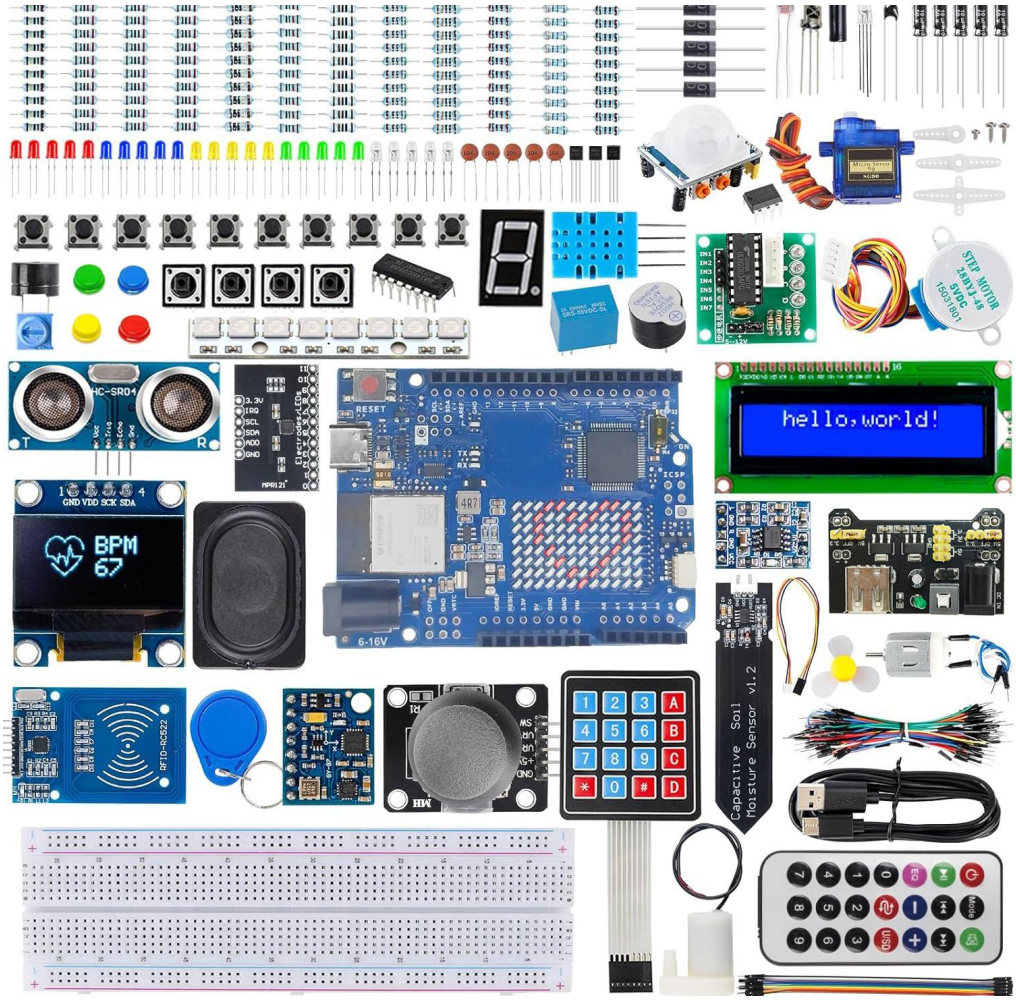
Elite Explorer Kit for Arduino Uno



V1.0.24.11.16

Content

Lesson 1 Installing IDE.....3

Lesson 2 Add Libraries..... 9

Lesson 3 Blink..... 11

Lesson 4 Flashing lights..... 17

Lesson 5 Button Control LED Delay Switch..... 19

Lesson 6 Active buzze..... 26

Lesson 7 LED water lamp experiment..... 29

Lesson 8 Infrared module controls LED lights..... 32

Lesson 9 Ultrasonic module experiment..... 36

Lesson 10 Soil temperature and humidity module experiment..... 39

Lesson 11 DHT11 temperature module experiment..... 42

Lesson 12 LCD1602 I2C screen experiment..... 45

Lesson 13PS2 remote sensing module experiment..... 47

Lesson 1 Installing IDE

Introduction

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

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

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

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



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

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



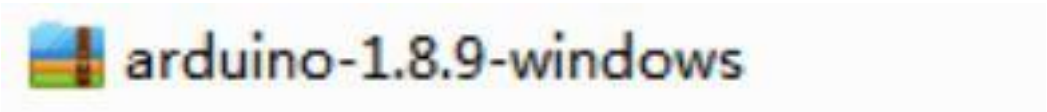
Click Windows Installer.



Click JUST DOWNLOAD.

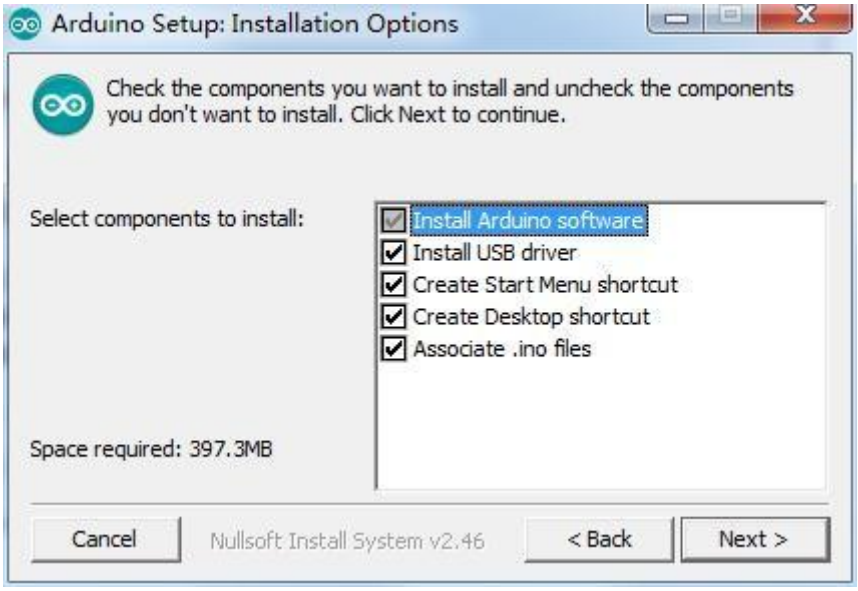
Also version 1.8.9is available in the material we provided, and the versions of our materials are the latest versions when this course was made.

Installing Arduino (Windows)

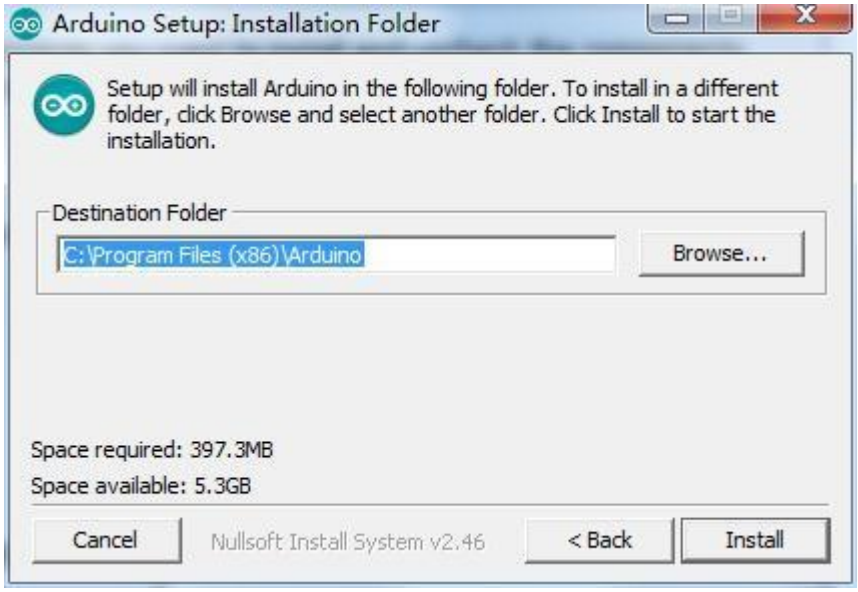




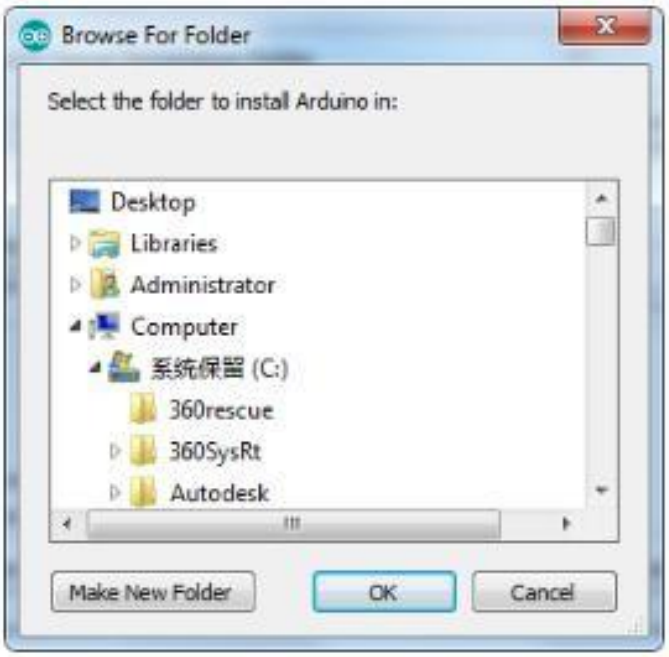
Click I Agree to see the following interface



