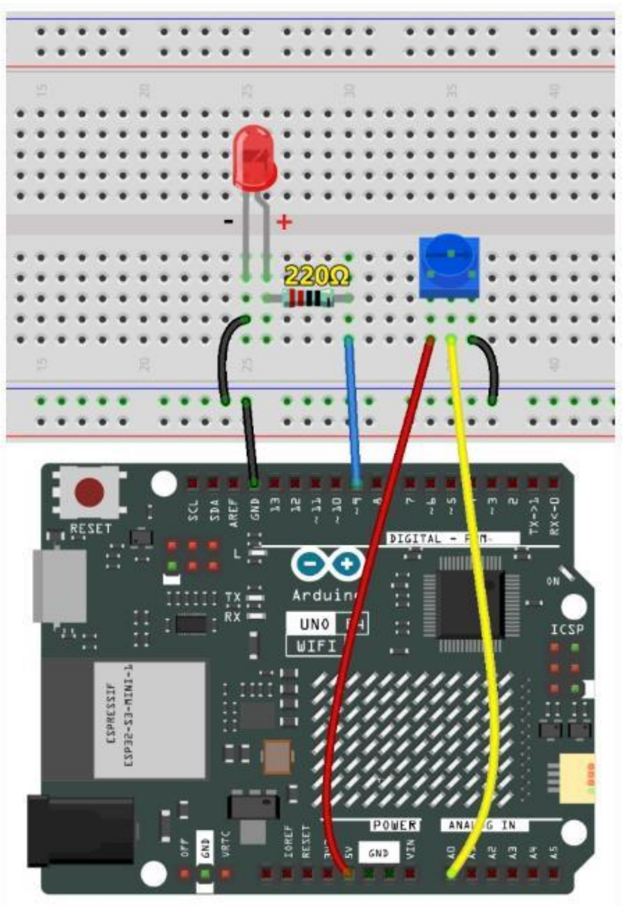




This push button is internally wired as indicated in above diagram. Horizontally (as angled above), the pins are interconnected. Vertically, however, are only connected when the button is pushed down. A capacitor is used in this circuit to prevent debouncing when the button is pressed. Debouncing is when a mechanical push button is pressed or released and doesn't make a clean, single transition between on and off states. Instead, it "bounces" and creates multiple on-off transitions before settling. This can cause the output to flicker. A capacitor can smooth out these bounces by temporarily storing and then releasing charge, providing a cleaner transition.

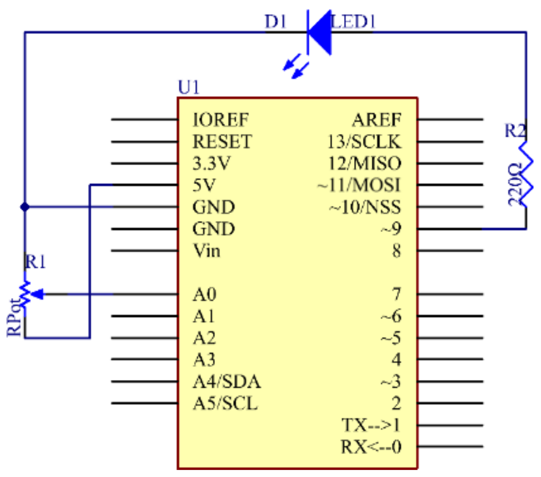
Upon uploading the code, an LED on the board will turn on or off depending on the status of the push button. If the button is pressed, it’s enabled. If released, it’s disabled.

Project 18: Potentiometer

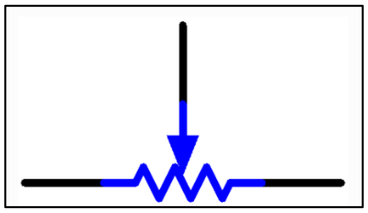


Required Components:

Arduino R4	1
Breadboard	1
Potentiometer	1
LED	1
220Ω Resistor	1



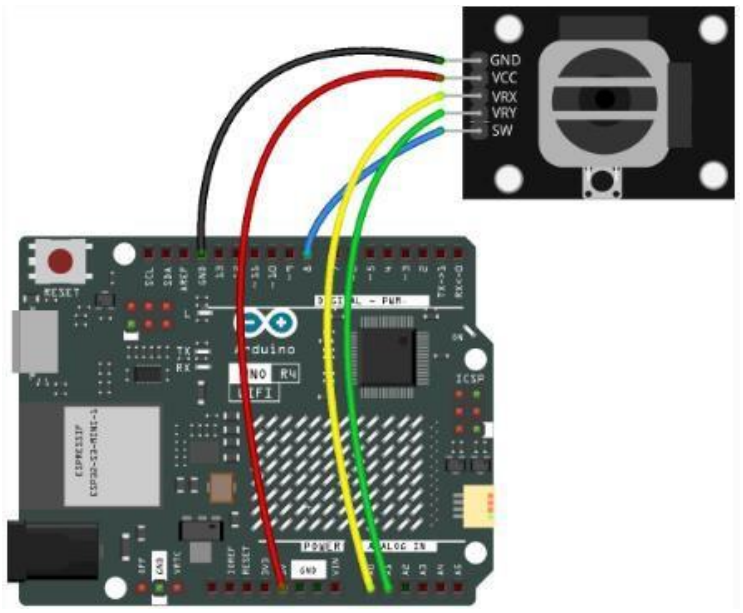
This project uses the 18-potentiometer.ino sketch.



A potentiometer functions just like a resistor, except with some notable differences. This device contains three terminals and a valve or knob that regulates its resistance value. Its minimum and maximum resistance varies from potentiometer to potentiometer. Despite the potentiometer already acting as a resistor, the 220Ω resistor is there as a barrier for the LED from overcurrent when the potentiometer's resistance is near zero.

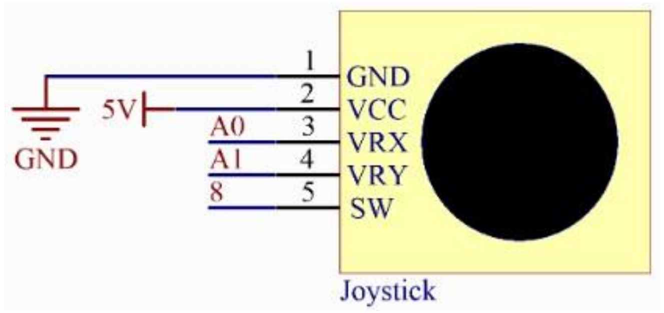
Upon uploading the code, the brightness of the LED can be regulated with the potentiometer's knob.

Project 19: Joystick Module

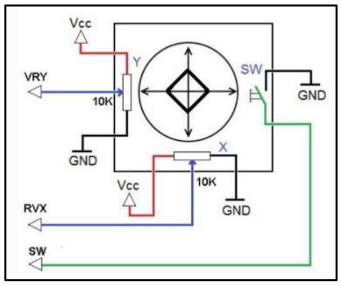


Required Components:

Arduino R4	1
Joystick Module	1



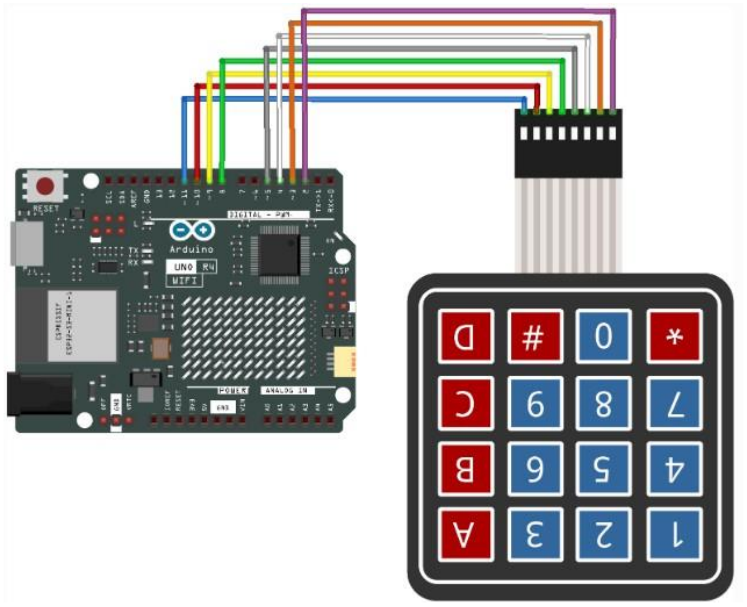
This project uses the 19-joystick.ino sketch.



This is a simplified module of a joystick similar of ones used for console controllers. This joystick module also has an integrated switch that can be used by pushing down on the it. The way the range of motion is communicated to the device is through the position of the joystick's x and y axis as shown in above diagram, with two potentiometers.

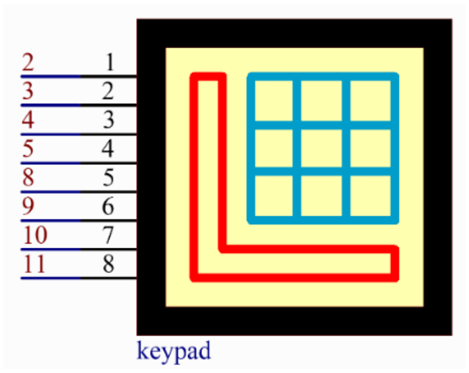
Upon uploading the code, the serial monitor will display the x and y position of the joystick, as well as the status of its switch.

Project 20: Keypad

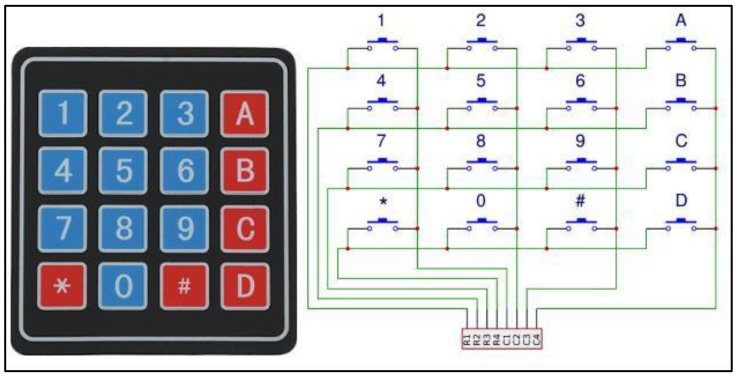


<u>Required Components:</u>

Arduino R4	1
Keypad	1



This project uses the 20-keypad.ino sketch.