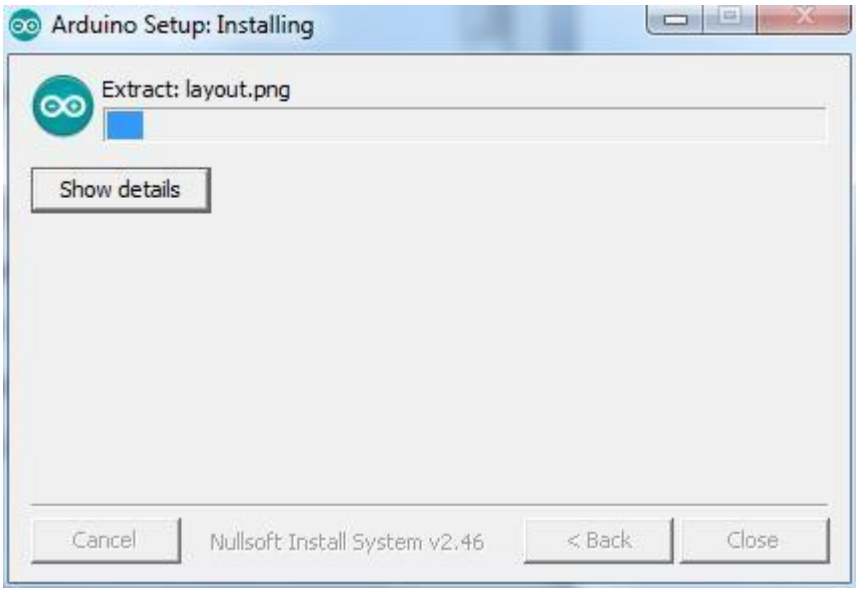
Click Next



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



Click Install to initiate installation



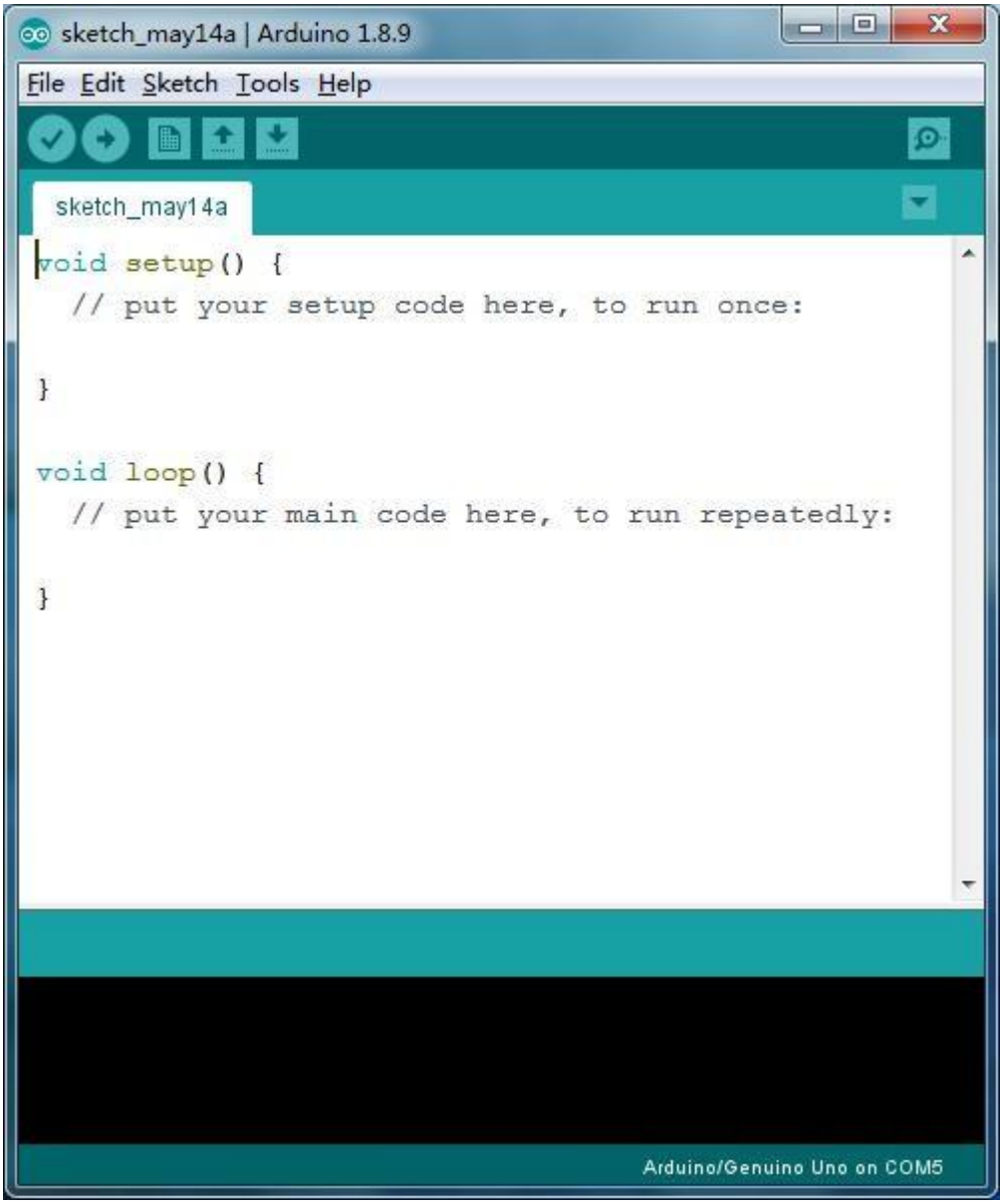
Finally, the following interface appears, click Install to finish the installation.



Next, the following icon appears on the desktop



Double-click to enter the desired development environment



Installing Arduino (Mac OS X)

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

Installing Arduino (Linux)

You will have to use the make install command. If you are using the Ubuntu system, it is recommended to install Arduino IDE from the software center of Ubuntu.

Lesson 2 Add Libraries

Installing Additional Arduino Libraries

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

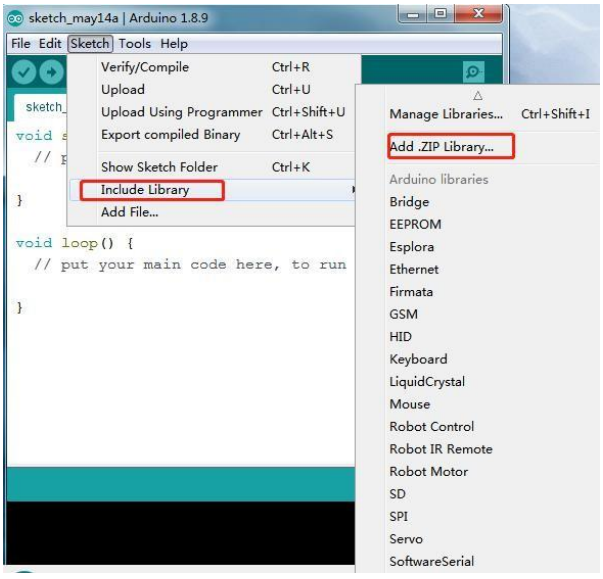
What are Libraries?

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

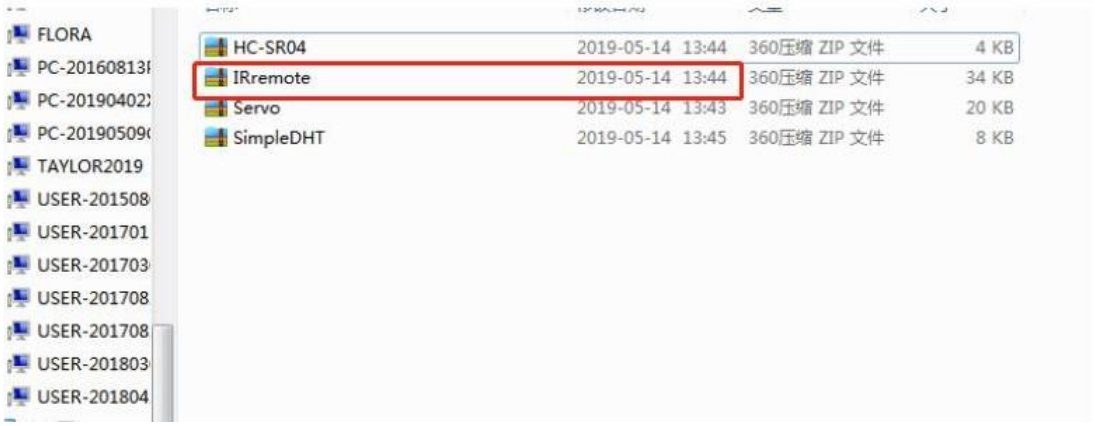
How to Install a Library

Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.9). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



and then Include Library > Manage Libraries.



Example: IRromote

Open arduino software - project - load library - add a .zip library

Add method two:

Copy the library folder to the Libraries folder in the Arduino installation directory. Restart Arduino and the added library will take effect.

Lesson 3 Blink

Overview:

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

Component Required:

1x NANO

Principle:

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

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



You may find that your UNO R3 board's 'L' LED already blinks when you connect it to a USB plug.

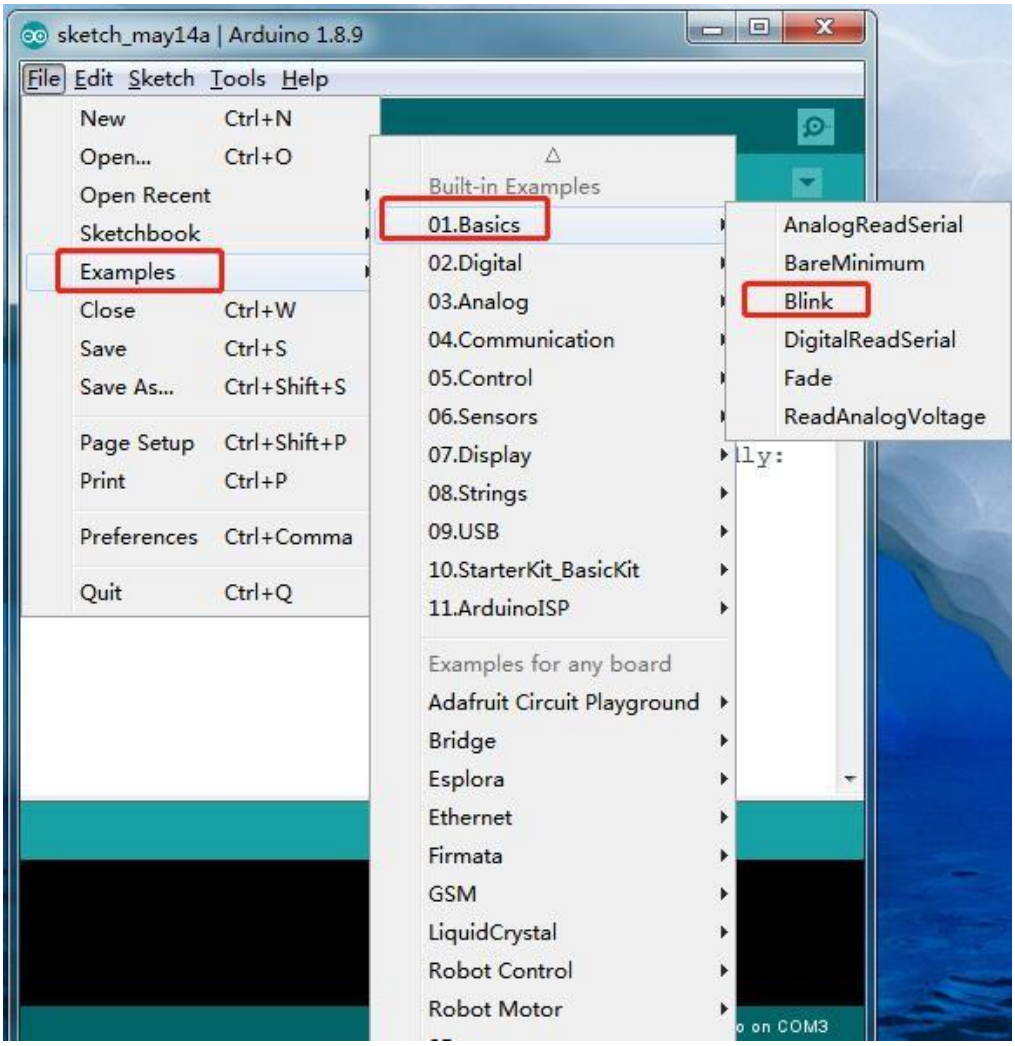
This is because the boards are generally shipped with the 'Blink' sketch pre-installed.