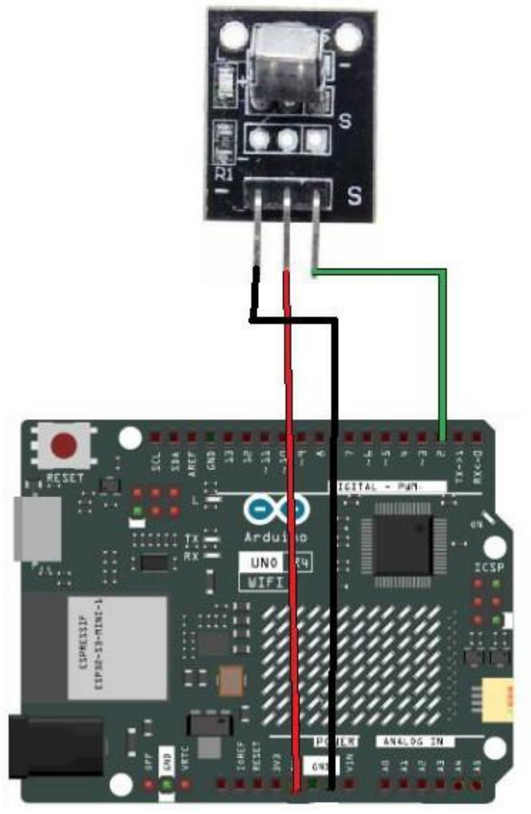


As seen in above diagram, this keypad is essentially an array of 12 tiny switches that are horizontally and vertically connected to merge into 8 pins that are responsible for detecting input from a key in its corresponding row and column.

Upon uploading the code, the serial monitor will simply display the button pressed and released on the keypad.

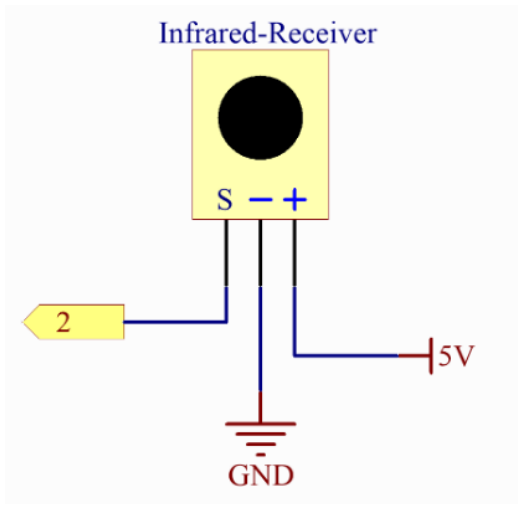
Note: In the Arduino Library Manager, search for **Adafruit Keypad** library and install it to successfully upload the code.

Project 21: Infrared Receiver Module

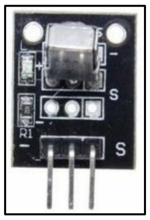


Required Components:

Arduino R4	1
IR Receiver Module	1
Remote Control	1



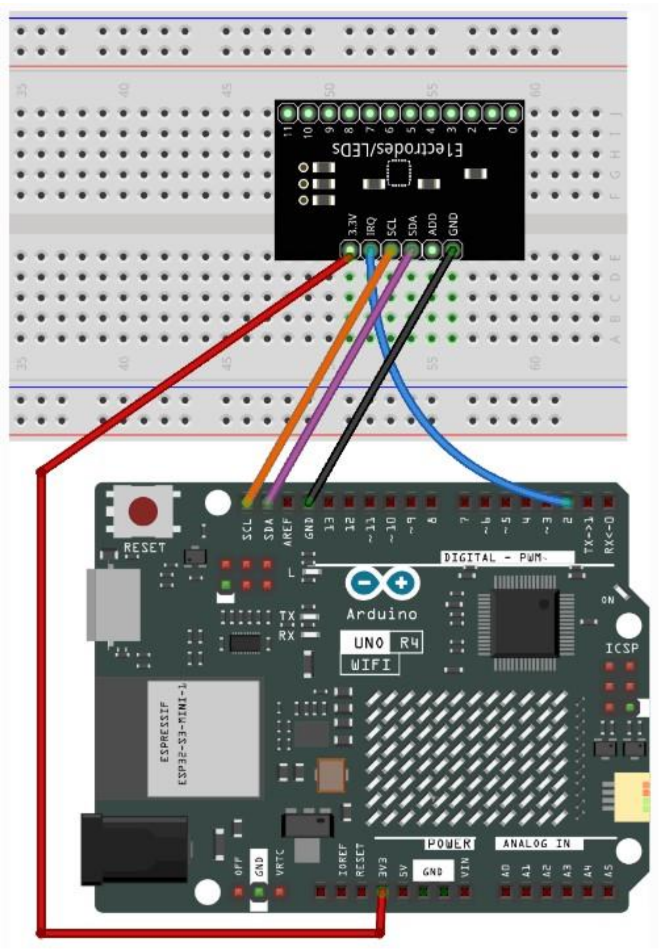
This project uses the 21-ir_receiver.ino sketch.



An IR receiver detects and decodes infrared signals sent by an IR emitter, which in this case is the remote control. They're widely used for wireless communication in television, audio equipment and electronics as such. Their common modulation frequencies are 36kHz, 38kHz and 40kHz.

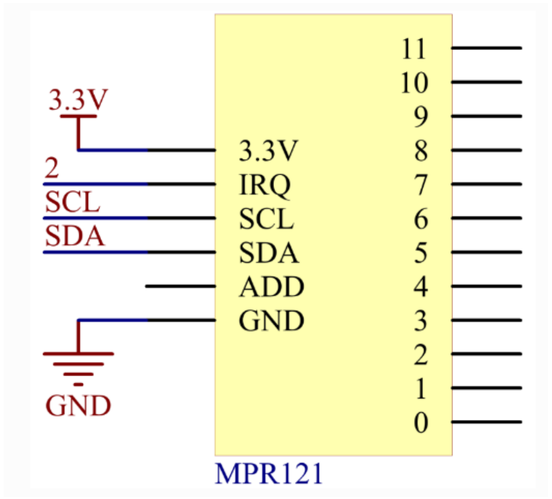
Upon uploading the code and opening the serial monitor, it will display the keys you hit on the remote as you point it at the IR receiver module. If you point it away from the module and hit any of the keys, the monitor will display error because the signal was not clear enough for the code (line 30 specifically) to decode the key pressed.

Project 22: MPR121 Touch Sensor

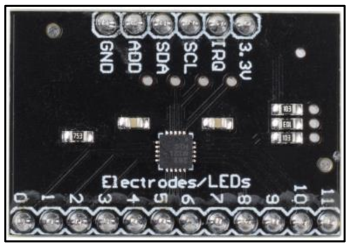


Required Components:

Arduino R4	1
Breadboard	1
MPR121 Touch Sensor	1



This project uses the 22-mpr121.ino sketch.



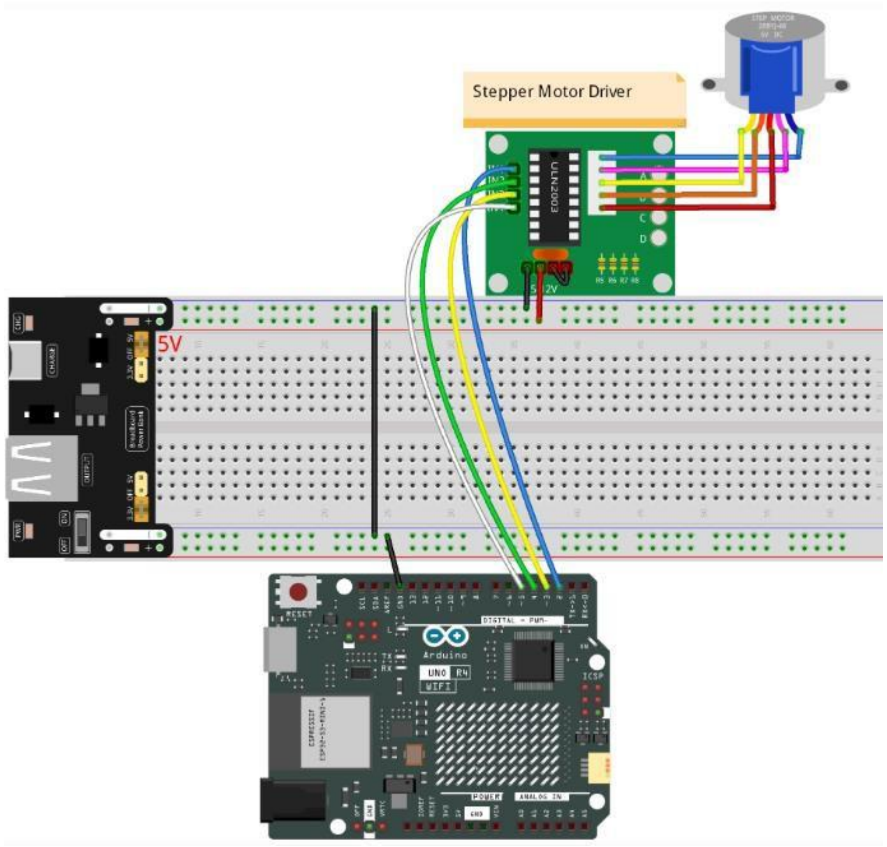
The MPR121 Touch Sensor offers 12 pins that can be interacted with just a human touch. It is operated with low power (1.7V – 3.6V). Its pins are as follow:

3.3V: Power supply.
IRQ: Open Collector Interrupt Output Pin, active low.
SCL: I2C Clock.
SDA: I2C Data.
ADD: I2C Address Select Input Pin. Connect the ADDR pin to the VSS, VDD, SDA or SCL line, the resulting I2C addresses are 0x5A, 0x5B, 0x5C and 0x5D respectively.
GND: Ground.
0~11: Electrodes.

Upon uploading the code and opening the serial monitor, the output of all 12 pins will be formatted as an array of bits. The pin touched will flip its corresponding bit to 1.

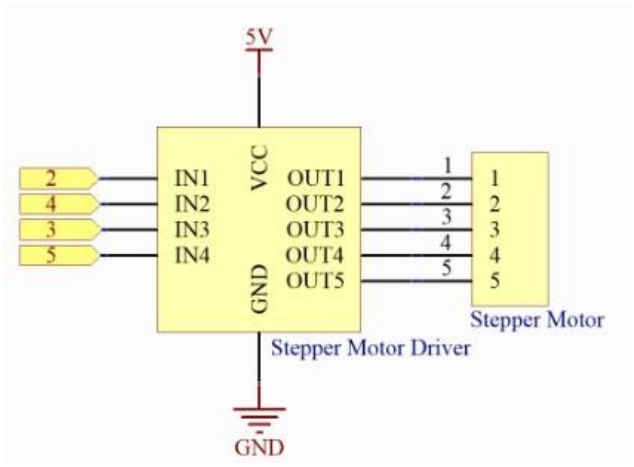
Note: In the Arduino Library Manager, search for **Adafruit MPR121** library and install it to successfully upload the code.

Project 23: Stepper Motor & Driver

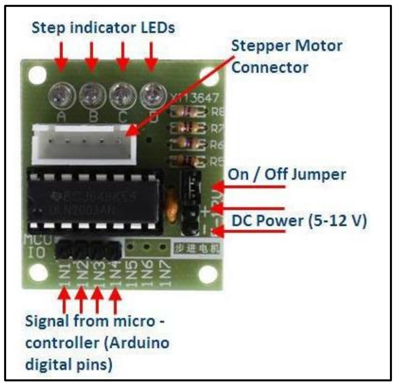


Required Components:

Arduino R4	1
Breadboard	1
Power Supply Module	1
Stepper Motor & ULN2003 Driver	1
9V Power Adapter	1

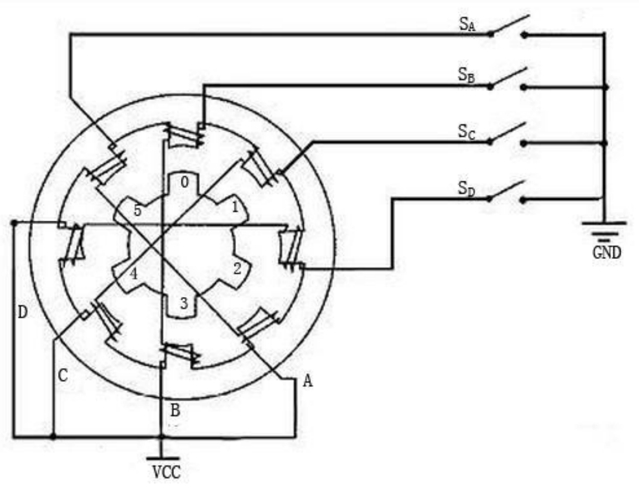


This project uses the 23-stepper_motor.rst.ino sketch.



This four-phase (known for possessing four coils) unipolar stepper motor moves in half-step mode (8 steps per cycle) for precise motion. The four coils are labeled A, B, C and D. They are energized in the following order in their half-step mode:

1. Energize Coil A.
2. Energize Coils A and B.
3. Energize Coil B.
4. Energize Coils B and C.
5. Energize Coil C.
6. Energize Coils C and D.
7. Energize Coil D.
8. Energize Coils D and A.

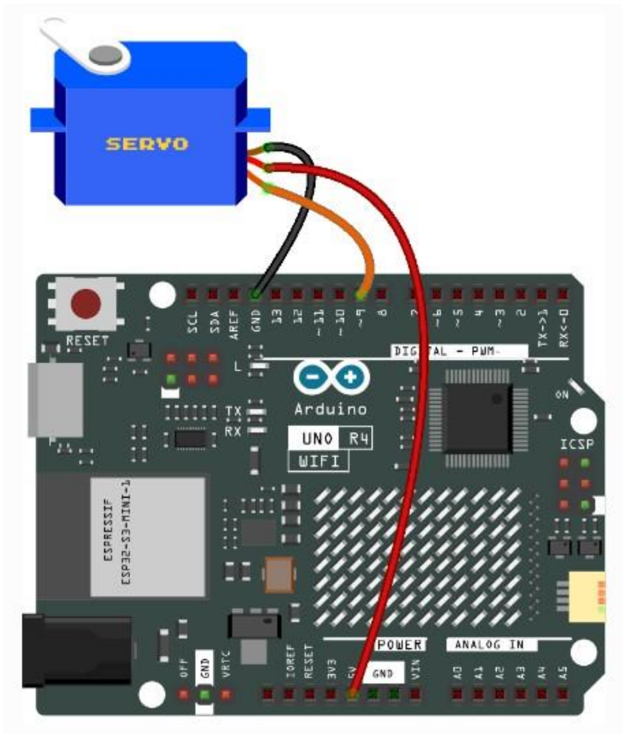


The driver used for this motor is a ULN2003. Drivers are necessary for reasons such as regulating the motor's current control and easily sequence its steps from an Arduino.

Stepper motors are notorious for their higher current draw, so this project will require a power supply module to be implemented into the breadboard. Both Arduino and driver will receive power from the module. 9V output AC Adapter will be used.

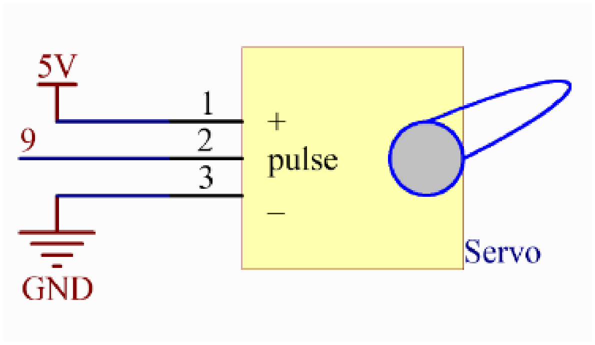
Upon uploading the code, the stepper motor will begin rotating in a clockwise direction at 5 RPM (revolutions per minute). Once it completes a revolution, it will pause for one second and rotate in a counterclockwise direction at 15 RPM for another complete revolution before stopping for another second. These rotations will loop.

Project 24: Micro Servo Motor

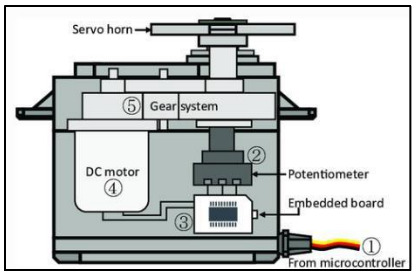


Required Components:

Arduino R4	1
9G Micro Servo	1



This project uses the 24-servo.ino sketch.

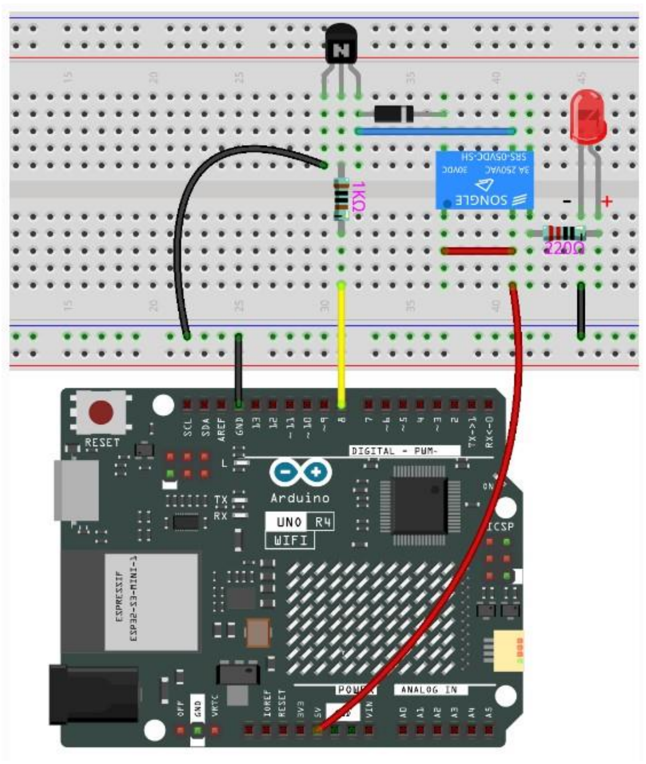


A micro servo is made up of a DC motor, gear system, embedded board, potentiometer, shaft and a case. The following is its functionality:

1. Microcontroller sends out PWM signals to the servo.
2. Embedded board in the servo receives these signals through the signal pin.
3. Embedded board controls the motor inside the servo to turn.
4. Motor drives the gear system.
5. Gear system, after deceleration, motivates the shaft.
6. Shaft is connected to a potentiometer.
7. When the shaft rotates, it drives the potentiometer.
8. Potentiometer outputs a voltage signal to the embedded board.
9. Embedded board determines the direction and speed of rotation based on the current position.
10. Servo stops exactly at the defined position and holds there.

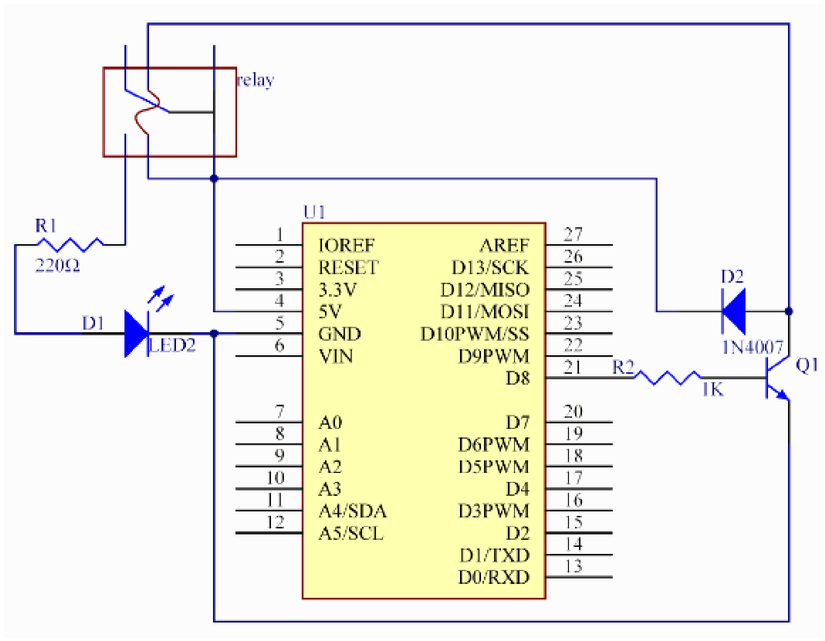
Upon uploading the code, the servo will rotate 180 degrees in one direction, pause for 15ms, rotate 180 degrees in the other direction, pause for 15ms, and repeat indefinitely.

Project 25: Relay



Required Components:

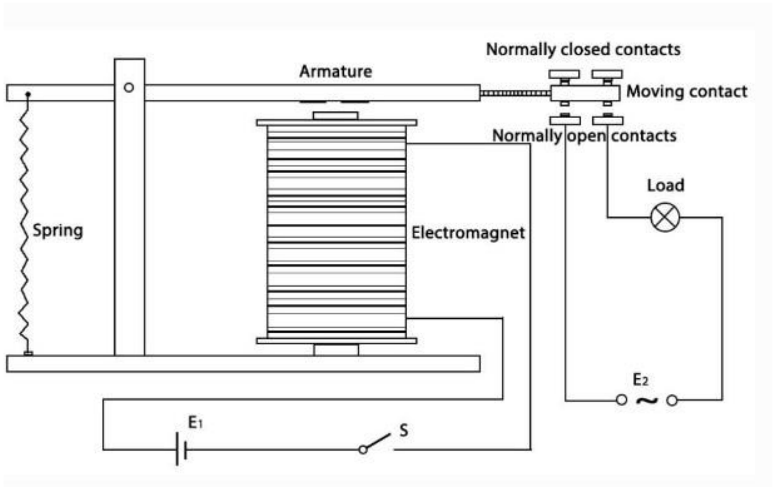
Arduino R4	1
Breadboard	1
1N4007 Diode	1
2N2222A Transistor	1
1kΩ Resistor	1
220Ω Resistor	1
5VDC 3A Relay	1
LED	1



This project uses the 25-relay.ino sketch.

Relays are switches that control multiple circuits provided by a separate low-power signal. It can separate low-power circuit from high-power circuit which provides safety by preventing interference (isolation), switch multiple circuits on and off (switching), and amply weak signal to control a larger current (amplification).