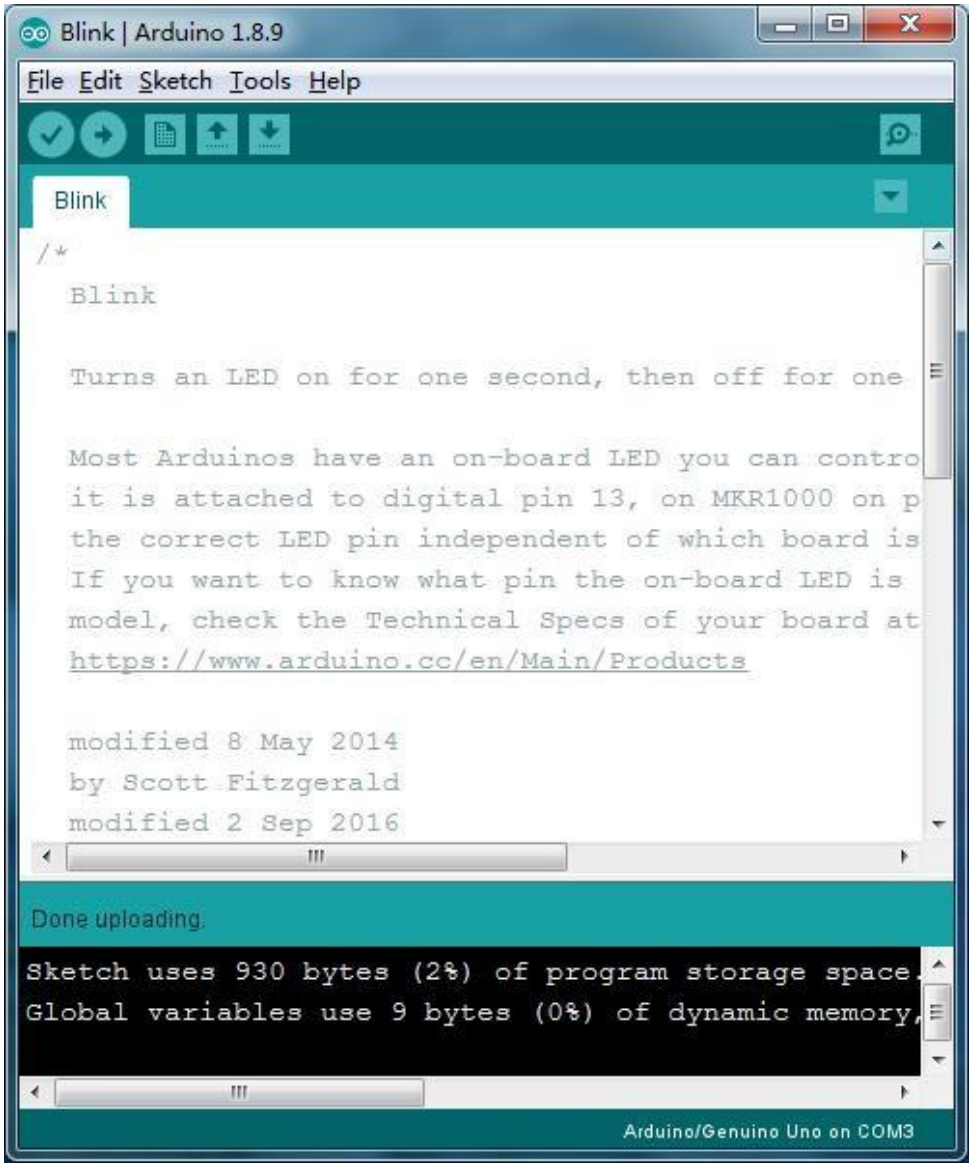
In this lesson, we will reprogram the UNO R3 board with our own Blink sketch and then change the rate at which it blinks.

In Lesson 1, you set up your Arduino IDE and made sure that you could find the right serial port for it to connect to your UNO R3 board. The time has now come to put that connection to the test and program your UNO R3 board.

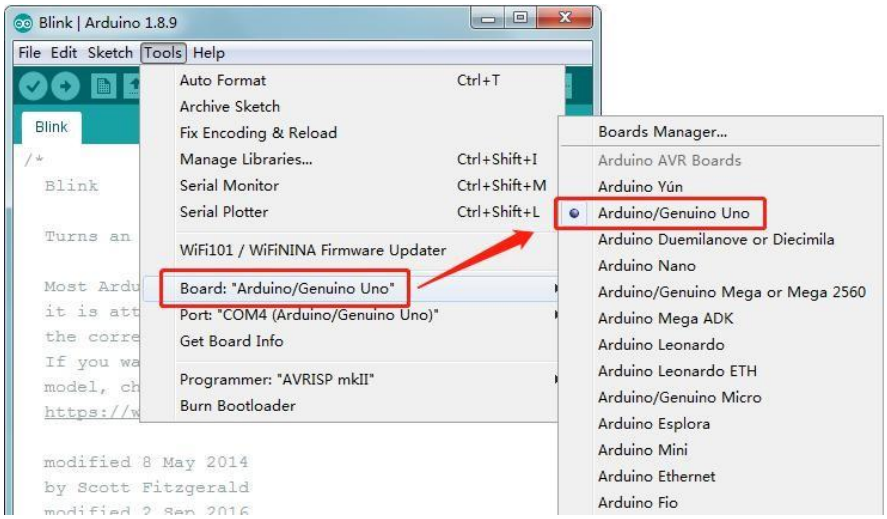
The Arduino IDE includes a large collection of example sketches that you can load up and use. This includes an example sketch for making the 'L' LED blink.

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



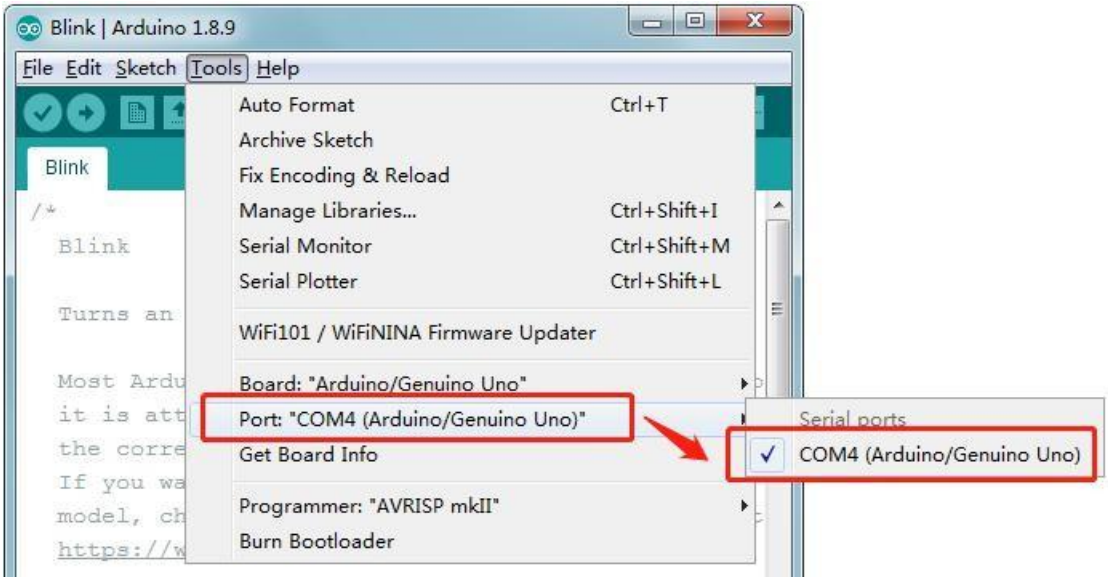


Upload code needs to select the board model Arduino/Genuino Uno



The upload code also needs to select the board port number. Our computer displays the port number of arduino as: "COM4 Arduio / Genuino Uno", you may be the other serial port number.

Note: A correct COM port should be COMX (arduino) XXX), which is certified by the standard.