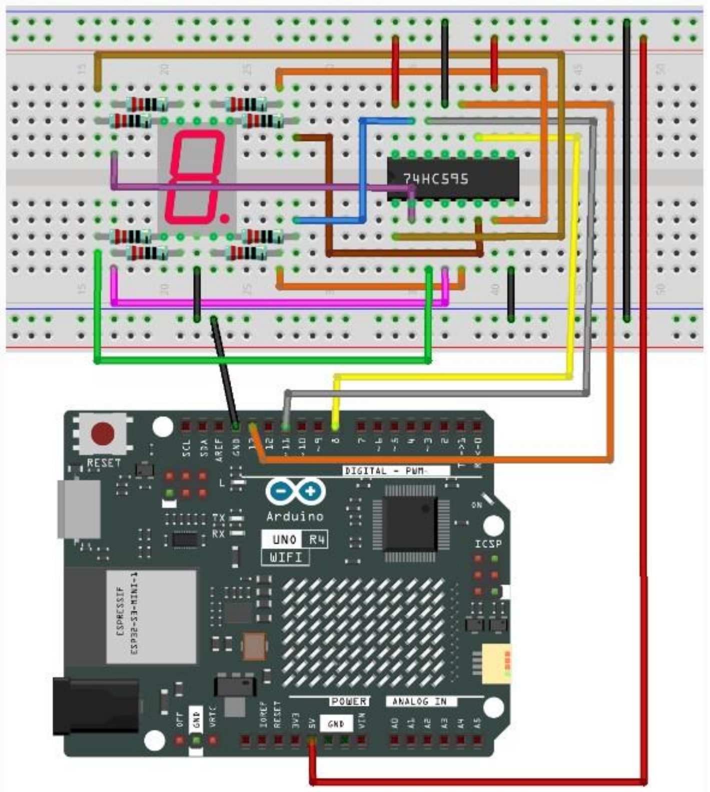
A relay consists of five parts:



Electromagnet: Consisting of an iron core wrapped in wires so that when electricity passes, it becomes magnetic.
Armature: A magnetic strip that is energized when current flows through it, behaving as a lever.
Spring: Spring that pulls on the armature to prevent current circulation throughout the circuit when there is no current flowing through the coil.
Molded Frame: Plastic casing for protection.
Contacts: Normally Open (connected when relay is active, disconnected when inactive), Normally Closed (disconnected when relay is active, connected when inactive).

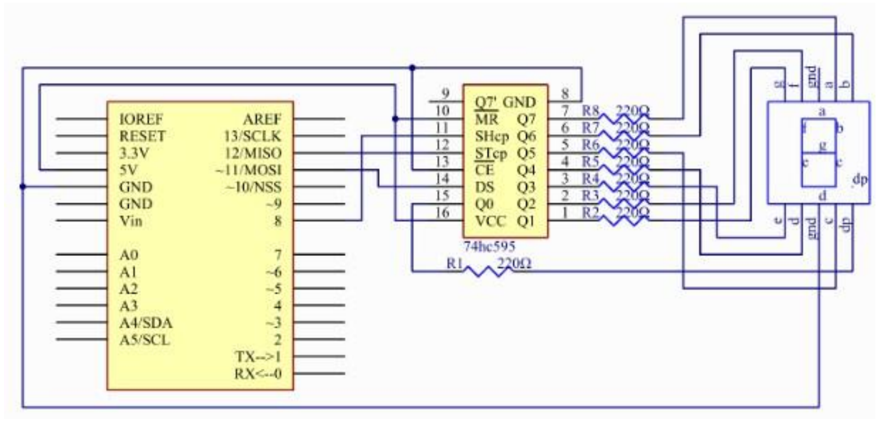
Upon uploading the code, the LED will alternate between turning on and off every second. Additionally, you will hear a sound like “tick-tock” of a clock. That is the relay opening (LED turns off) and closing (LED turns on).

Project 26: 74HC595

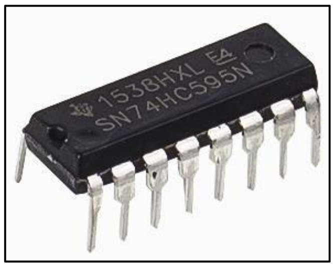


Required Components:

Arduino R4	1
Breadboard	1
7-Segment Display	1
74HC595	1
220Ω Resistor	8



This project uses the 26-74HC595.ino sketch.

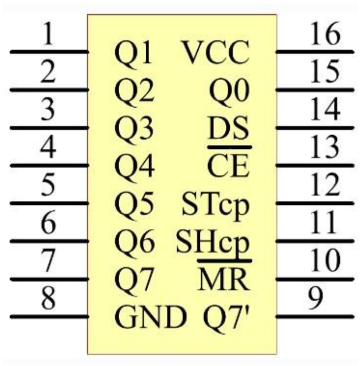


This is a microcontroller an 8-bit shift register that takes serial input into parallel outputs to save physical space on its IO ports. It also performs latching, which functions as a storage register that allows hold of the output for the next “update”. It’s mainly applied in LED arrays, 7-segment displays, keypad scanning and other similar applications.

Its pins are as follow:

Q0-Q7: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.

Q7’: Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series (daisy-chaining).



MR: Reset, active at low level.

SHcp: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.

STcp: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.

CE: Output enable pin, active at low level.

DS: Serial data input pin.

VCC: Voltage supply.

GND: Ground.

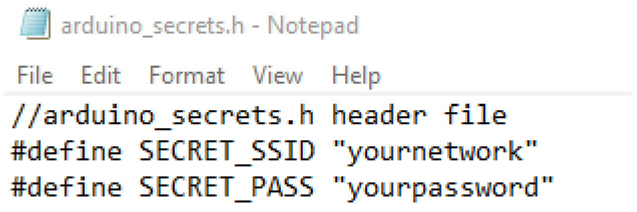
Upon uploading the code, the 7-segment display will begin counting from zero to fifteen (0 – F). The code performs this differently than in the other project. Here, a riding edge pulse is generated and a byte of data, one bit at a time, is shifted out in the shifting register. When all eight bits are shifted into the memory register, it is outputted to the bus (Q0-Q7) and the number, or letter, is displayed on the display. This loops 16 times for all characters and resetting.

R4 Project 1: Wi-Fi

This project uses the 01-wifi_connect.ino and 01-wifi_ap.ino sketches.

This project will focus on connecting your Arduino board to a Wi-Fi network. Through the serial monitor, you'll be able to view network details.

Before you upload the code, you must first modify the header file *arduino_secrets.h* in this project's folder. Change the texts within the quotations to your Wi-Fi name and your Wi-Fi password. The file can be opened and edited with Notepad.



```
//arduino_secrets.h header file
#define SECRET_SSID "yournetwork"
#define SECRET_PASS "yourpassword"
```

Now that that's done, 01-wifi_connect.ino will upload to the Arduino with no problems. Upon uploading the code, open the serial monitor and you'll be able to view your Arduino's network data, such as its IP address, MAC address, encryption type, as the network data it's connected to.

The second sketch creates a web server (web page) that you can connect to either with your computer or your phone. The upload of the sketch takes the same procedure as the first sketch – modifying the *arduino_secrets.h* file inside the 01-wifi_ap.ino folder this time. As per code, the IP address of your Arduino is 192.48.56.2, which is the address you paste in the URL of your browser.

Upon uploading the code and entering the web page, you'll be able to turn an LED on your Arduino on and off by interacting with the corresponding buttons on the site.

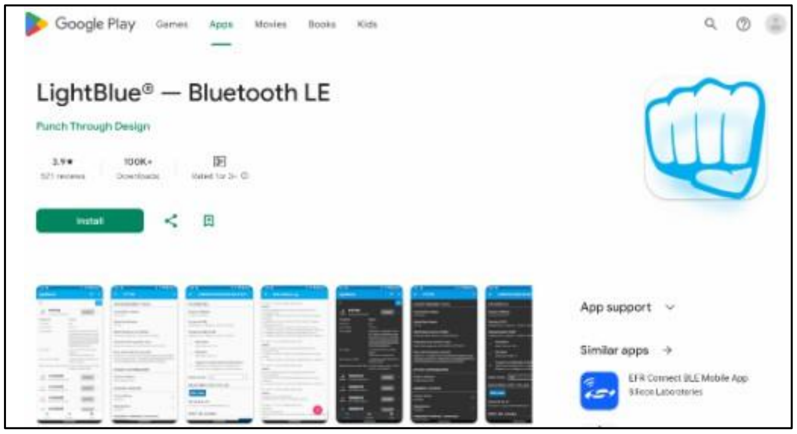
R4 Project 2: Bluetooth

This project uses the 02-bluetooth.ino sketch.

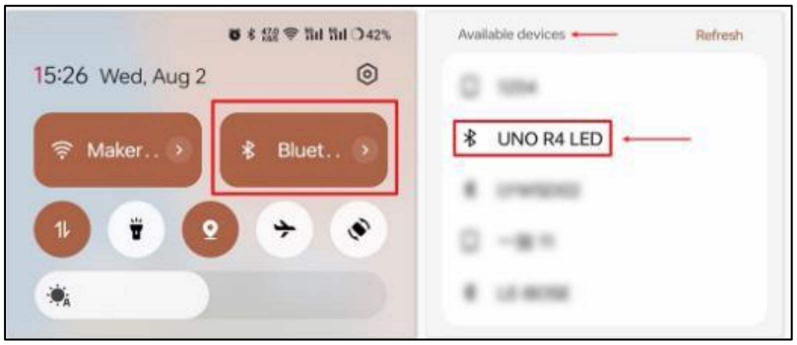
Note: In the Arduino Library Manager, search for **ArduinoBLE** library and install it to successfully upload the code.

Everyone's heard of Bluetooth. Its ease of use through pairing devices is widely popular across the world. In this project, you will be pairing your phone with the Arduino through Bluetooth and controlling the LED status through a mobile app.

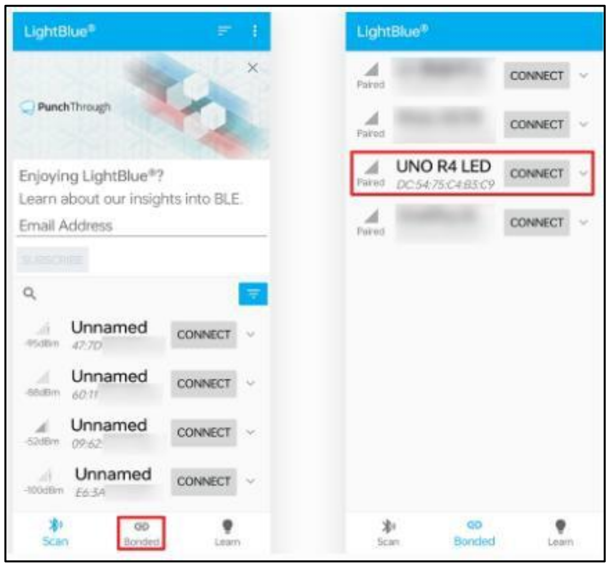
First, upload the sketch to your Arduino. Upon uploading the code, download the LightBlue – Bluetooth LE app on your phone. It is available for Android and iPhone systems.



Once it's installed, enable Bluetooth on your phone and connect to the Arduino device.



Next, launch LightBlue and navigate through the app until you've connected to your Arduino device through it as well.



Once connected, you’ll be able to access the detailed information about the Arduino Bluetooth device. Scroll until you encounter “Written Values”. This section will control the LED on the Arduino device. Writing 1 will turn the LED on and writing 0 will turn the LED off.