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

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

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

The first line of code is:

int led = 13;

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

Next, we have the 'setup' function. Again, as the comment says, this is executed when the reset

button is pressed. It is also executed whenever the board resets for any reason, such as power first being applied to it, or after a sketch has been uploaded.

```
void setup()
  {
// initialize the digital pin as an output.
  pinMode(13, OUTPUT);
}
```

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

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

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

```
void loop() {
digitalWrite(13, HIGH);     // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
digitalWrite(13, LOW);      // turn the LED off by making the voltage LOW
delay(1000);        // wait for a second
}
```

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

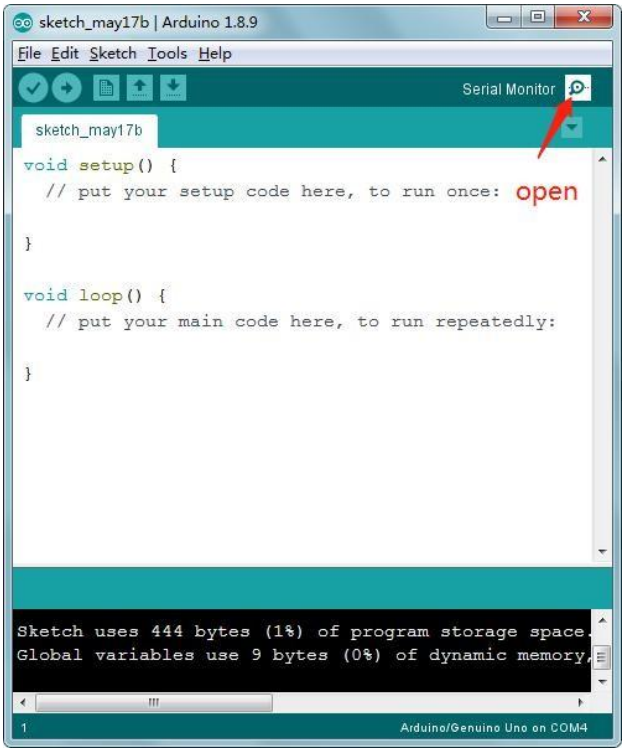
You are now going to make your LED blink faster. As you might have guessed, the key to this lies in changing the parameter in () for the 'delay' command.

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                       // wait for a second
}
```

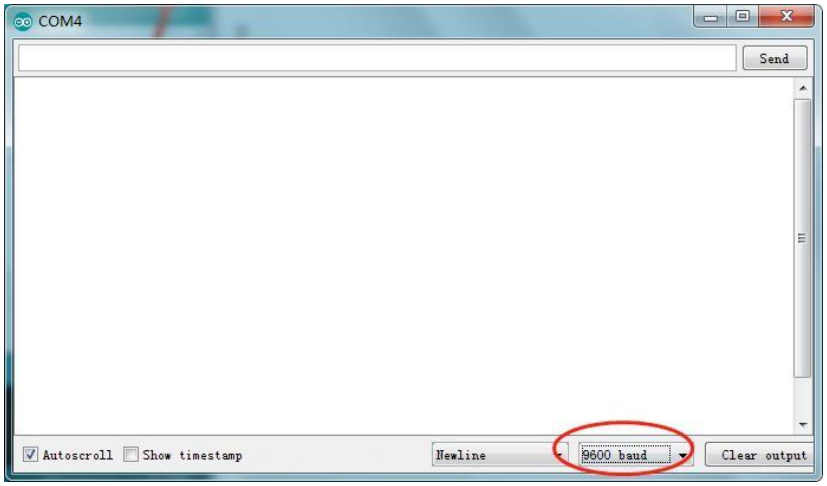
This delay period is in milliseconds, so if you want the LED to blink twice as fast, change the value from 1000 to 500. This would then pause for half a second each delay rather than a whole second.

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

Note: I want to add some information about how to open the serial monitor here.



Serial port is generally selected 9600HZ



Lesson 4 Flashing lights

Overview:

In this lesson, we expand on our previous LED lighting experiment. This time, instead of using the built-in LED, we will connect an external 5mm red LED to the development board and control it programmatically. This activity helps to enhance our hands-on circuit-building skills.

Wiring schematic:

- **1 x UNO R4 WIFI**
- **1 x 5mm red LED**
- **1 x 220-ohm resistor**
- **1 x 400 Tie Points Breadboard**
- **2 x M-M jumper wires (Male-to-Male wires)**

Wiring Schematic:

Connect the components as follows:

1. **UNO R4 WIFI Pin 8** → **Anode (long leg) of the LED** via a 220-ohm resistor.
2. **Cathode (short leg) of the LED** → **GND on the UNO R4 WIFI**.

Physical wiring diagram:



Code:

```
void setup() {
        pinMode(8,OUTPUT);
}

void loop() {
   digitalWrite(8,HIGH);
   delay(1000);
   digitalWrite(8,LOW);
   delay(1000);
}
```

Lesson 5 Button Control LED Delay Switch

Overview:

In this lesson, you'll learn how to use a button to control an LED with a delay function using the Arduino UNO R4 WiFi. The button will control the LED, and a delay will be applied to simulate a switch delay before turning the LED on or off.

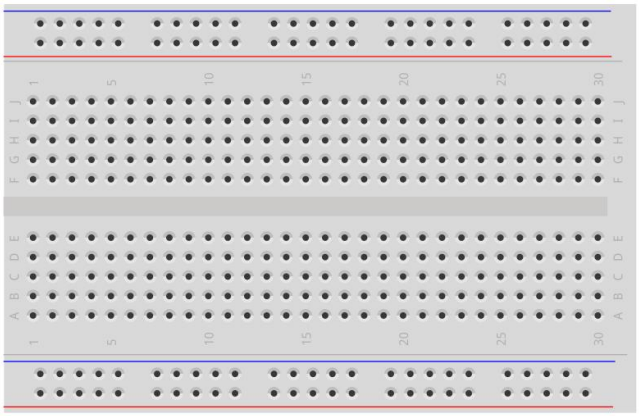
Component Required:

- 1 x UNO R4 WiFi
- 1 x 5mm Red LED
- 1 x 220-ohm Resistor
- 1 x 10k-ohm Resistor
- 1 x 400 Tie Points Breadboard
- 1 x Button
- 5 x M-M Wires (Male-to-Male jumper wires)

Component Introduction:

BREADBOARD MB-102：

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



Breadboards come in various sizes and configurations. The simplest kind is just a grid of holes in a plastic block. Inside are strips of metal that provide electrical connection between holes in the shorter rows. Pushing the legs of two different components into the same row joins them together electrically. A deep channel running down the middle indicates that there is a break in connections there, meaning, you can push a chip in with the legs at either side of the channel without connecting them together. Some breadboards have two strips of holes running along the long edges of the board that are separated from the main grid. These have strips running down the length of the board inside and provide a way to connect a common voltage. They are usually in pairs for +5 volts and ground. These strips are referred to as rails and they enable you to connect power to many components or points in the board.

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

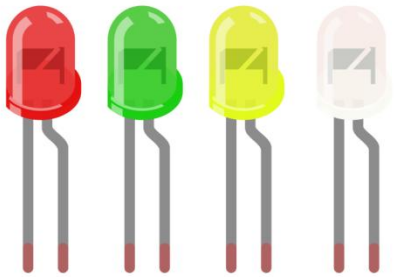
LED:

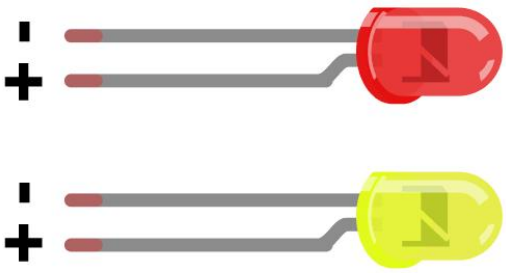
LED (Light Emitting Diode), which converts electrical energy into light energy, also has unidirectional conductivity and a reverse breakdown voltage of about 5V. Its forward volt-ampere characteristic curve is very steep, and the current-limiting resistor must be connected in series. In a 5V circuit, a resistor of about 400 ohms is generally used. The longer of the two pins of the LED is the positive pole. There are two ways to connect, when the positive pole of the led through the current limiting resistor and UNO R3

The I/O port is connected and the other end is grounded. At this time, when the UNO R3 output is high, the led is lit, and when the output is low, the led is off.

2. When the negative pole of the led is connected to the I/O port of the UNO R3, the

other end is connected to the 5V voltage through the current limiting resistor. At this time, UNO R3 When the output is low, the led is lit, and when the output is high, the led is off.





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

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

Firstly, the positive lead is longer.

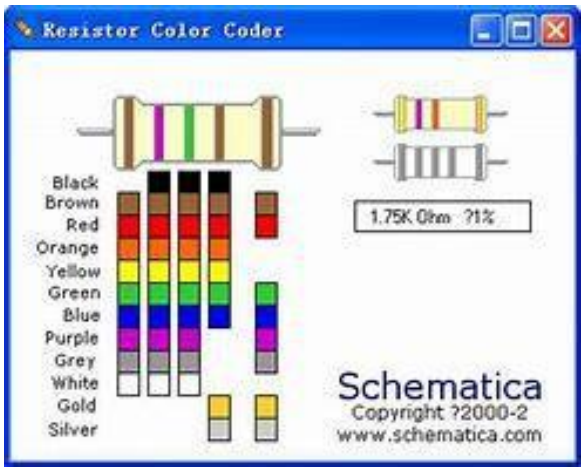
Secondly, where the negative lead enters the body of the LED, there is a flat edge to the case of the LED.

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

RESISTORS:

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



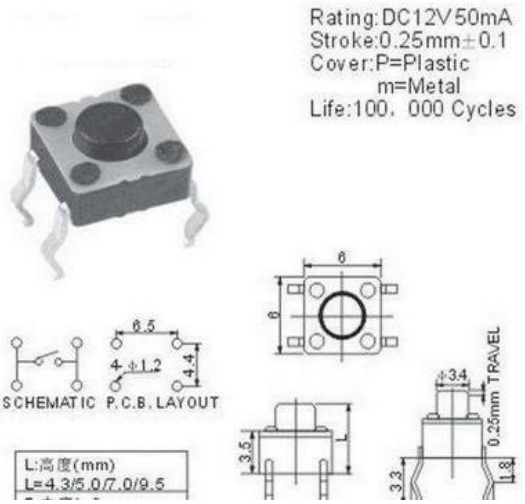


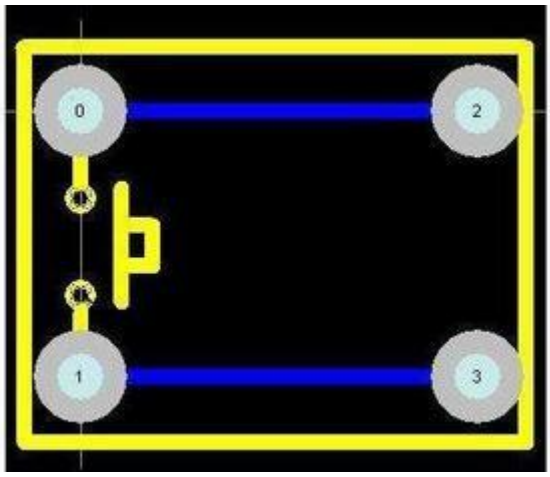
Unlike LEDs, resistors do not have a positive and negative lead. They can be connected either way around.

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

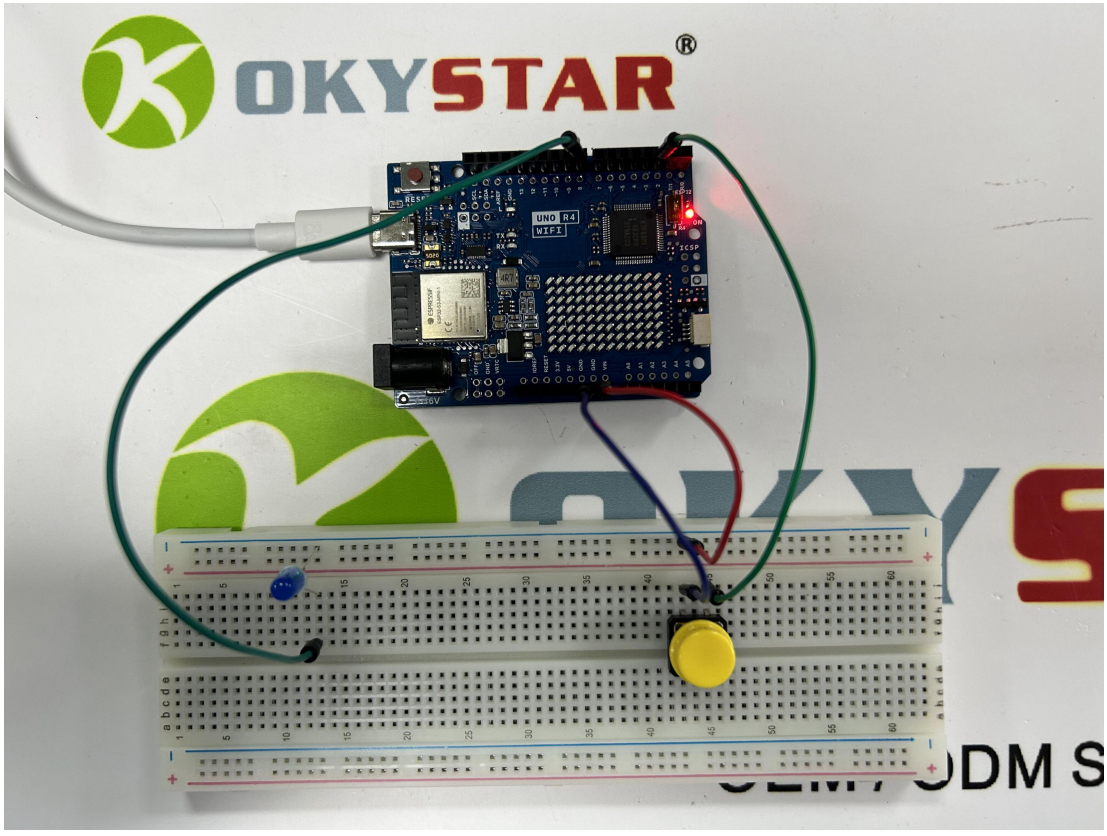
Button:

The button as an I/O port input device has seen the effect of the last use. It is actually used to control the on/off of the line, and the effect of changing the I/O port signal is achieved by adding a device such as a pull-up resistor.





Physical wiring diagram:



Code:

```
#define BUTTON_PIN 2    // Button connected to D2 pin
#define LED_PIN 8       // LED connected to D8 pin

void setup() {
  pinMode(LED_PIN, OUTPUT);              // Set LED pin as output
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Set button pin as input with internal pull-up resistor

  // Start serial communication with baud rate 9600
  Serial.begin(9600);
}

void loop() {
  int buttonState = digitalRead(BUTTON_PIN);    // Read the button state

  if (buttonState == LOW) {    // When the button is pressed (LOW due to pull-up resistor)
    digitalWrite(LED_PIN, HIGH);    // Turn on the LED
    Serial.println("LED ON");          // Print to the serial monitor
  } else {
    digitalWrite(LED_PIN, LOW);      // Turn off the LED
    Serial.println("LED OFF");        // Print to the serial monitor
  }

  delay(100);    // Add a small delay to avoid button bounce issues
}
```

Code Breakdown:

- Setup:
  - `pinMode(LED_PIN, OUTPUT)` sets the LED pin as an output pin.
  - `pinMode(BUTTON_PIN, INPUT_PULLUP)` enables the internal pull-up resistor for the button. The button will read LOW when pressed and HIGH when not pressed.
  - `Serial.begin(9600)` initializes serial communication at 9600 baud rate.

- Loop:
  - `digitalRead(BUTTON_PIN)` reads the state of the button.
  - If the button is pressed (reads LOW due to pull-up), the LED is turned on, and a message is printed to the serial monitor.
  - If the button is not pressed (reads HIGH), the LED is turned off, and the message "LED OFF" is printed.
  - `delay(100)` adds a short delay to prevent button debounce (false triggering).

Testing the Circuit:

1. Press the Button: The LED should light up and the serial monitor will display "LED ON".
2. Release the Button: The LED should turn off and the serial monitor will display "LED OFF".

Notes:

- Button Debounce: A small delay (like `delay(100)`) helps to prevent multiple readings from the button being triggered due to mechanical bouncing.
- LED Behavior: The LED will only turn on while the button is pressed and turn off when it is released.

Conclusion:

In this lesson, you have learned how to control an LED with a button using the UNO R4 WiFi. You also applied the concept of using internal pull-up resistors and simple serial communication to monitor the LED status in real time.

This exercise is a great foundation for more advanced projects where you can use buttons and LEDs to interact with your Arduino projects!

Lesson 6 Active buzzer

Overview：

In this lesson, you will learn how to generate a sound with an active buzzer using the UNO R4 WiFi.

Component Required:

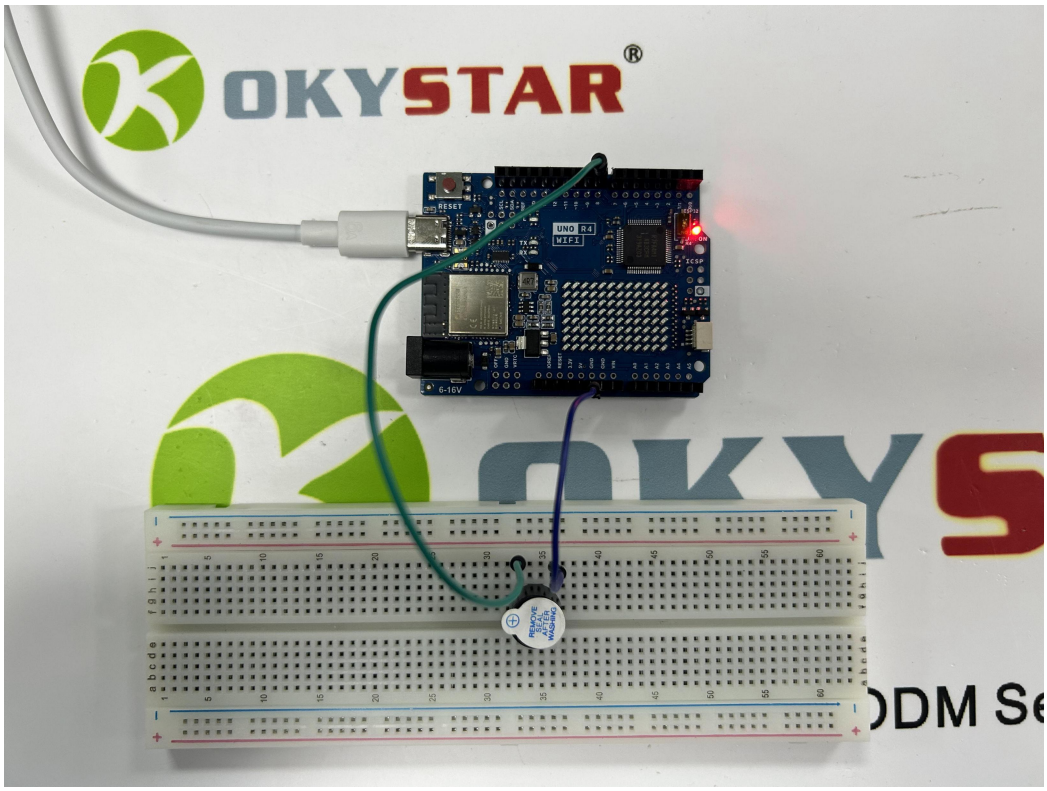
- 1 x UNO R4 WiFi
- 1 x Active Buzzer
- 1 x 400 Tie Points Breadboard
- 2 x M-M Wires (Male-to-Male Jumper Wires)

BUZZER:

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



Physical wiring diagram:



Code：

```
#define BUZZER_PIN 8    // Buzzer connected to pin D8

// Define some common note frequencies (in Hertz)
int melody[] = {
   262, 294, 330, 349, 392, 440, 494, 523, 587, 659, 698, 784
};

int noteDuration = 500;    // Duration of each note (milliseconds)

void setup() {
   pinMode(BUZZER_PIN, OUTPUT);    // Set buzzer pin as OUTPUT
}

void loop() {
   // Play the melody in a loop
   for (int i = 0; i < 12; i++) {
      tone(BUZZER_PIN, melody[i]);      // Play the current note
      delay(noteDuration);                  // Wait for the note to finish
```