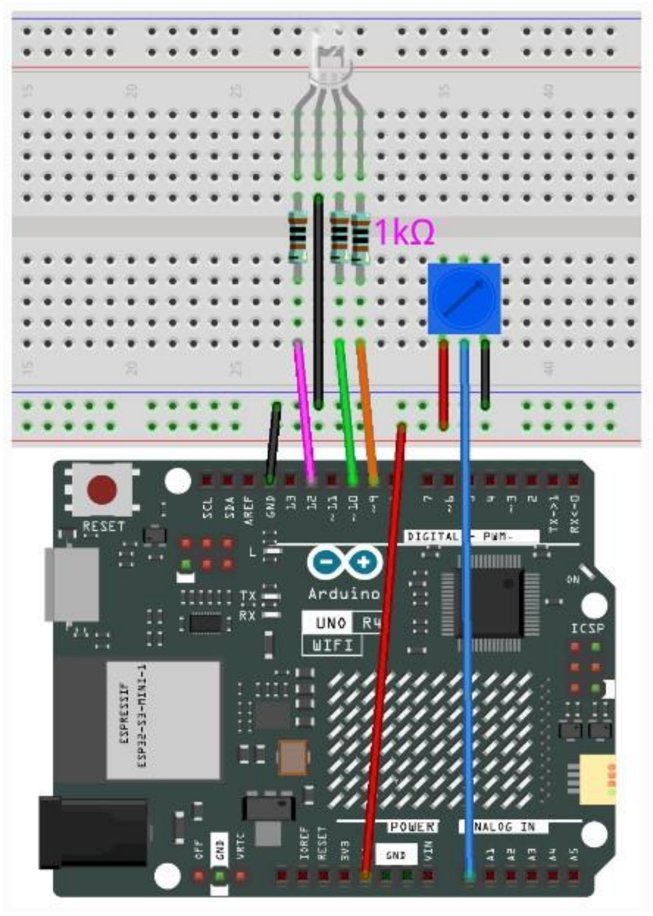
R4 Project 3: 12x8 LED Matrix

This project uses the 03-led_matrix.ino sketch.

The LED matrix refers to the 96 LEDs on the Arduino board that can be used to program a variety of graphics and animations.

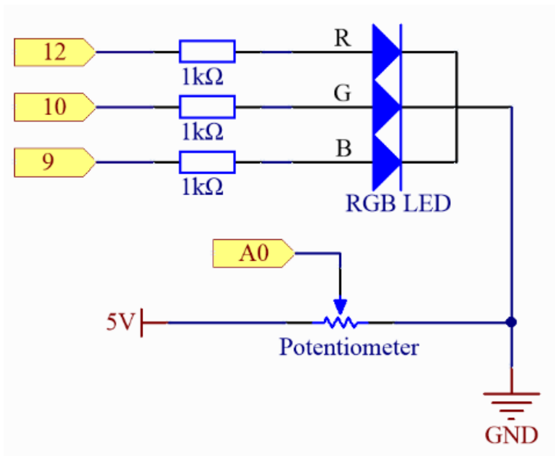
Upon uploading the code, an animation will play on the LED matrix for what looks like a loading bar. You can also use the online tool Arduino LED Matrix Editor to create your own animation and graphics.

Entertainment Project 1: Hue Dial



Required Components:

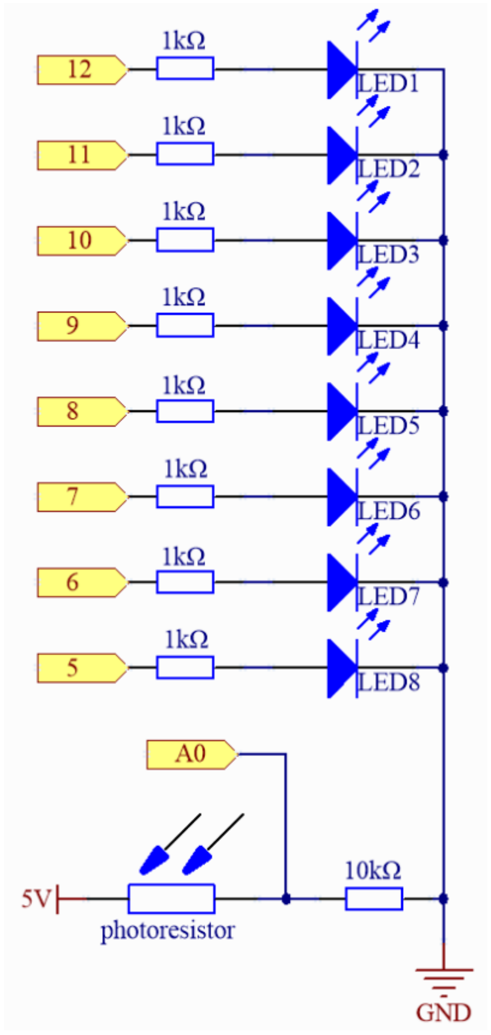
Arduino R4	1
Breadboard	1
RGB LED	1
Potentiometer	1
1kΩ Resistor	3

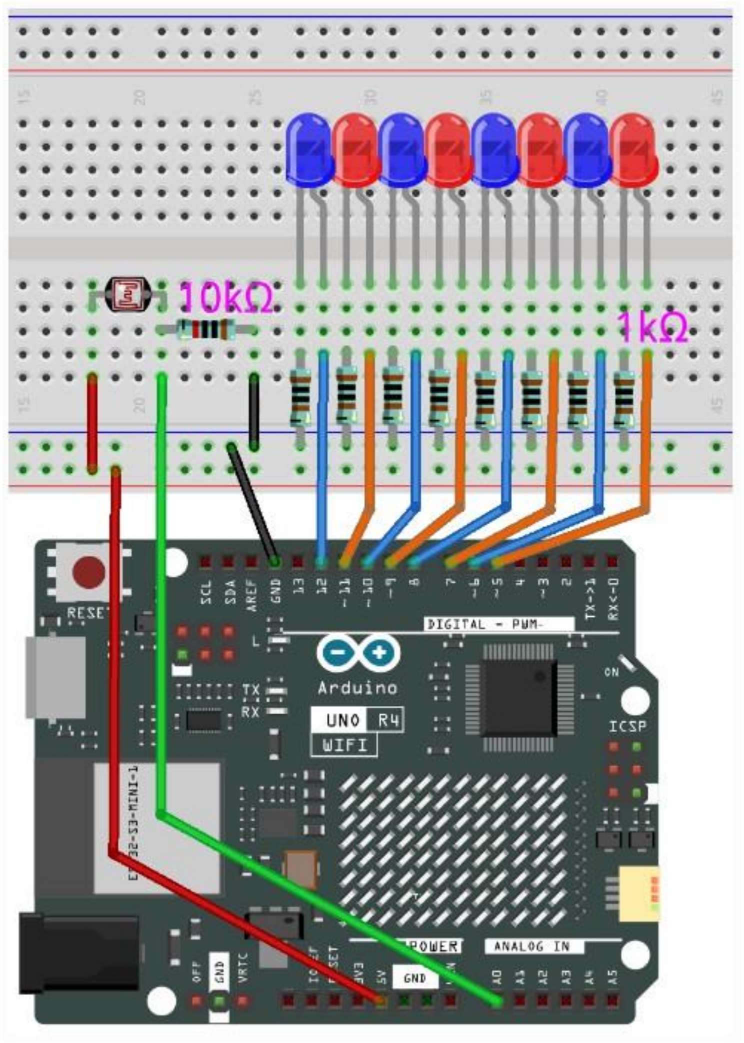


This project uses the 01-hue_dial.ino sketch.

This project is utilizing two major components we have learned previously: the RGB LED and the potentiometer. The RGB LED is a common cathode, so it’s wired to commonly ground all three of its colors. As mentioned, the potentiometer behaves like a resistor. Instead of controlling the brightness of the LED, the program is created to change the hue of it instead. At certain resistance of the potentiometer, it will correspond with the specific hue values that convert into RGB color values.

Entertainment Project 2: Light-Dependent Array





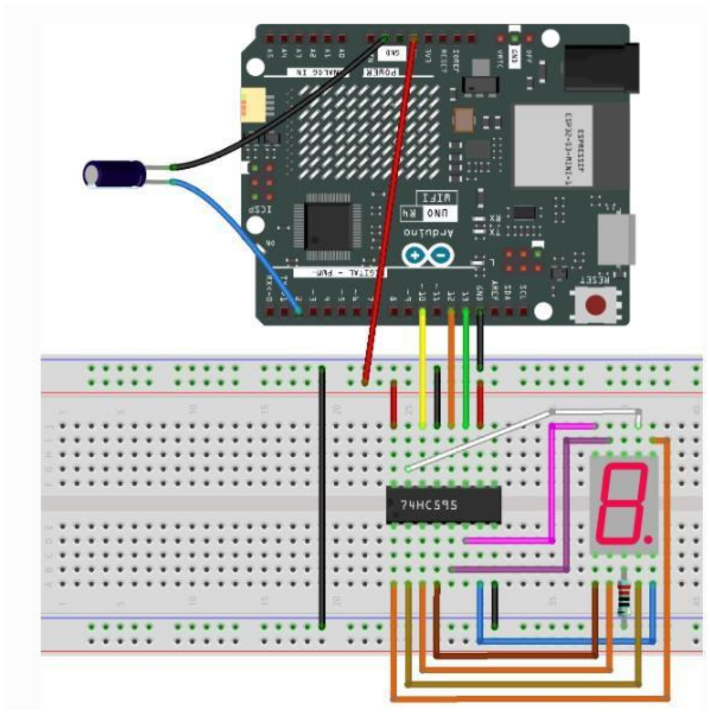
<u>Required Components:</u>

Arduino R4	1
Breadboard	1
LED	8
Photoresistor	1
1kΩ Resistor	8
10kΩ Resistor	1

This project uses the 02-light_depentent_array.ino sketch.

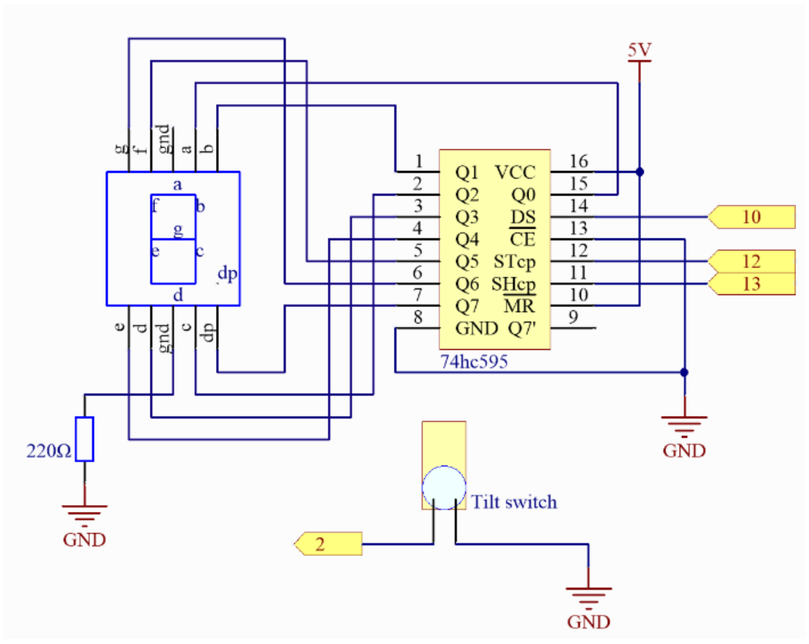
This project is utilizing two major components: a photoresistor and an LED. As mentioned previously, a photoresistor is a variable resistor where its resistance is dependent on the light it senses. The LED, as you may know, is a semiconducting diode. In this project, the photoresistor is programmed to light up and dim the LEDs in the array to imitate a bar that displays the amount of light that is reaching the photoresistor. Therefore, the more light the photoresistor senses, the more LEDs light up and vice versa.