```
      noTone(BUZZER_PIN);                  // Stop the tone
      delay(50);                           // Short pause between notes
   }

   delay(1000);    // Wait 1 second before replaying the melody
}
```

Notes:

1. Ensure the **UNO R4 WiFi** board is powered via USB or an external power source.
2. Use the **Arduino IDE** to upload the code to the board.
3. Once the program is uploaded, you should hear the buzzer play the melody repeatedly.

Lesson 7 LED water lamp experiment

Overview：

In this lesson, you will create a Running LED effect where four LEDs light up sequentially, simulating a "chasing" effect, using the UNO R4 WiFi.

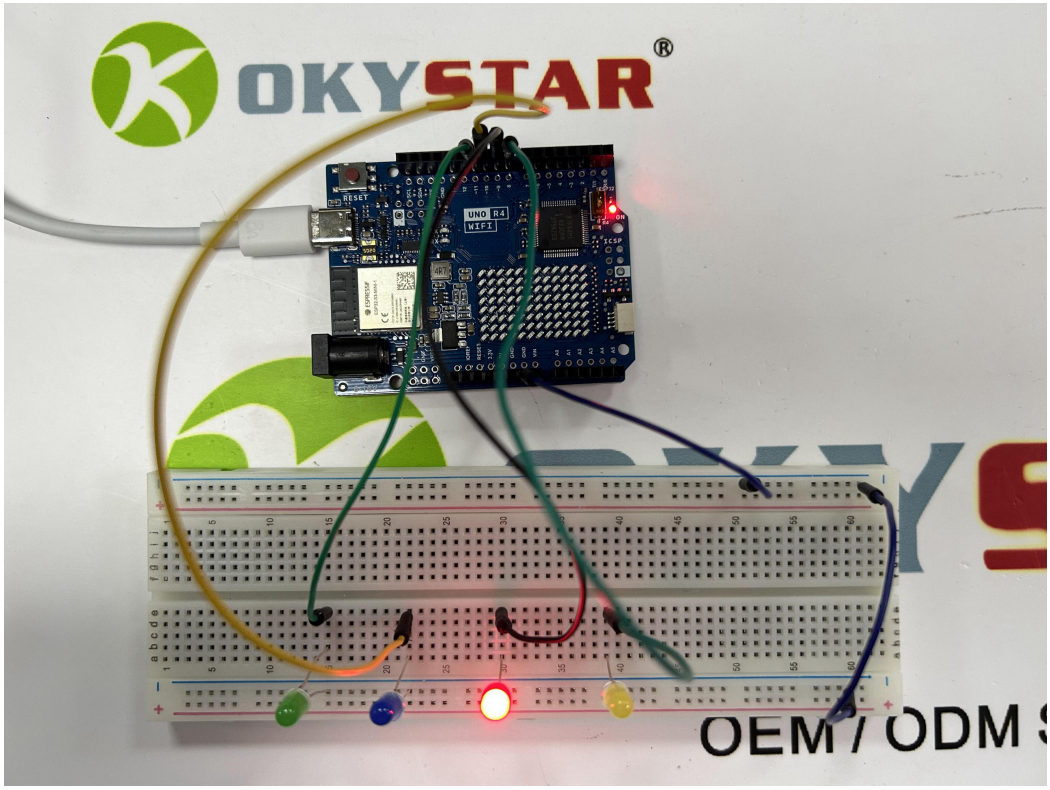
Component Required:

- 1 x UNO R4 WiFi
- 4 x LEDs
- 4 x 220Ω Resistors (to limit current for LEDs)
- 1 x 400 Tie Points Breadboard
- 4 x M-M Wires (Male-to-Male Jumper Wires)

LED:

LEDs (Light Emitting Diodes) are semiconductor devices that emit light when current flows through them. In this experiment, we will control four LEDs connected to different digital pins on the UNO R4 WiFi to create a running or chaser effect.

Physical wiring diagram:



Wiring Overview:

LED Pin	UNO R4 WiFi Pin	Resistor Value
LED 1	D8	220Ω
LED 2	D9	220Ω
LED 3	D10	220Ω
LED 4	D11	220Ω

Code:

```
void setup() {
  // Set pins as OUTPUT
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop() {
  // Run the LEDs in sequence (chaser effect)
  digitalWrite(8, HIGH);   // LED1 on
  digitalWrite(9, LOW);    // LED2 off
  digitalWrite(10, LOW);   // LED3 off
  digitalWrite(11, LOW);   // LED4 off
  delay(500);              // Wait for 500ms
  digitalWrite(8, LOW);    // LED1 off
  digitalWrite(9, HIGH);   // LED2 on
  digitalWrite(10, LOW);   // LED3 off
  digitalWrite(11, LOW);   // LED4 off
  delay(500);              // Wait for 500ms
  digitalWrite(8, LOW);    // LED1 off
  digitalWrite(9, LOW);    // LED2 off
```

```
    digitalWrite(10, HIGH); // LED3 on
    digitalWrite(11, LOW);    // LED4 off
    delay(500);                    // Wait for 500ms
    digitalWrite(8, LOW);      // LED1 off
    digitalWrite(9, LOW);      // LED2 off
    digitalWrite(10, LOW);    // LED3 off
    digitalWrite(11, HIGH); // LED4 on
    delay(500);                    // Wait for 500ms
}
```

Explanation:

Pin Mode Setup: In the `setup()` function, the pins **D8**, **D9**, **D10**, and **D11** are set to `OUTPUT` mode. This allows the microcontroller to control the LEDs connected to these pins.

LED Chaser Effect: In the `loop()` function, the LEDs are turned on one by one in a sequence with a delay of 500 milliseconds between each change. This creates the "chasing" effect, where one LED turns on, followed by the next in the sequence, creating the appearance of motion.

Delays: The `delay(500)` function is used to provide a pause between each LED state change, giving the effect of LEDs moving across the board.

Notes:

1. **UNO R4 WiFi:** Ensure the **UNO R4 WiFi** board is powered via USB or an external power source.
2. **LED Resistors:** Each LED should be connected with a **220Ω resistor** to prevent damaging the LED with excessive current.
3. **Arduino IDE:** Upload the code using the **Arduino IDE** to the **UNO R4 WiFi**.
4. The effect will repeat indefinitely, creating a continuous running LED pattern.

Lesson 8 Infrared module controls LED lights

Overview：

In this lesson, you will use an infrared (IR) receiver module to control an LED. The LED toggles between ON and OFF states each time an IR signal is received.

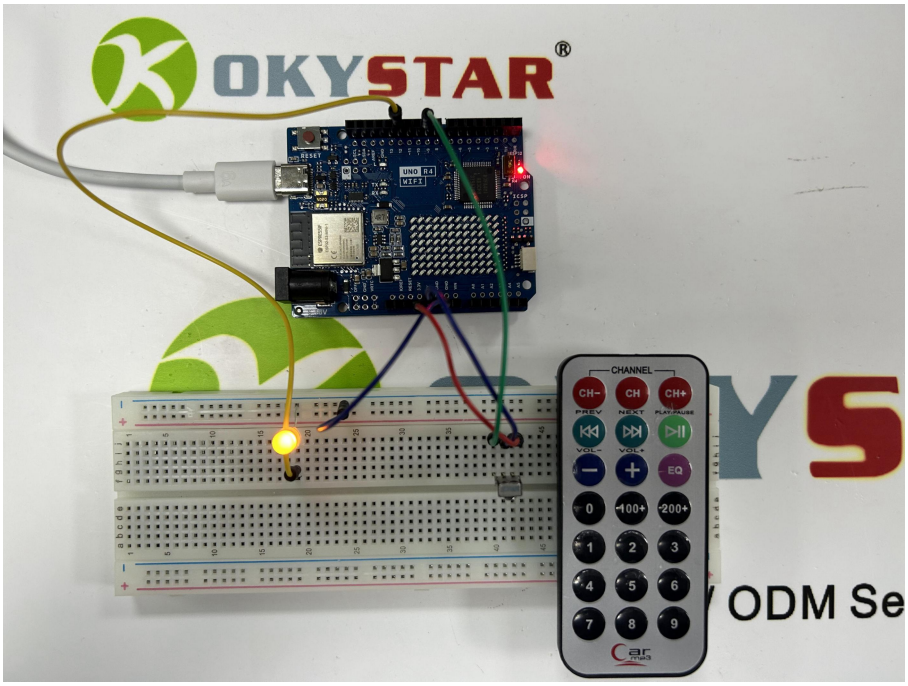
Component Required:

- 1 x **UNO R4 WiFi**
- 1 x **Infrared Receiver Module**
- 1 x **LED**
- 1 x **220Ω Resistor**
- 1 x **400 Tie Points Breadboard**
- 4 x M-M Wires (Male-to-Male Jumper Wires)

Infrared Receiver Module:

The infrared receiver detects signals from an IR remote control. When a button on the remote is pressed, the module outputs a signal to the microcontroller. In this experiment, the signal is used to toggle an LED on and off.

Physical wiring diagram:



Wiring Diagram:

1. Connect the **IR module signal pin** to **D9** on the **UNO R4 WiFi**.
2. Connect the **IR module GND** pin to **GND** on the UNO.
3. Connect the **IR module VCC** pin to **5V** on the UNO.
4. Connect the **LED** anode (longer leg) to **D12** through a **220Ω resistor**.
5. Connect the **LED cathode (shorter leg)** to **GND**.

Code:

```
 #define IR_PIN 9                // Infrared receiver signal pin
#define LED_PIN 12              // LED connected to pin D12

bool ledState = false;      // Variable to track LED state
bool lastSignal = HIGH;     // Previous signal state for debounce

void setup() {
   pinMode(IR_PIN, INPUT);        // Set IR pin as input
   pinMode(LED_PIN, OUTPUT);     // Set LED pin as output
   digitalWrite(LED_PIN, ledState); // Initialize LED state
   Serial.begin(9600);               // Start serial communication
}

void loop() {
   // Read the IR receiver's state
   bool currentSignal = digitalRead(IR_PIN);

   // Detect a falling edge signal (HIGH to LOW transition)
   if (currentSignal == LOW && lastSignal == HIGH) {
      // Toggle LED state
      ledState = !ledState;
```

```
        digitalWrite(LED_PIN, ledState); // Update LED state

        // Output the current state to the serial monitor
        Serial.print("LED is now ");
        Serial.println(ledState ? "ON" : "OFF");
    }

    // Update the previous signal state
    lastSignal = currentSignal;
}
```

Explanation:

1. IR Receiver Detection:

- The IR module reads signals from a remote and outputs a digital signal to the UNO R4 WiFi.

- When the output changes from HIGH to LOW, it indicates that a signal has been received.

2. Toggling LED State:

- The LED's state is toggled (`ON` to `OFF` or vice versa) whenever the IR receiver detects a signal.

3. Debouncing:

- The `lastSignal` variable ensures that the signal is processed only once per button press to avoid rapid toggling caused by signal noise.

4. Serial Monitor:

- The current state of the LED (`ON` or `OFF`) is displayed in the Serial Monitor for debugging purposes.

Notes:

1. Power Supply: Ensure the UNO R4 WiFi is powered via USB or an external power source.

2. IR Signal: Use an infrared remote to test the setup. Any standard remote can work.

3. Serial Monitor: Open the Serial Monitor in the Arduino IDE to view the LED's status after each IR signal.

Lesson 9 Ultrasonic module experiment

Overview:

In this lesson, you will use an ultrasonic sensor to measure the distance of objects and display the results on an LCD screen using the UNO R4 WiFi board.

Component Required:

- 1 x UNO R4 WiFi
- 1 x Ultrasonic sensor module (e.g., HC-SR04)
- 1 x LCD1602 I2C module
- 1 x Breadboard
- Jumper wires (M-M and M-F as needed)

Ultrasonic Sensor:

The ultrasonic sensor measures distance by sending out ultrasonic waves and detecting the reflected signal.

- **Trigger Pin (TRIG):** Sends a short burst of ultrasonic sound.
- **Echo Pin (ECHO):** Receives the reflected signal and calculates the time taken.

LCD1602 I2C Module:

The LCD module with I2C interface simplifies wiring and allows easy integration with Arduino.

- **Address:** The I2C address for the module is usually **0x27** or **0x3F** (adjustable if necessary).

Wiring Diagram:

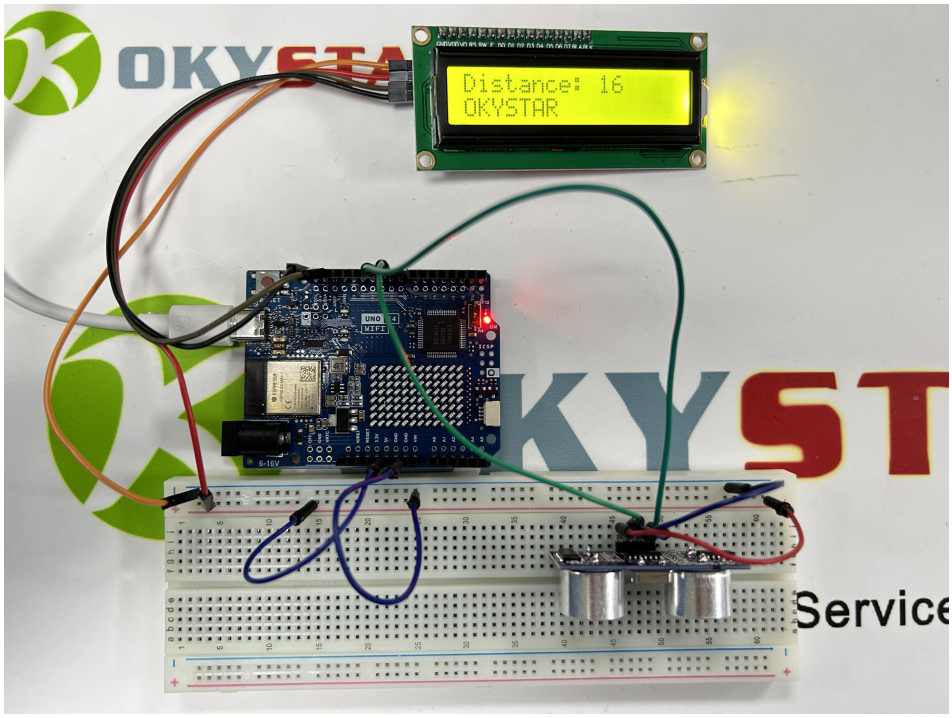
Ultrasonic Sensor:

Sensor Pin	UNO R4 WiFi Pin
VCC	5V
GND	GND
TRIG	D9
ECHO	D10

LCD1602 I2C Module:

LCD Pin	UNO R4 WiFi Pin
VCC	5V
GND	GND
SDA	SDA (A4)
SCL	SCL (A5)

Physical wiring diagram:



Code:

```
   #include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD Initialization: Address can be 0x27 or 0x3F based on your module
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Ultrasonic Sensor Pins
#define TRIG_PIN 9 // Trigger pin connected to D9
#define ECHO_PIN 10 // Echo pin connected to D10

void setup() {
   // Initialize Serial for debugging
```

```
    Serial.begin(9600);

    // Initialize LCD
    lcd.init();                  // Start LCD
    lcd.backlight();          // Turn on backlight
    lcd.setCursor(0, 0); // Move cursor to line 1
    lcd.print("Distance: -- cm"); // Placeholder
    lcd.setCursor(0, 1); // Move cursor to line 2
    lcd.print("OKYSTAR");                // Static text

    // Initialize Ultrasonic Sensor Pins
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
}

void loop() {
    // Send ultrasonic signal
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    // Read echo time
    long duration = pulseIn(ECHO_PIN, HIGH);

    // Calculate distance in cm (speed of sound = 34300 cm/s)
    long distance = duration * 0.034 / 2;

    // Print distance to Serial Monitor
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");

    // Update LCD display with distance
    lcd.setCursor(10, 0); // Move cursor to line 1, column 11
    lcd.print("          ");       // Clear old data
    lcd.setCursor(10, 0);
    lcd.print(distance);      // Display new distance

    // Update every 500ms
    delay(500);
}
```

Lesson 10 Soil temperature and humidity module experiment

Overview：

In this lesson, you will learn how to measure and display soil moisture levels using a soil moisture sensor and an LCD screen. The experiment uses the UNO R4 WiFi to read sensor data and display it on an LCD1602 I2C module.

Component Required:

- 1 x UNO R4 WiFi
- 1 x Soil Moisture Sensor
- 1 x LCD1602 I2C Module
- 1 x 10kΩ Resistor (optional for noise reduction)
- 1 x 400 Tie Points Breadboard
- 4 x M-M Wires (Male-to-Male Jumper Wires)

Soil Moisture Sensor:

The soil moisture sensor measures the moisture level in soil. It outputs an analog signal proportional to the detected moisture level:

- **Dry Soil:** High analog value
- **Wet Soil:** Low analog value

Wiring Diagram:

Soil Moisture Sensor:

Sensor Pin	UNO R4 WiFi Pin
VCC	5V
GND	GND
Signal (AO)	A0

LCD1602 I2C Module:

LCD Pin	UNO R4 WiFi Pin
VCC	5V
GND	GND

LCD Pin	UNO R4 WiFi Pin
SDA	A4
SCL	A5

Physical wiring diagram:



Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize the LCD object (Address: 0x27 or 0x3F; adjust based on your module)
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define soil moisture sensor pin
#define MOISTURE_SENSOR_PIN A0    // Analog pin connected to the soil moisture sensor

void setup() {
  // Initialize LCD
  lcd.init();
  lcd.backlight();    // Turn on the LCD backlight
  lcd.setCursor(0, 0);    // Set cursor to the first line
  lcd.print("Soil Moisture");    // Display title
}

void loop() {
  // Read soil moisture sensor value
  int moistureValue = analogRead(MOISTURE_SENSOR_PIN);

  // Display the soil moisture value on the LCD
  lcd.setCursor(0, 1);    // Set cursor to the second line
  lcd.print("Moisture: ");
  lcd.print(moistureValue);    // Print the sensor value

  // Delay for 500ms to prevent rapid updates
  delay(500);
}
```

Lesson 11 DHT11 temperature module experiment

Overview

In this lesson, you will learn how to measure the temperature using a DHT11 sensor and display the readings on an LCD. The temperature data will be shown on both the Serial Monitor and the LCD screen connected to the UNO R4 WiFi board.

Component Required:

- 1 x **UNO R4 WiFi**
- 1 x **DHT11 Temperature and Humidity Sensor**
- 1 x **16x2 LCD with I2C Interface**
- 1 x **400 Tie Points Breadboard**
- 4 x **M-M Wires** (Male-to-Male Jumper Wires)
- 1 x **220Ω Resistor** (if needed for the LCD backlight)

DHT11 Sensor:

The **DHT11** sensor is a basic digital temperature and humidity sensor. It has a simple interface and provides reliable temperature data. In this experiment, we will only use the temperature measurement capabilities of the sensor.

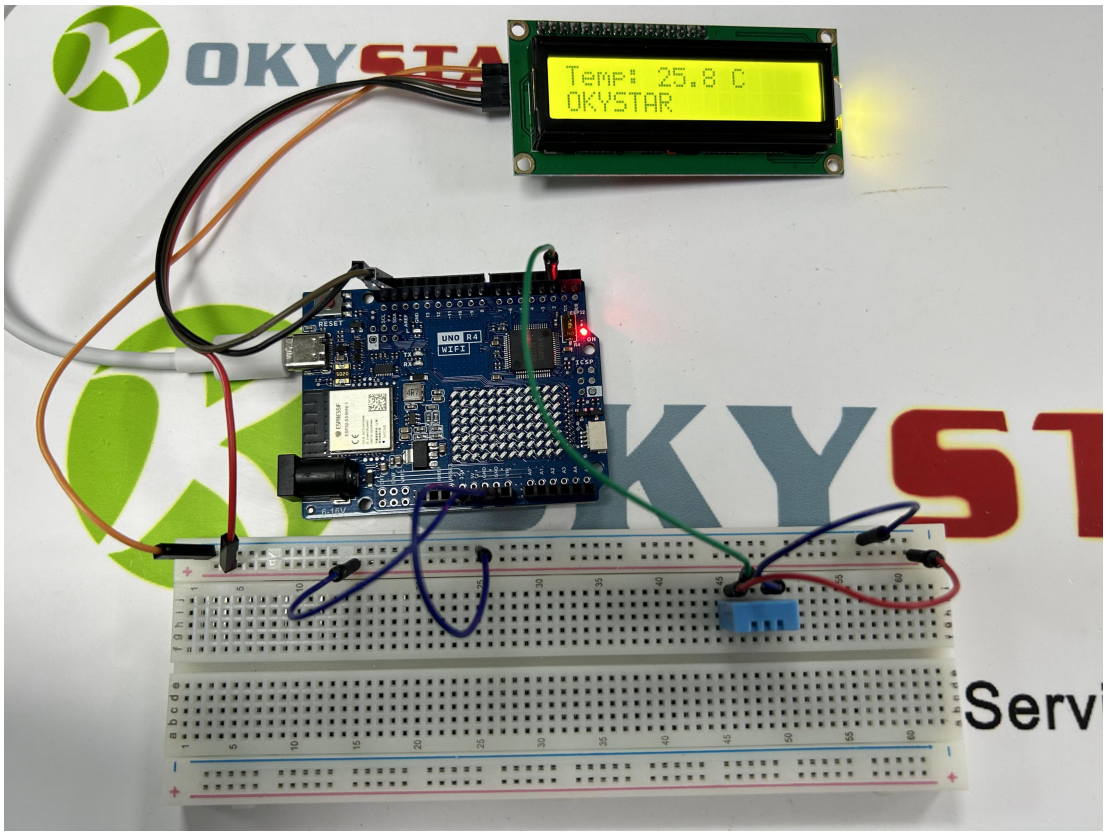
LCD with I2C:

The **16x2 LCD** with an I2C interface allows you to display data in a compact and easy-to-read format. It communicates with the **UNO R4 WiFi** board using the I2C protocol, which requires only two data pins (SDA and SCL).