Entertainment Project 3: Digital Dice



<u>Required Components:</u>

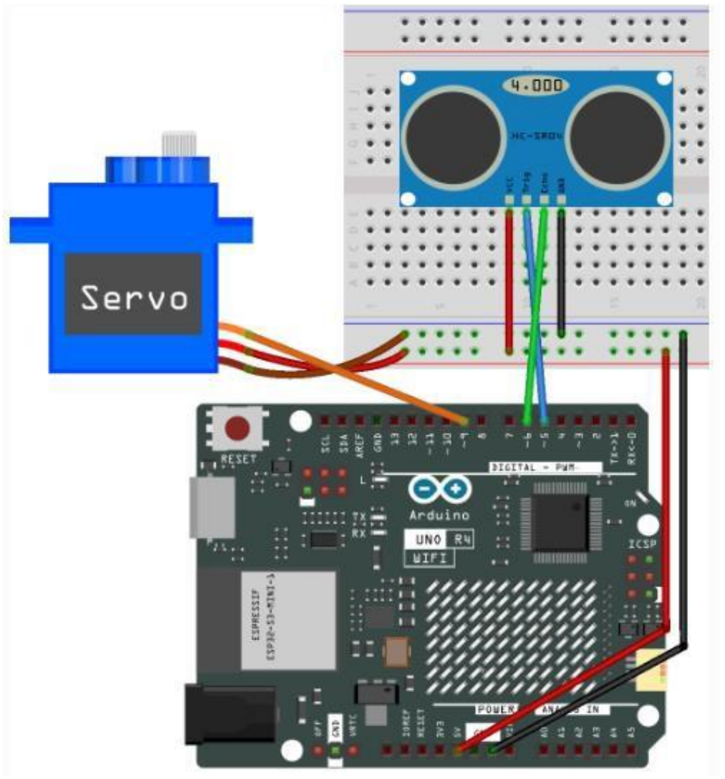
Arduino R4	1
Breadboard	1
Tilt Switch	1
7-Segment Display	1
74HC595	1
220Ω Resistor	1



This project uses the 03-digital_dice.ino sketch.

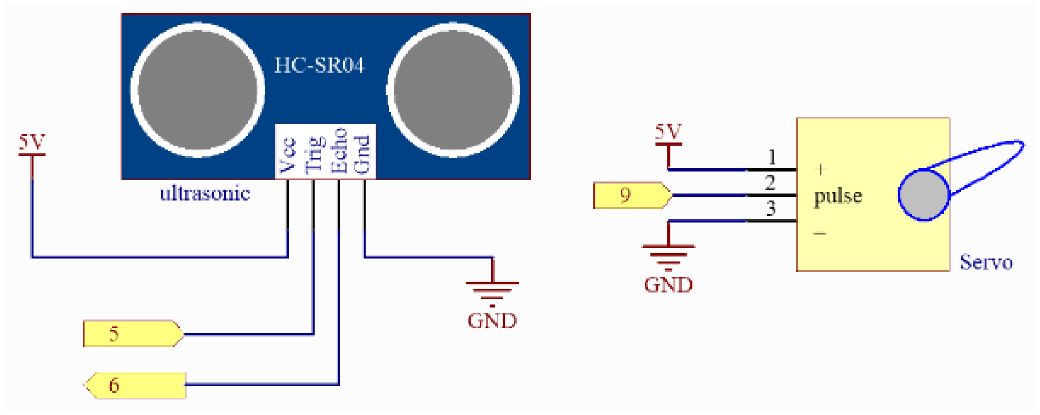
This project is utilizing three major components: a tilt switch, 74HC595 chip, and the seven-segment display. As covered, the tilt switch is a component containing a little ball, and its angle determines one of two outputs. The 74HC595 chip is a shift register where it takes an input and spreads them into multiple outputs. The seven-segment display is a display with seven LEDs that lights up a specific number at certain intervals. This project uses all three together to create a digital dice where shaking the tilt switch will imitate rolling of a dice, and upon stopping the shaking will output a number from 1 to 6.

Entertainment Project 4: Smart Servo



Required Components:

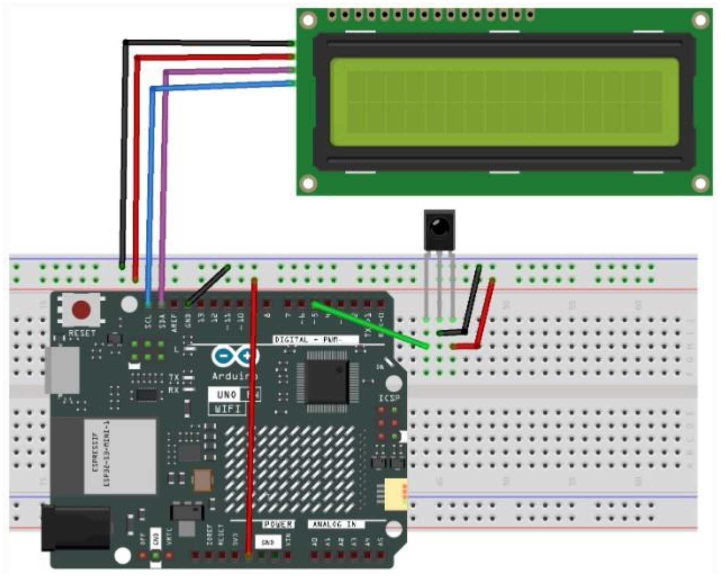
Arduino R4	1
Breadboard	1
Servo	1
Ultrasonic Sensor	1



This project uses the 04-smart_servo.ino sketch.

This project is utilizing two major components: a servo and an ultrasonic sensor. The servo is a micro servo motor, and an ultrasonic sensor is, basically, a distance-dependent sensor. This project is straightforward – when an object comes within 20cm of an ultrasonic sensor, the servo will rotate 90 degrees after a one second delay. Once the object is no longer within that distance threshold, the servo will rotate 90 degrees the other way. This code is simple and convenient for applications such as smart trash cans.

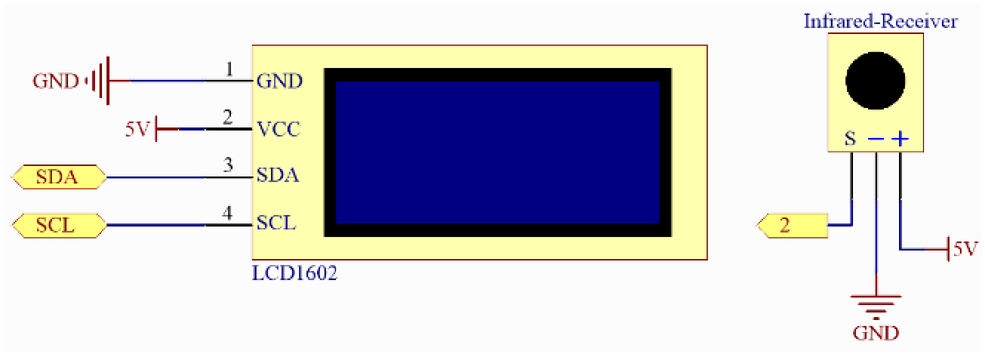
Entertainment Project 5: Guess Number



Required Components:

Arduino R4	1
Breadboard	1
IR Receiver Module	1
Remote Control	1
I2C LCD	1

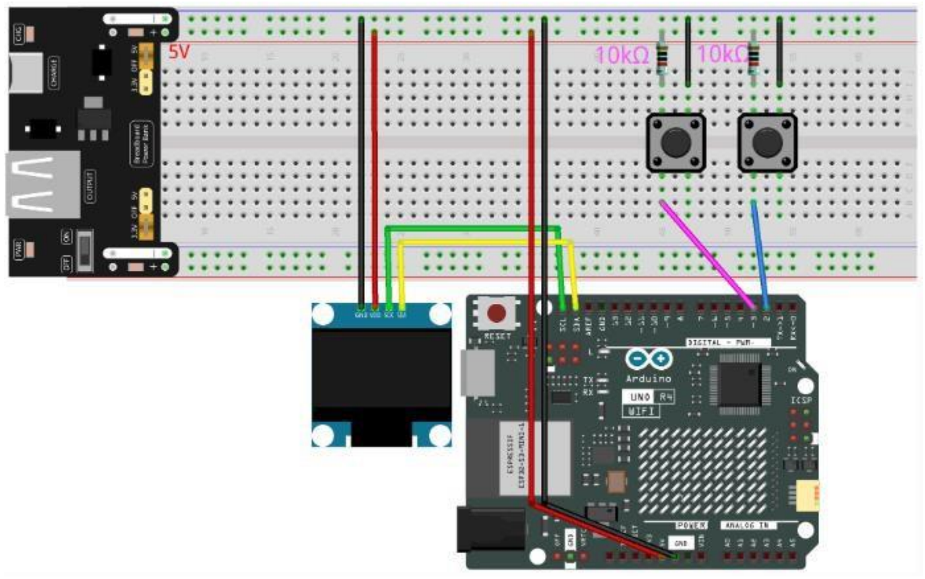
Note: Pay attention to IR Receiver Module's pinout. S – 2, - – GND, + – VCC.



This project uses the 05-guess_number.ino sketch.

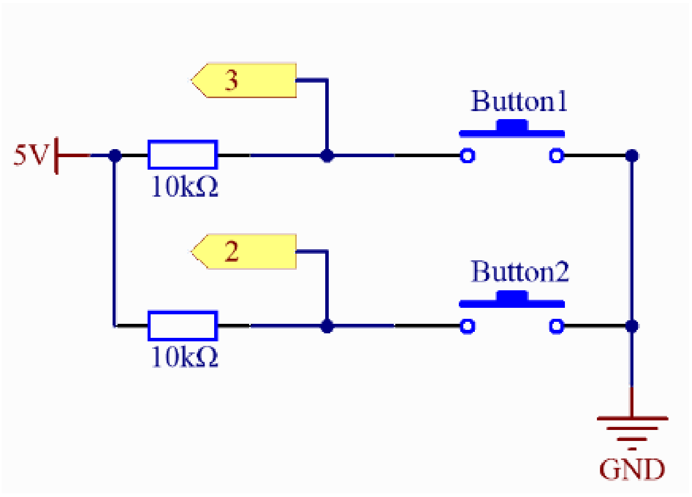
This project is utilizing two major components: the IR Receiver module with its remote, and the I2C LCD display. The premise is that you are trying to guess a number initially ranging through 1 – 99, as displayed on the LCD, by mashing your guesses on the remote as it's pointed to the receiver. For example, the lucky number you must guess is 60. If your input guess is 30 (less than the lucky number), the new range becomes 30 – 99. If your next guess is 70 (greater than the lucky number), the next range becomes 30 – 70. This continues until you input the lucky number. To input a one-digit guess, press the desired number and hit OK to register it. Pressing the asterisk (*) will reset the lucky number and the range.