Wiring Diagram:

- Connect the **DHT11 Sensor** to **D2** on the **UNO R4 WiFi**.
- Connect the **VCC** pin of the DHT11 to **5V** and the **GND** pin to **GND** on the **UNO R4 WiFi**.
- Connect the **LCD** to the **SDA** and **SCL** pins on the **UNO R4 WiFi** (SDA = Pin A4, SCL = Pin A5).

Physical wiring diagram:



Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

// LCD 初始化：地址根据实际模块调整（0x27 或 0x3F）
LiquidCrystal_I2C lcd(0x27, 16, 2);

// DHT 传感器设置
#define DHTPIN 2          // DHT 数据引脚连接到 Arduino 的 D2
#define DHTTYPE DHT11 // 传感器型号
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  // 初始化串口，用于调试
  Serial.begin(9600);
```

```
  // 初始化 LCD
  lcd.init();                  // 启动 LCD
  lcd.backlight();         // 开启背光
  lcd.setCursor(0, 0); // 移动光标到第 1 行
  lcd.print("Temp: -- C"); // 初始化占位符
  lcd.setCursor(0, 1); // 移动光标到第 2 行
  lcd.print("OKYSTAR");       // 显示固定文本

  // 初始化 DHT 传感器
  dht.begin();
}

void loop() {
  // 读取温度
  float temperature = dht.readTemperature();

  // 检查是否读取成功
  if (isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.setCursor(0, 0);
    lcd.print("Temp: ERROR ");
  } else {
    // 打印到串口监视器
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" *C");

    // 在 LCD 上更新温度
    lcd.setCursor(6, 0); // 移动光标到第 1 行第 7 列
    lcd.print("        ");     // 清除旧数据
    lcd.setCursor(6, 0);
    lcd.print(temperature, 1); // 显示温度（保留 1 位小数）
    lcd.print(" C");
  }

  // 每 2 秒更新一次
  delay(2000);
}
```

Lesson 12 LCD1602 I2C screen experiment

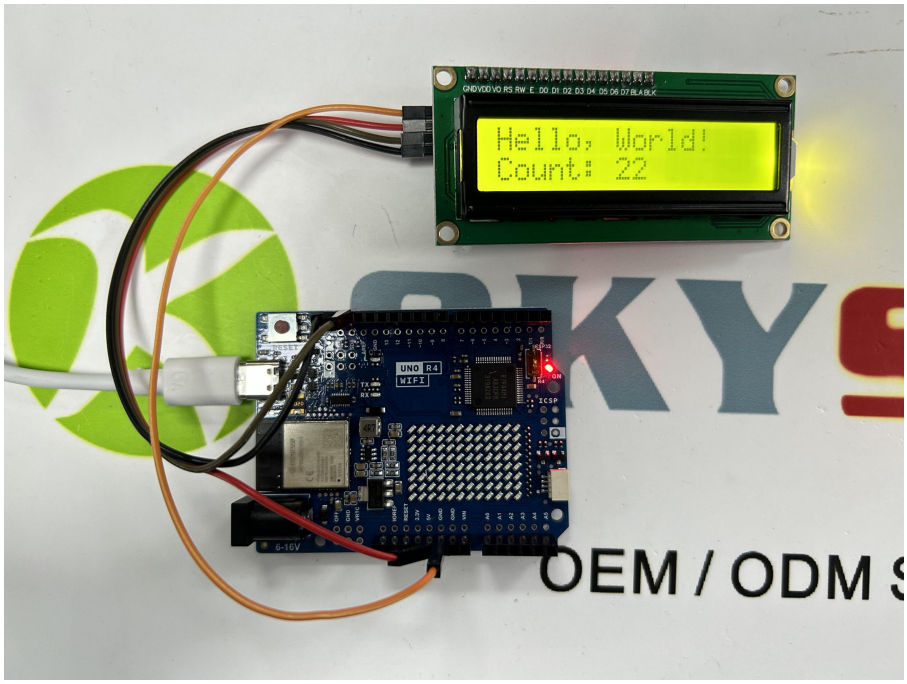
Overview:

In this lesson, you will learn how to use an LCD1602 I2C display with the UNO R4 WiFi to display static and dynamic text. We will use the I2C protocol to communicate with the LCD and display a simple message along with a dynamic counter.

Component Required:

- 1 x UNO R4 WiFi
- 1 x LCD1602 I2C Display
- 1 x 400 Tie Points Breadboard
- 4 x M-M Wires (Male-to-Male Jumper Wires)

Physical wiring diagram:



Code:

```
   #include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize the LCD object. The address may be 0x27 or 0x3F, depending on the module.
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);    // 16 columns, 2 rows

void setup() {
   // Initialize the LCD
   lcd.init();
   lcd.backlight();    // Turn on the backlight
   lcd.setCursor(0, 0);    // Move the cursor to the first row, first column
   lcd.print("Hello, World!");    // Display static text
}

void loop() {
   // Display dynamic counter
   static int counter = 0;
   lcd.setCursor(0, 1);    // Move the cursor to the second row, first column
   lcd.print("Count: ");
   lcd.print(counter++);
   delay(1000);    // Update the counter every second
}
```

Lesson 13 PS2 remote sensing module experiment

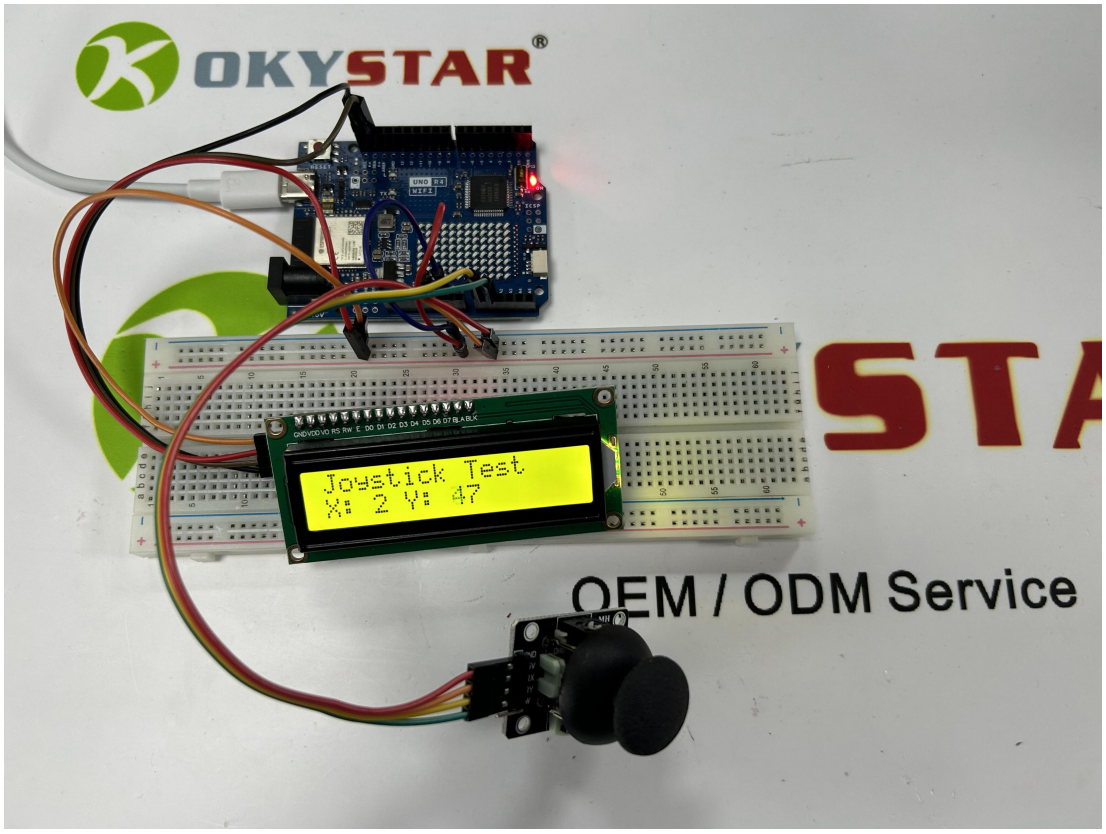
Overview:

In this lesson, you will learn how to use a PS2 joystick module to control a display and read joystick values using the UNO R4 WiFi board. The joystick values (X and Y axis) and button press status will be shown on a 16x2 LCD display.

Component Required:

- 1 x UNO R4 WiFi
- 1 x PS2 Joystick Module
- 1 x 16x2 I2C LCD
- 1 x 400 Tie Points Breadboard
- 3 x M-M Wires (Male-to-Male Jumper Wires)

Physical wiring diagram:



Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize LCD object (address might be 0x27 or 0x3F, check your module)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Joystick module pin definitions
#define VRX A0    // X-axis input
#define VRY A1    // Y-axis input
#define SW      2      // Button input

void setup() {
  // Initialize LCD
  lcd.init();
  lcd.backlight();    // Turn on the backlight
  lcd.setCursor(0, 0);    // Move cursor to row 1, column 1
  lcd.print("Joystick Test");    // Display text on LCD

  pinMode(SW, INPUT_PULLUP); // Set button pin as input with internal pull-up resistor
}

void loop() {
  // Read joystick module values for X, Y axis, and button status
  int xValue = analogRead(VRX);    // Read X-axis value
  int yValue = analogRead(VRY);    // Read Y-axis value
  int buttonState = digitalRead(SW);    // Read button status

  // Display joystick values on the LCD
  lcd.setCursor(0, 1);    // Move cursor to row 2, column 1
  lcd.print("X: ");
  lcd.print(xValue);    // Display X-axis value
  lcd.print(" Y: ");
  lcd.print(yValue);    // Display Y-axis value

  // If button is pressed, display "Button Pressed"
  if (buttonState == LOW) {
    lcd.setCursor(0, 0);    // Move cursor to row 1, column 1
    lcd.print("Button Pressed!    ");
  } else {
    lcd.setCursor(0, 0);    // Move cursor to row 1, column 1
    lcd.print("Joystick Test      ");
  }
```

```
    delay(200);    // Delay for 200ms to allow LCD update
}
```