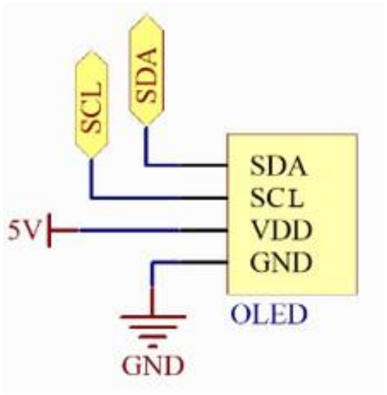
Entertainment Project 6: Pong



Required Components:

Arduino R4	1
Breadboard	1
OLED	1
Power Supply Module	1
Switches	2
10kΩ Resistor	2
9V Power Adapter	1





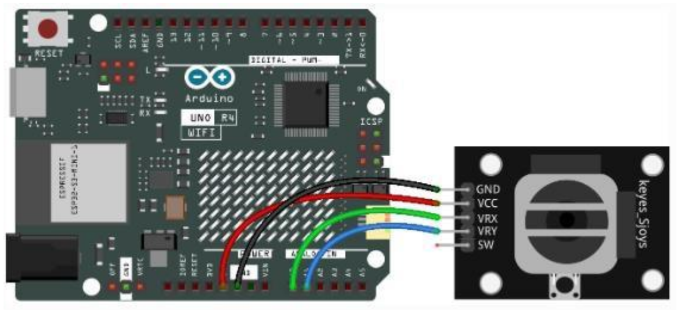
This project uses the 06-oled.ino sketch.

Pong is one of the earliest videogames that helped set foundation to the videogame industry. This project is a recreation of pong using the OLED for the screen and the two buttons for controls, where one moves the paddle up and the other moves it down. Each time the ball hits a paddle, its speed increases to maintain the game as challenging for both the player and the CPU as it’s programmed to always follow the ball at a fixed speed.

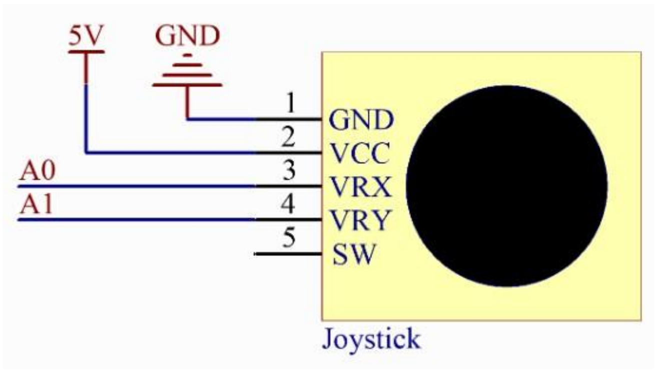
Entertainment Project 7: Snake

Required Components:

Arduino R4	1
Joystick Module	1

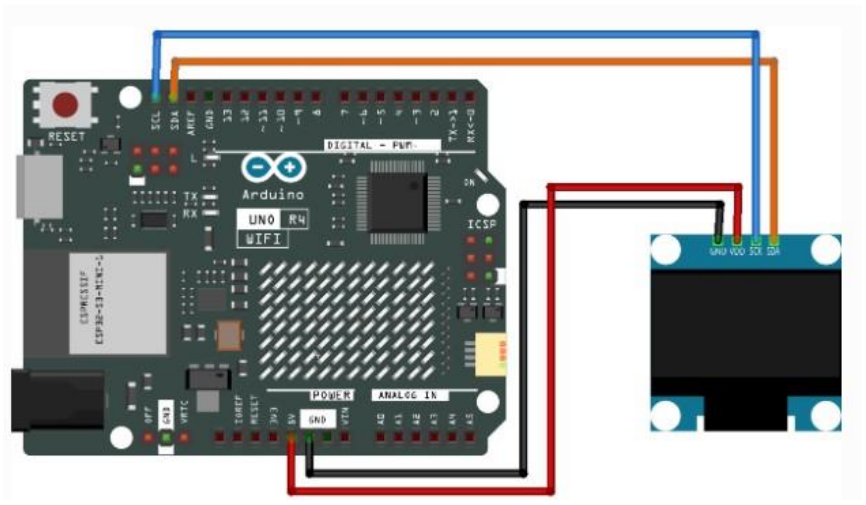


This project uses the 07-snake.ino sketch.



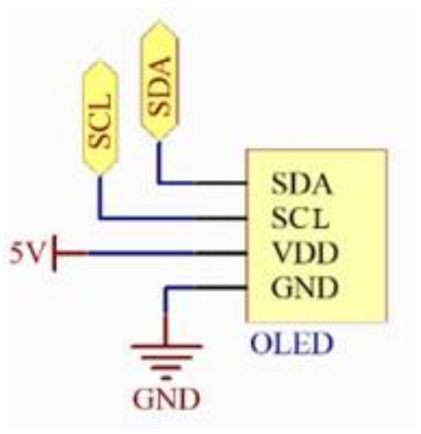
Snake is also one of the older games that are very well-known. This project uses the LED matrix on the Arduino R4 as the play area, while the joystick is utilized to control the snake. The goal of *Snake* is very simple – to last as long as possible collecting pellets that lengthen the snake with each one while avoiding it from colliding with itself. If the snake collides with itself, the game instantly restarts.

IoT Project 1: Weather & Time



Required Components:

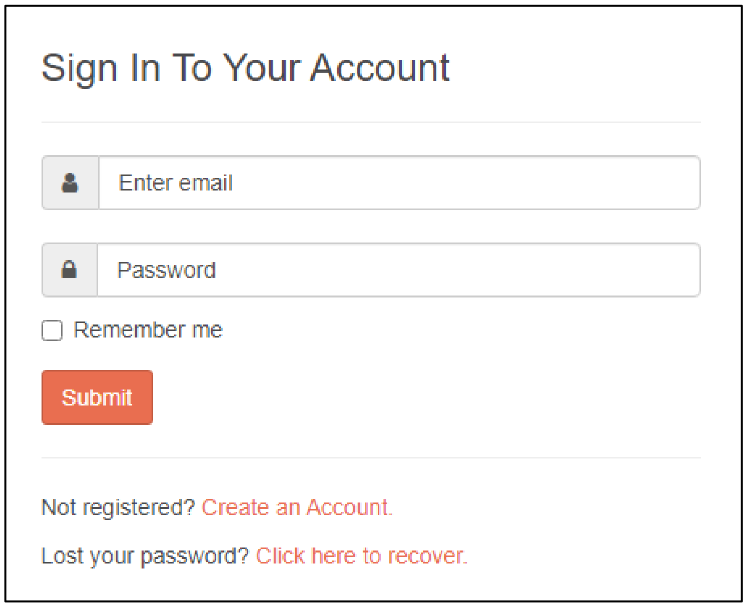
Arduino R4	1
OLED	1



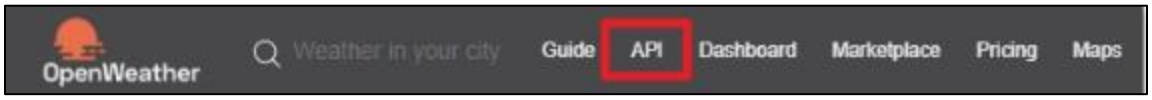
This project uses the 01-weather_oled.ino sketch.

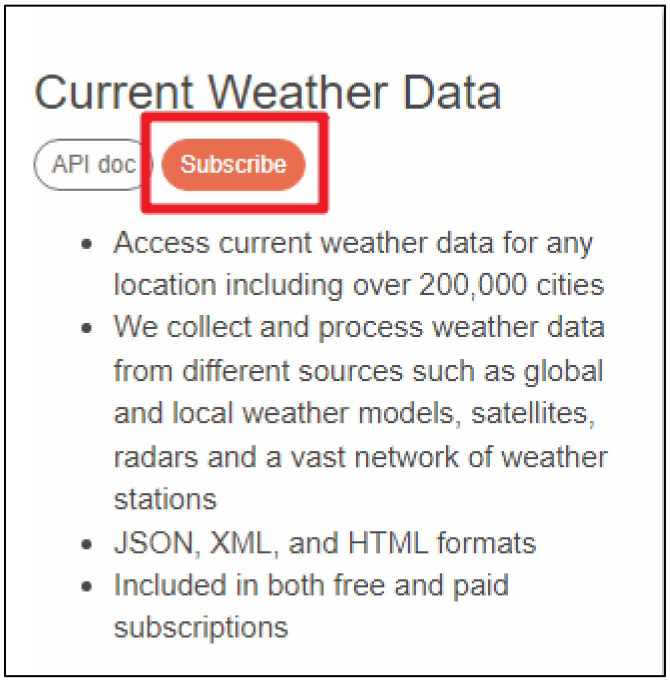
Note: to properly upload this code, install the ArduinoMqttClient, FastLED, Adafruit GFX, and Adafruit SSD1306 libraries.

For this project, you are required to create an OpenWeather account and obtain an API key. Visit the OpenWeather website and do so.


Sign In To Your Account
Enter email
Password
Remember me
Submit
Not registered? Create an Account.
Lost your password? Click here to recover.

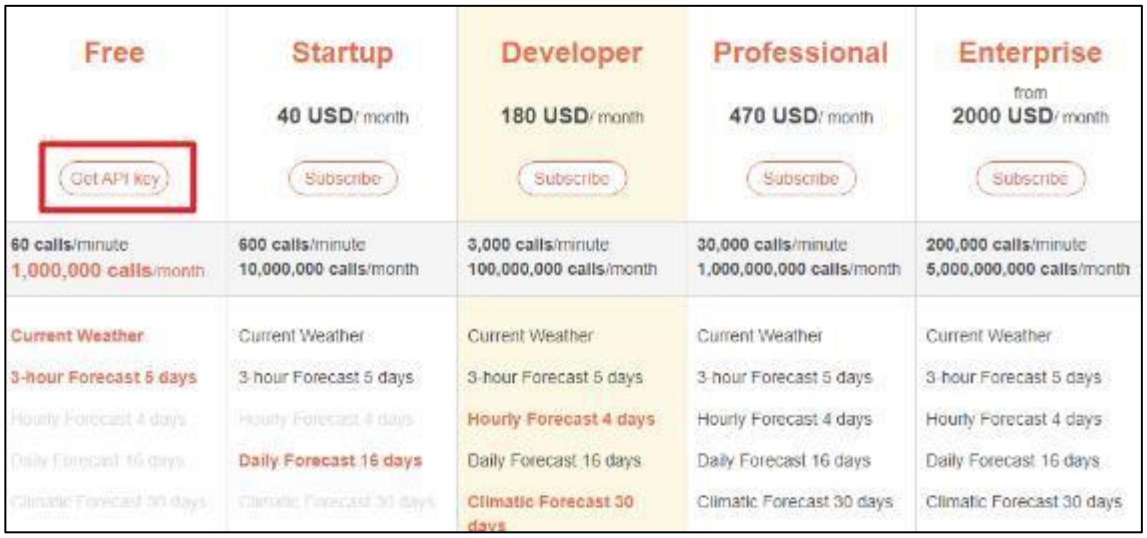
Once done, navigate to the API page and subscribe to it.





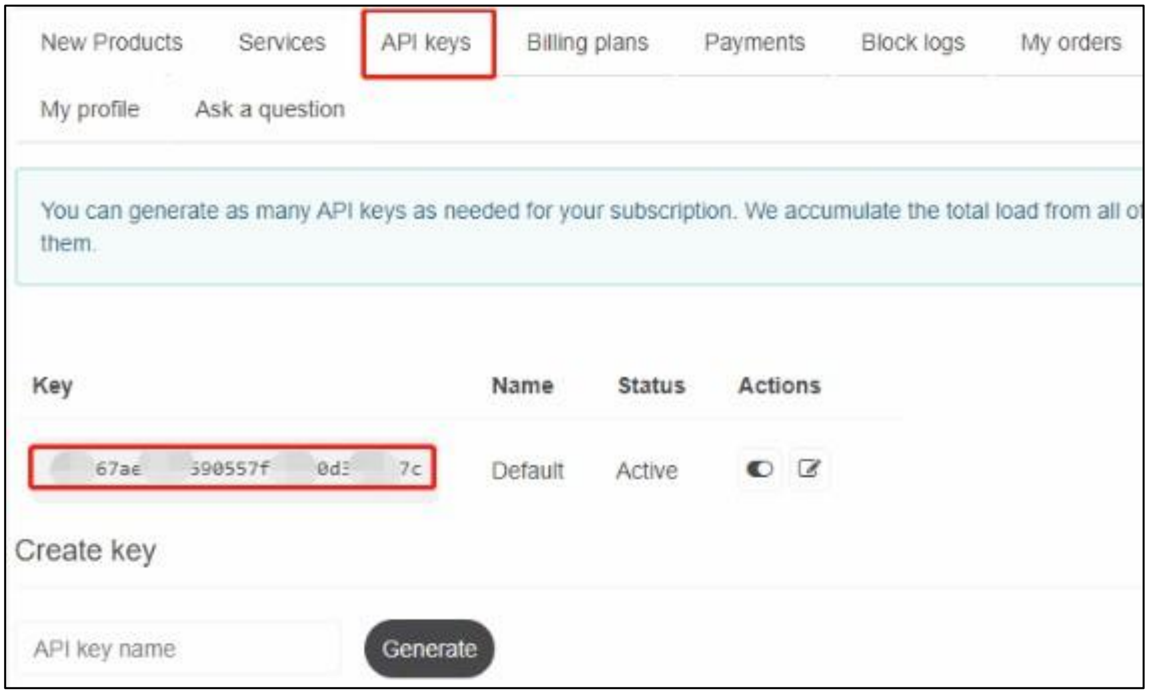
- Access current weather data for any location including over 200,000 cities
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations
- JSON, XML, and HTML formats
- Included in both free and paid subscriptions

Select the Free subscription.



Free	Startup	Developer	Professional	Enterprise
	40 USD/ month	180 USD/ month	470 USD/ month	from 2000 USD/ month
Get API key	Subscribe	Subscribe	Subscribe	Subscribe
60 calls/minute 1,000,000 calls/month	600 calls/minute 10,000,000 calls/month	3,000 calls/minute 100,000,000 calls/month	30,000 calls/minute 1,000,000,000 calls/month	200,000 calls/minute 5,000,000,000 calls/month
Current Weather	Current Weather	Current Weather	Current Weather	Current Weather
3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days
Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days
Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days
Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days

Navigate to API keys and copy the key generated.



New Products Services API keys Billing plans Payments Block logs My orders

My profile Ask a question

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions
[redacted]67ae[redacted]590557f[redacted]0d3[redacted]7c	Default	Active	

Create key

API key name Generate

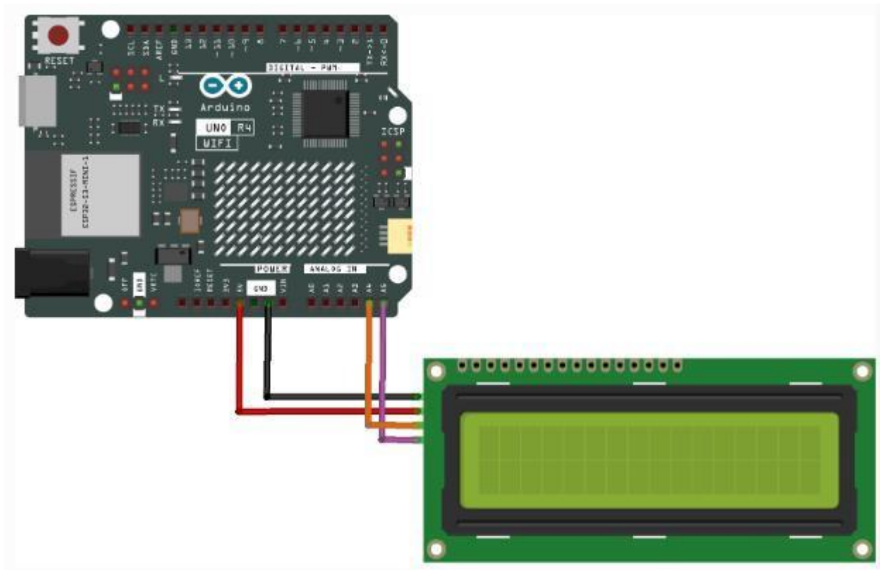
When copied, it must be pasted into the arduino_secrets.h file. At the same time, set it up with your Wi-Fi credentials and your location.

In the main code, timeClient.setTimeOffset() line must be modified to your time zone. If your time zone is GMT +1, for example, then in brackets you put 3600 * 1.

```
timeClient.setTimeOffset(3600 * 1);  // Adjust for your time zone (this is +1 hour)
```

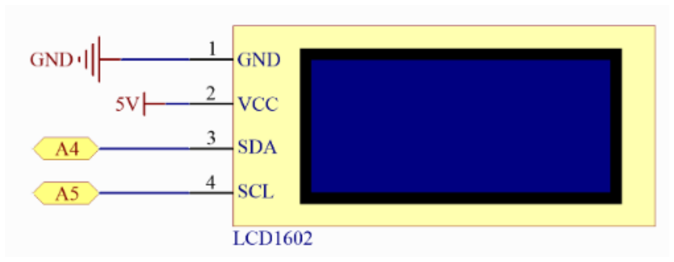
Upon uploading the code, the OLED will display the time and weather information.

IoT Project 2: Bluetooth LCD Text



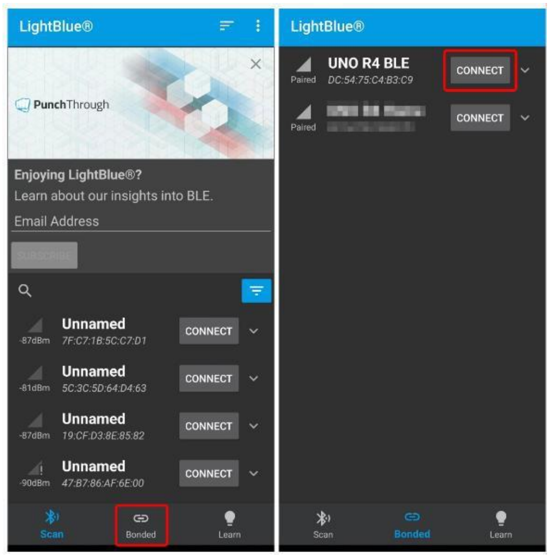
Required Components:

Arduino R4	1
I2C LCD	1

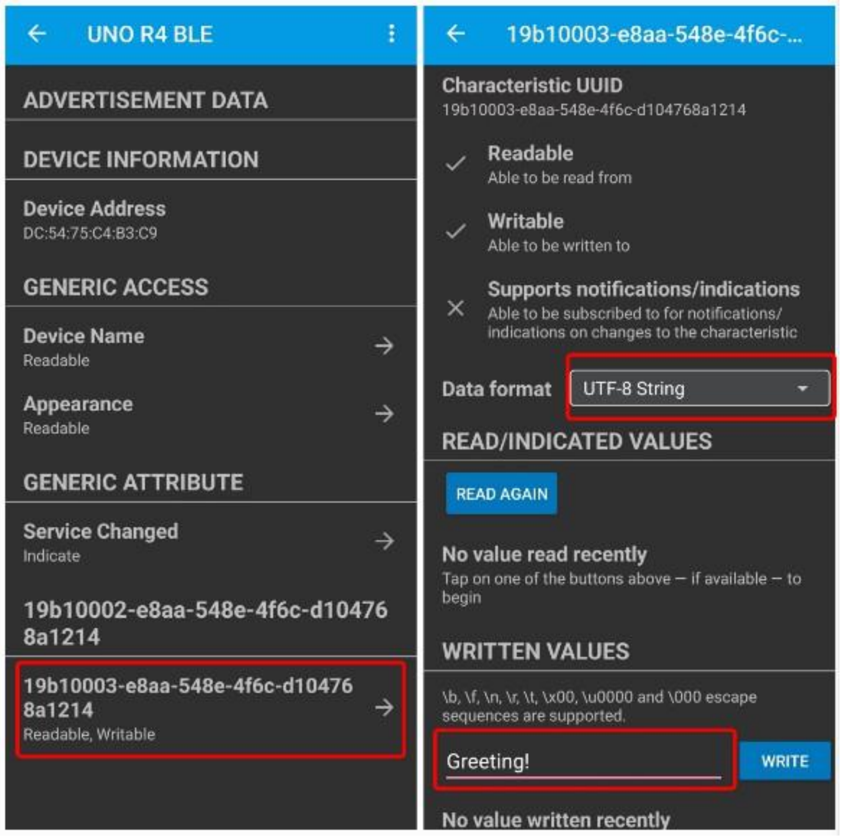


This project uses the 02-lightblue_lcd.ino sketch.

Assuming you already have the LightBlue app installed, go ahead and connect to your Arduino device through it.



Once connected, view the device details and scroll until you find its service UUID. It should be readable and writable. In the next window, change its data format to a UTF-8 String. Any text you write in the written values section will display on the LCD.


UNO R4 BLE

ADVERTISEMENT DATA

DEVICE INFORMATION

Device Address
DC:54:75:C4:B3:C9

GENERIC ACCESS

Device Name
Readable

Appearance
Readable

GENERIC ATTRIBUTE

Service Changed
Indicate

19b10002-e8aa-548e-4f6c-d104768a1214

19b10003-e8aa-548e-4f6c-d104768a1214
Readable, Writable

19b10003-e8aa-548e-4f6c-...

Characteristic UUID
19b10003-e8aa-548e-4f6c-d104768a1214

Readable
Able to be read from

Writable
Able to be written to

Supports notifications/indications
Able to be subscribed to for notifications/indications on changes to the characteristic

Data format: UTF-8 String

READ/INDICATED VALUES

READ AGAIN

No value read recently
Tap on one of the buttons above – if available – to begin

WRITTEN VALUES

\b, \f, \n, \r, \t, \x00, \u0000 and \000 escape sequences are supported.

Greeting!

WRITE

No value written recently